

---

# **Software Requirements Specification**

for

## **Security on mHealth applications (SECM-CAT)**

**Version 1.0 approved**

**Prepared by Software Engineering Research Group**

**Murcia University**

**Jan 2026**

## Table of Contents

<b>Table of Contents .....</b>	<b>ii</b>
<b>Revision History .....</b>	<b>xvi</b>
<b>1. Introduction.....</b>	<b>1</b>
1.1    SRS overview .....	1
1.1.1    Metainformation associated with the requirements .....	1
1.2    Purpose .....	3
1.3    Scope .....	4
1.4    Product perspective .....	7
1.4.1    System interfaces .....	7
1.4.2    User interfaces .....	7
1.4.3    Hardware interfaces .....	7
1.4.4    Software interfaces.....	7
1.4.5    Communications interfaces.....	7
1.4.6    Memory constraints .....	8
1.4.7    Operations .....	8
1.4.8    Site adaptation requirements.....	8
1.4.9    Interfaces with services.....	8
1.5    Product functions.....	8
1.6    User characteristics.....	8
1.7    Limitations.....	9
1.8    Assumptions and dependencies.....	9
1.9    Apportioning of requirements .....	9
<b>2. Requirements.....</b>	<b>9</b>
2.1    Specified requirements .....	9
2.2    External Interfaces.....	10
2.3    Functions .....	10
2.4    Usability requirements.....	11
2.5    Performance requirements .....	11
2.6    Logical database .....	11
2.7    Design constraints .....	11
2.8    Standards compliance.....	11
2.9    Software system attributes.....	12
2.9.1    Reliability.....	12
2.9.2    Availability .....	12
2.9.3    Security .....	12
2.9.3.1    Improper Credential Usage.....	12
<b>PUID: [SECM-CAT-ICU-001]</b> .....	12
<b>PUID: [SECM-CAT-ICU-002]</b> .....	14
<b>PUID: [SECM-CAT-ICU-003]</b> .....	15
<b>PUID: [SECM-CAT-ICU-004]</b> .....	16
2.9.3.2    Inadequate Supply Chain Security .....	17
<b>PUID: [SECM-CAT-ISU-001]</b> .....	17
<b>PUID: [SECM-CAT-ISU-002]</b> .....	18

<b>PUID:</b> [SECM-CAT-ISU-003]	19
<b>PUID:</b> [SECM-CAT-ISU-004]	20
<b>PUID:</b> [SECM-CAT-ISU-005]	22
<b>PUID:</b> [SECM-CAT-ISU-006]	23
<b>PUID:</b> [SECM-CAT-ISU-007]	24
<b>PUID:</b> [SECM-CAT-ISU-008]	25
<b>PUID:</b> [SECM-CAT-ISU-009]	26
<b>PUID:</b> [SECM-CAT-ISU-010]	28
<b>PUID:</b> [SECM-CAT-ISU-011]	29
<b>PUID:</b> [SECM-CAT-ISU-012]	30
<b>PUID:</b> [SECM-CAT-ISU-013]	32
<b>PUID:</b> [SECM-CAT-ISU-014]	33
<b>PUID:</b> [SECM-CAT-ISU-015]	34
<b>PUID:</b> [SECM-CAT-ISU-016]	36
<b>PUID:</b> [SECM-CAT-ISU-017]	37
<b>PUID:</b> [SECM-CAT-ISU-018]	38
<b>PUID:</b> [SECM-CAT-ISU-019]	39
<b>PUID:</b> [SECM-CAT-ISU-020]	40
<b>PUID:</b> [SECM-CAT-ISU-021]	41
<b>PUID:</b> [SECM-CAT-ISU-022]	42
<b>PUID:</b> [SECM-CAT-ISU-023]	43
<b>PUID:</b> [SECM-CAT-ISU-024]	45
<b>PUID:</b> [SECM-CAT-ISU-025]	46
<b>PUID:</b> [SECM-CAT-ISU-026]	47
<b>PUID:</b> [SECM-CAT-ISU-027]	48
<b>PUID:</b> [SECM-CAT-ISU-028]	50
<b>PUID:</b> [SECM-CAT-ISU-029]	52
<b>PUID:</b> [SECM-CAT-ISU-030]	54
<b>PUID:</b> [SECM-CAT-ISU-031]	55
<b>PUID:</b> [SECM-CAT-ISU-032]	57
<b>PUID:</b> [SECM-CAT-ISU-033]	58
<b>PUID:</b> [SECM-CAT-ISU-034]	59
<b>PUID:</b> [SECM-CAT-ISU-035]	61
<b>PUID:</b> [SECM-CAT-ISU-036]	62
<b>PUID:</b> [SECM-CAT-ISU-037]	63
<b>PUID:</b> [SECM-CAT-ISU-038]	64

**Requirement description:** The project must avoid exposing signing credentials in the source code and use external mechanisms (e.g., environment variables) to supply

keystore secrets during the build process, so that the signing flow can be integrated with secure key management systems (such as CI/CD pipelines, vaults, or HSMs)....	64
<b>PUID: [SECM-CAT-ISU-039] .....</b>	<b>65</b>
<b>PUID: [SECM-CAT-ISU-040] .....</b>	<b>66</b>
<b>PUID: [SECM-CAT-ISU-041] .....</b>	<b>68</b>
<b>PUID: [SECM-CAT-ISU-042] .....</b>	<b>69</b>
<b>PUID: [SECM-CAT-ISU-043] .....</b>	<b>69</b>
<b>PUID: [SECM-CAT-ISU-044] .....</b>	<b>71</b>
<b>PUID: [SECM-CAT-ISU-045] .....</b>	<b>72</b>
<b>PUID: [SECM-CAT-ISU-046] .....</b>	<b>73</b>
<b>PUID: [SECM-CAT-ISU-047] .....</b>	<b>75</b>
<b>PUID: [SECM-CAT-ISU-048] .....</b>	<b>76</b>
<b>PUID: [SECM-CAT-ISU-049] .....</b>	<b>77</b>
<b>PUID: [SECM-CAT-ISU-050] .....</b>	<b>78</b>
<b>PUID: [SECM-CAT-ISU-051] .....</b>	<b>79</b>
<b>PUID: [SECM-CAT-ISU-052] .....</b>	<b>80</b>
<b>PUID: [SECM-CAT-ISU-053] .....</b>	<b>81</b>
<b>PUID: [SECM-CAT-ISU-054] .....</b>	<b>82</b>
<b>PUID: [SECM-CAT-ISU-055] .....</b>	<b>84</b>
<b>PUID: [SECM-CAT-ISU-056] .....</b>	<b>85</b>
<b>PUID: [SECM-CAT-ISU-057] .....</b>	<b>87</b>
<b>PUID: [SECM-CAT-ISU-058] .....</b>	<b>88</b>
<b>PUID: [SECM-CAT-ISU-059] .....</b>	<b>89</b>
<b>2.9.3.3 Insecure Authentication/Authorization.....</b>	<b>90</b>
<b>    PUID: [SECM-CAT-IIA-001].....</b>	<b>90</b>
<b>    PUID: [SECM-CAT-IIA-002].....</b>	<b>92</b>
<b>    PUID: [SECM-CAT-IIA-003].....</b>	<b>93</b>
<b>    PUID: [SECM-CAT-IIA-004].....</b>	<b>94</b>
<b>    PUID: [SECM-CAT-IIA-005].....</b>	<b>96</b>
<b>    PUID: [SECM-CAT-IIA-006].....</b>	<b>97</b>
<b>    PUID: [SECM-CAT-IIA-007].....</b>	<b>98</b>
<b>    PUID: [SECM-CAT-IIA-008].....</b>	<b>100</b>
<b>    PUID: [SECM-CAT-IIA-009].....</b>	<b>101</b>
<b>    PUID: [SECM-CAT-IIA-010].....</b>	<b>102</b>
<b>    PUID: [SECM-CAT-IIA-011].....</b>	<b>104</b>
<b>    PUID: [SECM-CAT-IIA-012].....</b>	<b>105</b>
<b>    PUID: [SECM-CAT-IIA-012].....</b>	<b>107</b>
<b>    PUID: [SECM-CAT-IIA-013].....</b>	<b>108</b>

<b>PUID:</b> [SECM-CAT-IIA-014].....	110
<b>PUID:</b> [SECM-CAT-IIA-015].....	112
<b>PUID:</b> [SECM-CAT-IIA-016].....	113
<b>PUID:</b> [SECM-CAT-IIA-017].....	114
<b>PUID:</b> [SECM-CAT-IIA-018].....	115
<b>PUID:</b> [SECM-CAT-IIA-019].....	117
<b>PUID:</b> [SECM-CAT-IIA-020].....	118
<b>PUID:</b> [SECM-CAT-IIA-021].....	119
<b>PUID:</b> [SECM-CAT-IIA-022].....	121
<b>PUID:</b> [SECM-CAT-IIA-023].....	122
<b>PUID:</b> [SECM-CAT-IIA-024].....	124
<b>PUID:</b> [SECM-CAT-IIA-025].....	126
<b>PUID:</b> [SECM-CAT-IIA-026].....	127
<b>PUID:</b> [SECM-CAT-IIA-027].....	128
<b>PUID:</b> [SECM-CAT-IIA-028].....	129
<b>PUID:</b> [SECM-CAT-IIA-029].....	130
<b>PUID:</b> [SECM-CAT-IIA-030].....	132
<b>PUID:</b> [SECM-CAT-IIA-031].....	133
<b>PUID:</b> [SECM-CAT-IIA-032].....	134
<b>PUID:</b> [SECM-CAT-IIA-033].....	136
<b>PUID:</b> [SECM-CAT-IIA-034].....	137
<b>PUID:</b> [SECM-CAT-IIA-035].....	139
<b>PUID:</b> [SECM-CAT-IIA-036].....	140
<b>PUID:</b> [SECM-CAT-IIA-037].....	141
<b>PUID:</b> [SECM-CAT-IIA-038].....	143
<b>PUID:</b> [SECM-CAT-IIA-039].....	144
<b>PUID:</b> [SECM-CAT-IIA-040].....	145
<b>PUID:</b> [SECM-CAT-IIA-041].....	147
<b>PUID:</b> [SECM-CAT-IIA-042].....	148
<b>PUID:</b> [SECM-CAT-IIA-043].....	149
<b>PUID:</b> [SECM-CAT-IIA-044].....	151
<b>PUID:</b> [SECM-CAT-IIA-045].....	152
<b>PUID:</b> [SECM-CAT-IIA-046].....	153
<b>PUID:</b> [SECM-CAT-IIA-047].....	155
<b>PUID:</b> [SECM-CAT-IIA-048].....	156
<b>PUID:</b> [SECM-CAT-IIA-049].....	157
<b>PUID:</b> [SECM-CAT-IIA-050].....	159
<b>PUID:</b> [SECM-CAT-IIA-051].....	160

<b>PUID:</b> [SECM-CAT-IIA-052].....	162
<b>PUID:</b> [SECM-CAT-IIA-053].....	163
<b>PUID:</b> [SECM-CAT-IIA-054].....	164
<b>PUID:</b> [SECM-CAT-IIA-055].....	165
<b>PUID:</b> [SECM-CAT-IIA-056].....	166
<b>PUID:</b> [SECM-CAT-IIA-057].....	168
<b>PUID:</b> [SECM-CAT-IIA-058].....	169
<b>PUID:</b> [SECM-CAT-IIA-059].....	171
<b>PUID:</b> [SECM-CAT-IIA-060].....	173
<b>PUID:</b> [SECM-CAT-IIA-061].....	174
<b>PUID:</b> [SECM-CAT-IIA-062].....	176
<b>PUID:</b> [SECM-CAT-IIA-063].....	178
<b>PUID:</b> [SECM-CAT-IIA-064].....	179
<b>PUID:</b> [SECM-CAT-IIA-065].....	181
<b>PUID:</b> [SECM-CAT-IIA-066].....	183
<b>PUID:</b> [SECM-CAT-IIA-067].....	184
<b>PUID:</b> [SECM-CAT-IIA-068].....	185
<b>PUID:</b> [SECM-CAT-IIA-069].....	187
<b>PUID:</b> [SECM-CAT-IIA-070].....	189
<b>PUID:</b> [SECM-CAT-IIA-071].....	190
<b>PUID:</b> [SECM-CAT-IIA-072].....	192
<b>PUID:</b> [SECM-CAT-IIA-073].....	194
<b>PUID:</b> [SECM-CAT-IIA-074].....	195
<b>PUID:</b> [SECM-CAT-IIA-075].....	197
<b>PUID:</b> [SECM-CAT-IIA-076].....	198
<b>PUID:</b> [SECM-CAT-IIA-077].....	200
<b>PUID:</b> [SECM-CAT-IIA-078].....	201
<b>PUID:</b> [SECM-CAT-IIA-079].....	202
<b>PUID:</b> [SECM-CAT-IIA-080].....	204
<b>PUID:</b> [SECM-CAT-IIA-081].....	206
<b>PUID:</b> [SECM-CAT-IIA-082].....	207
<b>PUID:</b> [SECM-CAT-IIA-083].....	209
<b>PUID:</b> [SECM-CAT-IIA-084].....	210
<b>PUID:</b> [SECM-CAT-IIA-085].....	211
<b>PUID:</b> [SECM-CAT-IIA-086].....	212
<b>PUID:</b> [SECM-CAT-IIA-087].....	213
<b>PUID:</b> [SECM-CAT-IIA-088].....	214
<b>PUID:</b> [SECM-CAT-IIA-089].....	215

<b>PUID:</b> [SECM-CAT-IIA-090].....	216
<b>PUID:</b> [SECM-CAT-IIA-091].....	217
<b>PUID:</b> [SECM-CAT-IIA-092].....	218
<b>PUID:</b> [SECM-CAT-IIA-093].....	219
<b>PUID:</b> [SECM-CAT-IIA-094].....	220
<b>PUID:</b> [SECM-CAT-IIA-095].....	221
<b>PUID:</b> [SECM-CAT-IIA-096].....	222
<b>PUID:</b> [SECM-CAT-IIA-097].....	223
<b>PUID:</b> [SECM-CAT-IIA-098].....	224
<b>PUID:</b> [SECM-CAT-IIA-99].....	226
<b>PUID:</b> [SECM-CAT-IIA-100].....	227
<b>PUID:</b> [SECM-CAT-IIA-101].....	228
<b>PUID:</b> [SECM-CAT-IIA-102].....	229
<b>PUID:</b> [SECM-CAT-IIA-103].....	230
<b>PUID:</b> [SECM-CAT-IIA-104].....	232
<b>PUID:</b> [SECM-CAT-IIA-105].....	233
<b>PUID:</b> [SECM-CAT-IIA-106].....	234
<b>PUID:</b> [SECM-CAT-IIA-107].....	235
<b>PUID:</b> [SECM-CAT-IIA-108].....	236
<b>PUID:</b> [SECM-CAT-IIA-109].....	237
<b>PUID:</b> [SECM-CAT-IIA-110].....	238
<b>PUID:</b> [SECM-CAT-IIA-111].....	239
<b>PUID:</b> [SECM-CAT-IIA-112].....	240
<b>PUID:</b> [SECM-CAT-IIA-113].....	241
<b>PUID:</b> [SECM-CAT-IIA-114].....	242
<b>PUID:</b> [SECM-CAT-IIA-115].....	243
<b>PUID:</b> [SECM-CAT-IIA-116].....	244
<b>PUID:</b> [SECM-CAT-IIA-117].....	246
<b>PUID:</b> [SECM-CAT-IIA-118].....	247
<b>PUID:</b> [SECM-CAT-IIA-119].....	248
<b>PUID:</b> [SECM-CAT-IIA-120].....	249
<b>PUID:</b> [SECM-CAT-IIA-121].....	250
<b>PUID:</b> [SECM-CAT-IIA-122].....	251
<b>PUID:</b> [SECM-CAT-IIA-123].....	252
<b>PUID:</b> [SECM-CAT-IIA-124].....	254
2.9.3.4 Insufficient Input/Output Validation .....	255
<b>PUID:</b> [SECM-CAT-IOV-001].....	255
<b>PUID:</b> [SECM-CAT-IOV-002].....	257

<b>PUID:</b> [SECM-CAT-IOV-003].....	258
<b>PUID:</b> [SECM-CAT-IOV-004].....	259
<b>PUID:</b> [SECM-CAT-IOV-005].....	260
<b>PUID:</b> [SECM-CAT-IOV-006].....	262
<b>PUID:</b> [SECM-CAT-IOV-007].....	263
<b>PUID:</b> [SECM-CAT-IOV-008].....	265
<b>PUID:</b> [SECM-CAT-IOV-009].....	266
<b>PUID:</b> [SECM-CAT-IOV-010].....	267
<b>PUID:</b> [SECM-CAT-IOV-011].....	268
<b>PUID:</b> [SECM-CAT-IOV-012].....	271
<b>PUID:</b> [SECM-CAT-IOV-013].....	272
<b>PUID:</b> [SECM-CAT-IOV-014].....	273
<b>PUID:</b> [SECM-CAT-IOV-015].....	275
<b>PUID:</b> [SECM-CAT-IOV-016].....	276
<b>PUID:</b> [SECM-CAT-IOV-017].....	278
<b>PUID:</b> [SECM-CAT-IOV-018].....	279
<b>PUID:</b> [SECM-CAT-IOV-019].....	280
<b>PUID:</b> [SECM-CAT-IOV-020].....	281
<b>PUID:</b> [SECM-CAT-IOV-021].....	282
<b>PUID:</b> [SECM-CAT-IOV-022].....	283
<b>PUID:</b> [SECM-CAT-IOV-023].....	285
<b>PUID:</b> [SECM-CAT-IOV-024].....	286
<b>PUID:</b> [SECM-CAT-IOV-025].....	287
<b>PUID:</b> [SECM-CAT-IOV-026].....	288
2.9.3.5 Insecure Communication.....	289
<b>PUID:</b> [SECM-CAT-ICO-001].....	289
<b>PUID:</b> [SECM-CAT-ICO-002].....	290
<b>PUID:</b> [SECM-CAT-ICO-003].....	292
<b>PUID:</b> [SECM-CAT-ICO-004].....	293
<b>PUID:</b> [SECM-CAT-ICO-005].....	294
<b>PUID:</b> [SECM-CAT-ICO-006].....	295
<b>PUID:</b> [SECM-CAT-ICO-007].....	296
<b>PUID:</b> [SECM-CAT-ICO-008].....	298
<b>PUID:</b> [SECM-CAT-ICO-009].....	301
<b>PUID:</b> [SECM-CAT-ICO-010].....	302
<b>PUID:</b> [SECM-CAT-ICO-011].....	303
<b>PUID:</b> [SECM-CAT-ICO-012].....	305
<b>PUID:</b> [SECM-CAT-ICO-013].....	307

<b>PUID: [SECM-CAT-ICO-014]</b> .....	308
<b>PUID: [SECM-CAT-ICO-015]</b> .....	310
<b>PUID: [SECM-CAT-ICO-016]</b> .....	311
<b>PUID: [SECM-CAT-ICO-017]</b> .....	313
<b>PUID: [SECM-CAT-ICO-018]</b> .....	315
<b>PUID: [SECM-CAT-ICO-019]</b> .....	317
<b>PUID: [SECM-CAT-ICO-020]</b> .....	318
<b>PUID: [SECM-CAT-ICO-021]</b> .....	320
<b>PUID: [SECM-CAT-ICO-022]</b> .....	321
<b>PUID: [SECM-CAT-ICO-023]</b> .....	323
<b>PUID: [SECM-CAT-ICO-024]</b> .....	324
<b>PUID: [SECM-CAT-ICO-025]</b> .....	326
<b>PUID: [SECM-CAT-ICO-026]</b> .....	327
<b>PUID: [SECM-CAT-ICO-027]</b> .....	329
<b>PUID: [SECM-CAT-ICO-028]</b> .....	331
<b>PUID: [SECM-CAT-ICO-029]</b> .....	332
<b>PUID: [SECM-CAT-ICO-030]</b> .....	333
<b>PUID: [SECM-CAT-ICO-031]</b> .....	335
<b>PUID: [SECM-CAT-ICO-032]</b> .....	336
<b>2.9.3.6 Inadequate Privacy Controls</b> .....	337
<b>PUID: [SECM-CAT-IPC-001]</b> .....	337
<b>PUID: [SECM-CAT-IPC-002]</b> .....	339
<b>PUID: [SECM-CAT-IPC-003]</b> .....	341
<b>PUID: [SECM-CAT-IPC-004]</b> .....	342
<b>PUID: [SECM-CAT-IPC-005]</b> .....	343
<b>PUID: [SECM-CAT-IPC-006]</b> .....	344
<b>PUID: [SECM-CAT-IPC-007]</b> .....	346
<b>PUID: [SECM-CAT-IPC-008]</b> .....	347
<b>PUID: [SECM-CAT-IPC-009]</b> .....	348
<b>PUID: [SECM-CAT-IPC-010]</b> .....	350
<b>PUID: [SECM-CAT-IPC-011]</b> .....	351
<b>PUID: [SECM-CAT-IPC-012]</b> .....	352
<b>PUID: [SECM-CAT-IPC-013]</b> .....	354
<b>PUID: [SECM-CAT-IPC-014]</b> .....	355
<b>PUID: [SECM-CAT-IPC-015]</b> .....	356
<b>PUID: [SECM-CAT-IPC-016]</b> .....	358
<b>PUID: [SECM-CAT-IPC-017]</b> .....	359
<b>PUID: [SECM-CAT-IPC-018]</b> .....	360

<b>PUID:</b> [SECM-CAT-IPC-019].....	361
<b>PUID:</b> [SECM-CAT-IPC-020].....	363
<b>PUID:</b> [SECM-CAT-IPC-021].....	364
<b>PUID:</b> [SECM-CAT-IPC-022].....	365
<b>PUID:</b> [SECM-CAT-IPC-023].....	366
<b>PUID:</b> [SECM-CAT-IPC-024].....	368
<b>PUID:</b> [SECM-CAT-IPC-025].....	369
<b>PUID:</b> [SECM-CAT-IPC-026].....	371
<b>PUID:</b> [SECM-CAT-IPC-027].....	372
<b>PUID:</b> [SECM-CAT-IPC-028].....	374
<b>PUID:</b> [SECM-CAT-IPC-029].....	375
<b>PUID:</b> [SECM-CAT-IPC-030].....	376
<b>PUID:</b> [SECM-CAT-IPC-031].....	378
<b>PUID:</b> [SECM-CAT-IPC-032].....	380
<b>PUID:</b> [SECM-CAT-IPC-033].....	381
<b>PUID:</b> [SECM-CAT-IPC-034].....	383
<b>PUID:</b> [SECM-CAT-IPC-035].....	384
<b>PUID:</b> [SECM-CAT-IPC-036].....	385
<b>PUID:</b> [SECM-CAT-IPC-037].....	387
<b>PUID:</b> [SECM-CAT-IPC-038].....	388
<b>PUID:</b> [SECM-CAT-IPC-039].....	389
<b>PUID:</b> [SECM-CAT-IPC-040].....	390
<b>PUID:</b> [SECM-CAT-IPC-041].....	392
<b>PUID:</b> [SECM-CAT-IPC-042].....	394
2.9.3.7 Insufficient Binary Protections.....	395
<b>PUID:</b> [SECM-CAT-IBP-001].....	395
<b>PUID:</b> [SECM-CAT-IBP-002].....	396
<b>PUID:</b> [SECM-CAT-IBP-003].....	398
<b>PUID:</b> [SECM-CAT-IBP-004].....	399
<b>PUID:</b> [SECM-CAT-IBP-005].....	400
<b>PUID:</b> [SECM-CAT-IBP-006].....	401
<b>PUID:</b> [SECM-CAT-IBP-007].....	403
<b>PUID:</b> [SECM-CAT-IBP-008].....	405
<b>PUID:</b> [SECM-CAT-IBP-009].....	406
<b>PUID:</b> [SECM-CAT-IBP-010].....	407
<b>PUID:</b> [SECM-CAT-IBP-011].....	409
<b>PUID:</b> [SECM-CAT-IBP-012].....	411
<b>PUID:</b> [SECM-CAT-IBP-013].....	413

<b>PUID:</b> [SECM-CAT-IBP-014].....	414
<b>PUID:</b> [SECM-CAT-IBP-015].....	415
<b>PUID:</b> [SECM-CAT-IBP-016].....	416
<b>PUID:</b> [SECM-CAT-IBP-017].....	417
<b>PUID:</b> [SECM-CAT-IBP-018].....	419
<b>PUID:</b> [SECM-CAT-IBP-019].....	420
<b>PUID:</b> [SECM-CAT-IBP-020].....	422
<b>PUID:</b> [SECM-CAT-IBP-021].....	423
<b>PUID:</b> [SECM-CAT-IBP-022].....	425
<b>PUID:</b> [SECM-CAT-IBP-023].....	426
<b>PUID:</b> [SECM-CAT-IBP-024].....	427
<b>PUID:</b> [SECM-CAT-IBP-025].....	428
<b>PUID:</b> [SECM-CAT-IBP-026].....	429
<b>PUID:</b> [SECM-CAT-IBP-027].....	431
<b>PUID:</b> [SECM-CAT-IBP-028].....	432
<b>PUID:</b> [SECM-CAT-IBP-029].....	433
<b>PUID:</b> [SECM-CAT-IBP-030].....	435
<b>PUID:</b> [SECM-CAT-IBP-031].....	436
<b>PUID:</b> [SECM-CAT-IBP-032].....	437
<b>PUID:</b> [SECM-CAT-IBP-033].....	438
<b>PUID:</b> [SECM-CAT-IBP-034].....	439
<b>2.9.3.8 Security Misconfiguration</b> .....	440
<b>PUID:</b> [SECM-CAT-SMC-001] .....	440
<b>PUID:</b> [SECM-CAT-SMC-002] .....	442
<b>PUID:</b> [SECM-CAT-SMC-003] .....	443
<b>PUID:</b> [SECM-CAT-SMC-004] .....	445
<b>PUID:</b> [SECM-CAT-SMC-005] .....	446
<b>PUID:</b> [SECM-CAT-SMC-006] .....	447
<b>PUID:</b> [SECM-CAT-SMC-007] .....	448
<b>PUID:</b> [SECM-CAT-SMC-008] .....	449
<b>PUID:</b> [SECM-CAT-SMC-009] .....	451
<b>PUID:</b> [SECM-CAT-SMC-010] .....	452
<b>PUID:</b> [SECM-CAT-SMC-011] .....	453
<b>PUID:</b> [SECM-CAT-SMC-012] .....	454
<b>PUID:</b> [SECM-CAT-SMC-013] .....	455
<b>PUID:</b> [SECM-CAT-SMC-014] .....	457
<b>PUID:</b> [SECM-CAT-SMC-015] .....	458
<b>PUID:</b> [SECM-CAT-SMC-016] .....	460

<b>PUID:</b> [SECM-CAT-SMC-017] .....	461
<b>PUID:</b> [SECM-CAT-SMC-018] .....	462
<b>PUID:</b> [SECM-CAT-SMC-019] .....	464
<b>PUID:</b> [SECM-CAT-SMC-020] .....	465
<b>PUID:</b> [SECM-CAT-SMC-021] .....	466
<b>PUID:</b> [SECM-CAT-SMC-022] .....	467
<b>PUID:</b> [SECM-CAT-SMC-023] .....	468
<b>PUID:</b> [SECM-CAT-SMC-024] .....	469
<b>PUID:</b> [SECM-CAT-SMC-025] .....	470
<b>PUID:</b> [SECM-CAT-SMC-026] .....	471
<b>PUID:</b> [SECM-CAT-SMC-027] .....	473
<b>PUID:</b> [SECM-CAT-SMC-028] .....	474
<b>PUID:</b> [SECM-CAT-SMC-029] .....	475
<b>PUID:</b> [SECM-CAT-SMC-030] .....	476
<b>PUID:</b> [SECM-CAT-SMC-031] .....	477
<b>PUID:</b> [SECM-CAT-SMC-032] .....	478
<b>PUID:</b> [SECM-CAT-SMC-033] .....	480
<b>PUID:</b> [SECM-CAT-SMC-034] .....	481
<b>PUID:</b> [SECM-CAT-SMC-035] .....	482
<b>PUID:</b> [SECM-CAT-SMC-036] .....	483
<b>PUID:</b> [SECM-CAT-SMC-037] .....	484
<b>PUID:</b> [SECM-CAT-SMC-038] .....	485
<b>PUID:</b> [SECM-CAT-SMC-039] .....	487
<b>PUID:</b> [SECM-CAT-SMC-040] .....	488
<b>PUID:</b> [SECM-CAT-SMC-041] .....	489
<b>2.9.3.9 Insecure Data Storage</b> .....	490
<b>PUID:</b> [SECM-CAT-IDS-001] .....	490
<b>PUID:</b> [SECM-CAT-IDS-002] .....	492
<b>PUID:</b> [SECM-CAT-IDS-003] .....	493
<b>PUID:</b> [SECM-CAT-IDS-004] .....	494
<b>PUID:</b> [SECM-CAT-IDS-005] .....	495
<b>PUID:</b> [SECM-CAT-IDS-006] .....	497
<b>PUID:</b> [SECM-CAT-IDS-007] .....	498
<b>PUID:</b> [SECM-CAT-IDS-008] .....	499
<b>PUID:</b> [SECM-CAT-IDS-009] .....	500
<b>PUID:</b> [SECM-CAT-IDS-010] .....	501
<b>PUID:</b> [SECM-CAT-IDS-011] .....	503
<b>PUID:</b> [SECM-CAT-IDS-012] .....	504

<b>PUID:</b> [SECM-CAT-IDS-013]	505
<b>PUID:</b> [SECM-CAT-IDS-014]	506
<b>PUID:</b> [SECM-CAT-IDS-015]	507
<b>PUID:</b> [SECM-CAT-IDS-016]	509
<b>PUID:</b> [SECM-CAT-IDS-017]	510
<b>PUID:</b> [SECM-CAT-IDS-018]	511
<b>PUID:</b> [SECM-CAT-IDS-019]	512
<b>PUID:</b> [SECM-CAT-IDS-020]	513
<b>PUID:</b> [SECM-CAT-IDS-021]	514
<b>PUID:</b> [SECM-CAT-IDS-022]	515
<b>PUID:</b> [SECM-CAT-IDS-023]	516
<b>PUID:</b> [SECM-CAT-IDS-024]	517
<b>PUID:</b> [SECM-CAT-IDS-025]	519
<b>PUID:</b> [SECM-CAT-IDS-026]	520
<b>PUID:</b> [SECM-CAT-IDS-027]	521
<b>PUID:</b> [SECM-CAT-IDS-028]	522
<b>PUID:</b> [SECM-CAT-IDS-029]	525
<b>PUID:</b> [SECM-CAT-IDS-030]	527
<b>PUID:</b> [SECM-CAT-IDS-031]	528
<b>PUID:</b> [SECM-CAT-IDS-032]	530
<b>PUID:</b> [SECM-CAT-IDS-033]	531
<b>PUID:</b> [SECM-CAT-IDS-034]	532
<b>PUID:</b> [SECM-CAT-IDS-035]	533
<b>PUID:</b> [SECM-CAT-IDS-036]	534
<b>PUID:</b> [SECM-CAT-IDS-037]	535
<b>PUID:</b> [SECM-CAT-IDS-038]	537
<b>PUID:</b> [SECM-CAT-IDS-039]	538
<b>PUID:</b> [SECM-CAT-IDS-040]	539
<b>PUID:</b> [SECM-CAT-IDS-041]	540
<b>PUID:</b> [SECM-CAT-IDS-042]	541
<b>PUID:</b> [SECM-CAT-IDS-043]	542
<b>PUID:</b> [SECM-CAT-IDS-044]	544
<b>PUID:</b> [SECM-CAT-IDS-045]	545
<b>PUID:</b> [SECM-CAT-IDS-046]	546
<b>PUID:</b> [SECM-CAT-IDS-047]	547
<b>PUID:</b> [SECM-CAT-IDS-048]	548
<b>PUID:</b> [SECM-CAT-IDS-049]	550
<b>PUID:</b> [SECM-CAT-IDS-050]	551

<b>PUID:</b> [SECM-CAT-IDS-051] .....	552
<b>PUID:</b> [SECM-CAT-IDS-052] .....	553
<b>PUID:</b> [SECM-CAT-IDS-053] .....	554
<b>PUID:</b> [SECM-CAT-IDS-054] .....	555
<b>PUID:</b> [SECM-CAT-IDS-055] .....	557
<b>PUID:</b> [SECM-CAT-IDS-056] .....	558
<b>PUID:</b> [SECM-CAT-IDS-057] .....	559
<b>PUID:</b> [SECM-CAT-IDS-058] .....	560
<b>PUID:</b> [SECM-CAT-IDS-059] .....	561
<b>PUID:</b> [SECM-CAT-IDS-060] .....	562
<b>PUID:</b> [SECM-CAT-IDS-061] .....	563
<b>PUID:</b> [SECM-CAT-IDS-062] .....	565
<b>PUID:</b> [SECM-CAT-IDS-063] .....	566
<b>PUID:</b> [SECM-CAT-IDS-064] .....	567
<b>PUID:</b> [SECM-CAT-IDS-065] .....	568
<b>PUID:</b> [SECM-CAT-IDS-066] .....	569
<b>PUID:</b> [SECM-CAT-IDS-067] .....	570
<b>PUID:</b> [SECM-CAT-IDS-068] .....	572
<b>PUID:</b> [SECM-CAT-IDS-069] .....	573
<b>PUID:</b> [SECM-CAT-IDS-070] .....	574
<b>PUID:</b> [SECM-CAT-IDS-071] .....	575
<b>PUID:</b> [SECM-CAT-IDS-072] .....	577
<b>PUID:</b> [SECM-CAT-IDS-073] .....	578
<b>PUID:</b> [SECM-CAT-IDS-074] .....	579
<b>PUID:</b> [SECM-CAT-IDS-075] .....	581
<b>PUID:</b> [SECM-CAT-IDS-076] .....	582
<b>PUID:</b> [SECM-CAT-IDS-077] .....	583
<b>PUID:</b> [SECM-CAT-IDS-078] .....	585
<b>PUID:</b> [SECM-CAT-IDS-079] .....	586
<b>PUID:</b> [SECM-CAT-IDS-080] .....	588
<b>PUID:</b> [SECM-CAT-IDS-081] .....	589
<b>PUID:</b> [SECM-CAT-IDS-082] .....	590
<b>PUID:</b> [SECM-CAT-IDS-083] .....	591
<b>PUID:</b> [SECM-CAT-IDS-084] .....	592
2.9.3.10 Insufficient Cryptography.....	593
<b>PUID:</b> [SECM-CAT-ICR-001] .....	593
<b>PUID:</b> [SECM-CAT-ICR-002] .....	595
<b>PUID:</b> [SECM-CAT-ICR-003] .....	596

<b>PUID: [SECM-CAT-ICR-004]</b> .....	597
<b>PUID: [SECM-CAT-ICR-005]</b> .....	598
<b>PUID: [SECM-CAT-ICR-006]</b> .....	600
<b>PUID: [SECM-CAT-ICR-007]</b> .....	602
<b>PUID: [SECM-CAT-ICR-008]</b> .....	603
<b>PUID: [SECM-CAT-ICR-009]</b> .....	604
<b>PUID: [SECM-CAT-ICR-010]</b> .....	606
<b>PUID: [SECM-CAT-ICR-011]</b> .....	607
<b>PUID: [SECM-CAT-ICR-012]</b> .....	608
<b>PUID: [SECM-CAT-ICR-013]</b> .....	609
<b>PUID: [SECM-CAT-ICR-014]</b> .....	610
<b>PUID: [SECM-CAT-ICR-015]</b> .....	612
<b>PUID: [SECM-CAT-ICR-016]</b> .....	613
<b>PUID: [SECM-CAT-ICR-017]</b> .....	614
<b>PUID: [SECM-CAT-ICR-018]</b> .....	616
<b>PUID: [SECM-CAT-ICR-019]</b> .....	617
<b>PUID: [SECM-CAT-ICR-020]</b> .....	618
<b>PUID: [SECM-CAT-ICR-021]</b> .....	619
<b>PUID: [SECM-CAT-ICR-022]</b> .....	621
<b>PUID: [SECM-CAT-ICR-023]</b> .....	622
2.9.4    Maintainability .....	623
2.9.5    Portability .....	623
<b>3. Verification</b> .....	<b>624</b>
<b>4. Supporting Information</b> .....	<b>624</b>
<b>5. References</b> .....	<b>624</b>

## Revision History

Date	Revision	Description	Authors
08/01/2026	Revision 1.0	Initial	Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás

# 1. Introduction

This document describes a Software Requirements Specification (SRS) for developing secure m-Health applications. This SRS complies with the ISO/IEC/IEEE standard 29148-2018 [1].

## 1.1 SRS overview

*[[This clause defines the normative content of the software requirements specification (SRS). The project shall produce the following information item content in accordance with the project's policies with respect to the software requirements specification. Organization of the content such as the order and section structure may be selected in accordance with the project's information management policies.]]*

This document adheres to the organizational framework of the ISO/IEC/IEEE 29148:2018 standard (IEEE SA - IEEE/ISO/IEC 29148-2018, n.d.), which supersedes IEEE 830-1998. [2]

The Software Requirements Specification (SRS) requirements are detailed in section 2, specifically called "Requirements." A substantial portion of the subsections in this section are empty because this compendium comprises reusable requirements that pertain to a particular domain. All requirements for the security of this SRS can be found in [subparagraph 2.9.3](#).

OWASP Mobile top 10 2024 [3], a complete and standardized awareness project that addresses the various aspects of security and risk in mobile applications, has been considered to establish the main ten subcategories containing security requirements.

### 1.1.1 Metainformation associated with the requirements

This subsection explains the attributes associated with this catalog's requirements. The features marked as "Mandatory" need to be set when the requirement is created. The rest of the attributes can be empty. The IEEE standard defines the following attributes:

- ✓ **PUID (Project Unique IDentification).**  
Mandatory. This identifier corresponds to the requirement's placement within the hierarchical list.
- ✓ **Requirement description.**  
Mandatory. Full description.
- ✓ **Source**  
The attribute signifies the origin of the necessity. This origin may stem from a customer's demand, a technical remedy, or a prescribed guideline, among other possibilities.
- ✓ **Priority**  
The analyst determines this attribute, which signifies the sequence in which the requirements are developed. This numerical representation is derived from the preceding two values, each of which can be classified as low, medium, or high.
- ✓ **Rationale**  
This attribute indicates the motivation for the requirement.

Some other attributes indicate relationships between the requirements:

✓ **Child PUIDs and Parent PUIDs.**

A child PUID, also known as a trace-to relationship, pertains to a stipulation that possesses an inclusive dependency. Conversely, a parent PUID, also referred to as a trace-from relation, relates to a requirement with an inclusive dependence on it.

✓ **Exclusion PUIDs.**

A PUID exclusion refers to a condition that contradicts its requirements. The attribute's value initially comprises the count of PUIDs followed by the actual PUID values.:

# exclusion PUIDs: PUID, [PUID]\*

In addition, the following attributes are also defined:

✓ **Importance.**

This attribute denotes the level of significance the requirement holds for the client. The possible options for this attribute include low, medium, and high.

✓ **Current state.**

This attribute indicates the current condition of the requirement. A total of nine distinct states have been identified:

- ❖ **To\_be\_determined:** the requirement is included in the document but is not entirely described, or its description is not final.
- ❖ **Pending\_review:** the requirement was determined, but the customers have not reviewed it.
- ❖ **Discarded:** the requirement is no longer needed or is not feasible.
- ❖ **Approval:** the requirement is correct and was approved by the customers.
- ❖ **Analysis\_modeling:** the requirement was modeled in the analysis phase.
- ❖ **Design\_modeling:** the requirement was modeled in the design phase.
- ❖ **Implemented:** the requirement is implemented in the project.
- ❖ **Verified:** the requirement changes to this state when the technical team and the customer corroborate that the project accomplishes it.

✓ **Verification method.**

This characteristic denotes the approach that must be employed to validate the project's fulfillment of the stipulation. Four options are available: inspection, analysis, demonstration, or evidence.

✓ **Validation criteria.**

This characteristic signifies the prerequisites essential for substantiating the necessity. Typically, these prerequisites are encompassed within the documentation for STS (Software Test Specification).

✓ **Requested by.**

This particular attribute denotes the individual who initiates the solicitation of the necessity.

✓ **Responsible.**

This particular characteristic denotes the individual responsible for fulfilling the specified demand.

✓ **Configurable value.**

This attribute encompasses the potential values of the requirements' parameters. This attribute's initial value comprises the parameter count and the corresponding data types.

# Parameters: DataType, [DataType]\*

✓ **Version history.**

This characteristic encompasses the background information of the individual responsible for the stipulation, the specific date of creation, the iteration number, and a comprehensive explanation.

✓ **Author and date.**

This feature shows the last author and date of the request. Only the required PUID and specification are mandatory, and other identifiers are optional.

The attribute of configurable value enables the inclusion of requirements that can be parameterized. A requirement that can be adjusted according to parameters encompasses a component that possesses the capability to assume various values contingent upon the particular undertaking in which the catalog is being repurposed.

All requirements are also reused when the catalog is reused on a specific application. The attribute values have the potential to be modified with fresh values that are tailored to suit the particular application. The characteristics that may be attributed to a numerical value upon the reuse of the catalog encompass priority, criticality, present condition, method of verification, criteria for validation, originator, accountable entity, modifiable quantity, record of past iterations, and creator and date.

## 1.2 Purpose

*[[This subsection should: Delineate the purpose of the software to be specified.]]*

The Software Requirements Specification (SRS) is an elaborate delineation for a particular software product, program, or collection of programs engineered to carry out designated functionalities within a well-defined setting. The principal aim of this SRS is to delineate the essential prerequisites for developing a safeguarded m-Health application.

The present SRS pertains to a reusable catalog known as the security mobile requirements catalog (SECM-CAT), a repository of reusable requirements and a spiral process model. The requirements are systematically organized in catalogs that comply with the international recommendations on requirements, such as IEEE 830-1998 [2] or its updated version ISO/IEC/IEEE 29148:2018 [1], and also align with the guidelines outlined in IEEE 1233 ("IEEE Guide for Developing System Requirements Specifications," 1998).

This proposed security catalog has been meticulously crafted as a comprehensive, coherent, unequivocal, dependable, provable, attributable, and adaptable document, incorporating international laws and Standards, scientific research, guidelines, good practices, and recommendations on requirements. SECM-CAT was a meticulously designed catalog that can be employed to formulate the specifications for any m-Health application.

The target audience for this software requirements specification (SRS) encompasses individuals and sectors engaged in the secure planning, design, development, assessment, and auditing of mobile health (m-Health) applications.

## **1.3 Scope**

*[[This subsection should:*

- a) identifying the software product(s) to be produced by name (e.g., Host DBMS, Report Generator, etc.);*
- b) explaining what the software product(s) will do;*
- c) describing the application of the software being specified, including relevant benefits, objectives, and goals; and*
- d) being consistent with similar statements in higher-level specifications (e.g., a system requirements specification), if they exist.]]*

The scope of this document comprises security on m-Health applications.

The following standards provided by the International Organization for Standardization (ISO) were considered for this document:

- ✓ ISO 27001:2022 Information security, cybersecurity, and privacy protection. Information security management systems Requirements [5]. This standard is the base for establishing, implementing, maintaining, and continuously improving an information security management system.
- ✓ ISO 27002:2022 "Information security, cybersecurity, and privacy protection Information security controls" [6], designed to create industry and organization-specific guidelines for managing information security. This document considers the unique information security risks these industries and organizations face and follows the principles outlined in ISO 27001.
- ✓ ISO 27799:2016 Health informatics - Information security management in health using ISO/IEC 27002 [7]. This standard offers guidance to healthcare entities and other stewards of personal health data on the optimal approaches for safeguarding the confidentiality, integrity, and availability of said data due to the unique stipulations within the healthcare domain that necessitate adherence to privacy, accuracy, and verifiability, as well as accessibility of personal health data.

The following international laws, guides, and considerations were considered for this document:

- ✓ HIPAA [8]. This comprehensive regulation encompasses the requirements stipulated in Title II, subtitle F, sections 261 through 264 of the Health Insurance Portability and Accountability Act of 1996 (HIPAA), Public Law 104-191. These requirements mandate the adoption of protective measures to guarantee the confidentiality and integrity of this information while under the purview of HIPAA-covered entities (covered entities) and during its transmission among covered entities and between covered entities and external parties.
- ✓ HIPAA Security rules [9] This regulation encourages physicians to implement appropriate administrative, physical, and technical safeguards to ensure patient health information's confidentiality, integrity, and security are stored electronically (known as "ePHI").

- ✓ Privacy & Security Requirements and Considerations for Digital Health Solutions [10]. This recommendation delineates a collection of privacy and security considerations, which grants digital health solution providers the freedom to establish privacy and security requirements specific to their solutions. The format of the revised standard ISO/IEC 27002:2005 is adhered to in this document. All 11 security control objectives are encompassed in the subsequent sections:
  1. Risk Management
  2. Security Policy;
  3. Organizing Information Security;
  4. Asset Management
  5. Human Resources Security;
  6. Physical and Environmental Security;
  7. Communications and Operational Management;
  8. Access Control;
  9. Information Systems Acquisition, Development, and Maintenance;
  10. Security Incident Management;
  11. Business Continuity Management; and
  12. Compliance.
- ✓ NIST SP 800-53 rev. 5 [11]. Within the confines of this manuscript, there is a delineation of the security and privacy measures that must be adhered to by information systems and organizations. These measures safeguard the operations, assets, individuals, external organizations, and the entire Nation against various perils and risks. Such risks include, but are not limited to, malicious assaults, human fallibility, acts of nature, structural collapse, foreign intelligence agencies, and apprehensions regarding personal privacy.
- ✓ NIST SP 1800-1 [12] The NIST Cybersecurity Practice Guide: Securing Electronic Records on Mobile Devices seeks to outline how existing technologies can better protect information in electronic health record (EHR) systems, specifically for organizations that use mobile devices. Using commercial and open-source tools that support cybersecurity standards, a layered security approach is presented for more secure sharing of patient health records. The guide details how to implement automated security controls and integrate these solutions with existing IT infrastructure while complying with standards and regulations such as the HIPAA Security Rule. Although commercial products are used in the example, specific products are not promoted, and organizations are encouraged to adapt the proposed approach to their needs.
- ✓ CWE/SANS TOP 25 Most Dangerous Software Errors: 2021 [13]. In partnership with the National Cyber Security Division of the Department of Homeland Security, MITRE maintains the CWE (Common Weakness Enumeration) website. This online platform comprehensively explains the most critical software flaws alongside authoritative recommendations on preventing and reducing their impact. Additionally, it provides a wealth of information regarding over 700 other software errors, design flaws, and architectural weaknesses that have the potential to be exploited.
- ✓ OWASP Mobile top 10: 2024 [3], a nonprofit organization through a project that addresses the various aspects of security and risk in mobile apps, has created this guide, which establishes ten subcategories containing security requirements in mobile apps.
- ✓ CIS Critical Security Controls v 8.1. [14] The CIS Critical Security Controls (CIS Controls) are a prioritized set of safeguards designed to mitigate the most common cyberattacks on systems and networks, aligned with multiple legal and regulatory frameworks. Version 8.1 is an update to Version 8.0, guided by principles of context, which improves the applicability of

the safeguards with additional examples; clarity, which aligns the controls with other security frameworks while maintaining their unique characteristics; and consistency, which ensures continuity for current users with minimal changes to the implementation.

- ✓ Application Security and Development Security Technical Implementation Guide [15]. This Security Technical Implementation Guide is designed to enhance the security of Department of Defense (DoD) information systems. Its requirements are based on guidance from the National Institute of Standards and Technology (NIST), specifically the NIST 800-53 framework and related resources.
- ✓ Smartphone Secure Development Guidelines [16]. This document aims to provide technology-neutral guidance for developing secure mobile applications. While some technologies are mentioned, these references are not exhaustive. It is intended for mobile application developers and complements in-depth guidance on secure code development and server security, such as the software development lifecycle and defense. The document highlights specific features and functions of mobile applications that require attention to ensure their security. Security measures should be applied based on a risk assessment, as some may not be necessary if the application does not use specific resources that require protection.
- ✓ DISA STIG Mobile Application Security Requirements Guide: 2014 [17] This Guide endeavors to bolster the security of information systems within the Department of Defense (DoD). The stipulations are established based on the guidelines outlined in the NIST 800-53 and associated literature.
- ✓ Android Security best practices [18]. The official Android developer page shows several best practices that positively impact app security.
- ✓ OWASP Mobile Application Security Checklist [19] provides a comprehensive reference to the test cases outlined in the Mobile Application Security Testing Guide (MASTG) for each control of the Mobile Application Security Verification Standard (MASVS). This checklist addresses the measures and best practices needed to safeguard mobile applications from threats like malware, data theft, and vulnerabilities. Key security elements include data encryption, multi-factor authentication, code security, and prevention of data leaks. As part of the Open Web Application Security Project (OWASP), the MAS Checklist helps assess mobile app security across critical areas such as secure data storage (MASVS-STORAGE), proper cryptography (MASVS-CRYPTO), authentication (MASVS-AUTH), network protection (MASVS-NETWORK), platform security (MASVS-PLATFORM), secure coding (MASVS-CODE), attack resilience (MASVS-RESILIENCE), and user privacy (MASVS-PRIVACY).
- ✓ Mobile Application Security Verification Standard (MASVS) [20] is the industry standard for mobile app security. It can be used by mobile software architects and developers seeking to develop secure mobile applications and security testers to ensure completeness and consistency of test results.

A literature review was also performed to get security recommendations for m-health security. Of the entire collection of 599 scholarly papers, a mere fifteen articles [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31], [32], [33], [34] were ultimately chosen after undergoing a rigorous and meticulous selection process.

## 1.4 Product perspective

### 1.4.1 System interfaces

*[[List each system interface and identify the functionality of the software to accomplish the system requirement and the interface description to match the system.]]*

Not described.

### 1.4.2 User interfaces

*[[Specify the logical characteristics of each interface between the software product and its users.]]*

Not described.

### 1.4.3 Hardware interfaces

*[[Specify the logical characteristics of each interface between the software product and the hardware elements of the system. This includes configuration characteristics (number of ports, instruction sets, etc.). It also covers such matters as what devices are to be supported, how they are to be supported, and protocols. For example, terminal support may specify full-screen support as opposed to line-by-line support.]]*

Not described.

### 1.4.4 Software interfaces

*[[Specify the use of other required software products (e.g., a data management system, an operating system or a mathematical package), and interfaces with other application systems (e.g., the linkage between an accounts receivable system and a general ledger system). For each required software product, specify:*

- a) name;
- b) mnemonic;
- c) specification number;
- d) version number; and
- e) source.

*NOTE it is acceptable to specify required platforms or operating systems, but rarely feasible to require a specific version. Typically, a version number most recent version or any currently maintain version can be specified for software.*

*For each interface, specify:*

- a) discussion of the purpose of the interfacing software as related to this software product;
- b) definition of the interface in terms of message content and format. It is not necessary to detail any well-documented interface, but a reference to the document defining the interface is required.]]

Not described.

### 1.4.5 Communications interfaces

*[[Specify the various interfaces to communications such as local network protocols.]]*

Not described.

#### **1.4.6 Memory constraints**

*[[Specify any applicable characteristics and limits on primary and secondary memory.]]*

Not described.

#### **1.4.7 Operations**

*[[Specify the normal and special operations required by the user such as:*

- a) the various modes of operations in the user organization (e.g., user-initiated operations);*
- b) periods of interactive operations and periods of unattended operations;*
- c) data processing support functions; and*
- d) backup and recovery operations.*

*NOTE This is sometimes specified as part of the User Interfaces section.]]*

Not described.

#### **1.4.8 Site adaptation requirements**

*[[The site adaptation requirements include:*

- a) definition of the requirements for any data or initialization sequences that are specific to a given site, mission or operational mode (e.g., grid values, safety limits, etc.);*
- b) specification of the site or mission-related features that should be modified to adapt the software to a particular installation.]]*

Not described.

#### **1.4.9 Interfaces with services**

*[[Specify interactions with services, e.g., Software as a Service (SaaS) or cloud services.]]*

Not described.

### **1.5 Product functions**

*[[Provide a summary of the major functions that the software will perform. For example, an SRS for an accounting program may use this part to address customer account maintenance, customer statement and invoice preparation without mentioning the vast amount of detail that each of those functions requires...]]*

Not described.

### **1.6 User characteristics**

*[[Describe those general characteristics of the intended groups of users of the product including characteristics that may influence usability, such as educational level, experience, disabilities and technical expertise. This description should not state specific requirements, but rather should state the reasons why certain specific requirements are later specified in specific requirements in 9.6.9...]]*

Not described.

## **1.7 Limitations**

*[[Provide a general description of any other items that will limit the supplier's options, including:*

- a) regulatory requirements and policies;*
- b) hardware limitations (e.g., signal timing requirements);*
- c) interfaces to other applications;*
- d) parallel operation;*
- e) audit functions;*
- f) control functions;*
- g) higher-order language requirements;*
- h) signal handshake protocols (e.g., XON-XOFF, ACK-NACK);*
- i) quality requirements (e.g., reliability);*
- j) criticality of the application;*
- k) safety and security considerations;*
- l) physical/mental considerations; and*
- m) limitations that are sourced from other systems, including real-time requirements from the controlled system through interfaces.]]*

Not described.

## **1.8 Assumptions and dependencies**

*[[List each of the factors that affect the requirements stated in the SRS. These factors are not design constraints on the software but any changes to these factors can affect the requirements in the SRS. For example, an assumption may be that a specific operating system will be available on the hardware designated for the software product. If, in fact, the operating system is not available, the SRS would have to change accordingly]]*

Not described.

## **1.9 Apportioning of requirements**

*[[Apportion the software requirements to software elements. For requirements that will require implementation over multiple software elements, or when allocation to a software element is initially undefined, this should be so stated. A cross-reference table by function and software element should be used to summarize the apportionments. Identify requirements that may be delayed until future versions of the system (e.g., blocks and/or increments).]]*

Not described.

# **2. Requirements**

## **2.1 Specified requirements**

*[[Specify the software system requirements to a level of detail sufficient for software design, development and verification of the software increment or release in process.*

*The requirements should:*

- a) be stated in conformance with all the characteristics described in 5.2 of this document;*
- b) be cross-referenced to earlier versions or related documents;*
- c) be uniquely identifiable;*
- d) describe every input (stimulus) into the software system, every output (response) from the software system, and all functions performed by the software system in response to an input or in support of an output.]*

Not described.

## **2.2 External Interfaces**

*[[Define all inputs into and outputs from the software system. The description should complement the interface descriptions in 9.6.4.1 through 9.6.4.5, and should not repeat information there.*

*Each interface defined should include the following content:*

- a) name of item;*
- b) description of purpose;*
- c) source of input or destination of output;*
- d) valid range, accuracy and/or tolerance;*
- e) units of measure;*
- f) timing;*
- g) relationships to other inputs/outputs;*
- h) data formats;*
- i) command formats; and*
- j) data items or information included in the input and output. ]]*

Not described.

## **2.3 Functions**

*[[Define the fundamental actions that have to take place in the software in accepting and processing the inputs and in processing and generating the outputs, including:*

- a) validity checks on the inputs;*
- b) exact sequence of operations;*
- c) responses to abnormal situations, including:*
  - 1) overflow;*
  - 2) communication facilities;*
  - 3) hardware faults and failures; and*
  - 4) error handling and recovery;*
- d) effect of parameters;*
- e) relationship of outputs to inputs, including:*
  - 1) input/output sequences; and*
  - 2) formulas for input to output conversion.*

*It may be appropriate to partition the functional requirements into sub-functions or sub-processes. This does not imply that the software design will also be partitioned that way.]]*

Not described.

## **2.4 Usability requirements**

*[[Define usability and quality in use requirements and objectives for the software system that can include measurable effectiveness, efficiency, satisfaction criteria and avoidance of harm that could arise from use in specific contexts of use.*

*NOTE Additional guidance on usability requirements can be found in ISO/IEC TR 25060.]]*

Not described.

## **2.5 Performance requirements**

*[[Specify both the static and the dynamic numerical requirements placed on the software or on human interaction with the software as a whole.*

*Static numerical requirements may include the following:*

- a) the number of terminals to be supported;*
- b) the number of simultaneous users to be supported; and*
- c) the amount and type of information to be handled.*

*Static numerical requirements are sometimes identified under a separate section entitled Capacity. Dynamic numerical requirements may include, for example, the numbers of transactions and tasks and the amount of data to be processed within certain time periods for both normal and peak workload conditions.*

*The performance requirements should be stated in measurable terms.]]*

Not described.

## **2.6 Logical database**

*[[Specify the logical requirements for any information that is to be placed into a database, including:*

- a) types of information used by various functions;*
- b) frequency of use;*
- c) accessing capabilities;*
- d) data entities and their relationships;*
- e) integrity constraints;*
- f) security; and*
- g) data retention requirements.]]*

Not described.

## **2.7 Design constraints**

*[[Specify constraints on the system design imposed by external standards, regulatory requirements or project limitations.]]*

Not described.

## **2.8 Standards compliance**

*[[Specify the requirements derived from existing standards or regulations, including:*

- a) report format;*

- b) data naming;
- c) accounting procedures; and
- d) audit tracing.

*For example, this could specify the requirement for software to trace processing activity. Such traces are needed for some applications to meet minimum regulatory or financial standards. An audit trace requirement may, for example, state that all changes to a payroll database shall be recorded in a trace file with before and after values.]*

Not described.

## 2.9 Software system attributes

### 2.9.1 Reliability

*[[specify the factors required to establish the required reliability of the software system at the time of delivery.]]*

Not described.

### 2.9.2 Availability

*[[specify the factors required to guarantee a defined availability level for the entire system such as checkpoint, recovery and restart.]]*

Not described.

### 2.9.3 Security

*[[specify the requirements to protect the software from accidental or malicious access, use modification, destruction or disclosure. Specific requirements in this area could include the need to:*

- 1) utilize certain cryptographic techniques;
- 2) keep specific log or history data sets;
- 3) assign certain functions to different modules;
- 4) restrict communications between some areas of the programmer;
- 5) check data integrity for critical variables; and
- 6) assure data privacy.]

#### 2.9.3.1 Improper Credential Usage

<b>PUID:</b> [SECM-CAT-ICU-001]
<b>Requirement description:</b> The mobile application should not contain hardcoded details of external resources, credentials, or API keys to prevent unauthorized access and secure the app from vulnerabilities.
<b>Source:</b> <ul style="list-style-type: none"><li>✓ SRG-APP-000516-MAPP-000064: The mobile app code must not contain hardcoded references to resources external to the app [17].</li></ul>

<ul style="list-style-type: none"> <li>✓ Minimize API Key Exposure: Avoid hardcoding API keys into the app's codebase or configuration files [18].</li> <li>✓ SRG-APP-000033-MAPP-000012: A mobile app must not call APIs or otherwise invoke resources external to the mobile app unless such activity serves the documented purposes of the mobile app [17].</li> <li>✓ Avoid Using Hardcoded Credentials: Hardcoded credentials can be easily discovered by attackers and provide an easy access point for unauthorized users. Always avoid using hardcoded credentials in your mobile app's code or configuration files [3].</li> <li>✓ 3.9. Do not store any passwords or secrets in the application binary. Do not use a generic shared secret for integration with the backend server (like password embedded in code). Mobile application binaries can be easily downloaded and reverse engineered [16].</li> <li>✓ CWE-798: Use of Hard-coded Credentials</li> </ul> <p>The product contains hard-coded credentials, such as a password or cryptographic key, which it uses for its own inbound authentication, outbound communication to external components, or encryption of internal data [35].</p>
--

**Priority:** Not described

**Rationale:** This requirement helps decrease the potential threat of hardcoded credentials and data/coming out with mobile software. Reverse engineering: Hardcoded elements like API keys, passwords, or cryptographic secrets can be dropped via simple disassembly and are a common security threat. If left unsecured, unauthorized users could potentially tamper with these elements to access the backend systems, external services, or secured data within the application, which would result in a breakdown of how confidential and integral the whole application and its users are.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

- ✓ Perform a code review to check the codebase for hard-coded credentials or references to external resources.
- ✓ Securitize Your Keys, Passwords, and Secrets — Store these securely in environment variables or with an OS service like the secure key-management service.
- ✓ Make sure that credentials are never hardcoded and security practices are held to secure your confidential information.
- ✓ Look at the binary to see that someone cannot reverse-engineer it to extract secrets.
- ✓ Confirm that 2nd-level approval is mandated wherever required.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda
José L Fernández-Alemán
Juan Manuel Carrillo-de-Gea
Joaquín Nicolás
08/01/2026

<b>PUID:</b> [SECM-CAT-ICU-002]
<b>Requirement description:</b> Applications shall not store passwords or secrets in the application binary, nor inhibit backend integration by using generic shared secrets.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ 3.9. Do not store any passwords or secrets in the application binary. Do not use a generic shared secret for integration with the backend server (like password embedded in code). Mobile application binaries can be easily downloaded and reverse engineered [16].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> This is to secure sensitive information in mobile apps and avoid storing passwords or any secrets within the application binary. All applications that store such data right in their binary can be easily reverse engineered, which makes it easy for an attacker to extract such information as a password, API key, or cryptographic secret. This can result in unauthorized access to backend services, which will ultimately make your app and its users vulnerable to data breaches and hence introduce security threats. Protecting passwords and secrets from potential exposure through having them hard-coded into the codebase improves the security of an application.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ul style="list-style-type: none"> <li>✓ Ensure the application codebase contains no hardcoded passwords or secrets.</li> <li>✓ Are sensitive data like API keys or cryptographic secrets stored securely, such as by encrypted storage solutions or other more secure methods.</li> <li>✓ Conduct a rich code review to identify embedded secrets in the mobile app binary.</li> <li>✓ Verify the use of appropriate secure mechanisms like environment variables, secrets management for sensitive data.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda
José L Fernández-Alemán

Juan Manuel Carrillo-de-Gea  
Joaquín Nicolás  
08/01/2026

**PUID:** [SECM-CAT-ICU-003]

**Requirement description:** To avoid unauthorized access and reduce the security risks of the application, the requirement is to ensure that hardcoded default credentials are not used in the application.

**Source:**

- ✓ Default credentials: 1. Refrain from using hardcoded default credentials [3].

**Priority:** Not described

**Rationale:** This requirement aims to eliminate hardcoded default credentials in the application as a means of preventing unauthorized access. Default credentials hardcoded into the device are probably one of the most common vulnerabilities out there, as they can be discovered by attackers very quickly and could also easily provide a low-barrier entry point for unauthorized users. By doing so, the application mitigates a direct unauthorized access vector and thus provides improved authentication security.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

**1. Inspection for Hardcoded Credentials:**

- ✓ Check for hardcoded default credentials in the source code or config files of the application. Is there any presence? If yes, check it and block them.
- ✓ Customer-configured Test or Development developers using. Following are confirmed to be Removed or replaced. Use of default credentials with secure authentication methods in Production environments.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda  
José L Fernández-Alemán  
Juan Manuel Carrillo-de-Gea  
Joaquín Nicolás

08/01/2026

<b>PUID:</b> [SECM-CAT-ICU-004]
<b>Requirement description:</b> The application will validate access to the credentials used in check signatures, with which 2) only a valid app for the organization can be made, and correct API calls corresponding to what is described in mobile app documents.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ Application Verification: For credentials used only by applications developed by the same organization, verify access through check signatures to confirm that the requesting app is authorized [18].</li> <li>✓ SRG-APP-000033-MAPP-000012: A mobile app must not call APIs or otherwise invoke resources external to the mobile app unless such activity serves the documented purposes of the mobile app [17].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> This requirement is designed to prevent unauthorized use and ensure proper security by requiring applications to secure specific keys, which can only be used by apps published from the same organization. When the <code>checkSignatures</code> method is used in our app, it prevents random software from making requests to your API. It also helps to maintain that all external API calls are in line with the legitimate mobile application purposes written in your documentation, compared against potential misuse, hence keeping secure personal information.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li>1. <b>Signature Verification Check:</b></li> </ol> <ul style="list-style-type: none"> <li>✓ CheckSignatures is implemented to verify credentials are between apps in the same org.</li> <li>✓ Make sure the application limits API calls or external resource requests to what is explicitly described as being necessary for its functions.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda
José L Fernández-Alemán

Juan Manuel Carrillo-de-Gea
Joaquín Nicolás
08/01/2026

### 2.9.3.2 Inadequate Supply Chain Security

<b>PUID:</b> [SECM-CAT-ISU-001]
<b>Requirement description:</b> Development, testing, and production key usage (to prevent cross-environment exposure); Explicit intents when binding to services (to reduce intercept risk) — This means an application should generate environment-specific API keys for these key life stages so that they do not share the same value across development and production environments.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ Use of Environment-Specific API Keys: Generate and use separate API keys for development, testing, and production environments to prevent cross-environment exposure. Each environment should have its own isolated API keys to minimize risks, allowing the ability to disable or revoke keys in one environment without impacting others [18].</li> <li>✓ When binding to a Service, use explicit intents to ensure that the correct service is started. Implicit intents for services are a security risk, as they can be intercepted by unintended services. As of Android 5.0 (API level 21), using an implicit intent with bindService() will throw an exception. [18]</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> The main reason for this rule is to control the risk of a compromised key and decrease the possibility that if someone gains unauthorized access to an application and its keys, they also gain this level of access to another environment. Furthermore, when binding to services with explicit intents, only the specified service is going to be called, protecting against interception and any unauthorized access by unsolicited services as a result. For Android application this is moreover a security concern as implicit intents are interceptable.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined

<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<b>1. Environment-specific API Key verification:</b>
<ul style="list-style-type: none"><li>✓ Check whether unique API keys for dev, stage, and prod environments are created and used.</li><li>✓ Verify that the API keys can be switched off or revoked for each environment independently without affecting other environments.</li></ul>
<b>2. Service Binding Security Flip to on:</b>
<ul style="list-style-type: none"><li>✓ Explicit intents should be used with bindings to services, especially when attempting to enter bindService().</li><li>✓ Check the application is NOT using implicit intents in services; if not, then the correct conduct may intercept.</li></ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-ISU-002]
<b>Requirement description:</b> Applications must not change the SSL verification method that can allow self-signed certificates. Self-signed certificates or a local Certificate Authority (CA) must be used during development to keep with good SSL practices.
<b>Source:</b>
<ul style="list-style-type: none"><li>✓ General Best Practices: 13. During development cycles, avoid overriding SSL verification methods to allow untrusted certificates, instead try using self-signed certificates or a local development certificate authority (CA) [3].</li></ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Enforce secured development cycles to follow proper SSL/TLS practices. Like anything else, bypassing SSL verification and letting untrusted certificates be validated comes with security trade-offs: the app is opening itself up to man-in-the-middle (MITM) attacks among other risks. See the news with that self-signed certificate or local CA, without disrupting the verification process, is a very useful aspect for securely testing SSL connections in developers. This means that one has to present valid SSL verification when running on external URLs in order for any insecure configs not to be deployed by accident into production.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0

<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<p><b>1. SSL Verification Check:</b></p> <ul style="list-style-type: none"> <li>✓ Ensure that SSL Verification mechanisms are not overridden to trust untrusted certificates on dev/production builds.</li> <li>✓ To use a self-signed or local CA for SSL connections specially made during development.</li> <li>✓ Verify that the application never accepts untrusted certificates at any point of the software development life cycle.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-ISU-003]
<b>Requirement description:</b> The application shall not contain any unmanaged and unsigned mobile code.
<b>Source:</b> <ul style="list-style-type: none"> <li>✓ V-222665: The designer must ensure uncategorized or emerging mobile code is not used in applications. Remove uncategorized or emerging mobile code from the application or obtain a waiver and risk acceptance to operate [15].</li> <li>✓ V-222618: Unsigned Category 1A mobile code must not be used in the application in accordance with DoD policy. Configure the application so Category 1A mobile code is signed [15].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> This is done to mitigate security threats by making sure that the application uses mobile code which has been labeled, and only those apps are used. Capsules that are left uncategorized or unsigned mobile code can behave as a weakness because they may not have had complete security validation and thereby open up the system to unauthorized code execution or exploitation.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0

<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ul style="list-style-type: none"> <li>✓ By requiring signing for unverifiable execution modes on Category 1A mobile code.</li> <li>✓ Deleting or waiving those slots that currently offer an automatic verification out of the box.</li> <li>✓ Removing the ability to execute all uncategorized mobile code from non-native store applications with a CORI presented on behalf of signatory backing.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-ISU-004]
<b>Requirement description:</b> The application shall not contain any mobile code that is uncategorized or lacks a digital signature.
<b>Source:</b> <ul style="list-style-type: none"> <li>✓ 1.10. Consider the security of the whole data lifecycle in writing your application (collection over the wire, temporary storage, caching, backup, deletion, etc.) [16]</li> <li>✓ Use of Type-Safe Languages: The application should be developed using type-safe programming languages to reduce the likelihood of input validation vulnerabilities [18]</li> <li>✓ Follow security best practices: Implement general security best practices in your development process, including secure coding techniques, code reviews, and vulnerability scanning [18].</li> <li>✓ CIS Critical Security Control 16: Application Software Security: Manage the security life cycle of in-house developed, hosted, or acquired software to prevent, detect, and remediate security weaknesses before they can impact the enterprise [14].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> This expectation ensures that the application is secure from all aspects, be it development level or at database levels. Minimizing exposure and risk of mishandling: securing

data across the entire lifecycle from creation, transmission, storage to destruction. When we design applications with type-safe programming languages, they are less prone to input validation vulnerabilities, creating more robust and more secure applications. This translates to adopting secure coding standards and practices, such as securing dev environments with code reviews or facilitating scanning for vulnerabilities at regular intervals in the development lifecycle.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

#### 1. Data Lifecycle Security Check:

- ✓ Check if the app follows security practices at each step of the data lifecycle such as secure data collection, transient storage, caching, backup, and secure deletion.

#### 2. Verified Type-Safe Language Usage:

- ✓ Verify as well that these are implemented in type-safe programming languages to mitigate input validation flaws.

#### 3. Validation of Secure Development Practices:

- ✓ Make sure the development process focuses on secure coding practices, rather than just ticking required checkboxes.
- ✓ There should be at least regular code reviews and some level of vulnerability scanning during development phases.
- ✓ Ensure that any vulnerabilities found are patched quickly, which can then be released.

#### 4. Security Lifecycle Management:

- ✓ Verify that CIS Critical Security Control 16 is being followed in order to manage the security lifecycle of the application, so that security vulnerabilities or misconfigurations cannot make their way to production.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda

José L Fernández-Alemán

Juan Manuel Carrillo-de-Gea

Joaquín Nicolás 08/01/2026
-------------------------------

**PUID:** [SECM-CAT-ISU-005]

**Requirement description:** The application must remove all code that seems to accept every SSL/TLS certificate, e.g., org.apache.http.conn.ssl.AllowAllHostnameVerifier, SSLSocketFactory.ALLOW\_ALL\_HOSTNAME\_VERIFIER, post-development cycle for secure certificate validation

**Source:**

- ✓ Android Specific Best Practices: 1. Remove all code after the development cycle that may allow the application to accept all certificates such as org.apache.http.conn.ssl.AllowAllHostnameVerifier or SSLSocketFactory.ALLOW\_ALL\_HOSTNAME\_VERIFIER. These are equivalent to trusting all certificates [3].

**Priority:** Not described

**Rationale:** This rule is designed to protect SSL/TLS connections through rigorous certificate-checking functions. Code elements like AllowAllHostnameVerifier or ALLOW\_ALL\_HOSTNAME\_VERIFIER can be exploited to bypass security controls in applications using any SSL/TLS certificate. This is very dangerous, especially in production environments where it effectively opens the application to man-in-the-middle (MITM) attacks and other forms of interception. Elimination of such code during the development cycle and using very good SSL/TLS validation is important for ensuring data integrity and confidentiality in transit.

**Number of Children:** 0**Number of parents:** 0**Cycles:** 0**Number Audit:** 0**Child PUIDs:** Not described**Parent PUIDs:** Not described**Exclusion PUIDs:** Not described**Importance:** Not described**Current state:** To be determined**Verification method:** Demonstration/Analysis**Validation criteria:****1. Certificate Validation — Code Review:**

- ✓ Ensure that every newer import of AllowAllHostnameVerifier, SSLSocketFactory. Application level: ALLOW\_ALL\_HOSTNAME\_VERIFIER or relevant code is removed from the application after the development stage.

**2. SSL/TLS Security Check:**

- ✓ Ensure the application enforces SSL/TLS certificates in production, allowing neither insecure nor self-signed certificates unless so configured.

**3. Development Cycle Artifacts – Code Review:**

<ul style="list-style-type: none"> <li>✓ Make sure the production codebase is not allowing development-specific configurations to keep untrusted certificates.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda
José L Fernández-Alemán
Juan Manuel Carrillo-de-Gea
Joaquín Nicolás
08/01/2026

<b>PUID:</b> [SECM-CAT-ISU-006]
<b>Requirement description:</b> The application must incorporate look designs to distinguish unauthorized code changes and meddling with the binary.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ Insecure Authorization Prevention: 3. If offline authorization checks are necessary within the mobile app's code, developers should perform local integrity checks to detect unauthorized code changes [3].</li> <li>✓ Tampering detection via integrity checks: 1. The system must implement integrity checks to detect any tampering with the binary code. Upon detection, the application should become unusable by removing or disabling critical resources [3].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> The purpose behind this requirement is to improve security by instituting methods to determine and react if someone has tampered or modified the binary of an application without permission. If contained code changes are not detected, mobile applications could face exploitable attacks to the reversed side of passing authorization checks. The application finds any manipulation of its code or structure by conducting integrity checks. A protection—tampering with the object (all the more important part) can cause it to be disabled or removed, in case this is a very critical resource, and this will most likely make the application useless which will help to protect those same data that were sensitive from a leak, as well as a malformation from malicious actions.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<b>1. Personal Inform Local Integrity Check Validation:</b>

<ul style="list-style-type: none"> <li>✓ Verify that the application checks its internal integrity in offline authorization, i.e., whether any unauthorized code changes have occurred.</li> </ul> <p><b>2. A Tampering Detection Mechanism:</b></p> <ul style="list-style-type: none"> <li>✓ Ensure that the application has a mechanism to check for integrity, so it can identify changes with binary code.</li> </ul> <p><b>3. After discovering tampering: Functionality Test Phase:</b></p> <ul style="list-style-type: none"> <li>✓ Inject code changes, and verify that the application either disables or prevents access to vital controls as per the pre-configured response.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-ISU-007]
<b>Requirement description:</b> The application shall be secure from code signing and execution, including a deny-all, permit-by-exception (whitelist) software execution policy that verifies the source and full intent of software prior to allowing it any level of distribution on an Android platform.
<b>Source:</b> <ul style="list-style-type: none"> <li>✓ 9.6. Malicious Insider Threats: Ensure secure app signing and distribution processes to prevent attackers from signing and distributing malicious code [3].</li> <li>✓ V-222517: The application must employ a deny-all, permit-by-exception (whitelist) policy to allow the execution of authorized software programs. Configure the application to utilize a deny-all, permit-by-exception policy when allowing the execution of authorized software [15].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> This ensures that the app can be distributed and run only in authorized or non-frozen-term way and/or owned devices. The reason behind it: protect applications, to prevent unauthorized or malicious code to be signed so as to distribute appsiders into appsides (actually verified by tests done on) in adherence with secure application designation policies. It mitigates insider threats to the enterprise by only distributing signed applications. Also, with a deny-all, permit-by-exception approach in place, the system will only allow known and trusted applications to run while prohibiting any unapproved or dangerous code from launching in the application environment.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described

<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. App Signing Verification:</b> <ul style="list-style-type: none"> <li>✓ Ensure that the application is simply signed with safe, accepted signing methods during distribution.</li> <li>✓ The application shared must be the one that is the most secure and has been signed so that a malicious code that can be distributed</li> </ul> </li> <li><b>2. Whitelist Policy Configuration Detection:</b> <ul style="list-style-type: none"> <li>✓ Ensure that the application is simply signed with safe, accepted signing methods during distribution.</li> <li>✓ The application shared must be the one that is the most secure and has been signed so that a malicious code that can be distributed</li> </ul> </li> <li><b>3. Execution Control Test:</b> <ul style="list-style-type: none"> <li>✓ Try to run unapproved software in the application environment and verify that it's prohibited as per de deny-all, permit-by-exception-policy.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-ISU-008]
<b>Requirement description:</b> Ensure that any resources used outside of standard app-store mechanisms are signed and also verify the signature before accepting updates.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ 9.6. Resources used by apps that are outside of the app-store normal mechanism must be signed. Apps must verify the signature before accepting the updated resource [16].</li> <li>✓ V-222645: Application files must be cryptographically hashed prior to deploying to DoD operational networks. Application Admins validate cryptographic hashes prior to deploying the application to production [15].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> To protect the resources and files of the application, implemented through a signature check and cryptographic hashing. The more these resources are picked in a non-standard app-store

mechanism, the higher the chance of malicious tampering. This application enforces these resources to be signed and verified, ensuring that only trusted updates are accepted. In addition, all application files are cryptographically hashed and must be approved by the administrators before being deployed, preventing unauthorized changes to the files, thereby keeping the file integrity and security policies compliant.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

**1. Verification for External Resources using Signature:**

- ✓ The resources that your application uses, which are not in the standard app store mechanism, should be signed off.
- ✓ Check if the application verifies signatures for these resources prior to applying any updates

**2. Cryptographic Hashing Deployment:**

- ✓ Application files should be cryptographically hashed before they are deployed.
- ✓ Ensure application administrators verify these cryptographic hashes before releasing the same applications to the production environment

**3. Deployment Integrity Check:**

- ✓ Try to alter any one of the cryptographically hashed application files and check if it fails validation during implementation.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda

José L Fernández-Alemán

Juan Manuel Carrillo-de-Gea

Joaquín Nicolás

08/01/2026

**PUID:** [SECM-CAT-ISU-009]

**Requirement description:** The application shall be subject to periodic security assessments including code reviews, penetration tests, and cryptographic vulnerabilities assessments according to the results of which security holes must be remedied. All findings should be logged, tagged, and resolved before moving them into production.

<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ V-222648: An application code review must be performed on the application. Conduct and document code reviews on the application during development and identify and remediate all known and potential security vulnerabilities prior to releasing the application [15].</li> <li>✓ Regular Security Testing: 1. Conduct regular security assessments, including penetration testing and code reviews, to identify and address vulnerabilities [3].</li> <li>✓ General Best Practices: 14. During security assessments, it is advised to analyze application traffic to see if any traffic goes through plaintext channels [3].</li> <li>✓ Conduct Security Testing: 1. Perform thorough security testing, including cryptographic vulnerability assessments, penetration testing, and code reviews. Identify and remediate any weaknesses or vulnerabilities discovered during the testing process [3].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> To improve the security posture of your application by requiring continuous, complete, and thorough security testing to be performed on a regular basis throughout the development lifecycle. Code reviews, penetration testing, and cryptographic vulnerability assessments should be performed to find both existing and potential security vulnerabilities that could bring harm if neglected. Using plaintext channels for application traffic is often a vector of attack during assessments, and analyzing them helps prevent data exposure — particularly sensitive information. This ensures that the app can identify and fix any vulnerabilities before release, reducing the chances of getting compromised in production.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Code Review Documentation:</b> <ul style="list-style-type: none"> <li>✓ Verify that you review code as it exits development and record any findings.</li> <li>✓ Make sure the security vulnerabilities that you have identified are fixed before you roll out a release.</li> </ul> </li> <li><b>2. The post tagged: Regular Security Testing Verification:</b> <ul style="list-style-type: none"> <li>✓ Dissemination of the Secure Coding Knowledge Base to developers</li> <li>✓ Regular security assessments (penetration testing, cryptographic vulnerability assessments, etc.)</li> <li>✓ Testing confirms identified vulnerabilities and findings are documented and remediated.</li> </ul> </li> <li><b>3. Plaintext Channels — Traffic Analysis:</b> <ul style="list-style-type: none"> <li>✓ During security assessments, examine how applications traffic is processed to prevent sensitive information from being transmitted over plaintext channels.</li> </ul> </li> </ol>

- ✓ Verify that all plaintext traffic-related findings are documented and remediated.

#### 4. Confirmation of Remediation Before Release:

- ✓ Validate all discovered vulnerabilities have been resolved prior to releasing the application into production.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda  
José L Fernández-Alemán  
Juan Manuel Carrillo-de-Gea  
Joaquín Nicolás  
08/01/2026

**PUID:** [SECM-CAT-ISU-010]

**Requirement description:** The app should not be deployed using the ad-hoc signed certificates that are used during the development and testing phase. For deployment, only production-ready certificates should be used to ensure security and reliability.

**Source:**

- ✓ 9.7. Do not deploy apps with ad-hoc signing certificates used for development and testing [16].

**Priority:** Not described

**Rationale:** This is done so that there is minimum risk of application breaches, as in production only that certificate will be used which is certified and secure. These are the ad-hoc signing certificates developers usually use for a development and testing environment, but absolutely it is in no way that those meet the suitable security standard we should have in our production. An application with such certificates, when deployed, can be vulnerable to unauthorized modifications and misuse. This, therefore, authenticates our app and reduces the possibility of it being exploited by using production-level, verified certificates.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

#### 1. Check Certificate Verification:

- ✓ Confirm that the application is not installed with the ad-hoc signing certificates you have used for development or testing.
- ✓ Ensure that none of the signing certificates used for deployment are development-grade

## 2. Deployment Process Review:

- ✓ Validate the deployment process to ensure individual ad-hoc certificates are confined to D&T only.
- ✓ Clean up ad-hoc certificates or replace them with production-ready certificates before deployment.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda  
 José L Fernández-Alemán  
 Juan Manuel Carrillo-de-Gea  
 Joaquín Nicolás  
 08/01/2026

**PUID:** [SECM-CAT-ISU-011]

**Requirement description:** The application must continue with strong access control, and local integrity checks applied to determine if the code has been modified without permission.

**Source:**

- ✓ 164.308 Administrative safeguards.
  - (a) A covered entity or business associate must, in accordance with § 164.306:
    - (4) (ii) Implementation specifications:
      - (B) Access authorization (Addressable). Implement policies and procedures for granting access to electronic protected health information, for example, through access to a workstation, transaction, program, process, or other mechanism [9].
- ✓ Reinforce Authentication: 2. Due to offline usage requirements, mobile apps might need to perform local authentication or authorization checks. In such cases, developers should instrument local integrity checks to detect any unauthorized code changes. Consult additional guidance on detecting and reacting to binary attacks [3].
- ✓ CWE-863: Incorrect Authorization

The product performs an authorization check when an actor attempts to access a resource or perform an action, but it does not correctly perform the check. This allows attackers to bypass intended access restrictions [35].

- ✓ CWE-862: Missing Authorization: The product does not perform an authorization check when an actor attempts to access a resource or perform an action [35].

**Priority:** Not described

<b>Rationale:</b> To protect electronic protected health information (ePHI) by robustly requiring appropriate authorization for offline access scenarios. Local integrity checks will allow the app to catch unauthorized code modifications that could tamper with access controls. It helps prevent security flaws like Ships of Theseus attacks, where developers make mistakes in implementing the necessary authorization checks and allow unauthorized access to sensitive resources. This ensures that ePHI access is only available through authorized methods, helping to maintain compliance with the administrative safeguards and prevent circumvention of necessary access limitations.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Authorization Policy Review:</b> <ul style="list-style-type: none"> <li>✓ Verify that there are policies and procedures to give access to ePHI and access is working fine for workstations, transactions, and other processes.</li> </ul> </li> <li><b>2. 2nd Login: Local Integrity Check Verification:</b> <ul style="list-style-type: none"> <li>✓ Ensure that local checks of integrity are in place to detect unpermitted changes to the code (offline authorization).</li> <li>✓ The application must be able to detect and react upon binary tampering attacks targeting access control.</li> </ul> </li> <li><b>3. Authorization Control Testing:</b> <ul style="list-style-type: none"> <li>✓ Test to ensure that every access to resources undergoes proper authorization tests and ePHIs.</li> <li>✓ Attempt to access the unauthorized object to see that real-world restrictions are according to policy.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-ISU-012]
---------------------------------

**Requirement description:** The anti-tampering techniques to prevent code alterations and runtime inspections of your application must be implemented by the application.

**Source:**

- ✓ MASVS-RESILIENCE-4: The app implements anti-dynamic analysis techniques: Sometimes pure static analysis is very difficult and time consuming so it typically goes hand in hand with dynamic analysis. Observing and manipulating an app during runtime makes it much easier to decipher its behavior. This control aims to make it as difficult as possible to perform dynamic analysis, as well as prevent dynamic instrumentation which could allow an attacker to modify the code at runtime [20]

**Priority:** Not described

**Rationale:** This requirement aims to increase application resilience through protections against dynamic analysis and runtime manipulation. Dynamic analysis is the process of attackers observing and modifying app behavior at runtime, making it easier to reverse-engineer the application's functionality or inject malicious code. The application RANDS using anti-dynamic analysis techniques increases the security aspect by limiting the usage of runtime instrumentation that could potentially violate your security features or lead to code functionality changes or expose your credentials.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

**1. Verification of Anti-Dynamic Analysis Techniques:**

- ✓ Anti-Dynamic Analysis: Ensure the application implements anti-dynamic analysis measures such as anti-debugging to identify and prevent activity that attempts to analyze the application in a manner that would aid us.

**2. The second check is of Dynamic Instrumentation Prevention:**

- ✓ Ensure that the application contains methods to protect against dynamic instrumentation such as code obfuscation or anti-tampering checks, making it difficult for an attacker to manipulate the application at runtime.

**3. Test 5: Runtime Behavior Monitoring**

- ✓ Induce attempts of analyzing and modifying the app at runtime and ensure that the application detects and prevents these actions, and thus code behavior is not altered under such conditions.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda

José L Fernández-Alemán  
Juan Manuel Carrillo-de-Gea  
Joaquín Nicolás  
08/01/2026

<b>PUID:</b> [SECM-CAT-ISU-013]
<b>Requirement description:</b> The application should check that all of its elements will be detected and are necessary for the application to function. Constantly manage an inventory of all infrastructure-related assets—including those that are mobile, IoT, and cloud-based—to identify, protect, and remediate any unauthorized or unmanaged asset.
<b>Source:</b> <ul style="list-style-type: none"><li>✓ 1.1: Verify all application components are identified and are known to be needed [19].</li><li>✓ CIS Critical Security Control 1: Inventory and Control of Enterprise Assets: Actively manage (inventory, track, and correct) all enterprise assets (end-user devices, including portable and mobile; network devices; non-computing/Internet of Things (IoT) devices; and servers) connected to the infrastructure physically, virtually, remotely, and those within cloud environments, to accurately know the totality of assets that need to be monitored and protected within the enterprise. This will also support identifying unauthorized and unmanaged assets to remove or remediate [14].</li></ul>
<b>Priority:</b> Not described
<b>Rationale:</b> This helps to ensure that the application runs with minimal components, and all assets are accounted for. It helps prevent unauthorized access and ensures that the infrastructure is protected from unmanaged or possibly harmful devices by maintaining an inventory of all application components and enterprise assets. Security management system of the VMs will dynamically manage the life cycle of the VMs, and after monitoring and tracking the potentially compromised devices, they can be removed from the system in the event an attack is detected.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b> <ol style="list-style-type: none"><li><b>1. Verification of Application Component:</b><ul style="list-style-type: none"><li>✓ Be sure that you have identified, tracked, and validated that all application components are required for the application to work.</li></ul></li><li><b>2. Enterprise Asset Inventory Management—</b></li></ol>

- ✓ Ensure the active maintenance of an inventory of all assets, including mobile, IoT, network, and cloud-based components.
- ✓ Authorization includes mechanisms for monitoring, tracking, and correcting any unauthorized or unmanaged assets in the inventory process.

### 3. Test for Unauthorized Asset Detection

- ✓ Plug in an unauthorized asset into the network/infrastructure and verify that it is detected, documented, and flagged for remediation.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda

José L Fernández-Alemán

Juan Manuel Carrillo-de-Gea

Joaquín Nicolás

08/01/2026

**PUID:** [SECM-CAT-ISU-014]

**Requirement description:** If a class extending `SSLSocketFactory` is being used, the `checkServerTrusted` method of that class must be correctly implemented and must validate the server certificate

**Source:**

- ✓ Android Specific Best Practices: 2. If using a class which extends `SSLSocketFactory`, make sure `checkServerTrusted` method is properly implemented so that server certificate is correctly checked [3].

**Priority:** Not described

**Rationale:** This prescriptive rule aims to promote the resilience of the application against SSL/TLS-based attacks by mandating a robust cross-validation process for server certificate validation. Improper implementation of the `checkServerTrusted` method could allow the application to accept untrusted certificates, thus potentially exposing the user to man-in-the-middle (MITM) attacks. Overriding `checkServerTrusted` correctly in every subclass of `SSLSocketFactory` used by the application guarantees a correct SSL/TLS handshake, confirming that the server delivered an authentic certificate, and thus securing the interaction.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

#### 1. Review of `SSLSocketFactory` Implementation

<ul style="list-style-type: none"> <li>✓ Ensure that any custom class extending <code>SSLSocketFactory</code> correctly implements <code>checkServerTrusted</code> to check server certs.</li> </ul> <p><b>2. Certificate Verification Examination:</b></p> <ul style="list-style-type: none"> <li>✓ Check the untrusted/self-signed certificates, and this should automatically be detected and rejected by the application if not explicitly trusted.</li> </ul> <p><b>3. Resistance from MITM: Security Check:</b></p> <ul style="list-style-type: none"> <li>✓ You could try to create a man-in-the-middle (MITM) attack to check if the <code>checkServerTrusted</code> method does not let untrusted errors pass.</li> </ul>
--

<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-ISU-015]
<b>Requirement description:</b> OS-provided file encryption APIs and secure key storage should be used to encrypt sensitive data (e.g., passwords, authorization tokens) and keep keys secure on the device.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ 1.9. data encryption and secure key management are especially important [16]</li> <li>✓ Java Cryptography Architecture (JCA) Security Providers: Use the Google-provided JCA security providers in the specified order. Prioritize using the highest level of framework implementations available for your use case. [18]</li> <li>✓ 1.3. When storing sensitive data on the device, use a file encryption API provided by the OS or other trusted source. Some platforms (e.g., iOS and Android) provide file encryption API's which use a secret key protected by the device unlock code and delete-able on remote wipe. If this is available, it should be used as it increases the security of the encryption without creating extra burden on the end-user. It also makes stored data safer in the case of loss or theft. However, it should be borne in mind that even when protected by the device unlock key, if data is stored on the device, its security is dependent on the security of the device unlock code if remote deletion of the key is for any reason not possible [16].</li> <li>✓ SR7: Secure Storage Since mobile devices are prone to threats, there must be secure storage on the mobile device to store health records and cryptographic credentials [30].</li> <li>✓ 2.21. In the case passwords need to be stored on the device, leverage the encryption and key-store mechanisms provided by the mobile OS to securely store passwords, password equivalents and authorization tokens. Never store passwords in clear text. Do not store passwords or long-term session IDs without appropriate encryption [16].</li> </ul>
<b>Priority:</b> Not described

**Rationale:** This requirement aims to augment the SSL/TLS security of the application by comprehensively checking the server X509 certificates before establishing the SSL/TLS connection. If the `checkServerTrusted` method is not implemented properly, the application will accept untrusted certificates which may put the users at risk of man-in-the-middle (MITM) attacks. When the application uses an `SSLSocketFactory` instance, by implementing `checkServerTrusted` properly in any custom `SSLSocketFactory` class, the application can enforce a proper SSL/TLS handshake, verify the authenticity of the server's certificate, and enhance secure communication.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

**1. Performance of an Encryption Implementation Verification:**

- ✓ Ensure OS-provided file encryption APIs or other trusted encryption sources encrypt all sensitive data stored on the device.
- ✓ Encryption is implemented with a secret key linked to the device unlock code and that remote wipe is possible if relevant SoS Toolkit 3.0, SoS Toolkit 4.0, SoS Toolkit 4.1, SoS Toolkit 4.2.

**2. Secure Key Management Check:**

- ✓ Manage keys in a secure way and prevent storing keys in clear text or means in a non-secure way.
- ✓ Ensure encryption keys use secure key-store mechanisms the mobile OS provides.

**3. Review of Password and Token Storage:**

- ✓ Ensure that passwords, password equivalents, and authorization tokens are encrypted and stored securely using OS-provided mechanisms.
- ✓ Passwords should never be stored in plain text, and long-term session IDs should be properly hashed.

**4. Adherence to JCA Guidelines:**

- ✓ Ensure that the application is using Google-provided JCA security providers in the recommended order and favors the implementations of a higher available security framework.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda

José L Fernández-Alemán

Juan Manuel Carrillo-de-Gea

Joaquín Nicolás

08/01/2026

<b>PUID:</b> [SECM-CAT-ISU-016]
<b>Requirement description:</b> The application shall restrict the embedding of one data type within another through organization-defined limits to ensure the integrity of the data and to restrict unauthorized nesting of data.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ SRG-APP-000057-MAPP-000017: The mobile app must enforce organization-defined limitations on the embedding of data types within other data types [17].</li> <li>✓ Easily adjustable security settings for different types of data [26].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> This rule enforces data integrity by preventing the embedding of one data type inside another data type in accordance with organizational policy. Improper data embedding creates potential vulnerabilities like data injection or type mismatch attack vectors that are prone to using these routes to inject malicious content or bypass query validation. The application restricts unintentional nesting of data by imposing rigid constraints, reducing the possibility of unforeseen security threats.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Enforcing Limitations on Embedded Content:</b> <ul style="list-style-type: none"> <li>✓ Ensure the application enforces organization-defined limits on embedding one data type within another.</li> <li>✓ Monitoring, detection, and blocking of unauthorized or unsupported nested data.</li> </ul> </li> <li><b>2. Data Integrity Testing:</b> <ul style="list-style-type: none"> <li>✓ Empirical test cases for scenarios where one data type has the other one embedded inside it and to check that the application validates and enforces the limitations defined without any failure.</li> </ul> </li> <li><b>3. Code Review of Policies in Embed:</b> <ul style="list-style-type: none"> <li>✓ Review of application Code to ensure that rules on data embedding based on Organizational policies have been implemented.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>

Carlos M. Mejía-Granda  
 José L Fernández-Alemán  
 Juan Manuel Carrillo-de-Gea  
 Joaquín Nicolás  
 08/01/2026

**PUID:** [SECM-CAT-ISU-017]

**Requirement description:** If tampering or fraud is detected, the application backend server should take the appropriate action based on the integrity check result to implement preventive measures, block potential attacks, and help reduce abuse.

**Source:**

- ✓ Server Response to Integrity Checks: The system's backend server must respond appropriately to integrity check results by taking preventive actions to block potential attacks and reduce abuse when tampering or fraud is detected [18].

**Priority:** Not described

**Rationale:** Here this is a must as the backend server has to play an active role in maintaining the security of the application by doing integrity checks and taking preventive measures. When tampering or fraud is detected, the server's reaction is crucial for mitigating abuse, minimizing sensitive data leakage, and preventing further exploitation of vulnerabilities. Focusing on integrity issues improves the application's overall resilience and can lessen the impact that unauthorized activities may have.

**Number of Children:** 0**Number of parents:** 0**Cycles:** 0**Number Audit:** 0**Child PUIDs:** Not described**Parent PUIDs:** Not described**Exclusion PUIDs:** Not described**Importance:** Not described**Current state:** To be determined**Verification method:** Demonstration/Analysis**Validation criteria:****1. Check Response validation for Integrity Check:**

- ✓ Ensure that the backend server behaves correctly on integrity checks to detect any tampering or fraud.
- ✓ Within the server, set preventive types of actions to mitigate such risk (such as blocking or flagging suspicious activities).

**2. Server Activity Simulation With Attack Simulation:**

- ✓ Simulate integrity violations, verify the server responds by forcefully denying users access or limiting the likelihood of abuse.

**3. Integrity response mechanisms — code review**

- ✓ Ensure proper implementation in backend code for mechanisms like integrity checks to be validated and preventive steps to be invoked

**Requested by:** The organization**Responsible:** Developer

<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-ISU-018]
<b>Requirement description:</b> The application must validate and only accept server responses from backend APIs to ensure the data is both correct and secure.
<b>Source:</b>
✓ 12.11. Always validate server responses when using backend APIs. Introduce a whitelist model for accepted responses [16].
<b>Priority:</b> Not described
<b>Rationale:</b> Following this guideline is crucial for ensuring token-based authentication methods for mobile application and backend APIs, where data integrity and trustworthiness are key considerations especially when dealing with sensitive health information. This prevents the application from accepting or processing the data in the server response without the risk of malicious data or unexpected data, which could cause unauthorized access, corrupted data, and system vulnerabilities, as you can imagine. This is important to ensure sensitive patient data is protected and healthcare security standards are followed.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Server Response Validation:</b> <ul style="list-style-type: none"> <li>✓ Ensure the application verifies the integrity and correctness of all frontend/api server responses.</li> <li>✓ Do not process invalid responses in the application.</li> </ul> </li> <li><b>2. Step: Implementation of the Whitelist Model</b> <ul style="list-style-type: none"> <li>✓ Validate that a whitelist approach is followed for acceptable server responses.</li> <li>✓ Ensure the application processes only known, authorized response types.</li> </ul> </li> <li><b>3. Simulating the Unexpected — Responses:</b> <ul style="list-style-type: none"> <li>✓ Test can be performed by returning unexpected or malicious responses from the server to ensure the application rejects it and it does not affect the functionality or security.</li> </ul> </li> </ol>

<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda
José L Fernández-Alemán
Juan Manuel Carrillo-de-Gea
Joaquín Nicolás
08/01/2026

<b>PUID:</b> [SECM-CAT-ISU-019]
<b>Requirement description:</b> The app should not store any biometric data or private keys on the client device, thus would store and process sensitive data only on the server.
<b>Source:</b>
✓ 1.2. Store and process sensitive data on the server instead of the client-end device. Highly sensitive data (e.g., biometric data, private keys) should not be transported from the component that were initially created [16].
<b>Priority:</b> Not described
<b>Rationale:</b> There is an increasing necessity to protect sensitive data, especially for healthcare applications — the security and privacy of biometric data, private keys, and highly sensitive data is essential. Sensitive data is better protected against unauthorized access, theft, or tampering than on client devices when it is kept and processed on secure servers. Furthermore, sensitive data remain in the environment in which they were originally created, eliminating the need for transferring data that can be intercepted, in turn improving compliance toward healthcare data security standards.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Check for the Data Storage and Processing:</b> <ul style="list-style-type: none"> <li>✓ Ensure that sensitive data like biometric data and private keys are stored and processed only on the server.</li> <li>✓ Do not transfer sensitive data to (or process it on) a client device.</li> </ul> </li> <li><b>2. Server Security Assessment:</b> <ul style="list-style-type: none"> <li>✓ Ensure the server environment is up to security best practices, including but not limited to; access control and encryption, especially for sensitive data.</li> </ul> </li> <li><b>3. A test I struggle with: Data transfer limit</b></li> </ol>

✓ Attempt to exfiltrate sensitive data from the server to a client device and ensure that such operations are denied.
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-ISU-020]
<b>Requirement description:</b> There should be means of providing feedback (e.g., provide a security email address (e.g., security@domain.com)).
<b>Source:</b>
✓ 9.3. Provide feedback channels for users to report security problems with apps such as a security@ email address [16].
<b>Priority:</b> Not described
<b>Rationale:</b> Using this requirement, users must have access to an effective and direct way of reporting security issues, so the application must have better responsiveness about security vulnerabilities and threats. In healthcare applications, timely reporting and resolution of security problems becomes critical since they are responsible for securing sensitive patient information. A feedback system implemented correctly allows the organization to effectively correct problems and improve the security posture of the application.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Where To Get Feedback:</b> <ul style="list-style-type: none"> <li>✓ Confirm there is a configured and functional means of receiving security-related feedback (e.g., a security email address).</li> <li>✓ Show clearly how to provide feedback in the app or documentation (e.g., help section, privacy policy).</li> </ul> </li> <li><b>2. Feedback Workflow Testing:</b> <ul style="list-style-type: none"> <li>✓ Send a message to the security feedback channel and verify receipt of the message.</li> <li>✓ Verify if the feedback channel has an automated or manual acknowledgment mechanism to ensure that security reports are received.</li> </ul> </li> <li><b>3. Security Incident Tracking:</b></li> </ol>

<ul style="list-style-type: none"> <li>✓ Check ensuring that security reports submitted are logged with a unique identifier which can be used for tracking purposes.</li> <li>✓ Ensure that there is a mechanism that ensures all logged reports are reviewed and addressed, including documentation of actions taken to resolve issues.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-ISU-021]
<b>Requirement description:</b> Application should analyze and perform client-side data processing for sensitive operations (if applicable) to avoid sending confidential data to the server unnecessarily.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ Client-Side Data Processing: The application should evaluate whether sensitive data operations can be performed on the client-side to avoid unnecessary data transmission to a server [18].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> This is important to secure the handling of sensitive data and reduce the cost of moving it to a server. A common client-side operation involves the use of client-side analytics tools that can collect user behavior data without sending it to the server. This approach balances functionality and confidentiality through its consideration of client-side data processing, which reduces security risks while maintaining data protection principles.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Feasibility Analysis on Client Side:</b> <ul style="list-style-type: none"> <li>✓ Ensure that during the design phase, the application can determine if it is feasible to carry out sensitive data operations on the client side. Additional recommendations:</li> <li>✓ Have a framework for the documented decision-making process for when client-side data processing is appropriate.</li> </ul> </li> <li><b>2. Data Transmission Audit:</b></li> </ol>

- ✓ Conduct data flow and network log review to ensure that no other sensitive data is being sent to the server.
- ✓ Always perform sensitive operations client-side to minimize communication redundancy with the server.

### 3. Functionality Testing:

- ✓ Perform application tests to ensure that client-side processing of data does not affect the functionality, security, or performance of the application.
- ✓ Ensure that sensitive data processed on the client is kept secure and is not stored locally or in memory.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda  
 José L Fernández-Alemán  
 Juan Manuel Carrillo-de-Gea  
 Joaquín Nicolás  
 08/01/2026

**PUID:** [SECM-CAT-ISU-022]

**Requirement description:** The application must have at least one administrator signed up for receiving update notifications and security alerts to ensure timely patching/updates of both the application as well as its components.

**Source:**

- ✓ V-222669: At least one application administrator must be registered to receive update notifications or security alerts when automated alerts are available. Register administrators to receive update notifications so they can patch and update applications and application components [15].

**Priority:** Not described

**Rationale:** This requirement will send notifications when there are significant updates or security issues, and help apply patches and updates without delay. Prompt updates are necessary to fix vulnerabilities and preserve the security health of clinical applications, which normally manage private patient data. Automated notifications decrease the chances of these critical updates going unnoticed, protecting your organization from new vulnerabilities and maintaining compliance with security best practices.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

**1. Verify Administrator Registration:**

- ✓ Make sure at least one admin is enrolled on the application to receive update notifications and security alerts.

**2. Notification System Testing:**

- ✓ Check the notification system that sends updates and security alerts to registered administrators automatically. Ensure notifications provide guidance or references to patches or updates.

**3. Timeliness of Updates:**

- ✓ Ensure that administrators are alerted in a timely manner of available updates or patches. Ensure that administrators can monitor notifications and take action on them in an adequate time frame.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda  
José L Fernández-Alemán  
Juan Manuel Carrillo-de-Gea  
Joaquín Nicolás  
08/01/2026

**PUID:** [SECM-CAT-ISU-023]

**Requirement description:** The application must have a method of forcing updates of the app for critical security patches so that vulnerabilities are resolved before the app is used. It should also be structured in such a way that security updates can be made end-to-end in a timely fashion while respecting app-store approval processes.

**Source:**

- ✓ 164.308 Administrative safeguards.
  - (a) A covered entity or business associate must, in accordance with § 164.306:
    - (5)
      - (ii) Implementation specifications:
        - (A) Security reminders (Addressable)  
Periodic security updates [9].
  - ✓ MASVS-CODE-2: The app has a mechanism for enforcing app updates: Sometimes critical vulnerabilities are discovered in the app when it is already in production. This control ensures that there is a mechanism to force the users to update the app before they can continue using it [20].

- ✓ Implementation of countermeasures for critical risks: 1. The system must implement appropriate countermeasures for identified critical risks to mitigate the impact of potential attacks [3].
- ✓ 9.1. Applications must be designed and provisioned to allow updates for security patches, taking into account the requirements for approval by app-stores and the extra delay this may imply [16].
- ✓ Lack of Security Awareness: Establish security controls for app updates, patches, and releases to prevent attackers from exploiting vulnerabilities in the app [3].

**Priority:** Not described

**Rationale:** This is important for maintaining the security of healthcare applications by forcing the updates that patch important security holes. Applications deployed on production can be subject to emerging attacks, and it is essential to have mechanisms in place to encourage (with the risk of crippling production) or force users to update (with the risk of doing the same) to a patched version. Furthermore, designing the application to facilitate quick updates (while accounting for the approval process for app-stores) minimizes delays in the fix being deployed, thus ensuring that vulnerabilities do not get exploited.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

**1. Are you proving and verifying these update mechanisms:**

- ✓ Ensure that the application has a process for enforcing updates when critical vulnerabilities are discovered.
- ✓ Ensure users cannot skip the update enforcement and have to update their app in order to continue using it.

**2. Security Patch Deployment:**

- ✓ Check whether the application can receive and apply the security patch on time.
- ✓ Collaborate with app-store approval requirements to enable integration of patches and reduce delays.

**3. Test of the Critical Risk Mitigation**

- ✓ Conduct simulations on identified critical risks and verify that countermeasures, including enforced updates, are deployed and effective in mitigating the threat.

**4. Notice of Security Updates†**

- ✓ Provide clear and timely notifications to users about the need for updates (e.g., information on security patches that have been addressed).

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda  
 José L Fernández-Alemán  
 Juan Manuel Carrillo-de-Gea  
 Joaquín Nicolás  
 08/01/2026

**PUID: [SECM-CAT-ISU-024]**

**Requirement description:** Security updates must all be shipped using an encrypted connection and the security update must be checked from their proper publisher. This ensures the confidentiality and integrity of non-local maintenance and diagnostic communications using cryptographic mechanisms.

**Source:**

- ✓ 9.5. Out-of-appstore security updates should be shipped using an encrypted connection and their content should be verified before applying the update [16].
- ✓ V-222563: Applications used for non-local maintenance sessions must implement cryptographic mechanisms to protect the confidentiality of non-local maintenance and diagnostic communications. Configure the application to encrypt remote application maintenance sessions [15].
- ✓ V-222562: Applications used for non-local maintenance sessions must implement cryptographic mechanisms to protect the integrity of non-local maintenance and diagnostic communications. Configure the application to encrypt remote application maintenance sessions [15].

**Priority:** Not described

**Rationale:** This is very important for healthcare applications that are dealing with sensitive data — it guarantees the distribution of security updates by a secure mechanism outside of the app store that is applied directly. By encrypting connections, updates are safeguarded against interception during transfer, while integrity verification is used to ensure that only valid updates are applied, thereby reducing the possibility of tampering. Provides cryptographic protection for non-local maintenance and diagnostic communication to ensure the confidentiality and integrity of sensitive interactions to avoid exploitation as presented during remote sessions.

**Number of Children:** 0**Number of parents:** 0**Cycles:** 0**Number Audit:** 0**Child PUIDs:** Not described**Parent PUIDs:** Not described**Exclusion PUIDs:** Not described**Importance:** Not described**Current state:** To be determined**Verification method:** Demonstration/Analysis**Validation criteria:**

<p><b>1. Update transmission is encrypted:</b></p> <ul style="list-style-type: none"> <li>✓ Check if out-of-app-store updates are delivered on encrypted routes (e.g., TLS/SSL).</li> <li>✓ Provide an initial check with secure connection before the update.</li> </ul> <p><b>2. Integrity Verification for Updates:</b></p> <ul style="list-style-type: none"> <li>✓ Verify all updates outside the app store for authenticity and integrity before the application.</li> <li>✓ Succeeding with an affected update and be confident that it is identified and declined.</li> </ul> <p><b>3. Maintenance Sessions Are Secure with Cryptographic Protection:</b></p> <ul style="list-style-type: none"> <li>✓ Verify that strong cryptographic mechanisms are in place to encrypt all data in transit during remote maintenance sessions.</li> <li>✓ Ensure integrity of communications during remote sessions are cryptographically checked.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-ISU-025]
<b>Requirement description:</b> App must validate that resources outside the app-store mechanisms which will be consumed are signed and get validated before updating and applying.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ 9.6. Resources used by apps that are updated outside of the app-store normal mechanism must be signed. Apps must verify the signature before accepting the updated resource [16].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> This requirement helps to ensure the integrity and authenticity of content that can be updated by means other than the usual app-store mechanisms. Use of signature verification ensures that only trusted updates that have not been tampered with are accepted. This is especially critical in healthcare applications because unauthorized updates will affect sensitive data or application behavior, putting back-end systems and patient information at great risk.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li>1. <b>Signature: An Application to the External Resources</b></li> </ol>

<ul style="list-style-type: none"> <li>✓ Ensure all resources that are to be updated outside of app-store mechanisms are signed using a cryptographically strong signature.</li> <li>✓ Confirm that unsigned resources are rejected on the update path.</li> </ul> <p><b>2. Signature Validation Testing:</b></p> <ul style="list-style-type: none"> <li>✓ Validate the signature of updated resources to check the application.</li> <li>✓ Attempt to apply an update with a bad or missing signature [simulate] and verify that the bad update is detected and rejected.</li> </ul> <p><b>3. Resistance to Tampering Validation:</b></p> <ul style="list-style-type: none"> <li>✓ Trigger a tampered update and confirm the detection of changes by the application, ensuring the resource will not be applied.</li> </ul> <p><b>4. We organized periodic reviews of the Resource Update Workflow:</b></p> <ul style="list-style-type: none"> <li>✓ Implement an automatic signature verification in the update workflow that cannot be manually overridden before it applies the updates to any resources.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<p><b>Author and date:</b></p> <p>Carlos M. Mejía-Granda      José L Fernández-Alemán      Juan Manuel Carrillo-de-Gea      Joaquín Nicolás      08/01/2026</p>

<b>PUID:</b> [SECM-CAT-ISU-026]
<b>Requirement description:</b> The application must implement appropriate mechanisms to protect the integrity of its code and resources, preventing unauthorized modifications, and ensuring that only appropriate and authorized content is loaded within the application.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ MASVS-RESILIENCE-2: The app implements anti-tampering mechanisms: Apps run on a user-controlled device, and without proper protections it's relatively easy to run a modified version locally (e.g. to cheat in a game, or enable premium features without paying), or upload a backdoored version of it to third-party app stores. This control tries to ensure the integrity of the app's intended functionality by preventing modifications to the original code and resources. [20].</li> <li>✓ Tamper-proof access logs with different views [34].</li> <li>✓ Test ID 10: Verify that only mission-appropriate content may be uploaded within the application [37].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Helps ensure the integrity and security of the healthcare application by preventing tampering. However, in the absence of anti-tampering mechanisms, attackers can modify the application in ways that circumvent security controls, insert backdoors, or activate unwanted features. As it prevents uploading a certain type of content, it helps reduce the entry of potentially

harmful or inappropriate information in the application and ensures security and operation standards are maintained.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Verification of Anti-Tampering Mechanism:</b> <ul style="list-style-type: none"> <li>✓ Confirm that processes are in place for anti-tampering protections (e.g., checksum validation, code obfuscation); ensure no modification of the application code or resources can go undetected.</li> <li>✓ Attempt to tamper with the application's code and ensure that the modification is detected and blocked.</li> </ul> </li> <li><b>2. Testing Content Upload Restriction:</b> <ul style="list-style-type: none"> <li>✓ Validate the application to make sure overly aggressive content is strictly restricted.</li> <li>✓ Simulate an attempt to upload non-compliant or inappropriate content, and verify that the application does not allow it.</li> </ul> </li> <li><b>3. Tampering Response Validation:</b> <ul style="list-style-type: none"> <li>✓ Validate that the application maintains a proper response, and takes the respective action, for example, disable functionality or alert administrators.</li> </ul> </li> <li><b>4. Code Integrity Check Review:</b> <ul style="list-style-type: none"> <li>✓ Verify code integrity checks are implemented to ensure original code and resources are intact throughout the application lifecycle.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-ISU-027]
<b>Requirement description:</b> The user-facing components of the application must be logically or physically separated from the data storage and management interfaces and security functions. And security-related functionality is to be isolated; you can't have someone open an admin function in a different corner of your site.
<b>Source:</b>

- ✓ V-222574: The application user interface must be either physically or logically separated from data storage and management interfaces. Configure the application so user interface to the application and management interface to the application is separated [15].

#### ✓ SC-2 SEPARATION OF SYSTEM AND USER FUNCTIONALITY

**Control:** Separate user functionality, including user interface services, from system management functionality.

**Discussion:** System management functionality includes functions that are necessary to administer databases, network components, workstations, or servers. These functions typically require privileged user access. The separation of user functions from system management functions is physical or logical. Organizations may separate system management functions from user functions by using different computers, instances of operating systems, central processing units, or network addresses; by employing virtualization techniques; or some combination of these or other methods. Separation of system management functions from user functions includes web administrative interfaces that employ separate authentication methods for users of any other system resources. Separation of system and user functions may include isolating administrative interfaces on different domains and with additional access controls. The separation of system and user functionality can be achieved by applying the systems security engineering design principles in SA-8, including SA-8(1), SA-8(3), SA-8(4), SA-8(10), SA-8(12), SA-8(13), SA-8(14), and SA-8(18) [11].

- ✓ V-222590: The application must isolate security functions from non-security functions. Implement controls within the application that limits access to security configuration functionality and isolates regular application function from security-oriented function [15].

<b>Priority:</b> Not described
--------------------------------

**Rationale:** By separating sensitive operations from user-oriented features, this requirement mitigates the potential for security breaches and minimizes the attack surface by preventing unauthorized access to critical system functions. In healthcare applications, we have sensitive data and critical operations, so the separation of these functionalities is key to ensuring security and proper operation. Logical or physical separation of user and administrative interfaces helps you to mitigate privilege escalation attacks and enforce stricter access controls to management and security functions.

<b>Number of Children:</b> 0
------------------------------

<b>Number of parents:</b> 0
-----------------------------

<b>Cycles:</b> 0
------------------

<b>Number Audit:</b> 0
------------------------

<b>Child PUIDs:</b> Not described
-----------------------------------

<b>Parent PUIDs:</b> Not described
------------------------------------

<b>Exclusion PUIDs:</b> Not described
---------------------------------------

<b>Importance:</b> Not described
----------------------------------

<b>Current state:</b> To be determined
--

<b>Verification method:</b> Demonstration/Analysis
--

<b>Validation criteria:</b>
-----------------------------

1. Interface Separation Testing:

<ul style="list-style-type: none"> <li>✓ Ensure that interfaces for users are logically or physically separated from interfaces for storage and management of data.</li> <li>✓ Ensure user interfaces have no direct access to management functions or features.</li> </ul> <p><b>2. Validation of security function isolation:</b></p> <ul style="list-style-type: none"> <li>✓ Decouple security functions (configuration, access control) from the normal application functionality.</li> </ul> <p><b>3. Access control mechanism review:</b></p> <ul style="list-style-type: none"> <li>✓ Ensure proper segmentation and enforcement of access rules between management and security interfaces and the service plane to limit access to authorized personnel only.</li> <li>✓ Attempt to bypass any controls in place that prevent normal users from accessing privileged functionality.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-ISU-028]
<b>Requirement description:</b> Control the application response includes keeping all third-party libraries, frameworks, etc., up-to-date, free from known vulnerabilities and regularly assessing them for potential security threats. Also, it is necessary to perform periodic security updates and vulnerability assessments and keep a record of results for compliance verification purposes.
<b>Source:</b> <ul style="list-style-type: none"> <li>✓ V-222574: The application user interface must be either physically or logically separated from data storage and management interfaces. Configure the application so user interface to the application and management interface to the application is separated [15].</li> <li>✓ 164.308 Administrative safeguards.           <ul style="list-style-type: none"> <li>(a) A covered entity or business associate must, in accordance with § 164.306:               <ul style="list-style-type: none"> <li>(5) (ii) Implementation specifications: (B) Security reminders (Addressable) Periodic security updates [9].</li> </ul> </li> </ul> </li> <li>✓ Keep your security software up to date [32].</li> <li>✓ 1.2: Verify all third-party components used by the mobile app, such as libraries and frameworks, are identified, and checked for known vulnerabilities [19].</li> </ul>

<ul style="list-style-type: none"><li>✓ 1.8: Verify all third-party components have been assessed (associated risks) before being used or implemented. Additionally verify that a process is in place to ensure that each time a security update for a third-party component is published, the change is inspected and the risk evaluated [19].</li><li>✓ Stay Informed: 1. Stay up to date with the latest security threats and vulnerabilities in the mobile application landscape. Monitor security forums, security advisories, and mobile platform updates to ensure timely mitigation of emerging risks [19].</li><li>✓ 1.10: Verify that all components that are not part of the application but that the application relies on to operate, are clearly identified and the security implications of using those components are known [19].</li><li>✓ Keep your code up-to-date: Be sure to update your source code, including any third-party libraries and dependencies, to guard against the latest vulnerabilities [18].</li><li>✓ Stay informed: Stay updated on the latest security threats and best practices for API key management [18].</li><li>✓ Lack of Security in Third-Party Components:<ul style="list-style-type: none"><li>○ Implement secure coding practices, code review, and testing throughout the mobile app development lifecycle to identify and mitigate vulnerabilities.</li><li>○ Use only trusted and validated third-party libraries or components to reduce the risk of vulnerabilities[3].</li></ul></li><li>✓ MASVS-CODE-3: The app only uses software components without known vulnerabilities: To be truly secure, a full Whitebox assessment should have been performed on all app components. However, as it usually happens with e.g. for third-party components this is not always feasible and not typically part of a penetration test. This control covers “low-hanging fruit” cases, such as those that can be detected just by scanning libraries for known vulnerabilities [20].</li><li>✓ MASVS-CODE-1: The app requires an up-to-date platform version: Every release of the mobile OS includes security patches and new security features. By supporting older versions, apps stay vulnerable to well-known threats. This control ensures that the app is running on an up-to-date platform version so that users have the latest security protections [20].</li><li>✓ V-222515: An application vulnerability assessment must be conducted. Configure the application vulnerability scanners to test all components of the application, conduct vulnerability scans on a regular basis and remediate identified issues. Retain scan results for compliance verification [15].</li></ul>
<p><b>Priority:</b> Not described</p> <p><b>Rationale:</b> This requirement ensures the security of healthcare applications by addressing risks in third-party components and platform dependencies. Vulnerabilities in outdated or unassessed libraries can expose sensitive data and compromise system functionality. Regular updates and vulnerability scans reduce these risks, ensuring compliance with security best practices. Timely</p>

remediation and documentation of scan results further enhance the application's resilience and support regulatory requirements.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Identify Components and Check for Vulnerabilities:</b> <ul style="list-style-type: none"> <li>✓ Ensure that all third-party components of the application are identified and checked against known vulnerability databases.</li> <li>✓ Make sure only trusted and validated third-party components are used in the application.</li> </ul> </li> <li><b>2. Confirm Patches and Updates:</b> <ul style="list-style-type: none"> <li>✓ Ensure that application components — including source code, libraries, and dependencies — are up-to-date and that they are the latest secure versions.</li> <li>✓ Verify the application is enforcing that it uses an up-to-date version of the platform.</li> </ul> </li> <li><b>3. Vulnerability Assessment Process:</b> <ul style="list-style-type: none"> <li>✓ Determine that all application components undergo regular vulnerability scans.</li> <li>✓ Ensure that identified vulnerabilities are remediated in an acceptable timeframe and that scan results are kept for compliance indication.</li> </ul> </li> <li><b>4. Security Monitoring Logging and Alerts:</b> <ul style="list-style-type: none"> <li>✓ Application team keeping an eye on security advisories and forums for fields or requests.</li> <li>✓ Verify that there is an established process to address and assess the risk of newly identified vulnerabilities.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-ISU-029]
<b>Requirement description:</b> The app must timely notifications or alerts for product updates and security patches, containing a description of the issue, a summary of the associated risks, potential mitigations, and instructions on how to obtain it. To accomplish secure updates, a distribution mechanism must also be implemented.
<b>Source:</b>

<ul style="list-style-type: none"> <li>✓ V-222670: The application must provide notifications or alerts when product update and security related patches are available. Provide a distribution mechanism for obtaining updates to the application. Include a description of the issue, a summary of risk as well as potential mitigations and how to obtain the update [15].</li> <li>✓ 164.308 Administrative safeguards.           <ul style="list-style-type: none"> <li>(a) A covered entity or business associate must, in accordance with § 164.306:               <ul style="list-style-type: none"> <li>(5) (ii) Implementation specifications:</li> <li>(C) Security reminders (Addressable)</li> </ul> </li> </ul> </li> </ul> <p style="text-align: center;">Periodic security updates [9].</p>
<b>Priority:</b> Not described
<b>Rationale:</b> This requirement keeps users aware of updates to the healthcare application and any associated security patches to maintain the security and resiliency of the application. Specifically, clear risk, mitigation, and update communication allows users and administrators to take corrective action in a timely fashion, reducing their exposure to the vulnerability at hand. Based on this distribution method, one can always deliver the updates with high security and reliability, ensuring sensitive healthcare data and system integrity.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Check if your data is up-to-date</b> <ul style="list-style-type: none"> <li>✓ Ensure the app will have notifications or alerts when new product or security updates are available.</li> <li>✓ Ensure all notifications have problem details, risk, potential mitigations, and steps to get that fix.</li> </ul> </li> <li><b>2. Test: Secure Distribution Mechanism</b> Verify that the application offers one or more secure and reliable distribution mechanisms for updates.           <ul style="list-style-type: none"> <li>✓ Integrity and Authenticity Checking of Updates While Transmit and Apply</li> </ul> </li> <li><b>3. Risk Communication Verification:</b> <ul style="list-style-type: none"> <li>✓ Make notifications clearly indicate the urgency of the update in terms of the risk it mitigates.</li> <li>✓ Simulate an update process and ensure that users have all the information they need to perform a safe and timely update.</li> </ul> </li> <li><b>4. Periodic Update Reminder Test:</b> <ul style="list-style-type: none"> <li>✓ Ensure that the application notifies the user periodically for any pending updates or patches to take necessary action in a timely manner.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization

<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-ISU-030]
<b>Requirement description:</b> The app must keep all the SDKs, libraries, and platform components up-to-date with their latest version to patch vulnerabilities and adopt security improvements. Software is periodically updated through patches to cryptographic components and security measures.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ Make sure your SDKs and libraries are updated to the latest version [18].</li> <li>✓ MASVS-CODE-1: The app requires an up-to-date platform version: Every release of the mobile OS includes security patches and new security features. By supporting older versions, apps stay vulnerable to well-known threats. This control ensures that the app is running on an up-to-date platform version so that users have the latest security protections [20].</li> <li>✓ MASVS-CODE-3: The app only uses software components without known vulnerabilities: To be truly secure, a full whitebox assessment should have been performed on all app components. However, as it usually happens with e.g. for third-party components this is not always feasible and not typically part of a penetration test. This control covers “low-hanging fruit” cases, such as those that can be detected just by scanning libraries for known vulnerabilities [20].</li> <li>✓ Regularly Update Security Measures: 1. Stay informed about security updates, patches, and recommendations from cryptographic libraries, frameworks, and platform providers. Keep the mobile application and underlying cryptographic components up to date to address any identified vulnerabilities or weaknesses [3].</li> <li>✓ Make security updates regularly through patches [25].</li> </ul>

<b>Priority:</b> Not described
<b>Rationale:</b> All components — SDKs, libraries, and cryptographic frameworks — should be kept up to date, thereby improving the security posture of healthcare applications. This comprises npm installations, docker requirements, etc. Outdated components may leave the application open to known flaws and compromise the defenses. It also makes sure standard security updates and patches are made, helping the application to comply with best practices and reduce the risk of sensitive healthcare information being targeted by threats.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0

<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li>1. <b>Verify Component Update:</b> <ul style="list-style-type: none"> <li>✓ Ensure application SDKs, Libraries, and cryptographic frameworks are upgraded to the latest secure versions.</li> <li>✓ Ensure that there is a process to routinely check with component providers for updates.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-ISU-031]
<b>Requirement description:</b> Continuous updating of all software components and third-party libraries (e.g., SDKs, libraries, and cryptographic frameworks) ensures addressing known vulnerabilities by integrating regular security and platform updates to support the latest security features.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ V-222614: Security-relevant software updates and patches must be kept up to date. Check for application updates at least weekly and apply patches immediately or in accordance with POA&amp;Ms, IAVMs, CTOs, DTMs or other authoritative patching guidelines or sources [15].</li> <li>✓ Security Updates: Native libraries and tools used in the application must be kept up to date with the latest security patches to avoid exploitation through known vulnerabilities in outdated code. [18].</li> <li>✓ MASVS-CODE-1: The app requires an up-to-date platform version: Every release of the mobile OS includes security patches and new security features. By supporting older versions, apps stay vulnerable to well-known threats. This control ensures that the app is running on an up-to-date platform version so that users have the latest security protections [20].</li> <li>✓ MASVS-CODE-3: The app only uses software components without known vulnerabilities: To be truly secure, a full whitebox assessment should have been performed on all app components. However, as it usually happens with e.g. for third-party components this is not always feasible and not typically part of a penetration test. This control covers “low-hanging fruit” cases, such as those that can be detected just by scanning libraries for known vulnerabilities [20].</li> </ul>

- ✓ 6.2. Track third party frameworks/APIs used in the mobile application for security patches. Integrate security updates for third party code/libraries/frameworks/APIs on a regular basis together with your own code. Ask the provider for a security report [16].
- ✓ 164.308 Administrative safeguards.
  - (a) A covered entity or business associate must, in accordance with § 164.306:
    - (5)
    - (ii) Implementation specifications:
    - (C) Security reminders (Addressable)
    - Periodic security updates [9].
- ✓ Regularly Update and Patch Dependencies: 1. Keep all libraries, frameworks, and third-party dependencies up to date, as they may contain security vulnerabilities that could lead to insecure data storage. Regularly apply security patches and updates provided by the respective vendors [3].

**Priority:** Not described

**Rationale:** Software maintenance and update is a key preventive activity to protect healthcare-relevant applications from vulnerabilities. Regularly updating libraries, frameworks, and dependencies safeguards against the exploitation of known vulnerabilities, which minimizes risks to sensitive patient data as well as the integrity of the application itself. With specific patching guidelines and a regular cadence of updates, the application ensures it meets security best practices and minimizes the attack surface.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

**1. Platform Version Review:**

- ✓ Ensure that the application runs on the latest version of the platform and does not support obsolete versions & versions that have known vulnerabilities.

**2. Update Schedule Compliance:**

- ✓ Make sure that the application performs checks for security updates and patches at a minimum weekly.
- ✓ Keep applications up-to-date with the latest software patches following vendor-ironclad procedures.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda

José L Fernández-Alemán

Juan Manuel Carrillo-de-Gea

Joaquín Nicolás 08/01/2026
-------------------------------

<b>PUID:</b> [SECM-CAT-ISU-032]
<b>Requirement description:</b> Modify and maintain privileges for software libraries must be controlled within application owners.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ V-222514: The applications must limit privileges to change the software resident within software libraries.</li> </ul> <p>Configure the application OS file permissions to restrict access to software libraries and configure the application to restrict user access regarding software library update functionality to only authorized users or processes [15].</p>
<b>Priority:</b> Not described
<b>Rationale:</b> Ensuring this is a critical requirement for maintaining the integrity and security of software libraries used in healthcare applications. This enhances security by limiting who can modify or execute the application, thereby reducing the risk of unauthorized alterations that might introduce bugs or malicious software. By restricting access only at the OS-level and implementing proper permissions, we ensure that only application-level controls can modify the libraries, thus protecting sensitive data and maintaining system functionalities.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li>1. <b>Check if OS-Level Permission Configuration is set:</b> <ul style="list-style-type: none"> <li>✓ Ensure software libraries are only accessible to the appropriate users/processes via operating system file permissions.</li> <li>✓ Check that library files cannot be read, changed, or deleted by unauthorized users.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda
José L Fernández-Alemán
Juan Manuel Carrillo-de-Gea
Joaquín Nicolás
08/01/2026

**PUID:** [SECM-CAT-ISU-033]

**Requirement description:** The application shall remove out-of-date components after update installation, disable the use of unsupported components, and be decommissioned when there is no maintenance or vendor support available.

**Source:**

- ✓ V-222613: The application must remove organization-defined software components after updated versions have been installed. Configure or design the application to remove old components when updating [15].
- ✓ 6.5. Software components that are no longer supported by the vendor or developer must not be used [16].
- ✓ V-222659: The application must be decommissioned when maintenance or support is no longer available. Ensure there is maintenance for the application [15].
- ✓ SA-22 UNSUPPORTED SYSTEM COMPONENTS

## Control:

- a. Replace system components when support for the components is no longer available from the developer, vendor, or manufacturer; or
- b. Provide the following options for alternative sources for continued support for unsupported components [Selection (one or more): in-house support; [Assignment: organization-defined support from external providers]].

Discussion: Support for system components includes software patches, firmware updates, replacement parts, and maintenance contracts. An example of unsupported components includes when vendors no longer provide critical software patches or product updates, which can result in an opportunity for adversaries to exploit weaknesses in the installed components. Exceptions to replacing unsupported system components include systems that provide critical mission or business capabilities where newer technologies are not available or where the systems are so isolated that installing replacement components is not an option [11].

- ✓ V-222658: All products must be supported by the vendor or the development team. Remove or decommission all unsupported software products in the application [15].

**Priority:** Not described

**Rationale:** The need to do this forces the application to be secure or still has no risk of using frameworks that have reached End Of Life. Old systems can still harbor malware or other vulnerabilities that could be exploited by bad actors, so removing old components, disallowing unsupported software, and retiring applications that no longer have vendor support helps keep them secure and ensure integrity.

**Number of Children:** 0**Number of parents:** 0**Cycles:** 0**Number Audit:** 0**Child PUIDs:** Not described

<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Verification of Removal of Components:</b> <ul style="list-style-type: none"> <li>✓ Verify that older components are automatically or manually removed when newer versions are deployed.</li> <li>✓ Ensure no old files or obsolete code exist following an upgrade.</li> </ul> </li> <li><b>2. Unsupported Component Detection:</b> <ul style="list-style-type: none"> <li>✓ Perform application analysis to find all the technically feasible components and ensure that they are marked for removal/replacement.</li> <li>✓ Ensure unsupported units aren't usable in the system.</li> </ul> </li> <li><b>3. Process Validation for Decommissioning:</b> <ul style="list-style-type: none"> <li>✓ Ensure that there exists a process for safely decommissioning applications where vendor or developer support cannot be obtained.</li> <li>✓ Decommissioned applications should not maintain sensitive data or operational capabilities.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-ISU-034]
<b>Requirement description:</b> The application must preserve its own execution domains in each process, must comply with Android's application sandbox model, and must deploy to secure, dedicated platforms for sensitive or high-availability use.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ V-222591: The application must maintain a separate execution domain for each executing process. Design and configure applications to maintain a separate execution domain for each executing process [15].</li> <li>✓ Application Sandbox: Native code must respect the Android application sandbox, where each application operates under a unique User Identifier (UID) with restricted permissions. Developers must ensure that native code adheres to Android's sandboxing model to maintain application security [18]</li> </ul>

- ✓ V-222635: The application must not be hosted on a general-purpose machine if the application is designated as critical or high availability by the ISSO. Deploy mission critical applications on servers that are not shared by other less critical applications [15].
- ✓ MASVS-RESILIENCE-1: The app validates the integrity of the platform: Running on a platform that has been tampered with can be very dangerous for apps, as this may disable certain security features, putting the data of the app at risk. Trusting the platform is essential for many of the MASVS controls relying on the platform being secure (e.g. secure storage, biometrics, sandboxing, etc.). This control tries to validate that the OS has not been compromised and its security features can thus be trusted [20].

**Priority:** Not described

**Rationale:** This is designed for critical healthcare applications to be secure and isolated, sandboxing common applications, and deploying on secure platforms. Good separation reduces the risks of process interference, tampered platforms, and sharing of system resources with less valuable apps. Together, these controls can increase the resiliency of applications and ensure trust in platform security features.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

**1. Process Execution Isolation:**

- ✓ The execution of all the processes must be in a different execution domain, and the resources should not be shared to ensure security.

**2. Sandbox Adherence Check:**

- ✓ Ensure that the application's native code runs in Android's application sandbox subject to unique UID postfix and permissions.

**3. Deployment of Platform Dedicated:**

- ✓ Ensure your critical or high-availability application is deployed on a secure, dedicated server and not shared among low- or non-critical application hosted on general-purpose machines.

**4. Validate the Integrity of Your Platform:**

- ✓ Ensuring that the application validates that the OS has not been compromised and its integrity (and security features) is intact.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda

José L Fernández-Alemán

Juan Manuel Carrillo-de-Gea

Joaquín Nicolás  
08/01/2026

**PUID:** [SECM-CAT-ISU-035]

**Requirement description:** The application must be free of adware and known malware (via virus-checking against some established library of viruses), and should prevent the execution of malicious code, as well as disallow the uploading of malicious or dangerous-looking scripts or files.

**Source:**

- ✓ SRG-APP-000516-MAPP-000077: The mobile app source code must not contain adware or known malware [17].
- ✓ CIS Critical Security Control 10: Malware Defenses: Prevent or control the installation, spread, and execution of malicious applications, code, or scripts on enterprise assets [14].
- ✓ Test ID 9: The EHR application will not permit malicious code to be uploaded [37]

**Priority:** Not described

**Rationale:** This will help keep your healthcare applications safe from adware, malware, and malicious code that could sabotage your productivity. The application protects sensitive information and allows for compliance with healthcare security standards by preventing insertion and execution of malicious scripts, proactively detecting and removing harmful components from the source code.

**Number of Children:** 0**Number of parents:** 0**Cycles:** 0**Number Audit:** 0**Child PUIDs:** Not described**Parent PUIDs:** Not described**Exclusion PUIDs:** Not described**Importance:** Not described**Current state:** To be determined**Verification method:** Demonstration/Analysis**Validation criteria:****1. Source Code Malware Scan:**

- ✓ Conduct a complete static analysis of the application's source code to ensure it is adware & known malware-free.
- ✓ Ensure that the scan results show zero malicious elements found.

**2. Prevention of Malicious Code Execution:**

- ✓ Validate the application does not allow the execution of malicious code or scripts via testing.
- ✓ Attempt to inject and execute a malicious script and ensure that the system prevents this action.

**3. Malicious File Upload Testing:**

- ✓ Mock the upload of malicious scripts/files to the application and check whether they are being denied.

<ul style="list-style-type: none"> <li>✓ Ensure that the application raises alerts or logs when someone attempts to upload unauthorized files.</li> </ul> <p><b>4. Malware Defense Review:</b></p> <ul style="list-style-type: none"> <li>✓ Implement detection mechanisms for spying or malware spread/execution in carried out in the application.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-ISU-036]
<b>Requirement description:</b> The application should be built to run in untrusted environments and should only retrieve the absolute minimum amount of information needed to function—limiting the potential for data leakage or corruption.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ Execution in untrusted environments: 1. Applications should be designed to run in untrusted environments, obtaining only the minimum necessary information for functionality, reducing the risk of data leakage or tampering [3].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> This ensures that healthcare applications are robust when they run in possibly untrusted environments (both on personal devices and on public networks). Reducing the amount of data collected to only what is necessary to make the application function limits the attack surface and minimizes risks related to data exposure, manipulation, or unauthorized access. This method improves patient data privacy, as well as the security posture of the application.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Non-Usage of Minimal Data Collection Validation:</b> <ul style="list-style-type: none"> <li>✓ Ensure that the app only gathers the necessary data to operate.</li> <li>✓ Ensure that you are not requested for unnecessary data, and remove the respective part as well.</li> </ul> </li> <li><b>2. Training in an Untrusted Environment:</b></li> </ol>

<ul style="list-style-type: none"> <li>✓ Perform security testing of the application in the untrusted environment, e.g., unsecured network, and ensure it works securely without leaking sensitive data.</li> </ul>
<b>3. Data Leakage Testing:</b>
<ul style="list-style-type: none"> <li>✓ When the application sends data—verify that no unnecessary or excessive information is being sent.</li> <li>✓ Intercept the data and ensure no sensitive data got leaked.</li> </ul>
<b>4. Tampering Resistance Test:</b>
<ul style="list-style-type: none"> <li>✓ Use an untrusted execution environment to simulate tampering attempts and validate detection and mitigation of attack attempts on the application or its data.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda
José L Fernández-Alemán
Juan Manuel Carrillo-de-Gea
Joaquín Nicolás
08/01/2026

<b>PUID:</b> [SECM-CAT-ISU-037]
<b>Requirement description:</b> The application needs to implement protections against log injection by clients so that event histories in the backend system cannot be corrupted or forged.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ 5.6. Protect the back-end from client-initiated log injections that may corrupt or forge the history of events [16].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> This ensures the backend system that handles event logs in case a client tries to inject logs they are not supposed to inject. An example of this can be in a healthcare application where accurate event histories are needed for compliance, auditing, and operational analysis. These records may be corrupted, and it is an attacker opportunity as can be used to obfuscate attack the logs. The application protects the backend from such vulnerabilities by enforcing protections.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Testing for Log Injection Detection:</b></li> </ol>
<ul style="list-style-type: none"> <li>✓ Perform client-side log injection and assert the application should detect and prevent such requests.</li> </ul>

- ✓ Ensure that backend is not receiving any invalid or malicious log entries.
- 2. Log Validation Mechanism:**
- ✓ Confirm that the backend has mechanisms to verify incoming log entries for authenticity and integrity.
  - ✓ Detect and alert tampered or forged logs.
- 3. Backend Log Integrity Review:**
- ✓ Ensure event logs on the backend system after different client interactions are working as expected.
  - ✓ Ensure that legacy logs are protected from being tampered by unauthorized users.
- 4. Code Review for Log Handling:**
- ✓ Review the application's logging code to ensure that input is sanitized before logging.
  - ✓ Ensure logging operations comply with secure coding standards.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda

José L Fernández-Alemán

Juan Manuel Carrillo-de-Gea

Joaquín Nicolás

08/01/2026

**PUID:** [SECM-CAT-ISU-038]

**Requirement description:** The project must avoid exposing signing credentials in the source code and use external mechanisms (e.g., environment variables) to supply keystore secrets during the build process, so that the signing flow can be integrated with secure key management systems (such as CI/CD pipelines, vaults, or HSMs).

**Source:**

- ✓ 9.4. If an enterprise app store is used, protect the application signing key with the utmost care (e.g., use an HSM, air-gapped machine, etc.). [16].
- ✓ Follow Secure Key Management Practices: 1. Employ secure key management techniques, such as using key vaults or hardware security modules (HSMs) to securely store encryption keys. Protect keys from unauthorized access, including restricting access to authorized personnel, encrypting keys at rest, and using secure key distribution mechanisms [3].

**Priority:** Not described

**Rationale:** The application signing key is a sensitive asset that needs to be protected from being stolen or misused. With signing keys, you want to keep them protected using secure management techniques (e.g., HSMs, air-gapped systems, encrypted storage) so that the risk of compromise is minimized. The security of signing keys can significantly affect the integrity and protection of sensitive data in healthcare applications where there is a great deal of capacity and quality for trust.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Key Storage Validation:</b> <ul style="list-style-type: none"> <li>✓ Check that application signing keys are stored securely in hardware security modules (HSMs), air-gapped systems, or encrypted key vaults.</li> <li>✓ Ensure stored keys are accessible only to authorized personnel.</li> </ul> </li> <li><b>2. Access Control Testing:</b> <ul style="list-style-type: none"> <li>✓ Test that application signing keys cannot be accessed on the system by unauthorized personnel.</li> <li>✓ Ensure that all access is logged and monitored for any anomalies.</li> </ul> </li> <li><b>3. Key Distribution Security:</b> <ul style="list-style-type: none"> <li>✓ Make sure that the signing keys are sent through encrypted and authenticated channels.</li> <li>✓ Verify that the processes used for distribution guard against interception or unauthorized access.</li> </ul> </li> <li><b>4. Key Integrity Check:</b> <ul style="list-style-type: none"> <li>✓ Store signing keys in an encrypted form at rest, and provide integrity checks to detect tampering or compromise.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-ISU-039]
<b>Requirement description:</b> Implement mechanisms in the application to detect potentially risky or fraudulent interactions, including interactions coming from tampered versions of the application or untrustworthy environments.
<b>Source:</b> <ul style="list-style-type: none"> <li>✓ Detection of Risky Interactions: The system must detect potentially risky or fraudulent interactions, including those from tampered app versions or untrustworthy environments [18].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> This is required to help all healthcare applications be used in such a manner that they can find ways to identify and mitigate the interaction so that it doesn't harm the data integrity of the system. The application protects sensitive patient information and maintains operational trust

by detecting risky behaviors like interactions from tampering presence on app versions and environments that are not trusted. These mechanisms play a vital role in reducing the effects of fraud and preserving compliance with security regulations.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

#### Validation criteria:

##### 1. Testing of Risky Interaction Detection

- ✓ Simulate activities originating from manipulated app versions and verify that the system recognizes and identifies it as hazardous.
- ✓ Analyze the detection of interactions of untrusted environments like rooted/jailbroken devices.

##### 2. Fraudulent Activity Logging:

- ✓ Have adequate logging in place for all detected risky or fraudulent interactions, so they can be analyzed.
- ✓ Ensure logs are stored on a secured storage medium and only accessible to authorized personnel.

##### 3. System Response Validation:

- ✓ Validate that the application processes appropriately (e.g., blocks, alerts, or quarantines) at the point of a detected risky or fraudulent interaction.
- ✓ Verify that response mechanisms are adjustable based on risk levels.

##### 4. Review of Tampered App Detection

- ✓ Ensure that the application contains mechanisms that can attest to its integrity and identify alterations to its code or conduct.
- ✓ Require that users re-authenticate after a certain amount of time.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

#### Author and date:

Carlos M. Mejía-Granda

José L Fernández-Alemán

Juan Manuel Carrillo-de-Gea

Joaquín Nicolás

08/01/2026

**PUID:** [SECM-CAT-ISU-040]

**Requirement description:** The application shall implement procedures to notify the user upon the application being deprecated.

**Source:**

✓ V-222660: Procedures must be in place to notify users when an application is decommissioned. Create and establish procedures to notify users when an application is decommissioned [15].
<b>Priority:</b> Not described
<b>Rationale:</b> This proposal would add a new P&E section that specifies that applications should inform users if and when they are decommissioned in a timely manner to avert confusion and/or potential security risks. The anticipation of service termination leads to improved communication with users, allowing them to plan for the transition to supported applications or alternative solutions, thereby minimizing workflow disruption and enabling critical healthcare and other services to continue uninterrupted, especially when timely and reliable application functionality plays a vital role.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b> <ol style="list-style-type: none"> <li><b>Verification of the notification procedures:</b> <ul style="list-style-type: none"> <li>✓ Check there is a documented procedure in place to notify users of application decommissioning.</li> <li>✓ Adopt a formal approach with specific time frames and communication methods (in-app message, email, notification, etc.)</li> </ul> </li> <li><b>User Communication Testing:</b> <ul style="list-style-type: none"> <li>✓ Simulate a decommissioning scenario and verify users are notified in an appropriate and timely manner.</li> <li>✓ Confirm that the communication contains details regarding the timeline, rationale, and alternative solutions or applications for the decommissioning.</li> </ul> </li> <li><b>User Consent Validation:</b> <ul style="list-style-type: none"> <li>✓ Ensure that the system records user acknowledgment of decommissioning notifications.</li> <li>✓ Provide next steps or where users can get further assistance.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-ISU-041]
<b>Requirement description:</b> The application must automatically create a notification of detected integrity violations, send it to specialized services to find unauthorized versions of the distribution platforms and remove such versions.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ Detection Automatic violation notifications: 1. Detected integrity violations must be automatically reported to specialized services that can identify and remove unauthorized versions of the app from distribution platforms [3].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> This requirement ensures that the healthcare application can respond to integrity violations in a timely manner with automated identification and reporting. These versions are not official and can lead to dangerous outcomes such as exposing sensitive data or installing malware. This helps maintain the integrity and legitimacy of the application on distribution platforms by reporting these violations to relevant services.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Automated Testing of Notifications:</b> <ul style="list-style-type: none"> <li>✓ Check whether the program detects violations and sends them automatically to specialized services. The notifications should be specific enough to help identify and remove the offending app version.</li> </ul> </li> <li><b>2. Response Validation:</b> <ul style="list-style-type: none"> <li>✓ Ensure the specialized services receiving these notifications take action to identify and remove unauthorized app versions in distribution platforms.</li> <li>✓ Validate that the app and the services have timely and accurate communication.</li> </ul> </li> <li><b>3. Log Review for Notifications:</b> <ul style="list-style-type: none"> <li>✓ Create a log of any integrity violation notifications, including the breach detected, when the notification was made, and which service it was contacted with.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda
José L Fernández-Alemán
Juan Manuel Carrillo-de-Gea
Joaquín Nicolás
08/01/2026

<b>PUID:</b> [SECM-CAT-ISU-042]
<b>Requirement description:</b> The application shall enter into a mode of operation that is considered to be a safe state upon detection of conditions that cause the application to enter into a safe state per information defined in CCI-0002856 (CP-12), per the guidance in CCI-0002857 (CP-12).
<b>Source:</b>
✓ SRG-APP-000388-MAPP-000100: The mobile app, when conditions defined in CCI-0002856, CP-12 are detected, must enter a safe mode of operation defined in CCI-0002857, CP-12 [17].
<b>Priority:</b> Not described
<b>Rationale:</b> This is needed for the application to securely react to defined situations/incidents (for example, a security violation) and switch to a secure operational state. It only allows the application to run the minimum required processes, while keeping sensitive data safe and reducing risk potential. The integrated solution is critical for healthcare technology, which needs to ensure rapid, secure responses to operate while complying with the strictest patient safety and data privacy standards.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Incident Response Validation:</b> <ul style="list-style-type: none"> <li>✓ Ensure the application records the incident and continues to safe mode without affecting essential operations.</li> <li>✓ Ensure that users and admin are informed when safe mode is enabled.</li> </ul> </li> <li><b>2. Reset and Recovery Testing:</b> <ul style="list-style-type: none"> <li>✓ Ensure the app exits safe mode cleanly when the incident conditions are cleared.</li> <li>✓ Confirm that normal operations have been restored without degradation of security or data integrity.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-ISU-043]
---------------------------------

**Requirement description:** The application must (a) disable all of the functionalities of the application to the extent possible (b) use only approved ports, protocols, and services (c) ensure that any temporary compiled data cannot be compromised.

**Source:**

- ✓ 10.5. Strip unused functionalities from interpreters [16]
- ✓ CM-7 LEAST FUNCTIONALITY

Control:

a. Configure the system to provide only [Assignment: organization-defined mission essential capabilities]; and

b. Prohibit or restrict the use of the following functions, ports, protocols, software, and/or services: [Assignment: organization-defined prohibited or restricted functions, system ports, protocols, software, and/or services] [11].

- ✓ Compiled data should be temporary and be eliminated after the app objective is completed [25].
- ✓ V-222518: The application must be configured to disable non-essential capabilities. Disable application extraneous application functionality that is not required in order to fulfill the application's mission [15].
- ✓ V-222519: The application must be configured to use only functions, ports, and protocols permitted to it in the PPSM CAL. Configure the application to utilize application ports approved by the PPSM CAL [15].

**Priority:** Not described

**Rationale:** This requirement enforces the principle of least functionality, requiring the application to restrict operations to the capabilities that are necessary for its mission while minimizing the attack surface. The application also minimizes vulnerabilities by disabling unused functionalities and limiting unauthorized protocol connections and ports. Besides, the removal of temporarily prepared data eliminates the risk of its abuse and provides a secure data processing approach — a huge demand in healthcare solutions.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

1. **Validation of configuration and functionality:**

- ✓ Make sure the application is configured to activate only the necessary features that need to be functional in order for the application to run.
- ✓ Analyze system components (unused/unnecessary options must be disabled as well as interpreter features)

<p><b>2. Ports and Protocols Testing:</b></p> <ul style="list-style-type: none"> <li>✓ Verify that the application only utilizes the ports, protocols, and services that were approved in the PPSM CAL</li> <li>✓ Invoke unauthorized port or protocol usage and validate block.</li> </ul> <p><b>3. Step 1: Temporary Data Management Review</b></p> <ul style="list-style-type: none"> <li>✓ Ensure that the compiled data generated during the application utilization is temporary; it should be removed as soon as the application objective is met.</li> <li>✓ Ensure that no residual or sensitive data is accessible when the application is no longer needed.</li> </ul> <p><b>4. Configuration related sample check:</b></p> <ul style="list-style-type: none"> <li>✓ Ensure that system configuration follows the organization's least functionality requirements, disabling unnecessary features on the system and aligning with security requirements</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<p><b>Author and date:</b></p> <p>Carlos M. Mejía-Granda  José L Fernández-Alemán  Juan Manuel Carrillo-de-Gea  Joaquín Nicolás  08/01/2026</p>

<b>PUID:</b> [SECM-CAT-ISU-044]
<b>Requirement description:</b> Disable JavaScript for WebViews unless it is explicitly required for the functionality of the application and configure WebViews securely to avoid vulnerabilities such as XSS or sensitive data portion leakage.
<p><b>Source:</b></p> <ul style="list-style-type: none"> <li>✓ 6.5: Verify that JavaScript is disabled in WebViews unless explicitly required [19].</li> <li>✓ JavaScript Usage in WebView: The application must not enable JavaScript in WebView by calling setJavaScriptEnabled() unless it is absolutely necessary for the app's functionality. If JavaScript is not required, do not use setJavaScriptEnabled() to prevent potential cross-site scripting (XSS) vulnerabilities [18].</li> <li>✓ MASVS-PLATFORM-2: The app performs identity pinning for all remote endpoints under the developer's control: WebViews are typically used by apps that have a need for increased control over the UI. This control ensures that WebViews are configured securely to prevent sensitive data leakage as well as sensitive functionality exposure (e.g. via JavaScript bridges to native code) [20].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> This ensures that JavaScript executed in a WebView used in a healthcare application cannot perform actions that violate HIPAA or PHI exposure rules. To use JavaScript or not; the answer is clear: disable JavaScript by default and only use it when the need justifies it for the

functionality of your application. The use of WebViews with a secure configuration also provides additional mitigation against risks associated with unauthorized access to or exposure to sensitive functionality.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. How JavaScript Configuration Testing Works:</b> <ul style="list-style-type: none"> <li>✓ Ensure that JavaScript is disabled in all WebViews by default.</li> <li>✓ Make sure the setJavaScriptEnabled() method is invoked only when needed and documented.</li> </ul> </li> <li><b>2. To Test Cross-Site Scripting (XSS):</b> <ul style="list-style-type: none"> <li>✓ Perform XSS vulnerabilities test in WebViews and ensure the app is able to mitigate these attacks.</li> </ul> </li> <li><b>3. Code Review for WebView Use:</b> <ul style="list-style-type: none"> <li>✓ Reviewing Application Code for WebView instances and verify if it follows secure configuration practices</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-ISU-045]
<b>Requirement description:</b> Extreme caution must be used when using addJavaScriptInterface() and it must be limited to trusted web content embedded in the APK or internal sources, and WebViews must be configured correctly so that sensitive data does not leak and sensitive functionality is not exposed.
<b>Source:</b> <ul style="list-style-type: none"> <li>✓ Use of JavaScript to Interface Methods: The application must use addJavaScriptInterface() with extreme caution and only expose it to trusted web content. JavaScript interfaces must only be exposed to content embedded within the application's APK or trusted internal sources. If untrusted input is allowed, do not use addJavaScriptInterface(), as this could allow malicious JavaScript to invoke Android methods within the app [18].</li> </ul>

✓ MASVS-PLATFORM-2: The app performs identity pinning for all remote endpoints under the developer's control: WebViews are typically used by apps that have a need for increased control over the UI. This control ensures that WebViews are configured securely to prevent sensitive data leakage as well as sensitive functionality exposure (e.g. via JavaScript bridges to native code) [20].
<b>Priority:</b> Not described
<b>Rationale:</b> Most of the risks of exposing JavaScript interfaces to untrusted content, such as the unauthorized invocation of Android methods or leaking sensitive data. Limiting addJavaScriptInterface() usage to trusted sources and securing WebViews protects the application's attack surface and sensitive health information from being abused.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li>1. <b>Testing for JavaScript Interface Exposure:</b> <ul style="list-style-type: none"> <li>✓ Ensure that addJavaScriptInterface() is limited to content from the APK itself or trusted internal sources.</li> <li>✓ Ensure that JavaScript interfaces are not exposed to untrusted input.</li> </ul> </li> <li>2. <b>Validate the configuration of the secure WebView:</b> <ul style="list-style-type: none"> <li>✓ Prevents Data Leakage and Exposure of Functionality: Make sure that WebViews are set up to limit whether JavaScript bridges are only exposed to trusted hosts.</li> </ul> </li> <li>3. <b>Use Code Review to check usages of an interface.</b> <ul style="list-style-type: none"> <li>✓ Examine the codebase to confirm that addJavaScriptInterface() is used securely and is consistent with the application uses trusted sources</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-ISU-046]
<b>Requirement description:</b> The WebView must restrict JavaScript execution to only JS within the app package, and properly configured WebViews must not leak sensitive data or functionality.
<b>Source:</b>

<ul style="list-style-type: none"> <li>✓ 6.8: Verify that if Java objects are exposed in a WebView, verify that the WebView only renders JavaScript contained within the app package [19].</li> <li>✓ MASVS-PLATFORM-2: The app performs identity pinning for all remote endpoints under the developer's control: WebViews are typically used by apps that have a need for increased control over the UI. This control ensures that WebViews are configured securely to prevent sensitive data leakage as well as sensitive functionality exposure (e.g. via JavaScript bridges to native code) [20].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> This requirement ensures incorrect, untrusted Java objects are not exposed to the WebView; only trusted, app-packaged JavaScript is accessible. This reduces the risk of data leakage, unauthorized access to sensitive application functionality, and similar concerns. The application both preserves healthcare data integrity and controls operational security by enforcing secure WebView configurations and JavaScript sources.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b> <ol style="list-style-type: none"> <li>1. <b>These are as follows: Java Object Exposure Validation:</b> <ul style="list-style-type: none"> <li>✓ Ensure that any Java objects exposed in WebViews are only callable by the JavaScript code within the application package.</li> <li>✓ Ensure that untrusted JavaScript cannot access hooked Java objects.</li> </ul> </li> <li>2. <b>Testing Configuration of Secure WebView</b> <ul style="list-style-type: none"> <li>✓ Ensure WebViews are configured to mitigate sensitive data leakage and unauthorized access to app functionality.</li> <li>✓ Do not create any inadvertent JS bridges between your native code and your WebViews.</li> </ul> </li> <li>3. <b>Identity Pinning Testing:</b> <ul style="list-style-type: none"> <li>✓ Ensure the application is pinning identities in communication channels to all remote endpoints accessed through WebViews.</li> </ul> </li> <li>4. <b>Using JavaScript and WebView — Code Review</b> <ul style="list-style-type: none"> <li>✓ Perform a code review to validate the correct application of secure WebView configurations and limitations on JavaScript sources.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás

08/01/2026

<b>PUID:</b> [SECM-CAT-ISU-047]
<b>Requirement description:</b> The application shall limit access to domains that are not compliant within any third-party components used for embedded browsing that utilize a user-supplied agent (for example, WebViews) as well as disable any platform-supported features that are not used to reduce security risks.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ 12.1. In the case that the application includes embedded web browsing capabilities (e.g., WebViews), restrict access to third party domains that do not comply with the required security standards, disable any unused platform supported functionalities, such as the plugins, local file accessibility, local content provider (content URL) accessibility and the dynamic code (e.g., JavaScript) execution support. Furthermore, avoid using full screen web interfaces since these can be abused from attackers to create fake application screens [16]</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> This rule addresses the proper configuration of embedded web browsing components, such as WebViews, to prevent unauthorized domain access and potential wasting of unused features, as well as phishing attacks that might occur through the display of a screen similar to an application. The allomorph is using controls for access to secure domains, disabling unnecessary features, and avoiding full-screen interfaces that mitigate its vulnerability to attacks while providing a secure environment for the unexamined data.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Unused Functionality Review:</b> <ul style="list-style-type: none"> <li>✓ Ensure that platform-supported unused functionalities, such as plugins, local files access, content provider access, and JavaScript execution features, are all disabled in WebViews.</li> </ul> </li> <li><b>2. Validation of Interface in Full-Screen:</b> <ul style="list-style-type: none"> <li>✓ Verify the application does not expose full-screen web interfaces in embedded browsing components.</li> <li>✓ Assess risks associated with potential phishing attacks through full-screen interfaces and verify that the application is not vulnerable to the same.</li> </ul> </li> <li><b>3. Code Review: Proper WebView Security Configuration</b> <ul style="list-style-type: none"> <li>✓ Go through the codebase to check that said secure WebView configurations, such as access restrictions, disabled functionalities, and no full-screen interfaces are observed.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization

<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-ISU-048]
<b>Requirement description:</b> The application should manage webview caching. It should remove local cache using clearCache() after handling sensitive data and should use server-side headers like no-store to avoid caching sensitive content.
<b>Source:</b>
✓ Cache Management: The application must clear local WebView caches using clearCache() if sensitive data is processed through WebView. Server-side headers such as no-store must be used to ensure that sensitive content is not cached by the WebView [18].
<b>Priority:</b> Not described
<b>Rationale:</b> By mandating this setting, we make sure that sensitive healthcare-related data that will be processed through Webviews is neither locally nor server-side cached, thus minimizing the potential for data breaches or leakage. Query caching also allows for patient confidentiality and ensures secure coding adherence for any applications dealing with sensitive information.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Cache Clearing Validation:</b> <ul style="list-style-type: none"> <li>✓ Verify that the application uses clearCache() to clear all the WebView caches after sensitive data is processed</li> <li>✓ Mock data through WebView and make sure that the cached information is invalidated as soon as possible</li> </ul> </li> <li><b>2. Server-Side Header Testing:</b> <ul style="list-style-type: none"> <li>✓ Ensure no sensitive data gets cached into the WebView by ensuring the server-side headers like no-store, no-cache, etc.</li> <li>✓ Verify that the headers work as intended by testing the content delivery.</li> </ul> </li> <li><b>3. Check this code for Cache Management:</b> <ul style="list-style-type: none"> <li>✓ Review the application's use of WebView and make sure that clearCache() is called after processing sensitive content.</li> </ul> </li> </ol>

<ul style="list-style-type: none"> <li>✓ Ensure server-side headers are appropriately set to restrict sensitive data caching</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-ISU-049]
<b>Requirement description:</b> Display only trusted content in WebView (on older Android versions) to mitigate Man-In-The-Middle web attacks, Updatable Security Provider (for SSL vulnerabilities).
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ WebView Version and Security Patches: For devices running Android versions older than 4.4 (API level 19), the application must ensure that WebView only displays trusted content to avoid known security issues in the older webkit version. The application must use the updatable security Provider object to mitigate potential SSL vulnerabilities by keeping the security provider up to date [18].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> To prevent Man-In-The-Middle web attacks, the WebView application must restrict its display to only trusted content on devices running versions of Android prior to 4.4 (API level 19). The application must be using the updatable Security Provider object to mitigate specific vulnerabilities in SSL by keeping the security provider up to date. This requirement mitigates the potential security risks of older WebView versions, offering to ensure that applications operating on older devices are not compromised. These measures limit exposure by only hosting trusted content inside the WebView while keeping the security provider up to date whenever applicable, helping secure against SSL vulnerabilities and outdated webkit implementations, allowing sensitive healthcare data to remain safe in the application itself
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Testing for WebView Content Restrictions:</b> <ul style="list-style-type: none"> <li>✓ Ensure that WebView on pre-4.4 Android versions (API level 19) is only allowed to render trusted content.</li> <li>✓ Attempt to load untrusted content and ensure it is blocked from being loaded.</li> </ul> </li> <li><b>2. Security Provider Validation:</b></li> </ol>

- ✓ Make sure the application makes use of the updatable Security Provider object to mitigate any SSL vulnerabilities.
- ✓ Check and apply updates to the security provider.

### 3. Staging for Legacy Devices Compatibility Testing

- ✓ Verify WebView security configurations for Android version lower than 4.4.
- ✓ Verify that exposure to known vulnerabilities in legacy webkit versions is mitigated by application-specific safeguards.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda  
 José L Fernández-Alemán  
 Juan Manuel Carrillo-de-Gea  
 Joaquín Nicolás  
 08/01/2026

**PUID:** [SECM-CAT-ISU-050]

**Requirement description:** The application must make use of WebViews to enable only the minimum set of protocol handlers, ideally limited to HTTPS, and disable potentially dangerous handlers such as file, tel, and app-id.

**Source:**

- ✓ 6.6: Verify that WebViews are configured to allow only the minimum set of protocol handlers required (ideally, only https). Potentially dangerous handlers, such as file, tel and app-id, are disabled [19].

**Priority:** Not described

**Rationale:** Links for applications should be configured in such a way that only the protocol handlers needed for their functionality are registered for WebView. Disabling potentially dangerous handlers minimizes the risk of unauthorized access, leakage of sensitive data, or exploitation of system resources, enhancing the security posture of healthcare applications.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

#### 1. Testing Protocol Handler Configuration

- ✓ Ensure that no HTTP protocol handler is allowed for the WebView.
- ✓ Verify that potentially harmful handlers like file, tel, app-id, etc. are disabled.

#### 2. Unauthorized Protocol Testing:

- ✓ Attempt to invoke unauthorized or disabled protocol handlers and check that the event is blocked.

### 3. Secure Data Transmission Validation

- ✓ Ensure the WebView only allows secure communication channels (e.g., HTTPS) and deny non-secure connections.

### 4. Setting up protocol handler (via code review)

- ✓ Verify the WebView configuration of the application to ensure that only the minimal needed protocol handlers are configured.
- ✓ Confirm that potentially dangerous protocol handlers are not present in the codebase.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda  
José L Fernández-Alemán  
Juan Manuel Carrillo-de-Gea  
Joaquín Nicolás  
08/01/2026

**PUID:** [SECM-CAT-ISU-051]

**Requirement description:** The application must ensure that WebViews do not load user-supplied local resources to prevent unauthorized access or malicious content execution.

**Source:**

- ✓ 6.7: Verify that the app does not load user-supplied local resources into WebViews. [19].

**Priority:** Not described

**Rationale:** This requirement mitigates the risk of unauthorized access, malicious code execution, or data leakage by restricting WebViews from loading user-supplied local resources. Ensuring that only trusted and verified resources are loaded into WebViews helps protect the integrity of healthcare applications and the sensitive data they handle.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

#### 1. Local Resource Loading Testing:

- ✓ Verify that the WebView configuration prevents the loading of user-supplied local resources.

- ✓ Simulate attempts to load unauthorized local resources and confirm they are blocked.

## **2. Content Source Restriction Validation:**

- ✓ Confirm that WebViews are configured to load only predefined and trusted resources.
- ✓ Validate that the application restricts resource loading to prevent execution of user-supplied content.

## **3. Code Review for Resource Loading Logic:**

- ✓ Review the application's WebView implementation to ensure compliance with the restriction on loading user-supplied local resources.
- ✓ Check for potential vulnerabilities that may allow circumvention of this restriction.

## **4. Secure Resource Handling Testing:**

- ✓ Test WebView behavior under various scenarios to ensure secure handling of resources, including attempts to inject local files.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda

José L Fernández-Alemán

Juan Manuel Carrillo-de-Gea

Joaquín Nicolás

08/01/2026

**PUID:** [SECM-CAT-ISU-052]

**Requirement description:** The application must validate all input data in WebViews used for network content, ensuring that insecure protocols are not trusted and all incoming data is sanitized.

**Source:**

- ✓ WebView Data Handling: Applications using WebView for network content must validate all input data, ensuring that insecure protocols are not trusted and all incoming data is sanitized [18].

**Priority:** Not described

**Rationale:** This requirement ensures that WebView configurations in healthcare applications mitigate risks from untrusted protocols and unsanitized input. Proper validation and sanitization of data protect against injection attacks, data leakage, and unauthorized access, ensuring secure communication and preserving the integrity of sensitive data processed by the application.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>

**1. Input Data Validation Testing:**

- ✓ Verify that all input data received by WebView is validated for format, type, and content.
- ✓ Test for injection vulnerabilities by attempting to supply malformed or malicious input data.

**2. Protocol Trust Restriction:**

- ✓ Confirm that WebView does not trust insecure protocols, such as HTTP, by default.
- ✓ Test for proper rejection of insecure protocol-based content.

**3. Code Review for Data Handling Logic:**

- ✓ Conduct a review of the WebView implementation to ensure robust input validation and data sanitization are in place.
- ✓ Validate adherence to secure coding practices for data handling.

<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-ISU-053]
<b>Requirement description:</b> The application must implement object serialization using safe serialization APIs to prevent security vulnerabilities such as unauthorized data access or deserialization attacks.
<b>Source:</b> ✓ 6.9: Verify that object serialization, if any, is implemented using safe serialization APIs [19].
<b>Priority:</b> Not described

**Rationale:** This requirement ensures that object serialization in the application is performed securely, mitigating risks associated with unsafe serialization practices, such as deserialization attacks, data corruption, or unauthorized access. Using safe serialization APIs enhances data integrity and protects against the potential exploitation of serialized objects in healthcare applications.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

**1. Serialization API Validation:**

- ✓ Verify that the application uses safe and approved serialization APIs for object serialization.
- ✓ Confirm that unsafe or deprecated serialization methods are not utilized in the codebase.

**2. Code Review for Serialization Practices:**

- ✓ Review the application's codebase to ensure serialization and deserialization processes follow secure coding standards.
- ✓ Check for proper validation of serialized data during deserialization.

**3. Data Integrity Validation:**

- ✓ Confirm that the serialization process maintains the integrity and confidentiality of sensitive data.
- ✓ Validate that only authorized entities can serialize or deserialize objects.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda

José L Fernández-Alemán

Juan Manuel Carrillo-de-Gea

Joaquín Nicolás

08/01/2026

**PUID:** [SECM-CAT-ISU-054]

**Requirement description:** The application must securely implement object deserialization by validating untrusted data before deserialization and protecting against unauthorized gadget chain execution.

**Source:**

- ✓ 6.9: CWE-502: Deserialization of Untrusted Data:

The product deserializes untrusted data without sufficiently verifying that the resulting data will be valid.

It is often convenient to serialize objects for communication or to save them for later use. However, deserialized data or code can often be modified without using the provided accessor functions if it does not use cryptography to protect itself. Furthermore, any cryptography would still be client-side security -- which is a dangerous security assumption.

Data that is untrusted cannot be trusted to be well-formed.

When developers place no restrictions on "gadget chains," or series of instances and method invocations that can self-execute during the deserialization process (i.e., before the object is returned to the caller), it is sometimes possible for attackers to leverage them to perform unauthorized actions, like generating a shell [19].

**Priority:** Not described

**Rationale:** This requirement mitigates risks associated with the deserialization of untrusted data, such as unauthorized actions, code execution, or data corruption. By validating data prior to deserialization and preventing gadget chain execution, the application ensures robust protection against vulnerabilities that could compromise the security of healthcare data and systems.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

**1. Data Validation Testing:**

- ✓ Verify that all untrusted data is validated before deserialization to ensure it is well-formed and secure.
- ✓ Test the application for improper deserialization of invalid or malicious data inputs.

**2. Code Review for Deserialization Logic:**

- ✓ Review the deserialization implementation to ensure secure handling of untrusted data, including validation and cryptographic protections where applicable.
- ✓ Check for adherence to secure coding standards for deserialization.

<b>3. Runtime Behavior Validation:</b>
✓ Monitor the application's runtime behavior during deserialization to detect and prevent unauthorized actions or errors resulting from untrusted data.
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-ISU-055]
<b>Requirement description:</b> The application must detect and mitigate interactions from untrustworthy environments, such as rooted or jailbroken devices, and ensure operation only on secure, verified Android-powered devices.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ Mitigation of Untrustworthy Environments: The system must be able to detect and mitigate interactions from untrustworthy environments, ensuring that the app only operates in secure, verified Android-powered devices [18].</li> <li>✓ 6.10: Verify that the app detects whether it is being executed on a rooted or jailbroken device. Depending on the business requirement, users are warned, or the app is terminated if the device is rooted or jailbroken [19].</li> <li>✓ MASVS-RESILIENCE-1: The app validates the integrity of the platform: Running on a platform that has been tampered with can be very dangerous for apps, as this may disable certain security features, putting the data of the app at risk. Trusting the platform is essential for many of the MASVS controls relying on the platform being secure (e.g. secure storage, biometrics, sandboxing, etc.). This control tries to validate that the OS has not been compromised and its security features can thus be trusted [20].</li> <li>✓ 4.13. In the case of rooted or jailbroken devices, consider to integrate a custom or third-party secure container for the transmission channel, since the platform security controls that establish the TLS connection cannot be trusted [16].</li> </ul>

<b>Priority:</b> Not described
<b>Rationale:</b> This requirement ensures that healthcare applications operate in secure environments by detecting tampered devices, such as rooted or jailbroken devices. Operating on compromised platforms may disable critical security features, increasing risks of data breaches and unauthorized access. Mitigating these risks through device validation and secure containment enhances overall application resilience and data protection.
<b>Number of Children:</b> 0

<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<b>1. Rooted/Jailbroken Device Detection:</b>
<ul style="list-style-type: none"><li>✓ Verify that the application detects rooted or jailbroken devices during startup or execution.</li><li>✓ Confirm that warnings are displayed or that the application terminates operations on compromised devices based on business requirements.</li></ul>
<b>2. Secure Container Implementation:</b>
<ul style="list-style-type: none"><li>✓ Validate that a custom or third-party secure container is integrated for communication channels in scenarios where platform security cannot be trusted.</li></ul>
<b>3. Code Review for Environment Detection Logic:</b>
<ul style="list-style-type: none"><li>✓ Review the implementation of device integrity checks to ensure robust detection of untrustworthy environments.</li><li>✓ Confirm adherence to secure coding practices and validation standards.</li></ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-ISU-056]
<b>Requirement description:</b> The application must request the <code>READ_SMS</code> permission only when absolutely necessary, minimize its usage, and securely handle SMS data to prevent misuse or exploitation by malicious third parties.
<b>Source:</b> <ul style="list-style-type: none"><li>✓ Permissions Handling: The application must request the <code>READ_SMS</code> permission only if absolutely necessary and must minimize its use to reduce the risk of exposure of SMS data</li></ul>

to other applications. If the app needs to process SMS messages, it must validate and securely handle them to prevent misuse or exploitation by malicious third parties [18].
<b>Priority:</b> Not described
<b>Rationale:</b> This requirement ensures that the application adheres to the principle of least privilege by requesting <code>READ_SMS</code> permission only when essential. Minimizing the use of this permission reduces the risk of SMS data being exposed to unauthorized applications or malicious actors. Secure handling of SMS messages protects sensitive user data and ensures compliance with privacy standards in healthcare applications.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<p><b>1. Permission Request Validation:</b></p> <ul style="list-style-type: none"> <li>✓ Verify that the <code>READ_SMS</code> permission is requested only if it is essential for the application's functionality.</li> <li>✓ Confirm that the application includes a clear justification for requesting this permission.</li> </ul> <p><b>2. Permission Minimization Validation:</b></p> <ul style="list-style-type: none"> <li>✓ Confirm that the application minimizes the use of the <code>READ_SMS</code> permission and restricts its use to specific, necessary operations.</li> </ul> <p><b>3. Code Review for Permission Handling:</b></p> <ul style="list-style-type: none"> <li>✓ Conduct a code review to ensure that the request for <code>READ_SMS</code> permission aligns with the application's functionality and adheres to secure coding practices.</li> </ul> <p><b>4. Security Logging Validation:</b></p> <ul style="list-style-type: none"> <li>✓ Verify that the application logs all instances of SMS data access securely and monitors for any unusual or unauthorized use of the <code>READ_SMS</code> permission.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán

Juan Manuel Carrillo-de-Gea  
Joaquín Nicolás  
08/01/2026

**PUID:** [SECM-CAT-ISU-057]

**Requirement description:** The application must ensure that API keys are excluded from version control systems and securely injected during the build process using configuration management tools.

**Source:**

- ✓ Secure API Key in Continuous Delivery Pipelines: Exclude API keys from version control systems like Git, and use configuration management tools to inject them during the build process [18].
- ✓ Exclude API Keys from Source Control: API keys must not be committed to source code repositories (e.g., Git) to avoid accidental exposure in public or shared repositories [18].

**Priority:** Not described

**Rationale:** This requirement ensures the secure handling of API keys in the development and deployment lifecycle, protecting them from accidental exposure in version control systems. By excluding API keys from source control and securely injecting them during the build process, the application minimizes the risk of unauthorized access to backend systems or APIs, preserving data confidentiality and integrity.

**Number of Children:** 0**Number of parents:** 0**Cycles:** 0**Number Audit:** 0**Child PUIDs:** Not described**Parent PUIDs:** Not described**Exclusion PUIDs:** Not described**Importance:** Not described**Current state:** To be determined**Verification method:** Demonstration/Analysis**Validation criteria:****1. Version Control Inspection:**

- ✓ Verify that API keys are not present in source code repositories by inspecting version control history.
- ✓ Confirm that repository scanning tools are in place to detect accidental API key commits.

**2. Secure Build Process Testing:**

- ✓ Validate that API keys are securely injected during the build process using configuration management tools.
- ✓ Test the build pipeline to ensure no hardcoded keys are included in the final application.

### 3. Configuration Management Review:

- ✓ Ensure that configuration management tools are properly configured to store and manage API keys securely.
- ✓ Verify that only authorized personnel can access the tools or keys.

### 4. Code Review for API Key Handling:

- ✓ Review the application's codebase to confirm that API keys are excluded and not hardcoded within the source code.
- ✓ Check for adherence to secure coding practices regarding sensitive data handling.

### 5. Key Rotation and Access Control Validation:

- ✓ Confirm that processes for API key rotation and access control are implemented and periodically tested to maintain security.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda

José L Fernández-Alemán

Juan Manuel Carrillo-de-Gea

Joaquín Nicolás

08/01/2026

**PUID:** [SECM-CAT-ISU-058]

**Requirement description:** The application must securely handle API keys during the continuous integration and delivery process using encrypted environment variables or secure vaults. It must also manage keys in the project with tools like Gradle's secrets-gradle-plugin for injection at build time.

**Source:**

- ✓ Secure API Key in Continuous Delivery Pipelines: When working with infrastructure teams, ensure API keys are securely handled during the continuous integration and delivery process, using encrypted environment variables or secure vaults [18].
- ✓ Exclude API Keys from Source Control: Use tools like Gradle's secrets-gradle-plugin to manage API keys securely in your project and inject them during build time [18].

**Priority:** Not described

**Rationale:** This requirement ensures the secure handling of API keys throughout the continuous integration and delivery process, reducing the risk of accidental exposure. By using tools like Gradle's secrets-gradle-plugin and secure storage solutions such as encrypted environment variables or secure vaults, the application safeguards sensitive keys from unauthorized access during development and deployment.

**Number of Children:** 0

<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Secure Handling Validation:</b> <ul style="list-style-type: none"> <li>✓ Verify that API keys are handled securely during the continuous integration and delivery process using encrypted environment variables or secure vaults.</li> <li>✓ Test the build pipeline to ensure no sensitive keys are exposed during runtime.</li> </ul> </li> <li><b>2. Gradle Plugin Implementation Testing:</b> <ul style="list-style-type: none"> <li>✓ Confirm that tools like Gradle's secrets-gradle-plugin are used for secure API key management and injection at build time.</li> <li>✓ Validate the proper configuration of the plugin for secure handling of keys.</li> </ul> </li> <li><b>3. Access Control and Storage Testing:</b> <ul style="list-style-type: none"> <li>✓ Ensure that API keys are stored in encrypted environments or vaults accessible only to authorized personnel or processes.</li> <li>✓ Test the access controls to prevent unauthorized retrieval or exposure of keys.</li> </ul> </li> <li><b>4. Code Review for Key Management:</b> <ul style="list-style-type: none"> <li>✓ Review the application's codebase and build configurations to confirm API keys are managed securely and not exposed in the source code.</li> </ul> </li> <li><b>5. Pipeline Audit:</b> <ul style="list-style-type: none"> <li>✓ Conduct an audit of the continuous integration and delivery pipeline to validate compliance with secure key management practices.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-ISU-059]
<b>Requirement description:</b> The application must include execution flow diagrams and design documents demonstrating how deadlock and recursion issues in web services are mitigated during development.
<b>Source:</b>

✓ V-222625: Execution flow diagrams and design documents must be created to show how deadlock and recursion issues in web services are being mitigated. Develop web services to account for deadlock issues [15].
<b>Priority:</b> Not described
<b>Rationale:</b> This requirement ensures that deadlock and recursion issues, which can disrupt the functionality and reliability of web services, are effectively identified and mitigated during the application design and development phases. Providing execution flow diagrams and design documentation improves transparency and accountability, enabling secure and efficient web service implementation in healthcare applications.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Execution Flow Diagram Validation:</b> <ul style="list-style-type: none"> <li>✓ Verify that execution flow diagrams are created and accurately depict how the application handles potential deadlock and recursion scenarios in web services.</li> </ul> </li> <li><b>2. Design Document Review:</b> <ul style="list-style-type: none"> <li>✓ Confirm that design documents include a detailed explanation of mitigation strategies for deadlock and recursion issues.</li> <li>✓ Validate that the documentation aligns with best practices for web service development.</li> </ul> </li> <li><b>3. Code Review for Mitigation Strategies:</b> <ul style="list-style-type: none"> <li>✓ Review the application code to confirm that deadlock and recursion mitigation strategies, such as resource locks and recursion limits, are implemented as described in the design documents.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

### 2.9.3.3 Insecure Authentication/Authorization

<b>PUID:</b> [SECM-CAT-IAA-001]
<b>Requirement description:</b> The application must uniquely identify and authenticate non-organizational users and processes acting on their behalf to ensure secure access and prevent unauthorized usage.

**Source:**

- ✓ V-222556: The application must uniquely identify and authenticate non-organizational users (or processes acting on behalf of non-organizational users). Configure the application to identify and authenticate all non-organizational users [15].
- ✓ One unique account for each patient and each health care provider [26].

**Priority:** Not described

**Rationale:** This requirement ensures that non-organizational users and processes are properly identified and authenticated before accessing application resources. By implementing robust identification and authentication mechanisms, the application prevents unauthorized access and safeguards sensitive healthcare data.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:****1. User Authentication Testing:**

- ✓ Verify that non-organizational users are required to provide unique credentials for authentication.

**2. Process Authentication Validation:**

- ✓ Ensure that processes acting on behalf of non-organizational users are authenticated securely.
- ✓ Validate that these processes have restricted access based on their roles and permissions.

**3. Audit Trail Review:**

- ✓ Confirm that all authentication attempts are logged with details such as user identity, timestamp, and outcome.
- ✓ Validate that the audit logs are securely stored and accessible only to authorized personnel.

**4. Code Review for Authentication Mechanisms:**

- ✓ Review the implementation of identification and authentication mechanisms to ensure compliance with security standards.
- ✓ Confirm that authentication mechanisms are resistant to common vulnerabilities such as brute force or credential-stuffing attacks.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda  
 José L Fernández-Alemán  
 Juan Manuel Carrillo-de-Gea  
 Joaquín Nicolás  
 08/01/2026

<b>PUID:</b> [SECM-CAT-IAA-002]
<b>Requirement description:</b> The application must define and document user actions that can be performed without identification or authentication, ensuring they align with the organizational mission and business functions and provide supporting rationale in the security plan.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ AC-14 PERMITTED ACTIONS WITHOUT IDENTIFICATION OR AUTHENTICATION</li> </ul> <p>Control:</p> <ul style="list-style-type: none"> <li>a. Identify [Assignment: organization-defined user actions] that can be performed on the system without identification or authentication consistent with organizational mission and business functions; and</li> <li>b. Document and provide supporting rationale in the security plan for the system, user actions not requiring identification or authentication [11].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> This requirement ensures that any user actions permitted without identification or authentication are clearly defined, justified, and documented. By aligning these actions with organizational goals and documenting them in the security plan, the application reduces potential misuse and enhances transparency in access control policies.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<p><b>1. Action Definition Validation:</b></p> <ul style="list-style-type: none"> <li>✓ Verify that all user actions permitted without identification or authentication are explicitly defined.</li> <li>✓ Confirm that these actions are consistent with the organization's mission and business requirements.</li> </ul> <p><b>2. Security Plan Review:</b></p> <ul style="list-style-type: none"> <li>✓ Ensure that the security plan includes documentation of permitted actions and provides supporting rationale for each.</li> </ul>

- ✓ Validate that the rationale aligns with organizational security policies and standards.

### **3. Stakeholder Approval:**

- ✓ Confirm that stakeholders, including security and compliance teams, have reviewed and approved the documented permitted actions.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda

José L Fernández-Alemán

Juan Manuel Carrillo-de-Gea

Joaquín Nicolás

08/01/2026

**PUID:** [SECM-CAT-IIA-003]

**Requirement description:** The application must prevent non-privileged users from executing privileged functions, including disabling, circumventing, or altering implemented security safeguards or countermeasures.

**Source:**

- ✓ V-222429: The application must prevent non-privileged users from executing privileged functions to include disabling, circumventing, or altering implemented security safeguards/countermeasures [15].

**Priority:** Not described

**Rationale:** This requirement ensures that only authorized users with appropriate privileges can execute critical functions or modify security safeguards. Preventing non-privileged users from accessing or altering privileged functionalities mitigates risks such as unauthorized access, data breaches, and the compromise of application security measures, which is especially critical in healthcare applications.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

#### **1. Role-Based Access Testing:**

- ✓ Verify that privileged functions are restricted to users with appropriate roles and permissions.
- ✓ Attempt to execute privileged functions as a non-privileged user and confirm that access is denied.

## 2. Security Safeguard Integrity Validation:

- ✓ Confirm that non-privileged users cannot disable, circumvent, or alter security safeguards or countermeasures.
- ✓ Simulate attempts to tamper with security features and ensure they are blocked.

## 3. Code Review for Access Control Implementation:

- ✓ Review the application codebase to validate the proper implementation of role-based access controls for privileged functions.
- ✓ Ensure compliance with secure coding practices.

## 4. Audit Log Verification:

- ✓ Verify that all attempts to execute privileged functions are logged, including details such as user identity, timestamp, and the outcome of the attempt.
- ✓ Confirm that logs are securely stored and accessible only to authorized personnel.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda

José L Fernández-Alemán

Juan Manuel Carrillo-de-Gea

Joaquín Nicolás

08/01/2026

**PUID:** [SECM-CAT-IIA-004]

**Requirement description:** The application must terminate all existing user sessions immediately upon account deletion to prevent unauthorized access to application resources.

**Source:**

- ✓ V-222549: The application must terminate existing user sessions upon account deletion. Configure the application to terminate existing sessions of users whose accounts are deleted [15].

**Priority:** Not described

**Rationale:** This requirement ensures that deleted user accounts no longer retain active sessions that could allow unauthorized access to sensitive data or application resources. Terminating existing sessions upon account deletion prevents potential misuse, enhances security, and aligns with best practices for session management in healthcare applications.

<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<p><b>1. Session Termination Testing:</b></p> <ul style="list-style-type: none"> <li>✓ Simulate the deletion of a user account and verify that all active sessions associated with the account are terminated immediately.</li> <li>✓ Confirm that no further actions can be performed by the deleted account.</li> </ul> <p><b>2. Real-Time Termination Validation:</b></p> <ul style="list-style-type: none"> <li>✓ Test the application's ability to terminate sessions in real time when an account is deleted.</li> <li>✓ Ensure that the session termination propagates across all devices or platforms where the user was logged in.</li> </ul> <p><b>3. Audit Log Review:</b></p> <ul style="list-style-type: none"> <li>✓ Verify that the session termination is logged with details such as the user account, timestamp, and the initiating action.</li> <li>✓ Confirm that the audit logs are securely stored and accessible only to authorized personnel.</li> </ul> <p><b>4. Code Review for Session Management Logic:</b></p> <ul style="list-style-type: none"> <li>✓ Review the implementation of session termination logic to ensure compliance with the requirement.</li> <li>✓ Validate the robustness of session invalidation mechanisms.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IAA-005]
<b>Requirement description:</b> The application must automatically terminate non-privileged user sessions and log off users after a 15-minute period of inactivity to enhance security and prevent unauthorized access.
<b>Source:</b> <ul style="list-style-type: none"><li>✓ V-222389: The application must automatically terminate the non-privileged user session and log off non-privileged users after a 15-minute idle time period has elapsed. Design and configure the application to terminate the non-privileged users session after 15 minutes of inactivity [15].</li></ul>
<b>Priority:</b> Not described
<b>Rationale:</b> This requirement ensures that inactive sessions are automatically terminated to prevent unauthorized access to the application and sensitive data. Enforcing a 15-minute timeout period enhances security by minimizing risks posed by unattended sessions, particularly in environments handling healthcare data.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b> <ol style="list-style-type: none"><li><b>1. Idle Timeout Validation:</b><ul style="list-style-type: none"><li>✓ Verify that the application terminates non-privileged user sessions after 15 minutes of inactivity.</li><li>✓ Test various scenarios to confirm consistent enforcement of the idle timeout policy.</li></ul></li><li><b>2. Session Termination Effectiveness:</b><ul style="list-style-type: none"><li>✓ Simulate attempts to resume a session after the idle timeout period and confirm that reauthentication is required.</li></ul></li><li><b>3. Configuration Review:</b></li></ol>

- ✓ Validate that the timeout period is configurable and defaults to 15 minutes for non-privileged users.
- ✓ Ensure that the timeout period is properly documented and enforced across all application components.

#### 4. Code Review for Timeout Implementation:

- ✓ Review the codebase to ensure the idle timeout mechanism is implemented securely and aligns with the specified duration.
- ✓ Check for adherence to secure coding practices in session management.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda

José L Fernández-Alemán

Juan Manuel Carrillo-de-Gea

Joaquín Nicolás

08/01/2026

<b>PUID:</b> [SECM-CAT-IAA-006]
<b>Requirement description:</b> The application must automatically terminate administrator user sessions and log off users after a 10-minute period of inactivity to safeguard access to privileged functions.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ V-222390: The application must automatically terminate the admin user session and log off admin users after a 10-minute idle time period is exceeded. Design and configure the application to terminate the admin users session after 10 minutes of inactivity [15].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> This requirement ensures that administrator sessions are automatically terminated after a short idle period to minimize the risk of unauthorized access to privileged functions. By enforcing a 10-minute timeout for admin users, the application enhances security for critical operations and prevents misuse in environments handling sensitive healthcare data.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>

**1. Admin Idle Timeout Validation:**

- ✓ Verify that the application terminates administrator user sessions after 10 minutes of inactivity.
- ✓ Test scenarios with idle admin sessions to ensure consistent enforcement of the timeout policy.

**2. Reauthentication Requirement Testing:**

- ✓ Simulate attempts to resume an admin session after the idle timeout period and confirm that reauthentication is required.

**3. Configuration Review:**

- ✓ Validate that the timeout period for administrator sessions is configurable and defaults to 10 minutes.
- ✓ Ensure that the timeout configuration is documented and implemented across all application components.

**4. Code Review for Timeout Logic:**

- ✓ Review the codebase to ensure the idle timeout mechanism for admin sessions is implemented securely and adheres to best practices.
- ✓ Check for consistency in timeout enforcement across all privileged functions.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda

José L Fernández-Alemán

Juan Manuel Carrillo-de-Gea

Joaquín Nicolás

08/01/2026

**PUID:** [SECM-CAT-IAA-007]

**Requirement description:** The application must ensure that system or application identifiers are not displayed until the log-on process has been successfully completed.

**Source:**

- ✓ 9.4.2 Secure log-on procedures: a) not display system or application identifiers until the log-on process has been successfully completed; [7].

**Priority:** Not described

**Rationale:** This requirement protects the application from unauthorized information disclosure by ensuring that sensitive system or application identifiers are hidden until user authentication is successfully completed. By enforcing secure log-on procedures, the application minimizes exposure to unauthorized users and enhances the overall security posture.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

**1. Log-on Identifier Validation:**

- ✓ Verify that system or application identifiers are not displayed until after the user has successfully logged in.
- ✓ Test various log-on scenarios to confirm that identifiers remain hidden during the authentication process.

**2. Code Review for Log-on Procedures:**

- ✓ Review the implementation of log-on procedures to ensure compliance with secure practices for hiding identifiers.
- ✓ Validate adherence to best practices for user authentication processes.

**3. UI/UX Testing for Identifier Handling:**

- ✓ Confirm that the user interface does not display application or system identifiers during the log-on process.
- ✓ Validate those identifiers, which are revealed only after successful authentication.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda

José L Fernández-Alemán

Juan Manuel Carrillo-de-Gea

Joaquín Nicolás

08/01/2026

**PUID:** [SECM-CAT-IAA-008]

**Requirement description:** The application must prevent the display of help messages during the log-on procedure and obscure authentication feedback to avoid aiding unauthorized users or exposing sensitive authentication information.

**Source:**

- ✓ 9.4.2 Secure log-on procedures: c) not provide help messages during the log-on procedure that would aid an unauthorized user; [7].
- ✓ IA-6 AUTHENTICATION FEEDBACK

**Control:** Obscure feedback of authentication information during the authentication process to protect the information from possible exploitation and use by unauthorized individuals.

**Discussion:** Authentication feedback from systems does not provide information that would allow unauthorized individuals to compromise authentication mechanisms. For some types of systems, such as desktops or notebooks with relatively large monitors, the threat (referred to as shoulder surfing) may be significant. For other types of systems, such as mobile devices with small displays, the threat may be less significant and is balanced against the increased likelihood of typographic input errors due to small keyboards. Thus, the means for obscuring authentication feedback is selected accordingly. Obscuring authentication feedback includes displaying asterisks when users type passwords into input devices or displaying feedback for a very limited time before obscuring it [11].

**Priority:** Not described

**Rationale:** This requirement minimizes the risk of unauthorized access by ensuring that no helpful information is provided to attackers during the authentication process. By disabling log-on help messages and obscuring authentication feedback, the application reduces the likelihood of information leakage that could aid in compromising authentication mechanisms, enhancing the security of sensitive healthcare systems.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

### 1. Help Message Validation:

- ✓ Verify that no help messages are displayed during the log-on procedure.
- ✓ Test scenarios with incorrect log-on attempts to confirm that no detailed hints or information are provided to the user.

## 2. Authentication Feedback Obfuscation Testing:

- ✓ Confirm that authentication feedback, such as passwords, is obscured during entry (e.g., using asterisks or dots).
- ✓ Validate that feedback is not displayed long enough to be exploited by shoulder surfing.

## 3. Unauthorized Access Simulation:

- ✓ Simulate attempts by unauthorized users to gain insights into the authentication process and confirm that no exploitable information is provided.

## 4. Code Review for Log-on Security:

- ✓ Review the implementation of log-on procedures to ensure compliance with secure feedback and message-handling practices.
- ✓ Validate that feedback obfuscation and help message suppression are consistently applied across all authentication methods.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda

José L Fernández-Alemán

Juan Manuel Carrillo-de-Gea

Joaquín Nicolás

08/01/2026

**PUID:** [SECM-CAT-IAA-009]

**Requirement description:** The application must encrypt passwords during transmission to ensure that no passwords are sent in clear text over the network.

**Source:**

- ✓ 9.4.2 Secure log-on procedures: j) not transmit passwords in clear text over a network; [7].
- ✓ V-222543: The application must transmit only cryptographically-protected passwords. Configure the application to encrypt passwords when they are being transmitted [15].

**Priority:** Not described

**Rationale:** This requirement ensures the confidentiality and security of user credentials by mandating cryptographic protection for passwords during transmission. By encrypting passwords, the application prevents exposure to eavesdropping, man-in-the-middle attacks, and other network-based threats, maintaining the integrity of authentication mechanisms in healthcare systems.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<b>1. Transmission Encryption Validation:</b>
✓ Verify that passwords are encrypted using secure cryptographic protocols (e.g., TLS 1.2 or higher) during transmission.
✓ Test transmission scenarios to confirm that no passwords are sent in clear text.
<b>2. Network Traffic Analysis:</b>
✓ Use network monitoring tools to inspect data packets during password transmission and confirm that password fields are encrypted.
<b>3. Code Review for Encryption Implementation:</b>
✓ Review the application's authentication module to ensure encryption is applied to passwords before transmission.
✓ Validate the use of secure cryptographic libraries and protocols.
<b>4. Penetration Testing for Data Exposure:</b>
✓ Perform penetration tests to identify potential vulnerabilities in password transmission and ensure encryption mechanisms withstand attacks.
<b>5. Audit Log Verification:</b>
✓ Confirm that audit logs record successful and failed authentication attempts but do not store passwords in clear text.
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

**PUID:** [SECM-CAT-IIA-010]

**Requirement description:** The application must validate log-on information only after all input data has been completed and must not indicate which specific part of the input is correct or incorrect in case of an error.

**Source:**

- ✓ 9.4.2 Secure log-on procedures: d) validate the log-on information only on completion of all input data. If an error condition arises, the system should not indicate which part of the data is correct or incorrect; [7].

**Priority:** Not described

**Rationale:** This requirement protects the log-on process from enumeration attacks by ensuring that the application does not provide detailed error messages indicating the correctness of individual input fields. By validating log-on information after all input data is entered, the application minimizes the risk of information disclosure that could aid attackers in bypassing authentication mechanisms.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

**1. Validation Timing Testing:**

- ✓ Verify that log-on information is validated only after all required input fields are completed.
- ✓ Ensure the application does not perform partial validation during data entry.

**2. Error Message Inspection:**

- ✓ Confirm that error messages during the log-on process are generic and do not reveal whether specific input data (e.g., username or password) is correct or incorrect.

**3. Penetration Testing for Enumeration:**

- ✓ Conduct tests to simulate enumeration attacks and ensure the application does not disclose details about which input fields failed validation.

**4. Code Review for Validation Logic:**

- ✓ Review the implementation of log-on validation to confirm that error handling logic avoids specific input feedback.
- ✓ Ensure adherence to secure coding practices for authentication processes.

**Requested by:** The organization

<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IAA-011]
<b>Requirement description:</b> The application must enforce a limit of three consecutive invalid log-on attempts within a 15-minute time period and lock the account or take other defined actions when the maximum number of unsuccessful attempts is exceeded.
<b>Source:</b>
<ul style="list-style-type: none"><li>✓ AC-7 UNSUCCESSFUL LOGON ATTEMPTS<ul style="list-style-type: none"><li>a. Enforce a limit of [Assignment: organization-defined number] consecutive invalid logon attempts by a user during a [Assignment: organization-defined time period]; and</li><li>b. Automatically [Selection (one or more): lock the account or node for an [Assignment: organization-defined time period]; lock the account or node until released by an administrator; delay next logon prompt per [Assignment: organization-defined delay algorithm]; notify system administrator; take other [Assignment: organization-defined action]] when the maximum number of unsuccessful attempts is exceeded.; [11].</li></ul></li><li>✓ V-222432: The application must enforce the limit of three consecutive invalid logon attempts by a user during a 15-minute time period. Configure the application to enforce an account lock after 3 failed logon attempts occurring within a 15-minute window [15].</li></ul>
<b>Priority:</b> Not described
<b>Rationale:</b> This requirement mitigates risks associated with brute force and unauthorized access attempts by limiting the number of invalid log-on attempts. Enforcing account lockout or other protective actions after a set number of failed attempts strengthens security and helps prevent malicious actors from compromising user accounts in healthcare applications.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>

**1. Failed Log-on Attempt Simulation:**

- ✓ Simulate three consecutive invalid log-on attempts within a 15-minute window and verify that the account is locked or appropriate actions are taken.
- ✓ Ensure that attempts after the lockout period are logged and require proper authorization.

**2. Configuration Review:**

- ✓ Verify that the application enforces the lockout threshold (three invalid attempts) and the lockout duration (organization-defined).
- ✓ Confirm that these settings are configurable and documented.

**3. Notification Validation:**

- ✓ Ensure that administrators are notified when the lockout mechanism is triggered.
- ✓ Validate that the notification includes details such as the user account, timestamp, and triggering events.

**4. Code Review for Lockout Logic:**

- ✓ Review the application's implementation of log-on attempt limits and account lockout logic to ensure compliance with secure coding standards.
- ✓ Check for vulnerabilities that could bypass the lockout mechanism.

**5. Audit Log Verification:**

- ✓ Confirm that all invalid log-on attempts, lockout events, and subsequent activities are recorded in the audit logs.
- ✓ Validate that logs are securely stored and accessible only to authorized personnel.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda

José L Fernández-Alemán

Juan Manuel Carrillo-de-Gea

Joaquín Nicolás

08/01/2026

**PUID:** [SECM-CAT-IAA-012]

**Requirement description:** The application administrator must follow an approved process to unlock locked user accounts, including validating user identity prior to unlocking the account.

<b>Source:</b>
✓ V-222433: The application administrator must follow an approved process to unlock locked user accounts. Create a standard approved process for unlocking locked application accounts which includes validating user identity prior to unlocking the account. Use that process when unlocking application user accounts [15].
<b>Priority:</b> Not described
<b>Rationale:</b> This requirement ensures that the process for unlocking user accounts is secure and standardized to prevent unauthorized access. Validating the user's identity before unlocking an account reduces the risk of account compromise and enhances the overall security of the application, particularly in healthcare settings where sensitive information must be protected.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<p><b>1. Identity Validation Testing:</b></p> <ul style="list-style-type: none"> <li>✓ Verify that the account unlocking process includes steps to validate the user's identity using approved methods.</li> <li>✓ Test the process to ensure it cannot be bypassed by unauthorized users.</li> </ul> <p><b>2. Administrative Process Validation:</b></p> <ul style="list-style-type: none"> <li>✓ Confirm that the approved process for unlocking accounts is documented and communicated to all relevant personnel.</li> <li>✓ Ensure that administrators are trained to follow the approved process consistently.</li> </ul> <p><b>3. Audit Log Review:</b></p> <ul style="list-style-type: none"> <li>✓ Verify that all account unlocking events are logged with details such as the administrator performing the action, user account, timestamp, and reason for unlocking.</li> <li>✓ Validate that audit logs are securely stored and accessible only to authorized personnel.</li> </ul> <p><b>4. Policy Review:</b></p> <ul style="list-style-type: none"> <li>✓ Ensure that the account unlocking policy includes clear steps for validating user identity and criteria for approving unlock requests.</li> <li>✓ Validate that the policy aligns with organizational security standards and compliance requirements.</li> </ul>

<b>5. Code Review for Unlock Logic:</b>
<ul style="list-style-type: none"> <li>✓ Review the implementation of the unlocking mechanism to ensure compliance with secure coding practices and prevention of unauthorized access.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda
José L Fernández-Alemán
Juan Manuel Carrillo-de-Gea
Joaquín Nicolás
08/01/2026
<b>PUID:</b> [SECM-CAT-IIA-012]
<b>Requirement description:</b> The application administrator must follow an approved process to unlock locked user accounts, including validating user identity prior to unlocking the account.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ V-222433: The application administrator must follow an approved process to unlock locked user accounts. Create a standard approved process for unlocking locked application accounts which includes validating user identity prior to unlocking the account. Use that process when unlocking application user accounts [15].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> This requirement ensures that the process for unlocking user accounts is secure and standardized to prevent unauthorized access. Validating the user's identity before unlocking an account reduces the risk of account compromise and enhances the overall security of the application, particularly in healthcare settings where sensitive information must be protected.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Identity Validation Testing:</b></li> </ol> <ul style="list-style-type: none"> <li>✓ Verify that the account unlocking process includes steps to validate the user's identity using approved methods.</li> <li>✓ Test the process to ensure it cannot be bypassed by unauthorized users.</li> </ul>

**2. Administrative Process Validation:**

- ✓ Confirm that the approved process for unlocking accounts is documented and communicated to all relevant personnel.
- ✓ Ensure that administrators are trained to follow the approved process consistently.

**3. Audit Log Review:**

- ✓ Verify that all account unlocking events are logged with details such as the administrator performing the action, user account, timestamp, and reason for unlocking.
- ✓ Validate that audit logs are securely stored and accessible only to authorized personnel.

**4. Policy Review:**

- ✓ Ensure that the account unlocking policy includes clear steps for validating user identity and criteria for approving unlock requests.
- ✓ Validate that the policy aligns with organizational security standards and compliance requirements.

**5. Code Review for Unlock Logic:**

- ✓ Review the implementation of the unlocking mechanism to ensure compliance with secure coding practices and prevention of unauthorized access.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda

José L Fernández-Alemán

Juan Manuel Carrillo-de-Gea

Joaquín Nicolás

08/01/2026

**PUID:** [SECM-CAT-IAA-013]

**Requirement description:** The application must display a system use notification banner before granting access, warning users that the system is for authorized use only and providing legal and privacy notices.

**Source:**

- ✓ 9.4.2 Secure log-on procedures: b) display a general notice warning that the computer should only be accessed by authorized users [7].

- ✓ V-222434: The application must display the Standard Mandatory DoD Notice and Consent Banner before granting access to the application. Configure the application to present the standard DoD-approved banner prior to granting access to the application [15].
- ✓ Easy-to-understand privacy policy which clearly indicates that my personal data are well protected [26].
- ✓ AC-8 SYSTEM USE NOTIFICATION
 

Control:

  - a. Display [Assignment: organization-defined system use notification message or banner] to users before granting access to the system that provides privacy and security notices consistent with applicable laws, executive orders, directives, regulations, policies, standards, and guidelines and state that:
    1. Users are accessing a U.S. Government system;
    2. System usage may be monitored, recorded, and subject to audit;
    3. Unauthorized use of the system is prohibited and subject to criminal and civil penalties; and
    4. Use of the system indicates consent to monitoring and recording;
  - b. Retain the notification message or banner on the screen until users acknowledge the usage conditions and take explicit actions to log on to or further access the system; and
  - c. For publicly accessible systems:
    1. Display system use information [Assignment: organization-defined conditions], before granting further access to the publicly accessible system;
    2. Display references, if any, to monitoring, recording, or auditing that are consistent with privacy accommodations for such systems that generally prohibit those activities; and
    3. Include a description of the authorized uses of the system [11].
- ✓ Test ID 2: Only currently authorized users are able to access the EHR data [37].
- ✓ The application must display the Standard Mandatory DoD Notice and Consent Banner before granting access to the application. Configure the application to present the standard DoD-approved banner prior to granting access to the application [15].

**Priority:** Not described

**Rationale:** Displaying a system use notification banner ensures that users are informed of the terms and conditions of system access, including privacy, monitoring, and authorized use policies. This promotes compliance with organizational security policies and legal requirements, protecting sensitive healthcare data from unauthorized access.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

**1. Notification Banner Display Testing:**

- ✓ Verify that the system displays the notification banner before granting access.
- ✓ Confirm that the banner includes privacy and security notices consistent with applicable laws and policies.

**2. User Acknowledgment Validation:**

- ✓ Ensure that users explicitly acknowledge the notification banner before logging in or accessing the system.

**3. Banner Retention Testing:**

- ✓ Validate that the notification banner remains on the screen until user acknowledgment is received.

**4. Audit Log Review:**

- ✓ Verify that user acknowledgments of the notification banner are logged, including timestamps and user details where applicable.

**5. Code Review for Banner Implementation:**

- ✓ Review the codebase to confirm proper implementation of the notification banner and ensure compliance with secure coding practices.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda

José L Fernández-Alemán

Juan Manuel Carrillo-de-Gea

Joaquín Nicolás

08/01/2026

**PUID:** [SECM-CAT-IAA-014]

**Requirement description:** The application must retain the Standard Mandatory DoD Notice and Consent Banner on the screen until the user acknowledges the usage conditions and explicitly logs on for further access.

**Source:**

- ✓ V-222435: The application must retain the Standard Mandatory DoD Notice and Consent Banner on the screen until users acknowledge the usage conditions and take explicit actions to log on for further access. Configure the application to retain the standard DoD-approved

<p>banner until the user accepts the usage conditions prior to granting access to the application [15].</p> <ul style="list-style-type: none"> <li>✓ V-222436: The publicly accessible application must display the Standard Mandatory DoD Notice and Consent Banner before granting access to the application. Configure the application to present the standard DoD-approved banner prior to granting access to the application [15].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> This requirement ensures that users are informed of the usage conditions, privacy notices, and consent requirements before accessing the application. By retaining the notice until explicit acknowledgment, the application mitigates risks of unauthorized use and ensures compliance with legal and organizational policies.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<p><b>1. Banner Display Testing:</b></p> <ul style="list-style-type: none"> <li>✓ Verify that the application displays the Standard Mandatory DoD Notice and Consent Banner before granting access.</li> <li>✓ Confirm that the banner remains on the screen until the user explicitly acknowledges the usage conditions.</li> </ul> <p><b>2. User Acknowledgment Validation:</b></p> <ul style="list-style-type: none"> <li>✓ Test the log-on process to ensure that users must take explicit actions to acknowledge the banner before accessing the application.</li> </ul> <p><b>3. Code Review for Banner Logic:</b></p> <ul style="list-style-type: none"> <li>✓ Review the code implementation to ensure the banner retention and acknowledgment mechanisms are secure and align with organizational standards.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán

Juan Manuel Carrillo-de-Gea  
Joaquín Nicolás  
08/01/2026

**PUID:** [SECM-CAT-IIA-015]

**Requirement description:** The application must automatically remove or disable temporary user accounts 72 hours after account creation to prevent unauthorized access.

**Source:**

- ✓ V-222409: The application must automatically remove or disable temporary user accounts 72 hours after account creation. Configure temporary accounts to be automatically removed or disabled after 72 hours after account creation [15].

**Priority:** Not described

**Rationale:** This requirement ensures that temporary accounts do not persist beyond their intended usage period, reducing the risk of unauthorized access. Automatically removing or disabling these accounts enforces strict access control policies and prevents unused accounts from becoming security vulnerabilities.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

**1. Account Expiration Testing:**

- ✓ Verify that temporary accounts are automatically removed or disabled 72 hours after creation.
- ✓ Simulate account creation and confirm that expiration occurs within the defined time frame.

**2. Configuration Review:**

- ✓ Ensure that the application configuration supports defining and enforcing the 72-hour expiration period for temporary accounts.

**3. Code Review for Account Management Logic:**

- ✓ Review the implementation of temporary account expiration logic to ensure compliance with security standards.
- ✓ Validate that no exceptions or overrides are possible without appropriate authorization.

<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda
José L Fernández-Alemán
Juan Manuel Carrillo-de-Gea
Joaquín Nicolás
08/01/2026

<b>PUID:</b> [SECM-CAT-IAA-016]
<b>Requirement description:</b> The application must disable or delete unnecessary user accounts, including built-in accounts, and prevent the creation of unessential accounts during installation.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ V-222412: Unnecessary application accounts must be disabled, or deleted. Design the application so unessential user accounts are not created during installation. Disable or delete all unnecessary application user accounts [15].</li> <li>✓ V-222661: Unnecessary built-in application accounts must be disabled. Disable unnecessary built-in userids, use other strong authentication when possible and use strong passwords if accounts are necessary for application operation.</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> This requirement ensures that only essential user accounts exist within the application, reducing the attack surface and preventing unauthorized access. By disabling or deleting unnecessary accounts, including built-in ones, and enforcing strong authentication for essential accounts, the application minimizes the risk of compromise in healthcare environments.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Account Review and Deletion Testing:</b> <ul style="list-style-type: none"> <li>✓ Verify that all unnecessary user accounts, including built-in accounts, are disabled or deleted.</li> <li>✓ Ensure that unessential accounts are not created during the application installation process.</li> </ul> </li> </ol>

**2. Authentication Mechanism Validation:**

- ✓ Confirm that strong authentication methods are used for any built-in accounts that are essential for application operation.

**3. Configuration Review:**

- ✓ Validate that the application supports configuration options to disable or delete unnecessary accounts.

**4. Code Review for Account Management:**

- ✓ Validate that no unnecessary accounts are hardcoded or automatically created during installation.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda  
José L Fernández-Alemán  
Juan Manuel Carrillo-de-Gea  
Joaquín Nicolás  
08/01/2026

**PUID:** [SECM-CAT-IAA-017]

**Requirement description:** The application must replace default passwords with DoD-approved strong passwords and prioritize the use of strong authenticators instead of passwords wherever possible.

**Source:**

- ✓ V-222662: Default passwords must be changed. Configure the application to use strong authenticators instead of passwords when possible. Otherwise, change default passwords to a DoD-approved strength password and follow all guidance for passwords [15].

**Priority:** Not described

**Rationale:** This requirement ensures that default passwords, which are often well-known and vulnerable to exploitation, are replaced with strong passwords compliant with DoD standards. By promoting the use of strong authenticators, the application further enhances security and reduces reliance on traditional password mechanisms, aligning with modern security practices in healthcare applications.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<p><b>1. Default Password Validation:</b></p> <ul style="list-style-type: none"> <li>✓ Verify that all default passwords are replaced with strong, DoD-approved passwords during configuration or installation.</li> <li>✓ Confirm that no default passwords remain in use after deployment.</li> </ul> <p><b>2. Strong Authenticator Testing:</b></p> <ul style="list-style-type: none"> <li>✓ Validate that strong authenticators, such as biometric or multi-factor authentication, are prioritized over passwords wherever possible.</li> </ul> <p><b>3. Password Policy Enforcement:</b></p> <ul style="list-style-type: none"> <li>✓ Confirm that the application enforces strong password policies, including minimum length, complexity, and expiration requirements.</li> </ul> <p><b>4. Configuration Review:</b></p> <ul style="list-style-type: none"> <li>✓ Verify that the application supports configuration options for disabling or replacing default passwords.</li> </ul> <p><b>5. Code Review for Password Handling:</b></p> <ul style="list-style-type: none"> <li>✓ Review the codebase to ensure secure handling of passwords and compliance with password strength policies.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IAA-018]
<b>Requirement description:</b> The application must automatically disable user accounts after a 35-day period of inactivity to mitigate unauthorized access risks.

<b>Source:</b>
✓ V-222411: The application must automatically disable accounts after a 35-day period of account inactivity. Design and configure the application to expire user accounts after 35 days of inactivity [15].
<b>Priority:</b> Not described
<b>Rationale:</b> This requirement ensures that inactive accounts are automatically disabled, reducing the risk of unauthorized access through dormant or forgotten accounts. By enforcing account expiration after a 35-day inactivity period, the application strengthens access control and minimizes vulnerabilities in healthcare environments.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<b>1. Inactivity Period Validation:</b>
✓ Verify that user accounts are automatically disabled after 35 days of inactivity. ✓ Test scenarios with varying inactivity durations to confirm consistent enforcement.
<b>2. Reactivation Process Testing:</b>
✓ Ensure that a secure process is in place for reactivating accounts that have been disabled due to inactivity. ✓ Validate that reactivation requires identity verification and administrative approval.
<b>3. Audit Log Review:</b>
✓ Confirm that account disabling events are logged with details such as the user account, timestamp, and reason for the action. ✓ Validate that logs are securely stored and accessible only to authorized personnel.
<b>4. Configuration Review:</b>
✓ Verify that the inactivity period is configurable and defaults to 35 days.
<b>5. Code Review for Account Management:</b>
✓ Review the implementation of the inactivity disabling mechanism to ensure compliance with security best practices and organizational policies.

<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda
José L Fernández-Alemán
Juan Manuel Carrillo-de-Gea
Joaquín Nicolás
08/01/2026

<b>PUID:</b> [SECM-CAT-IAA-019]
<b>Requirement description:</b> The application must implement processes or features to prevent the removal or disabling of emergency accounts while the emergency situation is ongoing.
<b>Source:</b>
✓ V-222410: The application must have a process, feature or function that prevents removal or disabling of emergency accounts. Identify accounts that are created in an emergency situation and ensure procedures or processes are in place to prevent disabling or deleting the account while the emergency is underway [15].
<b>Priority:</b> Not described
<b>Rationale:</b> Emergency accounts provide critical access during urgent situations and must remain active for the duration of the emergency. Preventing the removal or disabling of these accounts ensures continuity of operations and supports effective incident response, particularly in healthcare environments where timely access to data and systems is essential.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<b>1. Emergency Account Testing:</b>
✓ Verify that emergency accounts are flagged and protected from removal or disabling during the emergency period.

- ✓ Test scenarios to confirm that account protection mechanisms remain effective throughout the emergency.

## 2. Process Validation:

- ✓ Ensure that processes for managing emergency accounts are documented and include safeguards to prevent unauthorized actions.

## 3. Configuration Review:

- ✓ Validate that the application provides options to configure and manage emergency account protections, including criteria for enabling and disabling the protections.

## 4. Code Review for Emergency Account Features:

- ✓ Review the implementation of emergency account protections to ensure compliance with secure coding practices.
- ✓ Validate that no bypasses or overrides are possible without proper authorization.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda

José L Fernández-Alemán

Juan Manuel Carrillo-de-Gea

Joaquín Nicolás

08/01/2026

**PUID:** [SECM-CAT-IAA-020]

**Requirement description:** The application must terminate shared or group account credentials immediately when group members leave, ensuring the security of the group account.

**Source:**

- ✓ V-222408: Shared/group account credentials must be terminated when members leave the group. Create a procedure for deleting either member accounts or the entire group account when members leave the group [15].

**Priority:** Not described

**Rationale:** This requirement ensures that shared or group accounts are protected from unauthorized access when members leave the group. By promptly terminating credentials or accounts associated with departed members, the application prevents potential misuse and maintains the integrity of sensitive healthcare systems.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<b>1. Group Account Deletion Testing:</b>
✓ Verify that the group account is deleted when all members leave or if the group is disbanded.
<b>2. Configuration Review:</b>
✓ Validate that the application provides a configurable mechanism for managing group account termination and individual access removal.
<b>3. Policy Validation:</b>
✓ Ensure documented procedures for managing group accounts include requirements for terminating credentials when members leave. ✓ Validate that the policy is communicated to and understood by administrators.
<b>4. Code Review for Group Account Management:</b>
✓ Review the implementation of group account management logic to ensure secure handling of membership changes and account termination.
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IAA-021]
<b>Requirement description:</b> The application must terminate all user sessions and network connections at the end of a session or after defined inactivity periods, including verifying remote disconnection for non-local maintenance sessions.
<b>Source:</b>

<ul style="list-style-type: none"> <li>✓ V-222566: The application must terminate all sessions and network connections when non-local maintenance is completed. Configure the application to expire idle user sessions after 10 minutes of inactivity for admin users and after 15 minutes of inactivity for regular users [15].</li> <li>✓ V-222564: Applications used for non-local maintenance sessions must verify remote disconnection at the termination of non-local maintenance and diagnostic sessions. Configure the application to verify termination of remote maintenance sessions [15].</li> <li>✓ V-222568: The application must terminate all network connections associated with a communications session at the end of the session. Configure or design the application to terminate application network sessions at the end of the session [15].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> This requirement ensures that sessions and network connections are securely terminated after use or inactivity, preventing unauthorized access and reducing the risk of session hijacking. Verifying disconnection for non-local maintenance sessions ensures proper session management in remote diagnostics and maintenance scenarios, aligning with secure practices for healthcare systems.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<p><b>1. Session Termination Testing:</b></p> <ul style="list-style-type: none"> <li>✓ Verify that user sessions are terminated after 10 minutes of inactivity for admin users and 15 minutes for regular users.</li> <li>✓ Confirm that all network connections are closed at the end of each session.</li> </ul> <p><b>2. Network Connection Termination Testing:</b></p> <ul style="list-style-type: none"> <li>✓ Test the termination of all network connections associated with a session to ensure no connections remain active after the session ends.</li> </ul> <p><b>3. Code Review for Session Management Logic:</b></p> <ul style="list-style-type: none"> <li>✓ Review the implementation of session and network termination mechanisms to ensure compliance with secure coding practices and organizational policies.</li> </ul>

<b>4. Policy Review:</b>
✓ Ensure that session termination policies, including inactivity thresholds and remote maintenance disconnection requirements, are documented and communicated to relevant stakeholders.
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IIA-022]
<b>Requirement description:</b> The application must provide the capability to limit the number of concurrent logon sessions per user based on organization-defined thresholds.
<b>Source:</b>
✓ V-222387: The application must provide a capability to limit the number of logon sessions per user. Design and configure the application to specify the number of logon sessions that are allowed per user [15].
✓ AC-10 CONCURRENT SESSION CONTROL
✓ Control: Limit the number of concurrent sessions for each [Assignment: organization-defined account and/or account type] to [Assignment: organization-defined number]; [11].
<b>Priority:</b> Not described
<b>Rationale:</b> This requirement prevents security risks associated with excessive or unauthorized concurrent sessions. By limiting the number of active sessions per user, the application reduces the attack surface and enforces secure session management, which is particularly important in healthcare systems where sensitive data must be protected.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>

**1. Concurrent Session Limitation Testing:**

- ✓ Verify that the application enforces the organization-defined limit on concurrent logon sessions per user.

**2. Configuration Validation:**

- ✓ Confirm that the number of allowed concurrent sessions is configurable and adheres to organizational policies.

**3. Code Review for Session Control Mechanisms:**

- ✓ Review the implementation of session control logic to ensure proper enforcement of concurrent session limits.
- ✓ Check for vulnerabilities that could bypass session limits.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda

José L Fernández-Alemán

Juan Manuel Carrillo-de-Gea

Joaquín Nicolás

08/01/2026

**PUID:** [SECM-CAT-IAA-023]

**Requirement description:** The application must uniquely identify and authenticate organizational users and processes acting on their behalf, ensuring that identifiers do not rely on personally identifiable device information and uniquely identify patients within the system.

**Source:**

- ✓ V-222522: The application must uniquely identify and authenticate organizational users (or processes acting on behalf of organizational users). Configure the application to uniquely identify and authenticate users and user processes [15].
- ✓ One unique account for each patient and each health care provider [26].
- ✓ IA-2 IDENTIFICATION AND AUTHENTICATION

Control: Uniquely identify and authenticate organizational users and associate that unique identification with processes acting on behalf of those users [11].

<ul style="list-style-type: none"> <li>✓ Unique Identifier Generation: If a Globally Unique Identifier (GUID) is needed, the application must generate a large, unique number and avoid using phone identifiers (e.g., phone number or IMEI) that may be linked to personal information [18].</li> <li>✓ SR4: Privacy It must retain the privacy for the patient's identity when stored on the devices as well as when used during communication [30].</li> <li>✓ 4.9.2. Correct Processing of Information</li> </ul> <p>Security Requirement 75 – Uniquely Identifying Patients/Persons: The EHRI and PoS systems connected to the EHRI must:</p> <ul style="list-style-type: none"> <li>a) Ensure that patients/persons are assigned an identifier (patient ID) that can uniquely identify the patient/person within the EHRI or within the PoS system [10];</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> This requirement ensures the secure and accurate identification of organizational users, processes, and patients within the system. By generating unique identifiers and avoiding the use of personal device information, the application enhances privacy and supports precise authentication and record matching in healthcare environments.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<p><b>1. User Identification Testing:</b></p> <ul style="list-style-type: none"> <li>✓ Verify that the application generates and assigns unique identifiers to all organizational users and processes acting on their behalf.</li> <li>✓ Confirm that authentication mechanisms associate users with their identifiers.</li> </ul> <p><b>2. Patient Identifier Validation:</b></p> <ul style="list-style-type: none"> <li>✓ Validate that patients are assigned unique identifiers within the system that ensure accurate record matching.</li> </ul> <p><b>3. Identifier Generation Testing:</b></p> <ul style="list-style-type: none"> <li>✓ Verify that unique identifiers are generated without relying on personal device information such as phone numbers or IMEI.</li> <li>✓ Ensure that Globally Unique Identifiers (GUIDs) are sufficiently random and secure.</li> </ul>
<b>Requested by:</b> The organization

<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IAA-024]
<b>Requirement description:</b> The application must implement a formal user registration and authentication process, ensuring that only authorized users can access electronic health information (EHR).
<b>Source:</b>
<ul style="list-style-type: none"><li>✓ § 164.312 Technical safeguards.<ul style="list-style-type: none"><li>(d) Standard: Person or entity authentication</li><li>✓ Implement procedures to verify that a person or entity seeking access to electronic protected health information is the one claimed [9].</li><li>✓ 4.5. Human Resources Security:<p>Security Requirement 13 – Verifying the Identity of Users All organizations connecting to the EHRI or hosting components of the EHRI must verify the identity and address of each permanent or temporary staff member or contractor who will become a registered user of a PoS system connected to the EHRI or who will have access to hosted components of the EHRI [10].</p><ul style="list-style-type: none"><li>✓ 1.3. Are there procedures to verify that any person or entity claiming access to electronic protected health information complies with its claim? [33].</li><li>✓ 9.2.1 User registration and de-registration: Access to health information systems that process personal health information shall be subject to a formal user registration process. User registration procedures shall ensure that the level of authentication required of claimed user identity is consistent with the level(s) of access that will become available to the user [7].</li><li>✓ 1.1. Is there any registration/log-in available in the app? [33].</li><li>✓ 4.2.8.1. User Registration: Security Requirement 53 – Registering Users: All organizations connecting to the EHRI must subject potential users of PoS systems that connect to the EHRI to a formal user-registration process. These user-registration procedures must ensure:<ol style="list-style-type: none"><li>a) The level of user identification that is provided is consistent with the assurance required, given the value of the information assets and the functions that will become available to the user;</li><li>b) Each potential user has a legitimate relationship with the organization; and</li><li>c) Each potential user has a legitimate need to access PHI via the EHRI [10].</li></ol></li></ul></li></ul></li></ul>

<ul style="list-style-type: none"> <li>✓ CWE-306: Missing Authentication for Critical Function The product does not perform any authentication for functionality that requires a provable user identity or consumes a significant amount of resources [35].</li> <li>✓ CWE-287: Improper Authentication When an actor claims to have a given identity, the product does not prove or insufficiently proves that the claim is correct. This weakness can lead to the exposure of resources or functionality to unintended actors, possibly providing attackers with sensitive information or even execute arbitrary code [35].</li> <li>✓ Test ID 11: Data within EHR is accessible only to authorized users and services [37].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> This requirement ensures that only authorized users can access e-health data by verifying their identity through a formal registration and authentication process, thereby preventing unauthorized access and ensuring that critical functions are only executed with proper authentication.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b> <ol style="list-style-type: none"> <li><b>1. Registration and authentication process:</b> <ul style="list-style-type: none"> <li>✓ Verify that the mobile application implements a formal user registration and authentication process</li> <li>✓ Verify that the user's identity is validated before granting access to sensitive data.</li> </ul> </li> <li><b>2. Role-based and permission-based access:</b> <ul style="list-style-type: none"> <li>✓ Verify that the application limits access to data and functionalities based on user roles</li> <li>✓ Ensure that only authorized users (e.g., administrators or physicians) can access sensitive health data.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IAA-025]
<b>Requirement description:</b> The application must enforce approved authorizations for controlling the flow of information within the system and between interconnected systems, adhering to organization-defined data flow control policies.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ V-222427: The application must enforce approved authorizations for controlling the flow of information within the system based on organization-defined information flow control policies. Configure the application to enforce data flow control in accordance with data flow control policies [15].</li> <li>✓ V-222428: The application must enforce approved authorizations for controlling the flow of information between interconnected systems based on organization-defined information flow control policies. Configure the application to enforce data flow control in accordance with data flow control policies [15].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> This requirement ensures that information flow within the application and between interconnected systems is controlled and authorized based on predefined organizational policies. Proper enforcement of information flow control prevents unauthorized data access, reduces the risk of data leaks, and supports compliance with regulatory requirements in healthcare environments.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Access Control Testing:</b> <ul style="list-style-type: none"> <li>✓ Confirm that only authorized users and processes can initiate or receive information flows within the application and across systems.</li> </ul> </li> <li><b>2. Configuration Review:</b> <ul style="list-style-type: none"> <li>✓ Ensure that the application provides configurable options to define and enforce information flow control policies.</li> </ul> </li> <li><b>3. Code Review for Data Flow Control Logic:</b> <ul style="list-style-type: none"> <li>✓ Review the implementation of data flow control mechanisms to ensure adherence to secure coding practices and compliance with organizational policies.</li> </ul> </li> </ol>

<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda
José L Fernández-Alemán
Juan Manuel Carrillo-de-Gea
Joaquín Nicolás
08/01/2026

<b>PUID:</b> [SECM-CAT-IAA-026]
<b>Requirement description:</b> The application must implement time-limited user registration for Point of Service (PoS) systems connected to the EHRi, requiring periodic renewal of user registrations to maintain access.
<b>Source:</b>
✓ 4.2.8.1. User Registration
Security Requirement 55 – Time-Limited User Registration: All organizations connecting to the EHRi must ensure that the registration of users of PoS systems that connect to the EHRi is time-limited (after which, the user's registration must be renewed) [10].
<b>Priority:</b> Not described
<b>Rationale:</b> This requirement ensures that user registrations for PoS systems connected to the EHRi remain valid and secure by requiring periodic renewal. Time-limited registration reduces the risk of unauthorized access and enforces regular verification of user permissions, aligning with information flow control policies.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<b>1. Registration Expiry Testing:</b>
✓ Verify that user registrations automatically expire after the defined time limit.
✓ Confirm that expired registrations are renewed only through a secure re-registration process.
<b>2. Access Restriction Validation:</b>

- ✓ Test scenarios where users attempt to access the system after their registration has expired, ensuring access is denied until renewal.

### 3. Configuration Review:

- ✓ Validate that the time limit for user registration is configurable and aligns with organizational policies.

### 4. Code Review for Registration Logic:

- ✓ Review the implementation of time-limited registration and renewal mechanisms to ensure compliance with secure coding practices.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda  
 José L Fernández-Alemán  
 Juan Manuel Carrillo-de-Gea  
 Joaquín Nicolás  
 08/01/2026

**PUID:** [SECM-CAT-IAA-027]

**Requirement description:** The application must restrict connection times to high-risk EHRI services to enhance security and minimize the risk of unauthorized access.

**Source:**

- ✓ 9.4.2 Secure log-on procedures: I) restrict connection times to provide additional security for high-risk applications and reduce the window of opportunity for unauthorized access [7].
- ✓ Security Requirement 69 – Restricting Connection Times to EHRI Applications: Where appropriate, the EHRI should restrict connection duration to EHRI application services to provide additional security for access to those applications [10].
- ✓ Test ID 12: Remote access to the EHR is monitored and controlled by access type, preventing unauthorized connections [37].

**Priority:** Not described

**Rationale:** This requirement ensures that connection times to EHRI applications are restricted, reducing the exposure window for unauthorized access and limiting potential attack opportunities. By managing session duration for high-risk applications, the application enhances access control and protects sensitive healthcare data.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<b>1. Connection Time Restriction Testing:</b>
✓ Verify that connection times to high-risk EHRi services are restricted as per organizational policies.
✓ Test scenarios to confirm that connections are terminated after the specified duration.
<b>2. Access Control Validation:</b>
✓ Ensure that remote access to EHRi applications is monitored and restricted based on access type and session duration.
<b>3. Code Review for Session Management:</b>
✓ Review the implementation of session duration controls to ensure secure handling of connection restrictions.
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IAA-028]
<b>Requirement description:</b> The application must implement authentication mechanisms that directly verify the end user's identity rather than relying solely on device identity.
<b>Source:</b>
✓ Use authentication that ties back to the end user identity (rather than only to the device identity) [16].
<b>Priority:</b> Not described
<b>Rationale:</b> This requirement ensures that authentication mechanisms are designed to identify the actual user, not just the device they are using. Tying authentication to user identity reduces the risk of unauthorized access if the device is compromised or shared, improving the overall security of healthcare applications.

<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. User Identity Verification Testing:</b> <ul style="list-style-type: none"> <li>✓ Verify that the application uses authentication methods that confirm the identity of the end user rather than relying solely on device-based credentials.</li> </ul> </li> <li><b>2. Configuration Review:</b> <ul style="list-style-type: none"> <li>✓ Confirm that authentication configurations enforce user identity verification over device-only authentication.</li> </ul> </li> <li><b>3. Code Review for Authentication Logic:</b> <ul style="list-style-type: none"> <li>✓ Review the implementation of authentication logic to ensure user identity verification is prioritized over device-based authentication.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IAA-029]
<b>Requirement description:</b> The application must validate certificates by constructing a certification path, including status information, to an accepted trust anchor when utilizing PKI-based authentication.
<b>Source:</b> <ul style="list-style-type: none"> <li>✓ V-222550: The application, when utilizing PKI-based authentication, must validate certificates by constructing a certification path (which includes status information) to an</li> </ul>

accepted trust anchor. Design the application to construct a certification path to an accepted trust anchor when using PKI-based authentication [15].
<b>Priority:</b> Not described
<b>Rationale:</b> This requirement ensures the integrity and trustworthiness of PKI-based authentication by validating certificates against an accepted trust anchor. Constructing a certification path provides assurance that the certificate is authentic, unrevoked, and valid, reducing the risk of unauthorized access or malicious actors compromising the application.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<p><b>1. Certificate Path Validation Testing:</b></p> <ul style="list-style-type: none"> <li>✓ Verify that the application constructs a complete certification path for all certificates used in PKI-based authentication.</li> <li>✓ Confirm that the certification path includes status information and validation against a trusted anchor.</li> </ul> <p><b>2. Revocation Status Check:</b></p> <ul style="list-style-type: none"> <li>✓ Ensure the application checks certificate revocation status using CRLs (Certificate Revocation Lists) or OCSP (Online Certificate Status Protocol).</li> </ul> <p><b>3. Configuration Review:</b></p> <ul style="list-style-type: none"> <li>✓ Validate that the application allows the configuration of trust anchors and supports updates to trusted certificates.</li> </ul> <p><b>4. Code Review for PKI Validation Logic:</b></p> <ul style="list-style-type: none"> <li>✓ Review the implementation of PKI-based authentication and certificate validation mechanisms to ensure compliance with secure coding practices.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán

Juan Manuel Carrillo-de-Gea  
Joaquín Nicolás  
08/01/2026

**PUID:** [SECM-CAT-IIA-030]

**Requirement description:** The application must implement a local cache of revocation data to support certification path discovery and validation when network access to revocation information is unavailable during PKI-based authentication.

**Source:**

- ✓ V-222553: The application, for PKI-based authentication, must implement a local cache of revocation data to support path discovery and validation in case of the inability to access revocation information via the network. Implement a Certificate Revocation List (CRL) import process and configure the application to check the CRL if OCSP is not available [15].

**Priority:** Not described

**Rationale:** This requirement ensures the reliability of PKI-based authentication even in scenarios where network access to revocation information is interrupted. By maintaining a local cache of revocation data, the application can validate certificates against a Certificate Revocation List (CRL), providing an additional layer of security and reducing dependency on external networks.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

**1. CRL Import Process Validation:**

- ✓ Verify that the application supports importing and maintaining a local CRL for revocation data.

**2. Certificate Validation Testing:**

- ✓ Confirm that the application checks the CRL for certificate revocation status when OCSP is unavailable.
- ✓ Validate that certificates marked as revoked are rejected during authentication.

**3. Cache Implementation Review:**

- ✓ Ensure that the application's local cache of revocation data is securely stored and updated as per organizational policies.

<b>4. Code Review for Revocation Handling:</b> <ul style="list-style-type: none"><li>✓ Review the implementation of revocation data caching and validation to ensure compliance with secure coding practices.</li></ul>
<b>5. Failover Mechanism Testing:</b> <ul style="list-style-type: none"><li>✓ Validate that the application automatically switches to CRL-based validation when OCSP is unavailable.</li></ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IIA-031]
<b>Requirement description:</b> The application must map authenticated identities to individual user or group accounts during PKI-based authentication to ensure accurate association of certificate information.
<b>Source:</b> <ul style="list-style-type: none"><li>✓ V-222552: The application must map the authenticated identity to the individual user or group account for PKI-based authentication. Configure the application to map certificate information to individual users or group accounts or create a process for automatically determining the individual user or group based on certificate information provided in the logs [15].</li></ul>
<b>Priority:</b> Not described
<b>Rationale:</b> This requirement ensures that the application accurately associates authenticated identities with specific user or group accounts during PKI-based authentication. Proper mapping of certificate information enhances accountability, supports access control policies, and ensures compliance with organizational security standards.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>

**1. Identity Mapping Testing:**

- ✓ Verify that the application maps authenticated identities to specific user or group accounts using certificate information.
- ✓ Test scenarios to ensure accurate mapping for both individual users and groups.

**2. Automated Mapping Validation:**

- ✓ Confirm that the application can automatically determine user or group identities based on certificate information provided in the logs.

**3. Configuration Review:**

- ✓ Validate that the application allows configuration of mapping rules for associating certificates with users or groups.

**4. Code Review for Mapping Logic:**

- ✓ Review the implementation of identity mapping mechanisms to ensure compliance with secure coding practices and alignment with organizational policies.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda  
José L Fernández-Alemán  
Juan Manuel Carrillo-de-Gea  
Joaquín Nicolás  
08/01/2026

**PUID:** [SECM-CAT-IAA-032]

**Requirement description:** The application must validate certificates by constructing a certification path, including status information, to an accepted trust anchor during PKI-based authentication.

**Source:**

- ✓ V-222551: The application, when utilizing PKI-based authentication, must validate certificates by constructing a certification path (which includes status information) to an accepted trust anchor. Design the application to construct a certification path to an accepted trust anchor when using PKI-based authentication [15].

**Priority:** Not described

**Rationale:** Validating certificates through a constructed certification path ensures that PKI-based authentication is trustworthy and secure. This process guarantees that certificates are verified

against a trusted source, preventing the use of invalid or compromised certificates and protecting sensitive healthcare data from unauthorized access.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

**1. Certification Path Validation Testing:**

- ✓ Verify that the application constructs a full certification path for all certificates used during PKI-based authentication.
- ✓ Ensure that the certification path includes trust anchor validation and revocation status checks.

**2. Certificate Status Validation:**

- ✓ Confirm that the application checks the status of certificates using trusted revocation mechanisms such as CRLs or OCSP.

**3. Configuration Review:**

- ✓ Validate that the application allows the configuration of trust anchors and supports updates to maintain their reliability.

**4. Code Review for PKI Authentication:**

- ✓ Review the implementation of PKI-based authentication and certification path construction to ensure compliance with secure coding practices

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda

José L Fernández-Alemán

Juan Manuel Carrillo-de-Gea

Joaquín Nicolás

08/01/2026

<b>PUID:</b> [SECM-CAT-IAA-033]
<b>Requirement description:</b> The application must implement robust mechanisms to secure local user authentication and mitigate the risk of client-side bypass vulnerabilities, especially on jailbroken or compromised devices.
<b>Source:</b>
✓ 2. Local user authentication can lead to client-side bypass vulnerabilities. If the application stores data locally, the authentication routine can be bypassed on jailbroken devices through runtime manipulation or binary modification. If offline authentication is a compelling business requirement, consult additional guidance on preventing binary attacks against the mobile app [3].
<b>Priority:</b> Not described
<b>Rationale:</b> This requirement addresses the risks of local user authentication, which can be exploited through client-side attacks such as runtime manipulation or binary modification on compromised devices. Implementing secure mechanisms for local authentication ensures that the application's authentication routines cannot be easily bypassed, safeguarding sensitive healthcare data stored locally.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<b>1. Jailbroken/Rooted Device Testing:</b>
✓ Verify that the application detects jailbroken or rooted devices and implements appropriate countermeasures to protect local authentication mechanisms.
<b>2. Offline Authentication Validation:</b>
✓ If offline authentication is required, confirm that the application includes integrity checks to prevent runtime manipulation or binary modification attacks.
<b>3. Client-Side Bypass Testing:</b>
✓ Test scenarios to ensure local authentication cannot be bypassed through manipulation of client-side processes or binaries.
<b>4. Configuration Review:</b>

- ✓ Validate that authentication mechanisms are configurable to address organizational policies for offline and local data storage.

## 5. Code Review for Authentication Logic:

- ✓ Review the implementation of local authentication to ensure it adheres to secure coding practices and includes protections against binary attacks.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda

José L Fernández-Alemán

Juan Manuel Carrillo-de-Gea

Joaquín Nicolás

08/01/2026

**PUID:** [SECM-CAT-IAA-034]

**Requirement description:** The application must implement robust authentication and authorization mechanisms that protect user credentials, ensure access is limited to necessary data, and enforce server-side verification for all client-side controls.

**Source:**

- ✓ 164.308 Administrative safeguards.

(a) A covered entity or business associate must, in accordance with § 164.306:

(4)

(ii) Implementation specifications:

(B) Access authorization (Addressable).

Implement policies and procedures for granting access to electronic protected health information, for example, through access to a workstation, transaction, program, process, or other mechanism [9].

- ✓ Security Requirement 70 – Robustly Authenticating Users: The EHRI and all PoS systems connected to the EHRI must robustly authenticate users [10].
- ✓ Security: 27. The authorization and authentication mechanisms protect users' credentials and allow access to their data [29].
- ✓ Reinforce Authentication: 1. Developers should assume that all client-side authorization and authentication controls can be bypassed by malicious users. Server-side reinforcement of these controls is critical [3].
- ✓ Local and backend security controls: 1. Security controls should be applied locally in conjunction with backend controls. Resources needed for protected functions should only be downloaded if both local and backend controls are successfully verified [3].

✓ Security: The authorization and authentication mechanisms protect the users' credentials and gives access to their data. It limits access to data that is only necessary for the user [21].
<b>Priority:</b> Not described
<b>Rationale:</b> This requirement ensures that user authentication and authorization processes are secure and reliable by combining client-side and server-side controls. Protecting user credentials and ensuring access is limited to necessary data mitigates risks of unauthorized access and enhances the security of sensitive healthcare data.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<p><b>1. Authentication Robustness Testing:</b></p> <ul style="list-style-type: none"> <li>✓ Verify that authentication mechanisms protect user credentials from unauthorized access or tampering.</li> <li>✓ Ensure that multi-factor authentication (MFA) is implemented where required.</li> </ul> <p><b>2. Authorization Validation:</b></p> <ul style="list-style-type: none"> <li>✓ Confirm that access controls restrict users to only the data and resources necessary for their role.</li> </ul> <p><b>3. Server-Side Reinforcement Testing:</b></p> <ul style="list-style-type: none"> <li>✓ Validate that server-side controls are in place to enforce client-side authentication and authorization logic.</li> </ul> <p><b>4. Access Request Review:</b></p> <ul style="list-style-type: none"> <li>✓ Confirm that resources for protected functions are only accessible after successful verification of both local and backend controls.</li> </ul> <p><b>5. Code Review for Security Logic:</b></p> <ul style="list-style-type: none"> <li>✓ Review the implementation of authentication and authorization mechanisms to ensure secure coding practices.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described

<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda
José L Fernández-Alemán
Juan Manuel Carrillo-de-Gea
Joaquín Nicolás
08/01/2026

<b>PUID:</b> [SECM-CAT-IAA-035]
<b>Requirement description:</b> The application must implement network access control and role-based mechanisms to ensure users and administrators access only the resources required for their roles across all system components and architecture segments.
<b>Source:</b>
<ul style="list-style-type: none"><li>✓ 4.3.1. access control<ul style="list-style-type: none"><li>Network access control: firewalls, application, or user roles are used to limit access to the needed resources for a notional administrator or patient to use the system at all segments and service components within the build architecture [9].</li><li>✓ Test ID 1: Architecture accounts for multiple user roles and the access privileges assigned to each role [37].</li></ul></li></ul>
<b>Priority:</b> Not described
<b>Rationale:</b> This requirement ensures that access to application resources is restricted based on user roles and responsibilities, minimizing the risk of unauthorized access. By limiting access through firewalls, role-based controls, and application configurations, the system protects sensitive data and maintains compliance with healthcare security standards.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"><li><b>1. Role-Based Access Validation:</b><ul style="list-style-type: none"><li>✓ Verify that user roles are defined and each role has specific access privileges.</li><li>✓ Ensure that access to resources is limited based on the assigned role.</li></ul></li><li><b>2. Network Access Control Testing:</b></li></ol>

- ✓ Confirm that network-level controls, such as firewalls, restrict access to architecture segments based on role and access requirements.

### 3. User Privilege Testing:

- ✓ Validate that users can only access resources necessary for their role and are restricted from accessing unauthorized areas.

### 4. Code Review for Role-Based Logic:

- ✓ Review the implementation of role-based and network access control mechanisms to ensure compliance with secure coding practices.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda

José L Fernández-Alemán

Juan Manuel Carrillo-de-Gea

Joaquín Nicolás

08/01/2026

**PUID:** [SECM-CAT-IAA-036]

**Requirement description:** The application must avoid using spoofable values, such as device identifiers or geolocation data, for user authentication.

**Source:**

- ✓ Avoid Weak Patterns: 7. Avoid using spoofable values for user authentication, including device identifiers or geo-location [3].

**Priority:** Not described

**Rationale:** This requirement ensures that authentication mechanisms are secure by eliminating reliance on easily spoofed values like device identifiers and geo-location data. Using strong and reliable authentication factors reduces the risk of unauthorized access and enhances the overall security of sensitive healthcare applications.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

**1. Authentication Bypass Testing:**

- ✓ Simulate authentication attempts using spoofed device identifiers and geo-location data.
- ✓ Verify that these spoofed values do not bypass the authentication process.

**2. Alternate Authentication Factor Testing:**

- ✓ Test the implementation of secure authentication mechanisms such as multi-factor authentication (MFA) or public key infrastructure (PKI).
- ✓ Validate that device identifiers and geo-location data are not used as primary or fallback authentication factors.

**3. Code Analysis for Spoofable Inputs:**

- ✓ Conduct a static code analysis to identify and verify that device identifiers, geo-location data, or similar spoofable inputs are not included in the authentication logic.
- ✓ Confirm that only secure authentication factors are implemented.

**4. Configuration and Hardening Verification:**

- ✓ Ensure configuration files and authentication settings do not allow the use of device identifiers or geo-location data for authentication.

**5. Session Hijacking and Replay Testing:**

- ✓ Attempt to hijack or replay sessions using captured device identifiers or geo-location data.
- ✓ Verify that such attacks are mitigated through robust authentication and session validation.

**6. Penetration Testing for Authentication Mechanisms:**

- ✓ Perform penetration testing specifically targeting the authentication mechanism.
- ✓ Confirm that authentication cannot be bypassed or manipulated through spoofed inputs.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda  
José L Fernández-Alemán  
Juan Manuel Carrillo-de-Gea  
Joaquín Nicolás  
08/01/2026

**PUID:** [SECM-CAT-IIA-037]

**Requirement description:** The application must use WS-Security to protect messages with time stamps, including creation and expiration times, and sequence numbers to prevent replay attacks and ensure message integrity.

**Source:**

- ✓ V-222399: Messages protected with WS\_Security must use time stamps with creation and expiration times. Design and configure applications using WS-Security messages to use time stamps with creation and expiration times and sequence numbers [15].
- ✓ SR5: User Anonymity Each device must have a unique virtual identity which is known only to the user. It may be generated using a password known to the user and credentials on the SE. The adversary must not be able to use it for replay attacks or to find the actual identity [30].

**Priority:** Not described

**Rationale:** This requirement ensures that WS-Security messages include time stamps and sequence numbers to prevent replay attacks and validate message integrity. Proper implementation enhances the security of communication and prevents unauthorized access or message tampering in healthcare applications.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

**1. Message Time Stamp Validation:**

- ✓ Verify that each WS-Security message includes a creation and expiration time stamp.
- ✓ Confirm that the application has rejected expired messages.

**2. Sequence Number Testing:**

- ✓ Ensure that WS-Security messages include sequence numbers to prevent replay attacks.
- ✓ Validate that the application rejects messages with duplicate sequence numbers.

**3. Configuration Review:**

- ✓ Confirm that the application's WS-Security configuration enforces the use of time stamps and sequence numbers.

**4. Replay Attack Simulation:**

- ✓ Simulate a replay attack by resending a previously valid message with the same sequence number or expired time stamp.

- ✓ Verify that the application detects and rejects the attack.

## 5. Code Review for WS-Security Implementation:

- ✓ Analyze the code to ensure WS-Security is correctly implemented with time stamps, sequence numbers, and proper validation logic.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda

José L Fernández-Alemán

Juan Manuel Carrillo-de-Gea

Joaquín Nicolás

08/01/2026

**PUID:** [SECM-CAT-IIA-038]

**Requirement description:** The application must ensure that each asserting party provides unique assertion ID references for every SAML assertion to maintain assertion integrity and prevent replay attacks.

**Source:**

- ✓ V-222401: The application must ensure each unique asserting party provides unique assertion ID references for each SAML assertion. Design and configure each SAML assertion authority to use unique assertion identifiers [15].

**Priority:** Not described

**Rationale:** This requirement ensures that SAML assertions include unique assertion IDs for each asserting party to prevent replay attacks, ensure assertion integrity, and enhance trust in federated authentication systems. Proper implementation guarantees secure identification and tracking of SAML assertions across applications.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

### 1. Assertion ID Uniqueness Testing:

- ✓ Verify that each SAML assertion includes a unique assertion ID generated by the asserting party.

- |  |
|--|
| <ul style="list-style-type: none"> <li>✓ Confirm that duplicate assertion IDs are detected and rejected by the application.</li> </ul> |
|--|

## 2. Configuration Review:

- |   |
|---|
| <ul style="list-style-type: none"> <li>✓ Validate that the SAML assertion authority is configured to generate unique identifiers for each assertion.</li> </ul> |
|---|

## 3. Code Review for Assertion Handling:

- |   |
|---|
| <ul style="list-style-type: none"> <li>✓ Ensure the application includes logic to check the uniqueness of assertion IDs and handles invalid or duplicate IDs securely.</li> </ul> |
|---|

## 4. Integration Testing with Asserting Parties:

- |   |
|---|
| <ul style="list-style-type: none"> <li>✓ Test the SAML assertion process with multiple asserting parties to ensure unique assertion IDs are consistently generated and verified.</li> </ul> |
|---|

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda  
José L Fernández-Alemán  
Juan Manuel Carrillo-de-Gea  
Joaquín Nicolás  
08/01/2026

**PUID:** [SECM-CAT-IAA-039]

**Requirement description:** The application must validate the validity periods of all WS-Security tokens and SAML assertions to ensure secure communication and prevent the use of expired or invalid credentials.

**Source:**

- |  |
|--|
| <ul style="list-style-type: none"> <li>✓ V-222400: Validity periods must be verified on all application messages using WS-Security or SAML assertions. Design and configure the application to use validity periods, ensure validity periods are verified on all WS-Security token profiles and SAML Assertions [15].</li> </ul> |
|--|

**Priority:** Not described

**Rationale:** This requirement ensures that validity periods on WS-Security tokens and SAML assertions are enforced to prevent replay attacks and ensure the use of current and valid credentials during authentication and communication. Implementing this control protects the application from unauthorized access and malicious attacks leveraging expired tokens or assertions.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<b>1. Validity Period Testing:</b> <ul style="list-style-type: none"><li>✓ Verify that the application checks the validity periods of WS-Security tokens and SAML assertions during authentication and message processing.</li><li>✓ Confirm that expired tokens or assertions are rejected.</li></ul>
<b>2. Configuration Review:</b> <ul style="list-style-type: none"><li>✓ Ensure that validity period enforcement is enabled for all WS-Security and SAML assertion configurations.</li></ul>
<b>3. Code Review for Validity Logic:</b> <ul style="list-style-type: none"><li>✓ Analyze the implementation of validity period checks to confirm secure handling and proper enforcement.</li></ul>
<b>4. Integration Testing with Federated Services:</b> <ul style="list-style-type: none"><li>✓ Test validity period enforcement in scenarios involving federated services using WS-Security or SAML assertions to ensure compatibility and security.</li></ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IAA-040]
<b>Requirement description:</b> The application must use FIPS-approved random number generators to generate SessionIndex in SAML AuthnStatements and ensure encrypted assertions or equivalent confidentiality protections when assertion data passes through intermediaries.
<b>Source:</b> <ul style="list-style-type: none"><li>✓ V-222573: Applications making SAML assertions must use FIPS-approved random numbers in the generation of SessionIndex in the SAML element AuthnStatement. Configure the application to use a FIPS-validated cryptographic module [15].</li><li>✓ V-222402: The application must ensure encrypted assertions, or equivalent confidentiality protections are used when assertion data is passed through an intermediary, and</li></ul>

confidentiality of the assertion data is required when passing through the intermediary. Encrypt assertions or use equivalent confidentiality when sensitive assertion data is passed through an intermediary [15].
<b>Priority:</b> Not described
<b>Rationale:</b> This requirement ensures the secure generation of SAML SessionIndex using FIPS-approved random numbers to prevent predictability and enhance authentication security. Additionally, encrypting SAML assertions or providing equivalent confidentiality ensures the protection of sensitive assertion data from exposure when transmitted through intermediaries.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<p><b>1. SessionIndex Randomness Testing:</b></p> <ul style="list-style-type: none"> <li>✓ Verify that SessionIndex in SAML AuthnStatements is generated using a FIPS-approved random number generator.</li> <li>✓ Confirm that the random numbers used are unpredictable and meet FIPS standards.</li> </ul> <p><b>2. Assertion Encryption Validation:</b></p> <ul style="list-style-type: none"> <li>✓ Test that all SAML assertions containing sensitive data are encrypted when transmitted through intermediaries.</li> <li>✓ Ensure equivalent confidentiality protections are implemented if encryption is not used.</li> </ul> <p><b>3. Configuration Review:</b></p> <ul style="list-style-type: none"> <li>✓ Validate that the application is configured to use a FIPS-validated cryptographic module for all cryptographic operations involving SAML assertions.</li> </ul> <p><b>4. Code Review for SAML Security Logic:</b></p> <ul style="list-style-type: none"> <li>✓ Analyze the implementation of cryptographic functions and ensure compliance with FIPS standards and assertion confidentiality requirements.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán

Juan Manuel Carrillo-de-Gea  
Joaquín Nicolás  
08/01/2026

**PUID: [SECM-CAT-IIA-041]**

**Requirement description:** The application must implement the <NotOnOrAfter> condition in the <SubjectConfirmation> element of SAML assertions to enforce time-bound validity and prevent the reuse of expired assertions.

**Source:**

- ✓ V-222403: The application must use the NotOnOrAfter condition when using the SubjectConfirmation element in a SAML assertion. Design and configure the application to use the <NotOnOrAfter> condition when using the <SubjectConfirmation> element in a SAML assertion [15].

**Priority:** Not described

**Rationale:** This requirement ensures that SAML assertions are time-bound, enhancing security by limiting their validity to a specific duration. Implementing the <NotOnOrAfter> condition prevents unauthorized reuse of assertions after their expiration, mitigating replay attacks and maintaining secure authentication flows.

**Number of Children:** 0**Number of parents:** 0**Cycles:** 0**Number Audit:** 0**Child PUIDs:** Not described**Parent PUIDs:** Not described**Exclusion PUIDs:** Not described**Importance:** Not described**Current state:** To be determined**Verification method:** Demonstration/Analysis**Validation criteria:****1. Assertion Validity Testing:**

- ✓ Verify that all SAML assertions include the <NotOnOrAfter> condition in the <SubjectConfirmation> element.
- ✓ Confirm that the application has rejected expired assertions.

**2. Configuration Review:**

- ✓ Validate that the application's SAML configuration enforces the inclusion of <NotOnOrAfter> conditions in all assertions.

**3. Code Review for SAML Validation Logic:**

- ✓ Ensure the implementation properly enforces <NotOnOrAfter> conditions and handles expired assertions securely.

<b>4. Integration Testing:</b>
✓ Test the application's interaction with federated systems to ensure <NotOnOrAfter> conditions are consistently enforced across systems.
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IAA-042]
<b>Requirement description:</b> The application must implement the <NotBefore> and <NotOnOrAfter> conditions or the <OneTimeUse> element within the <Conditions> element in SAML assertions to enforce time-bound validity and restrict reuse.
<b>Source:</b>
✓ V-222404: The application must use both the NotBefore and NotOnOrAfter elements or OneTimeUse element when using the Conditions element in a SAML assertion. Design and configure the application to implement the use of the <NotBefore> and <NotOnOrAfter> or <OneTimeUse> when using the <Conditions> element in a SAML assertion [15].
<b>Priority:</b> Not described
<b>Rationale:</b> This requirement enhances the security of SAML assertions by ensuring time-based validity and preventing unauthorized reuse. Using <NotBefore> and <NotOnOrAfter> elements enforces strict temporal boundaries, while the <OneTimeUse> element ensures that assertions cannot be reused, mitigating risks such as replay attacks.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<b>1. Assertion Validity Period Testing:</b>
✓ Verify that all SAML assertions include <NotBefore> and <NotOnOrAfter> elements to define valid time windows.

- ✓ Confirm that assertions outside these time windows are rejected.

## 2. One-Time Use Testing:

- ✓ Validate that assertions with <OneTimeUse> elements are rejected after a single successful use.

## 3. Configuration Review:

- ✓ Ensure the application is configured to enforce the inclusion of <NotBefore>, <NotOnOrAfter>, or <OneTimeUse> conditions in all SAML assertions.

## 4. Code Review for Assertion Validation Logic:

- ✓ Analyze the implementation to verify proper handling of <Conditions> elements and enforcement of their restrictions.

## 5. Integration Testing:

- ✓ Test the application's handling of SAML assertions in federated environments to ensure consistent enforcement of <Conditions> elements.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda

José L Fernández-Alemán

Juan Manuel Carrillo-de-Gea

Joaquín Nicolás

08/01/2026

**PUID:** [SECM-CAT-IAA-043]

**Requirement description:** The application must ensure that only one <OneTimeUse> element is included in the <Conditions> element of a SAML assertion to maintain the integrity of single-use conditions.

**Source:**

- ✓ V-222405: The application must ensure if a OneTimeUse element is used in an assertion, there is only one of the same used in the Conditions element portion of an assertion. When using OneTimeUse elements in a SAML assertion only allow one, OneTimeUse element to be used in the conditions element of a SAML assertion [15].

**Priority:** Not described

**Rationale:** This requirement ensures the integrity of SAML assertions by enforcing the use of a single <OneTimeUse> element in the <Conditions> section. By restricting multiple <OneTimeUse> elements, the application prevents ambiguity or misuse, maintaining the assertion's security and functionality.

<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<b>1. SAML Assertion Validation:</b>
✓ Verify that only one <OneTimeUse> element is present in the <Conditions> element of each SAML assertion.
✓ Confirm that assertions with multiple <OneTimeUse> elements are rejected.
<b>2. Replay Protection Testing:</b>
✓ Test the behavior of assertions with <OneTimeUse> to ensure they cannot be reused after successful validation.
<b>3. Configuration Review:</b>
✓ Validate that the application is configured to enforce the restriction of a single <OneTimeUse> element within the <Conditions> section.
<b>4. Code Review for Assertion Logic:</b>
✓ Analyze the implementation to confirm that logic explicitly restricts multiple <OneTimeUse> elements and ensures proper handling of single-use assertions.
<b>5. Integration Testing:</b>
✓ Test SAML assertions across federated systems to ensure <OneTimeUse> enforcement is consistent and reliable.
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda
José L Fernández-Alemán
Juan Manuel Carrillo-de-Gea
Joaquín Nicolás
08/01/2026

<b>PUID:</b> [SECM-CAT-IAA-044]
<b>Requirement description:</b> The application must implement persistent authentication as an opt-in feature, ensuring it is not enabled by default to enhance user control and security.
<b>Source:</b>
✓ 8. Persistent authentication within mobile applications should be implemented as an opt-in and not enabled by default [3].
<b>Priority:</b> Not described
<b>Rationale:</b> This requirement ensures that persistent authentication, which allows users to remain logged in for extended periods, is enabled only with explicit user consent. By defaulting to non-persistent sessions, the application minimizes the risk of unauthorized access due to unattended devices or compromised sessions, improving security for sensitive healthcare applications.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<b>1. Default State Validation:</b>
✓ Verify that persistent authentication is disabled by default for all users.
<b>2. Opt-In Testing:</b>
✓ Confirm that users are explicitly prompted to opt in for persistent authentication and that consent is required.
<b>3. Configuration Review:</b>
✓ Validate that the application's configuration enforces non-persistent sessions as the default setting.
<b>4. Session Duration Testing:</b>
✓ Test that sessions revert to standard expiration policies if persistent authentication is not enabled.
<b>5. Code Review for Default Behavior:</b>
✓ Analyze the application's implementation to confirm that persistent authentication is explicitly disabled by default and requires user action to enable it.
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer

<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda
José L Fernández-Alemán
Juan Manuel Carrillo-de-Gea
Joaquín Nicolás
08/01/2026

<b>PUID:</b> [SECM-CAT-IAA-045]
<b>Requirement description:</b> The application must perform all authentication requests server-side and ensure that application data is only available on the mobile device after successful authentication.
<b>Source:</b>
✓ Avoid Weak Patterns: 3. Perform all authentication requests server-side, where possible. Upon successful authentication, application data will be loaded onto the mobile device, ensuring application data availability only after successful authentication [3].
<b>Priority:</b> Not described
<b>Rationale:</b> This requirement ensures that authentication is securely processed on the server to minimize client-side vulnerabilities. By loading application data only after successful authentication, the application mitigates risks such as unauthorized data access and enhances overall security, especially for sensitive healthcare applications.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<b>1. Server-Side Authentication Validation:</b>
✓ Verify that all authentication requests are processed on the server. ✓ Confirm that no sensitive authentication logic is executed on the client side.
<b>2. Data Access Testing:</b>
✓ Attempt to access application data before authentication is completed. ✓ Ensure that application data is only accessible after successful authentication.
<b>3. Configuration Review:</b>

- ✓ Validate that the application is configured to restrict data loading to authenticated sessions only.

#### 4. Penetration Testing:

- ✓ Confirm that the application rejects all unauthorized access attempts.

#### 5. Code Review for Authentication Logic:

- ✓ Analyze the implementation to ensure all authentication is handled server-side and no sensitive logic is exposed on the client

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda  
José L Fernández-Alemán  
Juan Manuel Carrillo-de-Gea  
Joaquín Nicolás  
08/01/2026

**PUID:** [SECM-CAT-IAA-046]

**Requirement description:** The application must implement secure password management procedures, prohibit the use of weak authentication methods such as 4-digit PINs, and ensure local authentication mechanisms follow platform best practices.

**Source:**

- ✓ 164.308 Administrative safeguards.
  - (a) A covered entity or business associate must, in accordance with § 164.306:
    - (5) (ii) Implementation specifications:
      - (C) Password management (Addressable)
        - Procedures for creating, changing, and safeguarding passwords [9].
- ✓ MASVS- AUTH -2: The app performs local authentication securely according to the platform best practices: Many apps allow users to authenticate via biometrics or a local PIN code. These authentication mechanisms need to be correctly implemented. Additionally, some apps might not have a remote endpoint, and rely fully on local app authentication [20].
- ✓ Avoid Weak Patterns: 9. Where possible, refrain from allowing users to provide 4-digit PIN numbers for authentication passwords [3].

**Priority:** Not described

**Rationale:** This requirement enhances application security by enforcing robust password management policies, avoiding weak authentication methods, and ensuring local authentication mechanisms are implemented securely. Following platform best practices minimizes risks such as brute force attacks, unauthorized access, and compromised user credentials, particularly in healthcare applications.

<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<b>1. Password Management Testing:</b>
✓ Verify that the application enforces secure password creation, storage, and modification procedures.
✓ Confirm that users are required to create strong passwords following organization-defined criteria.
<b>2. Authentication Method Validation:</b>
✓ Ensure the application does not allow 4-digit PINs or other weak methods for authentication.
✓ Test alternative authentication methods like biometrics and ensure they follow platform security standards.
<b>3. Configuration Review:</b>
✓ Validate that password policies, such as complexity requirements and expiration settings, are correctly implemented in the application.
<b>4. Local Authentication Testing:</b>
✓ Test the local authentication mechanisms, ensuring that they are secure and follow platform best practices.
✓ Confirm that sensitive data is not accessible without successful authentication.
<b>5. Code Review for Authentication Logic:</b>
✓ Analyze the implementation to ensure adherence to password management procedures and the exclusion of weak authentication methods.
<b>6. Penetration Testing:</b>
✓ Simulate brute force attacks to test the strength of authentication mechanisms and ensure compliance with secure password policies.
<b>Requested by:</b> The organization

<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IAA-047]
<b>Requirement description:</b> The application must enforce a minimum password lifetime of 24 hours to prevent users from bypassing password history requirements by making frequent password changes.
<b>Source:</b>
✓ V-222544: The application must enforce 24 hours/1 day as the minimum password lifetime. Configure the application to have a minimum password lifetime of 24 hours [15].
<b>Priority:</b> Not described
<b>Rationale:</b> Enforcing a minimum password lifetime prevents users from bypassing password history requirements by rapidly cycling through passwords. This measure ensures that password policies are effective in maintaining security, particularly in applications managing sensitive data such as healthcare records.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<b>1. Password Policy Validation:</b>
✓ Verify that the application enforces a minimum password lifetime of 24 hours for all user accounts.
<b>2. Password Cycling Prevention Test:</b>
✓ Attempt to change a password multiple times within a 24-hour period. ✓ Confirm that the application restricts password changes within the minimum lifetime window.
<b>3. Configuration Review:</b>

- ✓ Ensure the application configuration includes a setting for enforcing the 24-hour minimum password lifetime.

#### 4. Code Review for Password Policy Logic:

- ✓ Analyze the code to ensure proper implementation of the minimum password lifetime rule.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda

José L Fernández-Alemán

Juan Manuel Carrillo-de-Gea

Joaquín Nicolás

08/01/2026

**PUID:** [SECM-CAT-IAA-048]

**Requirement description:** The application must enforce a maximum password lifetime of 60 days to ensure users regularly update their passwords and maintain strong account security.

**Source:**

- ✓ V-222545: The application must enforce a 60-day maximum password lifetime restriction. Configure the application to have a maximum password lifetime of 60 days [15].
- ✓ Regular password update [26].

**Priority:** Not described

**Rationale:** Implementing a maximum password lifetime of 60 days ensures that passwords are periodically updated, reducing the likelihood of unauthorized access due to compromised credentials. This practice is especially critical in securing applications that handle sensitive data, such as healthcare information.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

#### 1. Password Policy Validation:

- ✓ Verify that the application enforces a maximum password lifetime of 60 days for all user accounts.

## **2. Password Expiry Notification Test:**

- ✓ Confirm that users are notified in advance of their password expiration, allowing sufficient time for password updates.

## **3. Password Expiration Test:**

- ✓ Attempt to log in with an expired password after 60 days.
- ✓ Ensure the application prevents access and prompts the user to reset their password.

## **4. Configuration Review:**

- ✓ Validate that the application configuration includes settings to enforce the 60-day maximum password lifetime.

## **5. Code Review for Expiry Logic:**

- ✓ Analyze the code to confirm the proper implementation of password expiration rules.

## **6. Integration Testing with Identity Management Systems:**

- ✓ Test the password lifetime enforcement in scenarios involving federated authentication or external identity providers to ensure consistency.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda  
José L Fernández-Alemán  
Juan Manuel Carrillo-de-Gea  
Joaquín Nicolás  
08/01/2026

**PUID:** [SECM-CAT-IAA-049]

**Requirement description:** The application must enforce password complexity by requiring at least 15 characters, including one upper-case letter, one lower-case letter, one numeric character, and one special character, to ensure robust user authentication for accessing personal mobile health data.

**Source:**

- ✓ Use a password or other user authentication: Users use a combination of private and public keys that they must access for personal mobile health data [32].

<ul style="list-style-type: none"> <li>✓ V-222540: The application must enforce password complexity by requiring that at least one special character be used. Configure the application to require at least one special character in the password [15].</li> <li>✓ V-222539: The application must enforce password complexity by requiring that at least one numeric character be used. Configure the application to require at least one numeric character in the password [15].</li> <li>✓ V-222538: The application must enforce password complexity by requiring that at least one lower-case character be used. Configure the application to require at least one lower-case character in the password [15].</li> <li>✓ V-222537: The application must enforce password complexity by requiring that at least one upper-case character be used. Configure the application to require at least one upper-case character in the password [15].</li> <li>✓ V-222536: The application must enforce a minimum 15-character password length. Configure the application to require 15 characters in the password [15].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> This requirement ensures robust user authentication by enforcing strong password complexity rules. These measures protect personal mobile health data from unauthorized access and enhance the overall security posture of the application.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<p><b>1. Password Complexity Testing:</b></p> <ul style="list-style-type: none"> <li>✓ Attempt to set passwords that do not meet the complexity requirements (e.g., missing a special character, numeric character, upper-case letter, or lower-case letter) and verify that they are rejected.</li> </ul> <p><b>2. Password Length Validation:</b></p> <ul style="list-style-type: none"> <li>✓ Attempt to set a password shorter than 15 characters and confirm that it is rejected.</li> </ul> <p><b>3. User Authentication Testing:</b></p>

- ✓ Verify that passwords meeting the complexity and length requirements are accepted during account creation and login processes.

#### **4. Configuration Review:**

- ✓ Confirm that the application is configured to enforce all password complexity rules, including the use of special characters, numeric characters, upper-case and lower-case letters, and a minimum of 15 characters.

#### **5. Code Review for Authentication Logic:**

- ✓ Analyze the code to verify the implementation of password complexity validation and secure storage practices.

#### **6. Penetration Testing:**

- ✓ Attempt brute force attacks to verify the effectiveness of the password complexity policy in mitigating unauthorized access.

<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IAA-050]
<b>Requirement description:</b> The application must enforce a policy requiring at least eight characters to be changed when passwords are updated, ensuring password uniqueness and reducing the risk of reuse.
<b>Source:</b> ✓ V-222541: The application must require the change of at least 8 of the total number of characters when passwords are changed. Configure the application to require the change of at least 8 characters in the password when passwords are changed [15].
<b>Priority:</b> Not described
<b>Rationale:</b> Enforcing a requirement to change at least eight characters in a password during updates ensures that new passwords are significantly different from previous ones. This policy enhances security by reducing the likelihood of password reuse or easy guessing, particularly in applications managing sensitive data such as healthcare records.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0

<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<b>1. Password Update Validation:</b>
✓ Attempt to update a password by changing fewer than eight characters and confirm that the application rejects the update.
<b>2. Password Uniqueness Testing:</b>
✓ Test the new password against the previously used password to ensure at least eight characters have been changed.
<b>3. Configuration Review:</b>
✓ Verify that the application's password update configuration enforces the requirement to change at least eight characters.
<b>4. Code Review for Update Logic:</b>
✓ Analyze the implementation to confirm that the password change logic checks for a minimum of 8-character differences.
<b>5. Penetration Testing:</b>
✓ Attempt to reuse passwords by modifying fewer than eight characters and verify that the application prevents this.
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IAA-051]
<b>Requirement description:</b> The application must prohibit password reuse for at least the last five passwords to ensure strong password security and mitigate risks associated with recycled credentials.

<b>Source:</b>
✓ V-222546: The application must prohibit password reuse for a minimum of five generations. Configure the application to prohibit password reuse for up to 5 passwords [6].
<b>Priority:</b> Not described
<b>Rationale:</b> This requirement ensures that users cannot reuse their previous five passwords, reducing the risk of compromised credentials being reused and enhancing overall password security. Enforcing a password history policy strengthens the application's defense against unauthorized access.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<b>1. Password History Validation:</b>
✓ Attempt to reuse any of the last five passwords during a password change and confirm that the application rejects the attempt.
<b>2. Configuration Review:</b>
✓ Verify that the application's configuration enforces a five-password history policy for all user accounts.
<b>3. Code Review for Password History Logic:</b>
✓ Analyze the implementation to ensure that a user's last five passwords are stored securely and used for validation against reuse.
<b>4. User Notification Testing:</b>
✓ Ensure that users are informed if their attempted password is among the last five used and are prompted to create a unique password.
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás

08/01/2026

<b>PUID:</b> [SECM-CAT-IAA-052]
<b>Requirement description:</b> The application must support the use of temporary passwords for system logons, requiring an immediate change to a permanent password upon first use.
<b>Source:</b>
✓ V-222547: The application must allow the use of a temporary password for system logons with an immediate change to a permanent password. Configure the application to specify when a password is temporary and change the temporary password on the first use [15].
<b>Priority:</b> Not described
<b>Rationale:</b> Temporary passwords provide a secure mechanism for initial or recovery access to the application. Requiring immediate replacement with a permanent password ensures that temporary credentials cannot be exploited for prolonged unauthorized access, strengthening overall security.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<b>1. Temporary Password Assignment:</b>
✓ Verify that the application allows the assigning of temporary passwords and marks them as temporary in the system.
<b>2. Mandatory Password Change Testing:</b>
✓ Log in with a temporary password and confirm that the application enforces an immediate password change before granting access to application resources.
<b>3. Configuration Review:</b>
✓ Confirm that the application configuration includes settings to enable temporary passwords and enforce their immediate replacement.
<b>4. Code Review for Temporary Password Logic:</b>
✓ Analyze the implementation to ensure proper marking, expiration, and handling of temporary passwords.

**5. Security Testing:**

- ✓ Attempt to use a temporary password for multiple logins without changing it and confirm that the application restricts such behavior.

**6. User Notification Testing:**

- ✓ Confirm that users are informed of the temporary nature of the password and the requirement to change it immediately.

**Requested by:** The organization**Responsible:** Developer**Configurable value:** Not described**Version history:** v1.0**Author and date:**

Carlos M. Mejía-Granda  
 José L Fernández-Alemán  
 Juan Manuel Carrillo-de-Gea  
 Joaquín Nicolás  
 08/01/2026

**PUID:** [SECM-CAT-IIA-053]
**Requirement description:** The application must ensure that passwords can only be changed by the associated user or an administrator to prevent unauthorized modifications.
**Source:**

- ✓ V-222548: The application password must not be changeable by users other than the administrator or the user with which the password is associated [15].

**Priority:** Not described
**Rationale:** Restricting password changes to the associated user or an administrator ensures that unauthorized individuals cannot modify passwords, protecting user accounts from potential compromise and maintaining the integrity of access control mechanisms.
**Number of Children:** 0**Number of parents:** 0**Cycles:** 0**Number Audit:** 0**Child PUIDs:** Not described**Parent PUIDs:** Not described**Exclusion PUIDs:** Not described**Importance:** Not described**Current state:** To be determined**Verification method:** Demonstration/Analysis**Validation criteria:****1. Access Control Testing:**

- ✓ Verify that only the associated user or an administrator can change an account password.

<b>2. Unauthorized Access Simulation:</b>
✓ Attempt to change another user's password without proper administrative privileges and ensure that the application denies the action.
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IAA-054]
<b>Requirement description:</b> The application must enforce strong authentication mechanisms, such as CAC, to establish non-local maintenance and diagnostic sessions to ensure secure access.
<b>Source:</b>
✓ V-222565: The application must employ strong authenticators in the establishment of non-local maintenance and diagnostic sessions. Configure the application to use strong authentication (CAC) when accessing the application for maintenance purposes [15].
<b>Priority:</b> Not described
<b>Rationale:</b> Strong authentication mechanisms are essential for securing non-local maintenance and diagnostic sessions. Requiring methods such as CAC ensures that only authorized personnel can access the application remotely, reducing the risk of unauthorized access and potential security breaches.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<b>1. Authentication Method Validation:</b>
✓ Verify that non-local maintenance and diagnostic sessions require the use of strong authentication mechanisms, such as CAC.
<b>2. Remote Access Testing:</b>

- ✓ Attempt to establish a non-local maintenance session without using strong authentication and ensure that the application denies access.

### 3. Configuration Review:

- ✓ Confirm that the application's configuration mandates the use of CAC or equivalent strong authentication methods for remote sessions.

### 4. Code Review for Authentication Logic:

- ✓ Analyze the implementation to ensure proper enforcement of strong authentication requirements during remote access.

### 5. Policy Compliance Review:

- ✓ Validate that the authentication mechanisms align with organizational policies and industry standards for secure remote maintenance.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda

José L Fernández-Alemán

Juan Manuel Carrillo-de-Gea

Joaquín Nicolás

08/01/2026

**PUID:** [SECM-CAT-IAA-055]

**Requirement description:** The application must implement authorization security controls independent of authentication mechanisms to ensure user permissions are properly verified and not solely reliant on authentication.

**Source:**

- ✓ 2.13. Authentication should not be used as a replacement of authorization security controls. Authorization verifies the permissions of a user and presupposes strong authentication [16].

**Priority:** Not described

**Rationale:** Authentication confirms a user's identity, while authorization determines their permissions. Relying solely on authentication without robust authorization controls can lead to unauthorized access to sensitive functions or data. Separating these controls ensures a more secure application, particularly in environments handling sensitive information such as healthcare data.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<b>1. Authentication and Authorization Separation Test:</b>
<ul style="list-style-type: none"><li>✓ Verify that user authentication does not automatically grant access to restricted resources without separate authorization checks.</li></ul>
<b>2. Role-Based Access Testing:</b>
<ul style="list-style-type: none"><li>✓ Test various user roles to ensure that permissions are applied correctly based on the user's authorization level.</li></ul>
<b>3. Configuration Review:</b>
<ul style="list-style-type: none"><li>✓ Confirm that the application has distinct configuration settings for authentication and authorization controls.</li></ul>
<b>4. Code Review for Authorization Logic:</b>
<ul style="list-style-type: none"><li>✓ Analyze the implementation to ensure authorization is enforced independently of authentication.</li></ul>
<b>5. Access Control Validation:</b>
<ul style="list-style-type: none"><li>✓ Attempt to access restricted functions or data with valid authentication but without proper authorization, ensuring access is denied.</li></ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

**PUID:** [SECM-CAT-IAA-056]

**Requirement description:** The application must clear sensitive data, reset the application state, and terminate server-side sessions upon session termination, backgrounding, or user logout, ensuring no residual data remains accessible.

**Source:**

- ✓ 2.15. Clear any maintained sensitive data on session termination. Reset the application state and request for user re-authentication [16].
- ✓ 2.16. Clear any maintained sensitive data and attempt to also terminate any server-side session after application state change (e.g., termination, backgrounding). Consider a user request for application termination as a request to logout [16].
- ✓ 2.18. For platforms that support application component history stack (e.g., Android), always clear the stack on session or app termination and user's request to logout [16].

**Priority:** Not described

**Rationale:** Clearing sensitive data and terminating sessions upon logout or application termination reduces the risk of unauthorized access to residual data. Proper management of server-side and client-side session states enhances security, especially for applications that handle sensitive information such as healthcare records.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

**1. Session Termination Testing:**

- ✓ Verify that sensitive data is cleared and the application state is reset upon session termination or user logout.

**2. Server-Side Session Management:**

- ✓ Confirm that server-side sessions are terminated when the application is terminated or when the user logs out.

**3. Backgrounding Data Handling:**

- ✓ Test the application by moving it to the background and verify that no sensitive data is accessible without re-authentication.

**4. History Stack Validation:**

- ✓ On platforms like Android, verify that the application clears the component history stack upon session termination, logout, or application termination.

### 5. Code Review for Session Management Logic:

- ✓ Analyze the code to ensure proper implementation of data clearing, state resetting, and session termination mechanisms.

### 6. Penetration Testing:

- ✓ Attempt to access sensitive data after session termination or application closure, ensuring that no residual data is accessible.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda

José L Fernández-Alemán

Juan Manuel Carrillo-de-Gea

Joaquín Nicolás

08/01/2026

**PUID:** [SECM-CAT-IAA-057]

**Requirement description:** The application must provide a logout function that terminates the authenticated session, invalidates the session on the server side, and destroys session ID cookies upon logoff or application closure.

**Source:**

- ✓ 2.14. Apps that support user authentication must have a logout function which terminates the authenticated session. Upon logout, session should also be invalidated on the server side [16].
- ✓ V-222578: The application must destroy the session ID value and/or cookie on logoff or browser close. Configure the application to destroy session ID cookies once the application session has terminated [15].

**Priority:** Not described

**Rationale:** Providing a robust logout function ensures that authenticated sessions are properly terminated, minimizing the risk of unauthorized access from residual sessions. Destroying session IDs and invalidating server-side sessions enhances application security, especially for applications handling sensitive information like healthcare data.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<b>1. Logout Functionality Test:</b> <ul style="list-style-type: none"><li>✓ Verify that the application includes a logout function that terminates the user's authenticated session.</li></ul>
<b>2. Session Validation Verification:</b> <ul style="list-style-type: none"><li>✓ Confirm that server-side sessions are invalidated upon user logout or application closure.</li></ul>
<b>3. Session ID and Cookie Deletion Test:</b> <ul style="list-style-type: none"><li>✓ Attempt to reuse session cookies after logout or application closure and confirm that they have been destroyed.</li></ul>
<b>4. Configuration Review:</b> <ul style="list-style-type: none"><li>✓ Validate that the application configuration ensures proper destruction of session IDs and cookies upon logout.</li></ul>
<b>5. Code Review for Session Termination Logic:</b> <ul style="list-style-type: none"><li>✓ Analyze the implementation to ensure session termination and cookie destruction are correctly enforced.</li></ul>
<b>6. Penetration Testing:</b> <ul style="list-style-type: none"><li>✓ Simulate attempts to exploit residual session IDs or cookies after logout to verify that they cannot be reused.</li></ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

**PUID:** [SECM-CAT-IAA-058]

**Requirement description:** The application must implement robust authentication and authorization mechanisms to protect electronic protected health information (ePHI), ensuring that only authorized users and devices can access the system and unknown devices are appropriately challenged.

**Source:**

- ✓ 164.308 Administrative safeguards.

(a) A covered entity or business associate must, in accordance with § 164.306:

(4)

(ii) Implementation specifications:

(B) Access authorization (Addressable).

Implement policies and procedures for granting access to electronic protected health information, for example, through access to a workstation, transaction, program, process, or other mechanism [9].

- ✓ Security: 27. The authorization and authentication mechanisms protect users' credentials and allow access to their data [29].
- ✓ 4.3.4 Person or Entity Authentication

NAC and application person or entity authentication – at each point where a typical patient, provider, or health IT administrator must access a network or information, the person or device entity is challenged by using strong authentication methods [37].

- ✓ Test ID 3: Unknown devices are challenged when attempting to connect. Unknown devices are unable to connect to the EHR system [37]
- ✓ Authentication Implementation: The system must implement authentication mechanisms to control access to protected assets such as user data, app functionality, and other resources [18].
- ✓ MASVS-AUTH-1: The app uses secure authentication and authorization protocols and follows the relevant best practices: Most apps connecting to a remote endpoint require user authentication and also enforce some kind of authorization. While the enforcement of these mechanisms must be on the remote endpoint, the apps also have to ensure that it follows all the relevant best practices to ensure a secure use of the involved protocols [20].
- ✓ Test ID 19: The application accepts connections from only those devices hardened in compliance with security policy [37].

**Priority:** Not described

**Rationale:** Strong authentication and authorization mechanisms are essential for safeguarding sensitive healthcare data, including ePHI. By challenging unknown devices and enforcing secure protocols, the system minimizes the risk of unauthorized access and ensures compliance with healthcare security policies and standards.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<b>1. Authentication Validation:</b> <ul style="list-style-type: none"><li>✓ Test authentication mechanisms to confirm that only valid credentials allow access to the system.</li></ul>
<b>2. Authorization Testing:</b> <ul style="list-style-type: none"><li>✓ Verify that users can only access data and functionality that is consistent with their authorization levels.</li></ul>
<b>3. Device Challenge Verification:</b> <ul style="list-style-type: none"><li>✓ Attempt to connect with an unknown or non-compliant device and confirm that the system appropriately challenges and blocks the connection.</li></ul>
<b>4. Code Review for Authentication Protocols:</b> <ul style="list-style-type: none"><li>✓ Analyze the code to ensure that robust authentication and authorization protocols are implemented.</li></ul>
<b>5. Penetration Testing:</b> <ul style="list-style-type: none"><li>✓ Simulate attempts to bypass authentication and authorization mechanisms to verify their robustness.</li></ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IAA-059]
<b>Requirement description:</b> The application must implement and manage access rights to ensure that electronically protected health information (ePHI) is accessible only to authorized users or software programs and that access rights are reviewed, modified, and removed in compliance with organizational policies.
<b>Source:</b>

<ul style="list-style-type: none"> <li>✓ 5.18 Access rights: Access rights to information and other associated assets should be provisioned, reviewed, modified and removed in accordance with the organization's topic-specific policy on and rules for access control. It could be amplified in detail, review guidance section [6].</li> <li>✓ § 164.312 Technical safeguards.           <ul style="list-style-type: none"> <li>(a) (1) Standard: Access control. Implement technical policies and procedures for electronic information systems that maintain electronic protected health information to allow access only to those persons or software programs that have been granted access rights as specified in § 164.308(a)(4) [9]</li> </ul> </li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Managing and enforcing access rights is crucial for safeguarding ePHI and other sensitive assets. Ensuring that access is granted only to authorized individuals and regularly reviewing and updating access rights minimizes the risk of unauthorized access and helps maintain compliance with regulatory requirements.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<p><b>1. Access Provisioning Test:</b></p> <ul style="list-style-type: none"> <li>✓ Confirm that access rights are provisioned based on organizational policies and only granted to authorized users or software programs.</li> </ul> <p><b>2. Access Review Process Validation:</b></p> <ul style="list-style-type: none"> <li>✓ Verify that the application supports regular reviews of user and system access rights and that any necessary modifications are applied promptly.</li> </ul> <p><b>3. Access Revocation Testing:</b></p> <ul style="list-style-type: none"> <li>✓ Simulate user account termination or role changes and confirm that access rights are revoked or updated in accordance with policies.</li> </ul> <p><b>4. Configuration Review:</b></p> <ul style="list-style-type: none"> <li>✓ Ensure the application's access control configuration aligns with the organization's access management policies.</li> </ul>

**5. Code Review for Access Control Logic:**

- ✓ Analyze the implementation of access control mechanisms to ensure compliance with organizational policies.

**6. Penetration Testing:**

- ✓ Attempt to bypass access controls and confirm that unauthorized access to sensitive information is prevented.

**Requested by:** The organization**Responsible:** Developer**Configurable value:** Not described**Version history:** v1.0**Author and date:**

Carlos M. Mejía-Granda  
 José L Fernández-Alemán  
 Juan Manuel Carrillo-de-Gea  
 Joaquín Nicolás  
 08/01/2026

**PUID:** [SECM-CAT-IAA-060]

**Requirement description:** The application must implement federated identity management and modern authentication methods, such as passkeys and Credential Manager, to securely authenticate users and protect access to private data and sensitive operations.

**Source:**

- ✓ Authentication Use federated identity providers: Credential Manager supports federated authentication providers such as Sign in with Google. [18].
- ✓ Data Handling Practices: Access to private data must require user authentication, utilizing modern authentication methods such as passkeys and Credential Manager [18].
- ✓ MASVS-AUTH-3: The app secures sensitive operations with additional authentication: Some additional form of authentication is often desirable for sensitive actions inside the app. This can be done in different ways (biometric, pin, MFA code generator, email, deep links, etc) and they all need to be implemented securely [20].

**Priority:** Not described

**Rationale:** Federated identity management streamlines authentication while maintaining security by leveraging trusted identity providers. Requiring modern authentication methods ensures strong protection of private data and supports secure access for sensitive operations, reducing the risk of unauthorized access.

**Number of Children:** 0**Number of parents:** 0**Cycles:** 0**Number Audit:** 0**Child PUIDs:** Not described**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

**1. Federated Identity Testing:**

- ✓ Verify that the application integrates with federated identity providers, such as Sign in with Google, for user authentication.

**2. Modern Authentication Validation:**

- ✓ Confirm that modern authentication methods, such as passkeys and Credential Manager, are implemented for private data access.

**3. Sensitive Operation Authentication Test:**

- ✓ Test that additional authentication, such as biometrics or MFA, is required for sensitive operations within the application.

**4. Configuration Review:**

- ✓ Ensure that the application's configuration enforces federated identity and modern authentication requirements.

**5. Code Review for Authentication Mechanisms:**

- ✓ Analyze the implementation of federated identity and modern authentication methods to ensure compliance with security best practices.

**6. Penetration Testing:**

- ✓ Attempt to bypass authentication mechanisms for private data and sensitive operations, ensuring that they are robust against attacks.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda

José L Fernández-Alemán

Juan Manuel Carrillo-de-Gea

Joaquín Nicolás

08/01/2026

**PUID:** [SECM-CAT-IAA-061]

**Requirement description:** The application must implement asymmetric cryptography for authentication and authorization, ensuring private keys are securely generated and stored using platform-supported secure hardware, such as a Trusted Execution Environment (TEE) or Secure Element (SE).

**Source:**

- ✓ Consider using asymmetric cryptography for authentication and authorization purposes. Generate and use the private key directly within a platform-supported secure hardware (e.g., Trusted Execution Environment (TEE), Secure Element (SE)) [16].
- ✓ 6.4. Avoid using third-party libraries that contain main processor-only cryptographic implementations. Prefer using cryptographic framework provided by a platform supported secure hardware (e.g. TEE, SE) [16].

**Priority:** Not described

**Rationale:** Leveraging asymmetric cryptography within secure hardware ensures robust protection of sensitive keys against tampering or unauthorized access. This approach enhances the security of authentication and authorization processes by isolating cryptographic operations from the main processor.

**Number of Children:** 0**Number of parents:** 0**Cycles:** 0**Number Audit:** 0**Child PUIDs:** Not described**Parent PUIDs:** Not described**Exclusion PUIDs:** Not described**Importance:** Not described**Current state:** To be determined**Verification method:** Demonstration/Analysis**Validation criteria:****1. Key Generation Testing:**

- ✓ Verify that private keys are generated and stored within a Trusted Execution Environment (TEE) or Secure Element (SE).

**2. Cryptographic Operation Validation:**

- ✓ Test cryptographic operations to confirm that they utilize platform-supported secure hardware.

**3. Third-Party Library Review:**

- ✓ Ensure no third-party libraries are used for cryptographic implementations unless they are compatible with secure hardware.

**4. Configuration Inspection:**

- ✓ Verify the application configuration enforces the use of TEE or SE for cryptographic operations.

### 5. Code Review for Cryptographic Implementation:

- ✓ Analyze the code to ensure cryptographic operations are securely implemented, and hardware isolation is applied.

### 6. Penetration Testing:

- ✓ Simulate attempts to access private keys or intercept cryptographic operations, confirming the security provided by TEE or SE.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda

José L Fernández-Alemán

Juan Manuel Carrillo-de-Gea

Joaquín Nicolás

08/01/2026

**PUID:** [SECM-CAT-IAA-062]

**Requirement description:** The application must minimize repeated credential requests by implementing time-bounded, revocable authorization tokens secured in transit using TLS and adhering to the latest authorization standards, such as OAuth 2.0.

**Source:**

- ✓ Minimize Credential Requests: Avoid frequent credential requests. Implement the use of authorization tokens to reduce the need for repeated entry of user credentials. Ensure that only the minimum necessary credential information is requested for authentication and authorization [18].
- ✓ 2.20. Instead of passwords consider using longer term authorization tokens that can be securely stored on the device (as per the OAuth model8). Secure the tokens in transit (using TLS). Tokens can be issued by the backend service after verifying the user credentials initially. The tokens should be time bounded to the specific service as well as revocable (if possible server side), thereby minimizing the damage in loss scenarios. Use the latest versions of the authorization standards (such as OAuth 2.0). Make sure that these tokens expire as frequently as practicable [16].

**Priority:** Not described

**Rationale:** Replacing frequent credential requests with securely stored and time-bounded authorization tokens reduces the attack surface for credential compromise while enhancing user experience. Utilizing OAuth 2.0 and TLS ensures modern, secure practices for authentication and authorization.

**Number of Children:** 0

<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<b>1. Token Implementation Testing:</b>
✓ Verify that the application uses authorization tokens instead of frequent credential requests and ensures token storage is secure.
<b>2. Time-Bound Token Validation:</b>
✓ Test that tokens expire as configured and cannot be reused after expiration.
<b>3. Revocation Mechanism Validation:</b>
✓ Verify that tokens can be revoked server-side to prevent unauthorized access if a token is compromised.
<b>4. Secure Transmission Testing:</b>
✓ Confirm that all token exchanges occur over secure channels using TLS.
<b>5. OAuth 2.0 Compliance Validation:</b>
✓ Ensure the implementation adheres to the OAuth 2.0 standard, including token issuance and renewal processes.
<b>6. Configuration Review:</b>
✓ Inspect application configurations to confirm proper token management, including expiration and secure storage.
<b>7. Penetration Testing:</b>
✓ Simulate attempts to intercept or misuse tokens, ensuring robust token security mechanisms are in place.
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>

Carlos M. Mejía-Granda  
José L Fernández-Alemán  
Juan Manuel Carrillo-de-Gea  
Joaquín Nicolás  
08/01/2026

**PUID:** [SECM-CAT-IAA-063]

**Requirement description:** The application must accept FICAM-approved third-party credentials and conform to FICAM-issued technical profiles for services that rely on external identity providers.

**Source:**

- ✓ V-222559: The application must accept FICAM-approved third-party credentials. Configure applications intended to be accessible to non-federal government agencies to use FICAM-approved third-party credentials [15].
- ✓ The application must conform to FICAM-issued profiles. Configure the application to conform to FICAM-issued technical profiles when providing services that rely on external (Federal Government) identity providers [15].

**Priority:** Not described

**Rationale:** Replacing frequent credential requests with securely stored and time-bounded authorization tokens reduces the attack surface for credential compromise while enhancing user experience. Utilizing OAuth 2.0 and TLS ensures modern, secure practices for authentication and authorization.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

**1. Credential Acceptance Test:**

- ✓ Confirm that the application successfully accepts and validates FICAM-approved third-party credentials.

**2. FICAM Profile Conformance Test:**

- ✓ Verify that the application implements and adheres to FICAM-issued technical profiles for identity management.

**3. Configuration Inspection:**

- ✓ Ensure the application configuration enforces the acceptance of FICAM-compliant credentials.

**4. Interoperability Validation:**

- ✓ Test the integration with external Federal Government identity providers to ensure seamless interoperability.

**5. Code Review for Credential Handling:**

- ✓ Analyze the implementation of credential handling to ensure compliance with FICAM standards.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda

José L Fernández-Alemán

Juan Manuel Carrillo-de-Gea

Joaquín Nicolás

08/01/2026

**PUID:** [SECM-CAT-IAA-064]

**Requirement description:** The application must implement robust account and credential management processes, including secure storage, transmission, and handling of user credentials, password management mechanisms, and automated account management functions.

**Source:**

- ✓ 164.308 Administrative safeguards.
  - (a) A covered entity or business associate must, in accordance with § 164.306:
    - (5) (ii) Implementation specifications:
      - (C) Password management (Addressable)
        - Procedures for creating, changing, and safeguarding passwords [9].
  - ✓ Security: 24. It has password management mechanisms [29].
  - ✓ Minimize Use AccountManager for Service Integration: Implement AccountManager to securely connect to cloud-based services that can be accessed by multiple applications. Ensure that passwords are not stored on the device, relying on AccountManager for handling credentials securely [18].
  - ✓ Only legitimate users can access the health data stored at the medical server [27].
  - ✓ V-222619: The ISSO must ensure an account management process is implemented, verifying only authorized users can gain access to the application, and individual accounts designated

<p>as inactive, suspended, or terminated are promptly removed. Establish an account management process [15].</p>
<ul style="list-style-type: none"> <li>✓ CIS Critical Security Control 5: Account Management: Use processes and tools to assign and manage authorization to credentials for user accounts, including administrator accounts, as well as service accounts, to enterprise assets and software [14].</li> <li>✓ Credential Handling and Security: After retrieving an account using AccountManager, apply CREATOR before passing any credentials to avoid unintentionally exposing them to unauthorized applications [18].</li> <li>✓ V-222407: The application must provide automated mechanisms for supporting account management functions. Use automated processes and mechanisms for account management functions [15].</li> <li>✓ Properly Handle User Credentials: User credentials should always be stored, transmitted, and authenticated securely: <ul style="list-style-type: none"> <li>○ Encrypt credentials during transmission.</li> <li>○ Do not store user credentials on the device. Instead, consider using secure, revocable access tokens.</li> <li>○ Implement strong user authentication protocols.</li> <li>○ Regularly update and rotate any used API keys or tokens [3].</li> </ul> </li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Implementing secure account and credential management ensures authorized access, minimizes the risk of credential compromise, and facilitates compliance with regulatory requirements for handling sensitive data like health information.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<p><b>1. Account Management Testing:</b></p>
<ul style="list-style-type: none"> <li>✓ Verify that the application implements processes to create, update, and remove accounts promptly for inactive, suspended, or terminated users.</li> </ul>
<p><b>2. Credential Security Testing:</b></p>
<ul style="list-style-type: none"> <li>✓ Confirm that credentials are encrypted during transmission and not stored on the device.</li> </ul>

**3. Automated Mechanism Validation:**

- ✓ Validate that the application uses automated processes for managing account lifecycle events (e.g., creation, suspension, removal).

**4. AccountManager Usage Review:**

- ✓ Test that AccountManager is used for securely handling service integrations and that passwords are not stored on the device.

**5. Token Management Testing:**

- ✓ Confirm that the application uses secure, revocable access tokens and API keys or tokens are regularly updated and rotated.

**6. Penetration Testing:**

- ✓ Simulate attacks to bypass credential storage and management mechanisms, ensuring they are robust.

**7. Password Management Validation:**

- ✓ Confirm that the application enforces password creation, change, and safeguarding procedures.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda

José L Fernández-Alemán

Juan Manuel Carrillo-de-Gea

Joaquín Nicolás

08/01/2026

**PUID:** [SECM-CAT-IAA-065]

**Requirement description:** The application must implement a device-specific authentication token that the user can revoke to mitigate unauthorized access risks arising from lost or stolen devices.

**Source:**

- ✓ 6. Mobile applications should ideally use a device-specific authentication token that can be revoked within the mobile application by the user, mitigating unauthorized access risks from a stolen/lost device [16].

✓ SR12: Theft of device In the case of theft or loss of the device, there must be a provision to revoke old credentials and refurbish the health records on a new patient mobile device [30].
<b>Priority:</b> Not described
<b>Rationale:</b> Using a device-specific authentication token provides an additional layer of security by ensuring that access credentials are tied to the device. Revocability by the user reduces the risk of unauthorized access in case the device is lost or stolen.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Token Implementation Testing:</b> <ul style="list-style-type: none"> <li>✓ Verify that the application issues unique device-specific authentication tokens for each registered device.</li> </ul> </li> <li><b>2. Revocation Mechanism Testing:</b> <ul style="list-style-type: none"> <li>✓ Confirm that users can revoke the authentication token directly through the application.</li> </ul> </li> <li><b>3. Lost/Stolen Device Scenario Testing:</b> <ul style="list-style-type: none"> <li>✓ Simulate a lost or stolen device scenario to validate that revoked tokens are no longer usable for accessing the application.</li> </ul> </li> <li><b>4. Token Security Validation:</b> <ul style="list-style-type: none"> <li>✓ Ensure that the authentication tokens are securely stored on the device and transmitted only over secure channels (e.g., TLS).</li> </ul> </li> <li><b>5. Penetration Testing:</b> <ul style="list-style-type: none"> <li>✓ Attempt to reuse revoked tokens to ensure they are effectively invalidated.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán

Juan Manuel Carrillo-de-Gea  
 Joaquín Nicolás  
 08/01/2026

**PUID:** [SECM-CAT-IIA-066]

**Requirement description:** The application must purge credentials or keys from memory immediately after use to minimize exposure, using explicit memory management rather than relying on automatic garbage collection.

**Source:**

- ✓ 3.4. Consider purging credentials or keys from memory after use. Avoid automatic memory managed structures (e.g., controlled by garbage collector) and immutable objects for maintaining the keys. Prefer to immediately zero out the memory containing the data after use rather than waiting for the garbage collection mechanism [16].

**Priority:** Not described

**Rationale:** Purging credentials or keys from memory after use mitigates the risk of unauthorized access to sensitive data. Avoiding reliance on automatic memory management ensures proactive and secure handling of sensitive information.

**Number of Children:** 0**Number of parents:** 0**Cycles:** 0**Number Audit:** 0**Child PUIDs:** Not described**Parent PUIDs:** Not described**Exclusion PUIDs:** Not described**Importance:** Not described**Current state:** To be determined**Verification method:** Demonstration/Analysis**Validation criteria:****1. Memory Purge Testing:**

- ✓ Verify that credentials or keys are explicitly removed from memory after their use.

**2. Static Code Analysis:**

- ✓ Ensure no automatic memory management mechanisms are used for handling sensitive data like credentials or keys.

**3. Runtime Memory Inspection:**

- ✓ Test the application during runtime to confirm that sensitive data is not left in memory after it has been used.

**4. Immediate Zeroing Verification:**

- ✓ Validate that the memory allocated for credentials or keys is overwritten with zeros immediately after use.

## **5. Security Review of Key Handling:**

- ✓ Confirm that immutable objects are avoided when dealing with sensitive data to prevent unintentional persistence in memory.

## **6. Penetration Testing:**

- ✓ Simulate attacks to extract credentials or keys from application memory to validate that sensitive information is properly purged.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda

José L Fernández-Alemán

Juan Manuel Carrillo-de-Gea

Joaquín Nicolás

08/01/2026

**PUID:** [SECM-CAT-IIA-067]

**Requirement description:** The application must immediately release UI frames containing credentials or keys after their use to prevent unintended exposure.

**Source:**

- ✓ 3.5. If the credentials or keys appear in the user interface (UI) components, try to release the UI frames immediately after use. [16].

**Priority:** Not described

**Rationale:** Releasing UI frames containing sensitive data, such as credentials or keys, minimizes the risk of accidental exposure or unauthorized access through interface caching or memory retention.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

### **1. UI Frame Release Testing:**

- ✓ Verify that UI frames containing sensitive data are programmatically cleared or released immediately after use.

## 2. Runtime Memory Inspection:

- ✓ Confirm that sensitive data displayed in UI components is no longer present in memory after the UI frame is released.

## 3. Static Code Analysis:

- ✓ Analyze the application code to ensure that UI components displaying credentials or keys are promptly deallocated after use.

## 4. Simulated Access Attempt:

- ✓ Attempt to access previously displayed sensitive data in the UI to confirm that it is inaccessible after the frame is released.

## 5. User Interface Review:

- ✓ Ensure the application does not cache or retain sensitive data in UI components once its intended use is completed.

## 6. Penetration Testing:

- ✓ Simulate attacks to extract sensitive data from UI components to validate proper release mechanisms.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda

José L Fernández-Alemán

Juan Manuel Carrillo-de-Gea

Joaquín Nicolás

08/01/2026

**PUID:** [SECM-CAT-IAA-068]

**Requirement description:** The application must implement fine-grained URI access control using the `android:grantUriPermissions` attribute, `FLAG_GRANT_READ_URI_PERMISSION`, and `FLAG_GRANT_WRITE_URI_PERMISSION` in combination with the `<grant-uri-permission>` element, ensuring sensitive operations are secured with additional authentication.

**Source:**

- ✓ Granular URI Access Control: To enable fine-grained control over which URIs are accessed, use the `android:grantUriPermissions` attribute, in combination with `FLAG_GRANT_READ_URI_PERMISSION` and

<p>FLAG_GRANT_WRITE_URI_PERMISSION flags in Intent objects. Limit the scope of these permissions through the &lt;grant-uri-permission&gt; element [18].</p> <ul style="list-style-type: none"> <li>✓ MASVS-AUTH-1: The app uses secure authentication and authorization protocols and follows the relevant best practices: Most apps connecting to a remote endpoint require user authentication and also enforce some kind of authorization. While the enforcement of these mechanisms must be on the remote endpoint, the apps also have to ensure that it follows all the relevant best practices to ensure a secure use of the involved protocols [20].</li> <li>✓ MASVS-AUTH-3: The app secures sensitive operations with additional authentication: Some additional form of authentication is often desirable for sensitive actions inside the app. This can be done in different ways (biometric, pin, MFA code generator, email, deep links, etc) and they all need to be implemented securely [20].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Implementing granular URI access control minimizes unauthorized access risks by restricting permissions to specific resources. Adding authentication for sensitive operations enhances security by ensuring only authorized actions are performed.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<p><b>1. URI Permission Testing:</b></p> <ul style="list-style-type: none"> <li>✓ Verify that FLAG_GRANT_READ_URI_PERMISSION and FLAG_GRANT_WRITE_URI_PERMISSION are used appropriately for URI access within Intent objects.</li> </ul>
<p><b>2. Permission Scope Validation:</b></p> <ul style="list-style-type: none"> <li>✓ Confirm that the &lt;grant-uri-permission&gt; element limits the scope of granted permissions to the intended URIs.</li> </ul>
<p><b>3. Authentication for Sensitive Actions:</b></p> <ul style="list-style-type: none"> <li>✓ Test that sensitive operations involving URI access are secured with additional authentication, such as biometrics or MFA.</li> </ul>
<p><b>4. Static Code Analysis:</b></p>

- ✓ Ensure `android:grantUriPermissions` is implemented in the application manifest and used securely in Intent objects.

### 5. Runtime Validation:

- ✓ Simulate unauthorized access attempts to confirm that URI access permissions are correctly restricted and revoked after use.

### 6. Penetration Testing:

- ✓ Perform penetration tests to identify any potential misuse of granted URI permissions and validate secure configurations.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda  
 José L Fernández-Alemán  
 Juan Manuel Carrillo-de-Gea  
 Joaquín Nicolás  
 08/01/2026

**PUID:** [SECM-CAT-IIA-069]

**Requirement description:** The application must enforce a strong password policy and prohibit the reuse or recycling of session IDs to maintain robust authentication security.

**Source:**

- ✓ If a password-based authentication mechanism is used, ensure that a strong password policy is being followed. Consider enforcing restrictions about password length and formation, reuse of old user passwords, use of common passwords, password duration, etc. It may also be useful to provide feedback on the strength of the password when it is being entered for the first time. However, do not maintain any representation of the password strength in application storage or the back-end server as it may expose the password in preimage attacks [16].
- ✓ V-222582: The application must not re-use or recycle session IDs. Design the application to not re-use session IDs [15].

**Priority:** Not described

**Rationale:** A strong password policy ensures that user credentials are resilient against brute force and dictionary attacks while prohibiting session ID reuse or recycling, which prevents session fixation and unauthorized access.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

<b>Parent PUIDs:</b> Not described <b>Exclusion PUIDs:</b> Not described <b>Importance:</b> Not described <b>Current state:</b> To be determined <b>Verification method:</b> Demonstration/Analysis <b>Validation criteria:</b>
<b>1. Password Policy Testing:</b> <ul style="list-style-type: none"><li>✓ Verify that the application enforces password complexity requirements, including minimum length and use of upper-case, lower-case, numeric, and special characters.</li></ul>
<b>2. Password Reuse Testing:</b> <ul style="list-style-type: none"><li>✓ Confirm that users cannot reuse recent passwords, with a configurable restriction on the number of stored previous passwords.</li></ul>
<b>3. Session ID Validation:</b> <ul style="list-style-type: none"><li>✓ Ensure session IDs are unique for each user session and not recycled under any circumstances.</li></ul>
<b>4. Session Expiry Testing:</b> <ul style="list-style-type: none"><li>✓ Validate that session IDs expire upon logout and are replaced with new IDs upon subsequent logins.</li></ul>
<b>5. Strength Feedback Testing:</b> <ul style="list-style-type: none"><li>✓ Confirm that the application provides real-time feedback on password strength during setup without storing this feedback in logs or storage.</li></ul>
<b>6. Static Code Review:</b> <ul style="list-style-type: none"><li>✓ Analyze code to ensure proper password validation logic and unique session ID generation methods are implemented securely.</li></ul>
<b>7. Penetration Testing:</b> <ul style="list-style-type: none"><li>✓ Test for vulnerabilities such as session fixation and password policy bypasses.</li></ul>
<b>Requested by:</b> The organization <b>Responsible:</b> Developer <b>Configurable value:</b> Not described <b>Version history:</b> v1.0 <b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán

Juan Manuel Carrillo-de-Gea  
Joaquín Nicolás  
08/01/2026

**PUID:** [SECM-CAT-IIA-070]

**Requirement description:** The application must validate session identifiers and ensure their authenticity throughout the session lifecycle, invalidating them upon logout or session termination to prevent unauthorized reuse.

**Source:**

- ✓ V-222580: Applications must validate session identifiers. Configure the application to configure user session identifiers [15].
- ✓ SC-23 SESSION AUTHENTICITY

Control: Protect the authenticity of communications sessions.

Discussion: Protecting session authenticity addresses communications protection at the session level, not at the packet level. Such protection establishes grounds for confidence at both ends of communications sessions in the ongoing identities of other parties and the validity of transmitted information. Authenticity protection includes protecting against “man-in-the-middle” attacks, session hijacking, and the insertion of false information into sessions.

Control Enhancements:

(1) SESSION AUTHENTICITY | INVALIDATE SESSION IDENTIFIERS AT LOGOUT

Invalidate session identifiers upon user logout or other session termination.

Discussion: Invalidating session identifiers at logout curtails the ability of adversaries to capture and continue to employ previously valid session IDs [11].

**Priority:** Not described

**Rationale:** Validating session identifiers and ensuring their authenticity minimizes the risks of session hijacking and unauthorized reuse. Invalidation upon session termination prevents adversaries from exploiting active or cached session IDs.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

**1. Session ID Validation Testing:**

- ✓ Confirm that session identifiers are validated for authenticity at the start of each user session.

**2. Session Invalidations Testing:**

- ✓ Verify that session IDs are invalidated immediately upon user logout or session timeout.

**3. Session Hijacking Simulation:**

- ✓ Simulate a session hijacking attempt to ensure the application detects and invalidates compromised session IDs.

**4. Static Code Review:**

- ✓ Analyze the session management implementation to verify robust validation logic and secure session termination.

**5. Logout Behavior Verification:**

- ✓ Test the logout functionality to ensure the session ID is no longer valid after termination.

**6. Penetration Testing:**

- ✓ Conduct penetration testing to detect potential vulnerabilities in session management and identify remediation actions.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda  
José L Fernández-Alemán  
Juan Manuel Carrillo-de-Gea  
Joaquín Nicolás  
08/01/2026

**PUID:** [SECM-CAT-IIA-071]

**Requirement description:** The application must implement permission-based intent filtering to restrict the handling of sensitive data in intent to authorized applications only.

**Source:**

- ✓ Permission-Based Intent Filtering: When sending sensitive data in an intent, apply a permission check to ensure only authorized applications can receive the intent. Use a non-

null permission parameter when sending an intent, so that only applications with the required permission can handle the intent. [18].
<b>Priority:</b> Not described
<b>Rationale:</b> Applying permission-based intent filtering ensures sensitive data transmitted via intents is only accessible to authorized applications, reducing the risk of unauthorized data access or leakage.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<p><b>1. Intent Permission Validation:</b></p> <ul style="list-style-type: none"> <li>✓ Verify that all intents carrying sensitive data include a non-null permission parameter.</li> </ul> <p><b>2. Unauthorized Application Handling:</b></p> <ul style="list-style-type: none"> <li>✓ Test unauthorized applications attempting to handle protected intents to confirm access is denied.</li> </ul> <p><b>3. Static Code Analysis:</b></p> <ul style="list-style-type: none"> <li>✓ Ensure the implementation of permission checks in all intents of transmitting sensitive information.</li> </ul> <p><b>4. Runtime Testing:</b></p> <ul style="list-style-type: none"> <li>✓ Confirm that intents are only delivered to applications with the required permissions during application runtime.</li> </ul> <p><b>5. Security Review:</b></p> <ul style="list-style-type: none"> <li>✓ Validate that permission checks align with the defined security policies for intent handling.</li> </ul> <p><b>6. Penetration Testing:</b></p> <ul style="list-style-type: none"> <li>✓ Attempt to bypass intent permission filtering and confirm the application effectively blocks unauthorized access.</li> </ul>
<b>Requested by:</b> The organization

<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IAA-072]
<b>Requirement description:</b> The application must implement alternative access controls, such as signature-level protection for inter-process communication (IPC) and adaptive authentication techniques, to enhance security without over-reliance on user-confirmed permissions.
<b>Source:</b>
<ul style="list-style-type: none"><li>✓ Use of Alternative Access Controls: Where possible, access controls other than user-confirmed permissions must be used. For example, signature-level protection must be applied for IPC communication instead of prompting users for permission [18].</li><li>✓ MASVS-AUTH-3: The app secures sensitive operations with additional authentication: Some additional form of authentication is often desirable for sensitive actions inside the app. This can be done in different ways (biometric, pin, MFA code generator, email, deep links, etc) and they all need to be implemented securely [20].</li><li>✓ IA-10 ADAPTIVE AUTHENTICATION</li><li>✓ Control: Require individuals accessing the system to employ [Assignment: organization-defined supplemental authentication techniques or mechanisms] under specific [Assignment: organization-defined circumstances or situations].</li><li>✓ Discussion: Adversaries may compromise individual authentication mechanisms employed by organizations and subsequently attempt to impersonate legitimate users. To address this threat, organizations may employ specific techniques or mechanisms and establish protocols to assess suspicious behavior. Suspicious behavior may include accessing information that individuals do not typically access as part of their duties, roles, or responsibilities; accessing greater quantities of information than individuals would routinely access; or attempting to access information from suspicious network addresses. When pre-established conditions or triggers occur, organizations can require individuals to provide additional authentication information. Another potential use for adaptive authentication is to increase the strength of mechanism based on the number or types of records being accessed. Adaptive authentication does not replace and is not used to avoid the use of multi-factor authentication mechanisms but can augment implementations of multi-factor authentication [11].</li></ul>
<b>Priority:</b> Not described

<b>Rationale:</b> Enhancing access control with alternative mechanisms and adaptive authentication improves the security of sensitive operations, minimizes dependency on user interaction, and mitigates risks posed by compromised authentication methods or suspicious activities.
---

<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<b>1. Signature-Level Access Control Testing:</b>
<ul style="list-style-type: none"><li>✓ Confirm that signature-level protection is applied for IPC communication in place of user-permission prompts.</li></ul>
<b>2. Additional Authentication Testing:</b>
<ul style="list-style-type: none"><li>✓ Validate that sensitive operations within the application are protected by additional authentication mechanisms, such as biometrics or MFA.</li></ul>
<b>3. Adaptive Authentication Simulation:</b>
<ul style="list-style-type: none"><li>✓ Test the implementation of adaptive authentication by simulating suspicious behavior, such as accessing unusual data or exceeding normal access thresholds.</li></ul>
<b>4. Dynamic Authentication Behavior Validation:</b>
<ul style="list-style-type: none"><li>✓ Ensure that adaptive authentication is triggered under defined conditions, such as suspicious network addresses or excessive data access.</li></ul>
<b>5. Static Code Analysis:</b>
<ul style="list-style-type: none"><li>✓ Review source code to verify that alternative access controls and adaptive authentication techniques are implemented securely.</li></ul>
<b>6. Penetration Testing:</b>
<ul style="list-style-type: none"><li>✓ Attempt to bypass alternative access controls and adaptive authentication mechanisms to identify potential vulnerabilities.</li></ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán

Juan Manuel Carrillo-de-Gea  
Joaquín Nicolás  
08/01/2026

<b>PUID:</b> [SECM-CAT-IIA-073]
<b>Requirement description:</b> The application must implement permission-based access control for sensitive resources.
<b>Source:</b>
✓ Control access to information (contacts, camera, localization) through permission [25].
<b>Priority:</b> Not described
<b>Rationale:</b> Restricting access to sensitive resources through permission-based controls ensures that unauthorized applications cannot access critical data or hardware, protecting user privacy and maintaining application security.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<b>1. Permission Request Validation:</b>
✓ Ensure the application requests appropriate permissions before accessing sensitive resources such as contacts, cameras, or location services.
<b>2. Unauthorized Access Testing:</b>
✓ Confirm that resources cannot be accessed without the required user authorization.
<b>3. Permission Configuration Analysis:</b>
✓ Verify the application configuration restricts access to sensitive resources through defined permission settings.
<b>4. Runtime Testing:</b>
✓ Simulate scenarios where permissions are granted and denied to confirm the application behaves securely in both cases.

<p><b>5. Static Code Review:</b></p> <ul style="list-style-type: none"><li>✓ Review code to verify that permission checks are correctly implemented and enforced for sensitive data and resources.</li></ul> <p><b>6. Penetration Testing:</b></p> <ul style="list-style-type: none"><li>✓ Attempt to bypass permission controls and verify the application blocks unauthorized access to sensitive resources.</li></ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<p><b>PUID:</b> [SECM-CAT-IIA-074]</p> <p><b>Requirement description:</b> The application must validate and enforce approved authorizations for logical access to information and system resources based on access control policies, ensuring only authorized users and services can access protected data and resources.</p>
<p><b>Source:</b></p> <ul style="list-style-type: none"><li>✓ Control access to information (contacts, camera, localization) through permission [25].</li><li>✓ Security Best Practices: Always validate the permissions for services to avoid unauthorized access. Use declarative permissions and android for better visibility and control over service access [18].</li><li>✓ Test ID 5: System components are configured to allow only authorized access to information [37].</li><li>✓ Authorize access to the system based on:<ol style="list-style-type: none"><li>1. A valid access authorization;</li><li>2. Intended system usage; and</li><li>3. [Assignment: organization-defined attributes (as required)] [11];</li></ol></li><li>✓ Control: Enforce approved authorizations for logical access to information and system resources in accordance with applicable access control policies [11].</li><li>✓ V-222425: The application must enforce approved authorizations for logical access to information and system resources in accordance with applicable access control policies. Design or configure the application to enforce access to application resources [15].</li></ul>
<b>Priority:</b> Not described

**Rationale:** Proper enforcement of access controls is essential to protect sensitive information and system resources from unauthorized access. This ensures compliance with security policies and reduces the risk of data breaches and misuse of system functionalities.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

**1. Permission Validation Test:**

- ✓ Ensure all declared permissions are validated at runtime to confirm authorized access to application resources.

**2. Access Control Configuration Review:**

- ✓ Verify the application enforces logical access controls based on approved access policies.

**3. Role-Based Access Testing:**

- ✓ Simulate access requests from different roles and ensure users only access resources they are authorized to access.

**4. Unauthorized Access Prevention Test:**

- ✓ Test attempts to access restricted resources without proper authorization and confirm such requests are denied.

**5. Static Code Analysis:**

- ✓ Review code to ensure proper implementation of declarative permissions and authorization checks.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda

José L Fernández-Alemán

Juan Manuel Carrillo-de-Gea

Joaquín Nicolás

08/01/2026

**PUID:** [SECM-CAT-IAA-075]

**Requirement description:** The application must control the export behavior and access permissions of BroadcastReceiver components to prevent unauthorized access and ensure secure communication, particularly for sensitive data and critical functionality.

**Source:**

- ✓ Default Export Behavior: By default, BroadcastReceiver components are exported, making them accessible to any external application. This default behavior can expose the receiver to untrusted sources.

Receiver Export Control: If a BroadcastReceiver is not intended for use by external applications, it must be explicitly marked as `android:exported="false"` in the `<receiver>` manifest declaration to prevent unauthorized external access.

Permission Protection: When the BroadcastReceiver is intended for use by other applications, the application must enforce security permissions by specifying the `android:permission` attribute in the `<receiver>` manifest element. This ensures that only applications with the correct permissions can send intents to the receiver.

Restricted Access to Broadcast Intents: If sensitive data is being handled, additional security mechanisms must be applied, such as specifying the permission required to send an Intent to the BroadcastReceiver. This prevents untrusted or malicious applications from invoking the receiver.

Explicit Receivers for Critical Functionality: For critical actions or services, explicit BroadcastReceiver implementations must be used, where the Intent explicitly identifies the target receiver by name, avoiding reliance on implicit broadcasts. This ensures that the correct receiver processes the Intent.

Use of Ordered Broadcasts: If ordered broadcasts are used, the application must ensure that permissions are properly enforced to prevent lower-priority receivers from intercepting or modifying the broadcast data without proper authorization [18].

**Priority:** Not described

**Rationale:** Proper management of BroadcastReceiver components and their permissions is essential to avoid unauthorized access, mitigate security vulnerabilities, and ensure that sensitive data is handled securely. Controlling receiver export behavior and enforcing permissions reduces exposure to untrusted or malicious applications.

<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<b>1. Manifest Configuration Review:</b>
<ul style="list-style-type: none"><li>✓ Verify that all non-public <code>BroadcastReceiver</code> components are explicitly marked with <code>android:exported="false"</code> in the manifest.</li></ul>
<b>2. Permission Attribute Check:</b>
<ul style="list-style-type: none"><li>✓ Confirm the <code>android:permission</code> attribute is specified for all exported <code>BroadcastReceiver</code> components to restrict access.</li></ul>
<b>3. Explicit Receiver Testing:</b>
<ul style="list-style-type: none"><li>✓ Test critical services to ensure only explicitly defined <code>BroadcastReceiver</code> components are invoked via Intent targeting.</li></ul>
<b>4. Code Review for Broadcast Intent Handling:</b>
<ul style="list-style-type: none"><li>✓ Inspect code to confirm that sensitive broadcast intents include security checks and proper permissions.</li></ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IAA-076]
<b>Requirement description:</b> The application must implement secure practices for intent broadcasting, using explicit intents or controlled mechanisms to ensure broadcasts are delivered securely and only to intended recipients.
<b>Source:</b>

<ul style="list-style-type: none"> <li>✓ Secure Intent Broadcasting: When broadcasting intents, consider using sendBroadcast or sendOrderedBroadcast based on application requirements.           <ul style="list-style-type: none"> <li>○ If a specific recipient must receive the broadcast, use an explicit intent and specify the receiver by name.</li> <li>○ Be cautious with ordered broadcasts, as they can be consumed by a recipient and might not be delivered to all intended applications [18].</li> </ul> </li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Secure broadcasting practices prevent unauthorized applications from intercepting or altering intent data. By using explicit intents and managing ordered broadcasts carefully, the application ensures data integrity and secure communication between components.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<p><b>1. Explicit Intent Verification:</b></p> <ul style="list-style-type: none"> <li>✓ Verify that explicit intents are used for broadcasts requiring specific recipient targeting.</li> <li>✓ Confirm the receiver's name is explicitly specified in the intent.</li> </ul>
<p><b>2. Ordered Broadcast Handling Test:</b></p> <ul style="list-style-type: none"> <li>✓ Validate that ordered broadcasts include appropriate permission checks to prevent unauthorized interception or consumption of data.</li> </ul>
<p><b>3. Unauthorized Access Simulation:</b></p> <ul style="list-style-type: none"> <li>✓ Attempt to intercept broadcasts from untrusted sources to ensure such attempts are blocked.</li> </ul>
<p><b>4. Receiver Behavior Review:</b></p> <ul style="list-style-type: none"> <li>✓ Confirm that all receivers handle ordered broadcasts securely without altering or blocking other recipients' access.</li> </ul>
<p><b>5. Static Code Analysis:</b></p> <ul style="list-style-type: none"> <li>✓ Inspect the codebase to confirm the correct usage of sendBroadcast and sendOrderedBroadcast based on security requirements.</li> </ul>

<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda
José L Fernández-Alemán
Juan Manuel Carrillo-de-Gea
Joaquín Nicolás
08/01/2026

<b>PUID:</b> [SECM-CAT-IIA-077]
<b>Requirement description:</b> The application must secure Inter-Process Communication (IPC) calls by implementing <code>checkCallingPermission()</code> for individual IPC requests or using declarative permissions in the manifest to enhance security and maintainability.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ Individual IPC Call Protection: Protect individual IPC calls made to the service by using <code>checkCallingPermission()</code> before processing any request. However, declarative permissions in the manifest are recommended for better maintainability and security [18].</li> <li>✓ MASVS-PLATFORM-1: The app uses IPC mechanisms securely: Apps typically use platform provided IPC mechanisms to intentionally expose data or functionality. Both installed apps and the user are able to interact with the app in many different ways. This control ensures that all interactions involving IPC mechanisms happen securely [20].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Properly securing IPC calls prevents unauthorized applications from accessing sensitive data or functionality exposed through IPC mechanisms. By validating permissions through <code>checkCallingPermission()</code> or declarative manifest entries, the application enforces robust access control policies.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Permission Verification Testing:</b></li> </ol> <ul style="list-style-type: none"> <li>✓ Verify that <code>checkCallingPermission()</code> is implemented for all IPC calls requiring access to sensitive functionality.</li> </ul> <ol style="list-style-type: none"> <li><b>2. Manifest Configuration Review:</b></li> </ol>

- ✓ Confirm that declarative permissions are correctly defined in the manifest for all IPC-exposed components.

### 3. Unauthorized IPC Simulation:

- ✓ Simulate IPC requests from untrusted sources and ensure unauthorized requests are denied.

### 4. Code Review for IPC Mechanisms:

- ✓ Inspect the codebase to ensure all IPC interactions include proper permission validation using `checkCallingPermission()` or equivalent manifest declarations.

### 5. Dynamic Testing:

- ✓ Execute dynamic tests to verify the correct application of permissions during runtime, ensuring no unauthorized IPC access.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda  
José L Fernández-Alemán  
Juan Manuel Carrillo-de-Gea  
Joaquín Nicolás  
08/01/2026

**PUID:** [SECM-CAT-IAA-078]

**Requirement description:** The application must enforce explicit access control checks within Binder or Messenger interfaces to ensure secure communication, as these objects do not support declarative permissions in the manifest.

**Source:**

- ✓ Interface Permissions: Binder and Messenger objects aren't declared in the manifest, and therefore cannot directly use declarative permissions. They inherit permissions declared in the manifest for the Service or Activity they are part of. If access control or authentication is required for the interface, the developer must implement explicit checks within the Binder or Messenger interface [18].

**Priority:** Not described

**Rationale:** Binder and Messenger objects operate outside the scope of declarative manifest permissions, potentially exposing sensitive interfaces to unauthorized access. Implementing explicit access control checks within these interfaces ensures that only authorized applications or processes can interact with them.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<b>1. Explicit Access Control Implementation:</b>
✓ Verify that explicit access control checks are implemented within Binder and Messenger objects to validate permissions before processing requests.
<b>2. Service and Activity Permission Validation:</b>
✓ Confirm that Binder and Messenger objects correctly inherit permissions from their parent Service or Activity.
<b>3. Code Review:</b>
✓ Conduct a static code analysis to ensure explicit checks are included within Binder and Messenger interfaces to handle authentication and authorization.
<b>4. Dynamic Testing:</b>
✓ Test the runtime behavior of Binder and Messenger interfaces under varying access scenarios to confirm proper enforcement of access controls.
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IAA-079]
<b>Requirement description:</b> The application must implement access control mechanisms for interfaces by using methods like <code>checkCallingPermission()</code> to verify permissions of calling entities, ensuring secure and restricted access to sensitive services and activities.
<b>Source:</b>
✓ 164.308 Administrative safeguards.  (a) A covered entity or business associate must, in accordance with § 164.306: <b>(4)</b>

<p><b>(ii) Implementation specifications:</b></p> <p><b>(B) Access authorization (Addressable).</b></p> <p>Implement policies and procedures for granting access to electronic protected health information, for example, through access to a workstation, transaction, program, process, or other mechanism [9].</p> <ul style="list-style-type: none"> <li>✓ Access Control Implementation: For interfaces requiring access controls, the <code>checkCallingPermission()</code> method must be used to verify whether the calling entity has the required permission. This check is critical when services or activities are accessed on behalf of the caller, as the application identity is passed to other interfaces. [18].</li> <li>✓ CIS Critical Security Control 6: Access Control Management: Use processes and tools to create, assign, manage, and revoke access credentials and privileges for user, administrator, and service accounts for enterprise assets and software [14].</li> <li>✓ MASVS-AUTH-1: The app uses secure authentication and authorization protocols and follows the relevant best practices: Most apps connecting to a remote endpoint require user authentication and also enforce some kind of authorization. While the enforcement of these mechanisms must be on the remote endpoint, the apps also have to ensure that it follows all the relevant best practices to ensure a secure use of the involved protocols [20].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Access controls on interfaces are essential to prevent unauthorized access to sensitive functionalities and data. Using <code>checkCallingPermission()</code> or equivalent methods ensures the application validates the permissions of callers, maintains the security of protected assets, and ensures compliance with organizational and regulatory access control policies.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<p><b>1. Permission Check Implementation:</b></p> <ul style="list-style-type: none"> <li>✓ Confirm the use of <code>checkCallingPermission()</code> or equivalent methods in services and activities requiring access control.</li> </ul> <p><b>2. Unauthorized Access Attempt:</b></p> <ul style="list-style-type: none"> <li>✓ Attempt unauthorized access to restricted services or activities and verify that access is denied.</li> </ul>

**3. Permission Validation Test:**

- ✓ Test scenarios where callers with appropriate permissions are allowed access while those without are rejected.

**4. Code Inspection:**

- ✓ Conduct a code review to ensure `checkCallingPermission()` or similar access control mechanisms are consistently applied.

**5. Dynamic Analysis:**

- ✓ Simulate runtime behavior to validate proper enforcement of access controls for all services and activities.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda  
José L Fernández-Alemán  
Juan Manuel Carrillo-de-Gea  
Joaquín Nicolás  
08/01/2026

**PUID:** [SECM-CAT-IAA-080]

**Requirement description:** The application must implement strong access controls, including secure user authentication, role-based access control (RBAC), and validation of user permissions before granting access to sensitive information or functionality.

**Source:**

- ✓ Employ Proper Access Controls: 1. Implement strong access controls to restrict unauthorized access to sensitive data. Authenticate users securely, enforce role-based access controls, and validate user permissions before granting access to sensitive information [3].
- ✓ SR8: Selective Access Since the health folder may have a collection of different types of health information, they must be accessed using selective RBAC by medical professionals based on their role [30].
- ✓ IA-9 SERVICE IDENTIFICATION AND AUTHENTICATION
- ✓ Control: Uniquely identify and authenticate [Assignment: organization-defined system services and applications] before establishing communications with devices, users, or other services or applications.
- ✓ Discussion: Services that may require identification and authentication include web applications using digital certificates or services or applications that query a database. Identification and authentication methods for system services and applications include

information or code signing, provenance graphs, and electronic signatures that indicate the sources of services. Decisions regarding the validity of identification and authentication claims can be made by services separate from the services acting on those decisions. This can occur in distributed system architectures. In such situations, the identification and authentication decisions (instead of actual identifiers and authentication data) are provided to the services that need to act on those decisions [11].

✓ 9.2.3 Management of privileged access rights

- 1. Health information systems should associate users (including health professionals, supporting staff and others) with the records of subjects of care and allow future access based on this association.
- 2. Systems containing personal health information should support role-based access control capable of mapping each user to one or more roles and each role to one or more system functions.
- 3. A user of a health information system containing personal health information shall access its services in a single role [7].

✓ 4.8.2.2. Privilege Management

(a) Role-Based Access Control

Security Requirement 57 – Granting Access to Users by Role: The EHRi and all PoS systems connected to the EHRi must support role-based access control (RBAC) capable of mapping each user to one or more roles and each role to one or more system functions

Consideration CH26 – Select identity-proofing level of assurance by role: Consider what robust mechanism(s) should be used to initially prove the identity of the registrant and whether that level of assurance is required and/or adequate for all roles accessing the solution.

Consideration CH27 – Use a single identity credential where possible: Consider the use of a single identity (i.e. single sign-on) for users (i.e. patients or providers) where the portal provides access to other applications/services that have separate authentication mechanisms.

Consideration ID-14 – Consider Establishing Authentication Protocols for Use Among Member Organizations [10].

**Priority:** Not described

**Rationale:** Implementing strong access controls ensures that only authorized users can access sensitive information and system functionalities. By utilizing role-based access control and robust authentication mechanisms, the system minimizes risks of unauthorized access and ensures compliance with healthcare data protection standards.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<p><b>1. Role-Based Access Testing:</b></p> <ul style="list-style-type: none"> <li>✓ Verify that the application supports RBAC by mapping users to roles and roles to specific system functions.</li> <li>✓ Ensure that users can only access data and functionalities based on their assigned roles.</li> </ul> <p><b>2. Permission Validation:</b></p> <ul style="list-style-type: none"> <li>✓ Attempt to access sensitive resources with insufficient permissions and confirm that access is denied.</li> <li>✓ Ensure that users with valid permissions can access their designated resources.</li> </ul> <p><b>3. Authentication Mechanism Check:</b></p> <ul style="list-style-type: none"> <li>✓ Validate that all users are authenticated securely before accessing sensitive data or system functionalities.</li> <li>✓ Verify that the application supports single sign-on (SSO) where appropriate.</li> </ul> <p><b>4. Privilege Escalation Testing:</b></p> <ul style="list-style-type: none"> <li>✓ Attempt privilege escalation attacks and confirm that unauthorized access is not granted.</li> </ul> <p><b>5. Code Review:</b></p> <ul style="list-style-type: none"> <li>✓ Review the implementation of access control mechanisms, including RBAC and permission validation, to ensure alignment with best practices.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IAA-081]
<b>Requirement description:</b> The application must support workgroup-based access control by assigning users to working groups and granting access to records based on group membership while ensuring timely revocation of access privileges when required.
<b>Source:</b> ✓ b) Workgroup-Based Access Control Security Requirement 59 – Granting Access to Users in Workgroups: The EHRI and all PoS systems connected to the EHRI must be capable of assigning users to working groups and granting access to records based on working groups.
Security Requirement 60 – Timely Revocation of Access Privileges: The EHRI and all PoS systems connected to the EHRI must support the revocation of user access privileges in a timely manner (i.e. immediately prevent the user from logging on after access privileges have been revoked) [10].

<b>Priority:</b> Not described
<b>Rationale:</b> Implementing workgroup-based access control facilitates efficient management of user access by organizing users into working groups simplifying permissions for shared records. Ensuring timely revocation of access privileges is critical to mitigate risks of unauthorized access after a user's permissions are rescinded.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<p><b>1. Workgroup Assignment Testing:</b></p> <ul style="list-style-type: none"><li>✓ Verify that the application allows administrators to assign users to specific workgroups.</li><li>✓ Confirm that access permissions for records are correctly granted based on group membership.</li></ul> <p><b>2. Access Control Enforcement:</b></p> <ul style="list-style-type: none"><li>✓ Attempt to access records outside the assigned workgroup and confirm access is denied.</li><li>✓ Verify that users within a workgroup can access the assigned records without issues.</li></ul> <p><b>3. Timely Revocation Testing:</b></p> <ul style="list-style-type: none"><li>✓ Revoke the access privileges of a user and attempt to log in with the revoked account.</li><li>✓ Confirm that the application immediately prevents the user from accessing the system after privilege revocation.</li></ul> <p><b>4. Audit Logging:</b></p> <ul style="list-style-type: none"><li>✓ Ensure that all changes in workgroup assignments and access revocations are logged for audit purposes.</li><li>✓ Review logs to confirm the accuracy and completeness of recorded events.</li></ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

**PUID:** [SECM-CAT-IAA-082]

**Requirement description:** The application must enforce discretionary access control policies, associating users with patient records to allow or restrict access based on pre-existing relationships while ensuring access privileges are reported for transparency.

**Source:**

- ✓ V-222426: The application must enforce organization-defined discretionary access control policies over defined subjects and objects. Design and configure the application to enforce discretionary access control policies [15].
- ✓ (c) Discretionary Access Control

Security Requirement 61 – Granting Access By Association: The EHRI and all PoS systems connected to the EHRI:

- a) Must be capable of associating users (i.e. healthcare providers) with the records of patients/persons and allowing future access based on this association (i.e., they must be capable of granting discretionary access to records based on a registered user with legitimate and pre-existing access to a patient's record(s) granting access rights for that (those) record(s) to another registered user); and
- b) Must not allow users to grant other users access to a record if the granting users themselves do not possess such access with respect to the record.

Note that granting other users access to a record does not override the role-based access control restrictions of those other users.

Security Requirement 62 – Reporting the Access Privileges of a User: The EHRI must, and PoS systems connected to the EHRI should, provide functionality that can report, for a given user:

- a) Which records the user can access;
- b) Which portions of the record(s) the user can access; and
- c) Which privileges (e.g. viewing, modification) the user has with respect to each of these records [10].

**Priority:** Not described

**Rationale:** Enforcing discretionary access control policies ensures that access to sensitive patient records is granted based on legitimate relationships, reducing unauthorized access risks. Reporting access privileges promotes transparency and facilitates compliance with access control policies.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

1. **Access Association Testing:**
  - ✓ Verify that the application allows healthcare providers to associate with patient records, granting access based on these associations.
  - ✓ Ensure that users without valid associations are denied access to records.
2. **Access Delegation Testing:**
  - ✓ Attempt to grant access to records for which the granting user does not have permissions.
  - ✓ Confirm that the application blocks access delegation in such cases.
3. **Privilege Reporting Testing:**

- ✓ Verify that the application provides functionality to report a user's access privileges, including records, portions of records, and allowed actions (e.g., view, modify).
- ✓ Validate that reported privileges match the access control policies configured in the system.

#### 4. Audit Logging:

- ✓ Confirm that all access and delegation attempts are logged.
- ✓ Ensure logs include details about the user, record access, privileges, and any delegation actions.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda  
José L Fernández-Alemán  
Juan Manuel Carrillo-de-Gea  
Joaquín Nicolás  
08/01/2026

**PUID:** [SECM-CAT-IAA-083]

**Requirement description:** The application must handle failures of unauthorized calls to service interfaces, ensuring appropriate responses are provided when the calling process lacks necessary permissions.

**Source:**

- ✓ Handling Unauthorized Calls: If the calling process lacks permission to invoke an interface provided by a service, the bindService() invocation will fail. The application must handle these failures and provide appropriate responses [18].

**Priority:** Not described

**Rationale:** Properly handling unauthorized interface calls enhances application security by preventing misuse of service interfaces, providing clear feedback to legitimate users, and logging unauthorized attempts for audit purposes.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

#### 1. Unauthorized Call Testing:

- ✓ Simulate calls to service interfaces from unauthorized processes.
- ✓ Verify that the bindService() invocation fails and the application responds with an appropriate error or notification message.

#### 2. Failure Handling Validation:

<ul style="list-style-type: none"> <li>✓ Check that the application logs unauthorized call attempts, including details such as the calling process and attempted interface.</li> <li>✓ Ensure that no service functionality is exposed or executed for unauthorized calls.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IAA-084]
<b>Requirement description:</b> The application must securely implement identity overriding mechanisms using <code>clearCallingIdentity()</code> and <code>restoreCallingIdentity()</code> to ensure temporary privilege escalation is safely managed during external process interactions.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ Overriding Caller Identity: If an external process needs to interact with the app but lacks the necessary permissions, use <code>clearCallingIdentity()</code> to temporarily override the caller's identity, allowing the app to perform the call as if the app itself initiated it. After completing the call, use <code>restoreCallingIdentity()</code> to restore the original caller's identity and permissions. [18].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Proper use of identity-overriding mechanisms is critical to maintaining security during interactions with external processes. It ensures that temporary privilege escalation is controlled, reducing the risk of unauthorized access or privilege abuse.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Identity Overriding Test:</b> <ul style="list-style-type: none"> <li>✓ Verify that the application correctly invokes <code>clearCallingIdentity()</code> to override the caller's identity.</li> <li>✓ Ensure the application performs the required operation successfully.</li> </ul> </li> <li><b>2. Identity Restoration Test:</b> <ul style="list-style-type: none"> <li>✓ Check that the application invokes <code>restoreCallingIdentity()</code> immediately after completing the privileged operation.</li> <li>✓ Verify that the original caller's identity and permissions are fully restored.</li> </ul> </li> <li><b>3. Unauthorized Use Prevention:</b></li> </ol>

<ul style="list-style-type: none"> <li>✓ Attempt unauthorized access or actions using <code>clearCallingIdentity()</code> without proper controls in place.</li> <li>✓ Confirm that the application logs and rejects unauthorized privilege escalations.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IAA-085]
<b>Requirement description:</b> The application must verify permissions before accessing services provided by external applications and handle denied requests gracefully.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ Accessing External Services: Ensure that any call to a service provided by an external app checks for proper permissions before initiating the request. If necessary, fail gracefully when access is denied. [18].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Validating permissions for external service calls prevents unauthorized access and mitigates potential security risks. Gracefully handling denied requests ensures application stability and enhances the user experience.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Permission Validation Test:</b> <ul style="list-style-type: none"> <li>✓ Simulate a request to an external service requiring permissions.</li> <li>✓ Confirm that the application verifies the necessary permissions before initiating the request.</li> </ul> </li> <li><b>2. Denied Request Handling Test:</b> <ul style="list-style-type: none"> <li>✓ Attempt a request to an external service without the required permissions.</li> <li>✓ Verify that the application fails gracefully by providing appropriate error messages or fallback mechanisms.</li> </ul> </li> <li><b>3. Unauthorized Access Prevention:</b> <ul style="list-style-type: none"> <li>✓ Ensure that unauthorized requests to external services are logged and prevented from execution.</li> </ul> </li> </ol>

<b>4. Permission Change Response Test:</b>
✓ Test how the application reacts to changes in granted permissions during runtime.
✓ Confirm the application adjusts its behavior based on updated permission statuses.
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda
José L Fernández-Alemán
Juan Manuel Carrillo-de-Gea
Joaquín Nicolás
08/01/2026

<b>PUID:</b> [SECM-CAT-IAA-086]
<b>Requirement description:</b> The application must comply with Android's guidelines for bound services when using Binder or Messenger to perform inter-process communication (IPC).
<b>Source:</b>
✓ Compliance with Bound Services: Follow Android's guidelines for bound services when using Binder or Messenger to perform IPC with a service. [18].
<b>Priority:</b> Not described
<b>Rationale:</b> Adhering to Android's guidelines for bound services ensures secure and efficient inter-process communication (IPC). This minimizes the risks of unauthorized access, data leakage, and potential application instability due to misconfiguration.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<b>1. Service Binding Test:</b>
✓ Validate that the application uses Android's recommended methods for binding to services.
✓ Confirm that the binding process follows Android's documented guidelines for Binder and Messenger objects.
<b>2. Permission Validation Test:</b>
✓ Verify that the application enforces appropriate access controls when interacting with bound services.
<b>3. Error Handling Validation:</b>
✓ Simulate binding failures to a service.
✓ Ensure the application provides proper error handling and fallback mechanisms.
<b>4. Security Review:</b>

<ul style="list-style-type: none"> <li>✓ Inspect the implementation to confirm that IPC communication via Binder or Messenger adheres to Android's security best practices.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IIA-087]
<b>Requirement description:</b> The application must ensure secure intent broadcasting by using explicit intents for specific recipients and applying appropriate controls when using ordered broadcasts to prevent unauthorized interception
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ Secure Intent Broadcasting: When broadcasting intents, consider using sendBroadcast or sendOrderedBroadcast based on application requirements.           <ul style="list-style-type: none"> <li>○ If a specific recipient must receive the broadcast, use an explicit intent and specify the receiver by name.</li> <li>○ Be cautious with ordered broadcasts, as they can be consumed by a recipient and might not be delivered to all intended applications [18].</li> </ul> </li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Securing intent broadcasting prevents unauthorized interception and misuse of sensitive information shared between application components. This ensures that broadcasts reach the intended recipient and maintains the integrity and confidentiality of the transmitted data.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Broadcast Security Test:</b> <ul style="list-style-type: none"> <li>✓ Verify that explicit intents are used when broadcasting sensitive data.</li> <li>✓ Confirm that unauthorized receivers cannot intercept the broadcast.</li> </ul> </li> <li><b>2. Ordered Broadcast Validation:</b> <ul style="list-style-type: none"> <li>✓ Simulate scenarios with ordered broadcasts to ensure only authorized recipients process the intent.</li> <li>✓ Confirm fallback mechanisms for undelivered broadcasts.</li> </ul> </li> </ol>

<b>3. Intent Usage Review:</b>
✓ Perform a code review to verify the use of sendBroadcast and sendOrderedBroadcast and ensure they align with the defined security practices.
<b>4. Recipient Validation Test:</b>
✓ Validate that the application correctly specifies intended recipients when broadcasting explicit intents.
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IAA-088]
<b>Requirement description:</b> The application must integrate with Android's Credential Manager to provide unified access to authentication methods, including passkeys, passwords, and federated sign-in solutions, ensuring seamless and secure user authentication.
<b>Source:</b>
✓ The system should integrate with Android's Credential Manager to support unified API access for major authentication methods, including passkeys, passwords, and federated sign-in solutions like Google Sign-in [18].
<b>Priority:</b> Not described
<b>Rationale:</b> Integrating with Android's Credential Manager ensures consistent and secure handling of various authentication methods while streamlining user experience. This approach centralizes authentication processes, reduces security risks, and enhances compatibility with modern authentication standards.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>Integration Test:</b> <ul style="list-style-type: none"> <li>✓ Verify that the application correctly integrates with Android's Credential Manager and supports authentication methods like passkeys, passwords, and federated sign-in solutions.</li> </ul> </li> <li><b>Unified Interface Validation:</b> <ul style="list-style-type: none"> <li>✓ Confirm that all authentication requests are processed through a unified access interface.</li> </ul> </li> </ol>

<ul style="list-style-type: none"> <li>✓ Validate the functionality of the service-request handler module in routing requests securely and accurately.</li> </ul>
<b>3. Authentication Compatibility Test:</b>
<ul style="list-style-type: none"> <li>✓ Ensure compatibility with major authentication providers and methods supported by Android Credential Manager.</li> <li>✓ Validate that user credentials are securely stored and accessed via the Credential Manager.</li> </ul>
<b>4. Security Review:</b>
<ul style="list-style-type: none"> <li>✓ Perform a security audit to verify that the Credential Manager integration adheres to Android's security standards.</li> <li>✓ Confirm that sensitive data is not exposed during authentication processes.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IAA-089]
<b>Requirement description:</b> The application must implement passkeys as a secure and user-friendly alternative to traditional passwords to enhance authentication processes and user experience.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ The Implement passkeys: Enable passkeys as a more secure and user-friendly upgrade to passwords [18].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Passkeys offer enhanced security compared to traditional passwords by leveraging cryptographic mechanisms that reduce the risk of credential theft, phishing attacks, and other common vulnerabilities. They also improve user convenience by simplifying authentication.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Passkey Enablement Test:</b> <ul style="list-style-type: none"> <li>✓ Verify that the application supports the creation and use of passkeys for authentication.</li> </ul> </li> <li><b>2. Cryptographic Validation:</b> <ul style="list-style-type: none"> <li>✓ Ensure that passkeys are generated using secure cryptographic methods and stored securely on the user's device or in a secure hardware element.</li> </ul> </li> </ol>

<b>3. Authentication Flow Testing:</b>
✓ Validate that the application seamlessly uses passkeys during authentication, ensuring compatibility across supported devices and platforms.
<b>4. Security Assessment:</b>
✓ Conduct a security review to ensure passkeys cannot be intercepted, duplicated, or exploited by malicious actors.
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda
José L Fernández-Alemán
Juan Manuel Carrillo-de-Gea
Joaquín Nicolás
08/01/2026

<b>PUID:</b> [SECM-CAT-IIA-090]
<b>Requirement description:</b> The application must integrate with Android's autofill framework to streamline authentication processes, securely enabling users to store and retrieve complex passwords via password managers while reducing user errors and friction.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ Autocomplete Framework Integration: The system must integrate with Android's autocomplete framework to streamline the sign-up and sign-in process, allowing users to securely store and retrieve complex passwords via password managers [18].</li> <li>✓ Error Reduction via Autocomplete: The system must reduce user friction and error rates during authentication by leveraging autocomplete to automatically fill in credentials, supporting the use of randomized, complex passwords [18].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Integrating with Android's autocomplete framework enhances both security and user experience by enabling the use of randomized, complex passwords while minimizing manual input errors. This approach supports modern password management practices, promoting better overall application security and usability.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Integration Testing:</b></li> </ol> <ul style="list-style-type: none"> <li>✓ Verify the application integrates with Android's autocomplete framework, enabling autocomplete functionality for credentials during the sign-up and sign-in processes.</li> </ul>

<p><b>2. Password Manager Compatibility Check:</b></p> <ul style="list-style-type: none"> <li>✓ Confirm that the application supports secure interaction with popular password managers for storing and retrieving complex passwords.</li> </ul> <p><b>3. Error Rate Assessment:</b></p> <ul style="list-style-type: none"> <li>✓ Measure the reduction in authentication errors by comparing manual input processes to autofill-enabled workflows.</li> </ul> <p><b>4. Security Validation:</b></p> <ul style="list-style-type: none"> <li>✓ Ensure credentials autofilled into the application are transmitted securely using encrypted communication channels (e.g., HTTPS).</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IAA-091]
<b>Requirement description:</b> The application must support biometric authentication methods, such as fingerprint scans or facial recognition, to enhance security for sensitive data in finance, healthcare, or identity management applications.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ The application must support biometric authentication methods, such as fingerprint scans or facial recognition, for enhanced security, particularly in apps dealing with sensitive data like finance, healthcare, or identity management [18].</li> <li>✓ Use of access passwords, PIN scramble, or fingerprint to verify user identity [25].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Biometric authentication provides a secure and user-friendly way to verify identity, reducing reliance on traditional passwords and PINs. This is especially important in applications handling sensitive information, where additional layers of protection are required.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<p><b>1. Biometric Enrollment Testing:</b></p> <ul style="list-style-type: none"> <li>✓ Verify that the application allows users to register biometric credentials (e.g., fingerprint or facial recognition) securely.</li> </ul>

<p><b>2. Biometric Authentication Testing:</b></p> <ul style="list-style-type: none"> <li>✓ Confirm the application supports biometric authentication during login or sensitive actions (e.g., accessing financial or healthcare data).</li> </ul> <p><b>3. Fallback Authentication Testing:</b></p> <ul style="list-style-type: none"> <li>✓ Ensure users can access the application using secure fallback methods, such as PIN or password if biometric methods are unavailable.</li> </ul> <p><b>4. Security Testing:</b></p> <ul style="list-style-type: none"> <li>✓ Validate that biometric data is stored and transmitted securely, adhering to platform-specific best practices and encryption standards.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IAA-092]
<b>Requirement description:</b> The application must implement FaceID and TouchID to unlock biometrically locked secrets and securely protect sensitive authentication materials, such as session tokens.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ Reinforce Authentication: 3. Use FaceID and TouchID to unlock biometrically locked secrets and securely protect sensitive authentication materials, like session tokens [3].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Using biometric methods like FaceID and TouchID enhances the security of sensitive authentication materials, ensuring that session tokens and other critical secrets are protected against unauthorized access. This approach minimizes risks associated with stolen or leaked credentials.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Biometric Unlock Testing:</b></li> </ol> <ul style="list-style-type: none"> <li>✓ Verify that FaceID and TouchID can successfully unlock biometrically secured secrets, such as session tokens when configured.</li> </ul> <ol style="list-style-type: none"> <li><b>2. Fallback Method Testing:</b></li> </ol> <ul style="list-style-type: none"> <li>✓ Ensure a secure fallback method, such as a PIN or password, is available when biometric unlocking is not accessible.</li> </ul>

<p><b>3. Data Security Testing:</b></p> <ul style="list-style-type: none"><li>✓ Confirm that sensitive materials, such as session tokens, are encrypted and stored securely in the device's secure enclave or equivalent hardware-backed storage.</li></ul> <p><b>4. Unauthorized Access Testing:</b></p> <ul style="list-style-type: none"><li>✓ Test that access to biometrically locked secrets is denied when an invalid biometric attempt is made, ensuring no data leaks occur.</li></ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IIA-093]
<b>Requirement description:</b> The application must validate the presence of enrolled biometric data, such as registered fingerprints or iris scans, before enabling API access for biometric authentication purposes.
<b>Source:</b>
<ul style="list-style-type: none"><li>✓ 13.3. Ensure that there are enrolled data using the biometric sensor (e.g., user's fingerprints and/or user's iris are registered) before using the API for authentication purposes [16].</li></ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Requiring enrolled biometric data before enabling API-based authentication ensures that biometric features are effectively configured and reduces the risk of authentication failures or unauthorized access.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"><li><b>1. Biometric Data Verification:</b><ul style="list-style-type: none"><li>✓ Confirm that the application verifies the existence of enrolled biometric data (e.g., fingerprints or iris scans) before activating biometric authentication APIs.</li></ul></li><li><b>2. Enrollment Prompt Testing:</b><ul style="list-style-type: none"><li>✓ Verify that users are prompted to enroll their biometrics if no data is detected when attempting to enable biometric authentication.</li></ul></li><li><b>3. API Restriction Testing:</b></li></ol>

<ul style="list-style-type: none"> <li>✓ Ensure that the biometric authentication API is inaccessible unless valid biometric data has been enrolled on the device.</li> </ul> <p><b>4. Unauthorized Access Testing:</b></p> <ul style="list-style-type: none"> <li>✓ Tests that attempt to authenticate using biometric APIs without prior enrollment are blocked securely.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IAA-094]
<b>Requirement description:</b> The application must implement secure biometric authentication methods, such as fingerprint or facial recognition, ensuring compatibility with biometric sensors and providing alternative authentication controls when biometric hardware is unavailable.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ The application must support biometric authentication methods, such as fingerprint scans or facial recognition, for enhanced security, particularly in apps dealing with sensitive data like finance, healthcare, or identity management [18].</li> <li>✓ Add biometrics: Offer the ability to use biometric authentication such as fingerprint or facial recognition for added security [18].</li> <li>✓ 4.6: Verify that biometric authentication, if any, is not event-bound (i.e. using an API that simply returns "true" or "false"). Instead, it is based on unlocking the keychain/keystore [19].</li> <li>✓ 13.1. Always verify that there is a biometric sensor (e.g., Fingerprint reader) present and available on the device before using the API for authentication purposes. In the case that the sensor is not available, an alternative authentication control should be provided [16].</li> <li>✓ MASVS- AUTH -2: The app performs local authentication securely according to the platform best practices: Many apps allow users to authenticate via biometrics or a local PIN code. These authentication mechanisms need to be correctly implemented. Additionally, some apps might not have a remote endpoint, and rely fully on local app authentication [20].</li> <li>✓ MASVS-AUTH-3: The app secures sensitive operations with additional authentication: Some additional form of authentication is often desirable for sensitive actions inside the app. This can be done in different ways (biometric, pin, MFA code generator, email, deep links, etc) and they all need to be implemented securely [20].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Implementing robust biometric authentication methods strengthens application security for sensitive operations, ensuring that user credentials are protected and fallback authentication mechanisms are available in the absence of biometric hardware.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0

<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Biometric Sensor Validation:</b> <ul style="list-style-type: none"> <li>✓ Verify that the application detects the presence of a biometric sensor before activating biometric authentication.</li> </ul> </li> <li><b>2. Fallback Mechanism Testing:</b> <ul style="list-style-type: none"> <li>✓ Confirm that alternative authentication methods (e.g., PIN, password) are provided when biometric hardware is unavailable.</li> </ul> </li> <li><b>3. Secure API Validation:</b> <ul style="list-style-type: none"> <li>✓ Ensure biometric authentication APIs utilize secure mechanisms such as unlocking the keychain/keystore rather than simple "true/false" responses.</li> </ul> </li> <li><b>4. Authentication Flow Testing:</b> <ul style="list-style-type: none"> <li>✓ Test that biometric authentication supports sensitive operations, such as accessing user data, with additional authentication when required.</li> </ul> </li> <li><b>5. Unauthorized Access Prevention:</b> <ul style="list-style-type: none"> <li>✓ Validate that unauthorized users cannot bypass biometric authentication or fallback mechanisms to gain access to sensitive app features.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IAA-095]
<b>Requirement description:</b> The application must enforce step-up authentication for actions involving sensitive data or transactions, requiring additional authentication before granting access or execution.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ 4.9: Verify that step-up authentication is required to enable actions that deal with sensitive data or transactions [19].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Step-up authentication provides an additional layer of security for critical operations involving sensitive data or high-value transactions, mitigating the risk of unauthorized access and enhancing overall application security.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0

<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<b>1. Sensitive Action Authentication Testing:</b> ✓ Confirm that the application requires additional authentication (e.g., biometric, PIN, or MFA) for actions involving sensitive data or high-value transactions.
<b>2. Authorization Flow Validation:</b> ✓ Verify that unauthorized users cannot perform sensitive actions without passing the step-up authentication process.
<b>3. Session Authentication Testing:</b> ✓ Test that step-up authentication is re-initiated after session expiration or when critical thresholds for inactivity are reached.
<b>4. Fallback Mechanism Testing:</b> ✓ Ensure that the step-up authentication process provides alternative methods (e.g., OTP or password) if the primary method fails or is unavailable.
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IAA-096]
<b>Requirement description:</b> The application must ensure authentication requirements for mobile applications are consistent with their web application counterparts, maintaining equivalent or greater authentication factors.
<b>Source:</b> ✓ 4.9: Avoid Weak Patterns: 1. If you are porting a web application to a mobile equivalent, ensure the authentication requirements of mobile applications match that of the web application component. It should not be possible to authenticate with fewer factors than the web browser [3].
<b>Priority:</b> Not described
<b>Rationale:</b> Ensuring equivalent authentication requirements across web and mobile platforms prevents potential security gaps, such as using fewer authentication factors on mobile applications, which could lead to unauthorized access or compromised security.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0

<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Authentication Factor Comparison:</b> <ul style="list-style-type: none"> <li>✓ Confirm that the authentication factors required by the mobile application are equal to or stronger than those required by the web application counterpart.</li> </ul> </li> <li><b>2. Mobile Authentication Testing:</b> <ul style="list-style-type: none"> <li>✓ Validate that the mobile application enforces all authentication steps present in the web version, including multi-factor authentication (if applicable).</li> </ul> </li> <li><b>3. Session Synchronization Testing:</b> <ul style="list-style-type: none"> <li>✓ Verify that user sessions initiated on the mobile app reflect the same authentication security measures as those initiated via the web application.</li> </ul> </li> <li><b>4. Fallback Mechanism Evaluation:</b> <ul style="list-style-type: none"> <li>✓ Ensure no fallback mechanism on the mobile application circumvents the authentication steps present in the web application.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IIA-097]
<b>Requirement description:</b> The mobile application must implement organization-defined out-of-band authentication to enhance security under specified conditions.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ SRG-APP-000393-MAPP-000100: The mobile app must implement organization-defined out-of-band authentication under organization-defined conditions [17].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Out-of-band authentication provides an additional layer of security by leveraging an independent communication channel, reducing the risk of compromise in cases of session hijacking, phishing, or other credential-related attacks. It ensures that critical operations or sensitive access requests are verified through an external, secure mechanism.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described

<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Out-of-Band Trigger Testing:</b> <ul style="list-style-type: none"> <li>✓ Verify that the application correctly triggers out-of-band authentication under organization-defined conditions, such as high-risk transactions or unusual login attempts.</li> </ul> </li> <li><b>2. Communication Channel Validation:</b> <ul style="list-style-type: none"> <li>✓ Ensure that the out-of-band authentication mechanism uses an independent and secure communication channel (e.g., SMS, email, or a dedicated app).</li> </ul> </li> <li><b>3. Authentication Confirmation Check:</b> <ul style="list-style-type: none"> <li>✓ Confirm that the user must successfully complete the out-of-band authentication process before being granted access or permission for the specified action.</li> </ul> </li> <li><b>4. Fallback Mechanism Testing:</b> <ul style="list-style-type: none"> <li>✓ Verify the presence of a secure fallback mechanism in cases where the out-of-band channel is unavailable, ensuring no bypass of authentication requirements.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IAA-098]
<b>Requirement description:</b> The application must enforce multifactor authentication (MFA) for network access to privileged accounts and sensitive data, incorporating at least two distinct authentication factors for enhanced security.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ V-222523: The application must use multifactor (Alt. Token) authentication for network access to privileged accounts. Configure the application to use an Alt. Token when providing network access to privileged application accounts [15].</li> <li>✓ 4.3.1. access control:</li> <li>✓ multifactor authentication: each system where a typical patient, doctor, or health IT administrator must interact with patient records, systems, or networks requires at least a certificate, username, and password to access [37].</li> <li>✓ 9.4.1 Information access restriction: Health information systems processing personal health information shall authenticate users and should do so by means of authentication involving at least two factors [7].</li> <li>✓ MASVS-AUTH-3: The app secures sensitive operations with additional authentication: Some additional form of authentication is often desirable for sensitive actions inside the app.</li> </ul>

<p>This can be done in different ways (biometric, pin, MFA code generator, email, deep links, etc) and they all need to be implemented securely [20].</p> <ul style="list-style-type: none"> <li>✓ 1.6. Consider re-evaluating access authorization to sensitive data based on contextual information such as location (e.g., require further authentication if location data shows device is outside of the expected region) [16].</li> <li>✓ 2.9. Use context to add security to authentication (e.g., geo location, IP location, etc). Ensure that any collected data is in compliance with the local laws and regulatory requirements [16].</li> <li>✓ 2.10. Consider using additional authentication factors for applications giving access to sensitive data or interfaces where possible: <ul style="list-style-type: none"> <li>○ Knowledge factors - Something you know (i.e. user's secret question)</li> <li>○ Possession factors - Something you have (i.e. hardware token, grid, sim card)</li> <li>○ Inherence factors - Something you are (i.e. fingerprint, voice, facial, retina recognition) [16]</li> </ul> </li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Multifactor authentication (MFA) ensures secure access to privileged accounts and sensitive data by requiring at least two different authentication factors. This mitigates risks associated with compromised credentials, enhances user identity validation, and complies with regulations such as HIPAA for health information systems.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b> <ol style="list-style-type: none"> <li>1. <b>MFA Policy Enforcement Testing:</b> <ul style="list-style-type: none"> <li>✓ Verify that MFA is enabled for all privileged accounts and sensitive data access points.</li> <li>✓ Confirm that at least two authentication factors (knowledge, possession, inherence) are required during login.</li> </ul> </li> <li>2. <b>Context-Aware Access Testing:</b> <ul style="list-style-type: none"> <li>✓ Test location-based authentication by attempting to log in from authorized and unauthorized regions. Ensure that additional authentication steps are triggered in unexpected locations.</li> </ul> </li> <li>3. <b>Token and Certificate Validation:</b> <ul style="list-style-type: none"> <li>✓ Validate the use of alternate tokens or certificates for network access to privileged accounts, ensuring they are FIPS-compliant and securely managed.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda

José L Fernández-Alemán  
 Juan Manuel Carrillo-de-Gea  
 Joaquín Nicolás  
 08/01/2026

**PUID:** [SECM-CAT-IIA-99]

**Requirement description:** The application must accept and electronically verify Personal Identity Verification (PIV) credentials, enforce multifactor authentication (MFA) using CAC or Alt. Token for local and network access, and support PIV credentials from other federal agencies.

**Source:**

- ✓ V-222524: The application must accept Personal Identity Verification (PIV) credentials. Configure the application to require CAC authentication [15].
- ✓ V-222525: The application must electronically verify Personal Identity Verification (PIV) credentials. Configure the application to require CAC authentication [15].
- ✓ V-222526: The application must use multifactor (e.g., CAC, Alt. Token) authentication for network access to non-privileged accounts. Configure the application to require CAC or Alt. Token authentication for non-privileged network access to non-privileged accounts [15].
- ✓ SRG-APP-000392-MAPP-000100: The mobile app must electronically verify Personal Identity Verification (PIV) credentials [17].
- ✓ V-222527: The application must use multifactor (Alt. Token) authentication for local access to privileged accounts. Configure the application to only use Alt. Tokens when locally accessing privileged application accounts [15].
- ✓ V-222528: The application must use multifactor (e.g., CAC, Alt. Token) authentication for local access to non-privileged accounts. Configure the application to require CAC or Alt. Token authentication for non-privileged network access [15].
- ✓ V-222557: The application must accept Personal Identity Verification (PIV) credentials from other federal agencies. Configure the application to accept PIV credentials when utilizing authentication provided by Federal (Non-DoD) agencies [15].
- ✓ V-222558: The application must electronically verify Personal Identity Verification (PIV) credentials from other federal agencies. Configure the application to verify the PIV credentials presented when utilizing authentication provided by Federal (Non-DoD) agencies [15].

**Priority:** Not described

**Rationale:** Implementing multifactor authentication with PIV or CAC credentials ensures secure access to application resources, supports interoperability with federal agencies, and complies with DoD authentication requirements. This approach mitigates unauthorized access risks and strengthens the application's security posture.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. PIV Credential Acceptance and Verification:</b> <ul style="list-style-type: none"> <li>✓ Verify that the application accepts and electronically verifies valid PIV credentials for both privileged and non-privileged users.</li> <li>✓ Confirm interoperability by validating PIV credentials issued by other federal agencies.</li> </ul> </li> <li><b>2. MFA Implementation Testing:</b> <ul style="list-style-type: none"> <li>✓ Ensure multifactor authentication is enforced for all privileged and non-privileged accounts during local and network access, using CAC or Alt. Token credentials.</li> </ul> </li> <li><b>3. Access Controls Testing:</b> <ul style="list-style-type: none"> <li>✓ Confirm that non-privileged accounts cannot access privileged resources without MFA.</li> <li>✓ Validate that session initiation is blocked for invalid or expired PIV credentials.</li> </ul> </li> <li><b>4. Interoperability Validation:</b> <ul style="list-style-type: none"> <li>✓ Test the application's ability to accept and authenticate PIV credentials from Federal (Non-DoD) agencies without errors or restrictions.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IAA-100]
<b>Requirement description:</b> The application must enforce server-side authorization controls to prevent bypass of client-side authorization mechanisms, ensuring secure access to protected resources.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ Insecure Authorization Prevention: 2. Assume that all client-side authorization can be bypassed, hence reinforcing server-side authorization controls whenever possible [3].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Client-side authorization mechanisms are inherently vulnerable to bypass attacks, as malicious actors can manipulate client-side code or data to gain unauthorized access. Enforcing robust server-side authorization ensures that all access requests are validated, mitigating the risks of unauthorized access to sensitive resources or operations.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis

<b>Validation criteria:</b>
<b>1. Server-Side Authorization Validation:</b>
✓ Ensure all critical actions and data access requests are validated against server-side authorization policies before processing.
✓ Verify that no client-side modifications (e.g., changing parameters) can bypass server-side validation.
<b>2. Penetration Testing:</b>
✓ Conduct tests simulating client-side bypass attempts, such as manipulating API requests or session tokens, and confirm the server rejects unauthorized requests.
<b>3. Source Code Review:</b>
✓ Review server-side implementation to verify that authorization checks are consistently applied for all protected endpoints.
<b>4. Role-Based Access Control Testing:</b>
✓ Confirm that users only access resources and operations they are explicitly authorized for, based on server-side role and permission checks.
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IAA-101]
<b>Requirement description:</b> The application must implement a clear separation of permissions for client-side and server-side components, ensuring accurate definition and enforcement of permissions for both invoking and granting applications.
<b>Source:</b>
✓ Client and Server Permission Separation: Ensure that permissions are properly defined for both the client and server sides. Verify that the app invoking the service has the required permissions, and confirm that your app grants the correct permissions to the calling app [18].
<b>Priority:</b> Not described
<b>Rationale:</b> Proper permission separation between client and server components is essential to prevent unauthorized access and ensure that only trusted applications can invoke sensitive services. This minimizes security risks, such as privilege escalation and data exposure, by verifying permissions at both ends of the communication.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined

<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Permission Validation on Invocation:</b> <ul style="list-style-type: none"> <li>✓ Verify that the app invoking the service has the required permissions to access the service.</li> <li>✓ Simulate scenarios where an app without the correct permissions attempts to invoke the service and confirm that access is denied.</li> </ul> </li> <li><b>2. Server-Side Permission Enforcement:</b> <ul style="list-style-type: none"> <li>✓ Validate that the server-side components strictly enforce permission checks for all incoming requests.</li> <li>✓ Confirm that permissions are verified before any data or functionality is accessed.</li> </ul> </li> <li><b>3. Granular Permission Assignment:</b> <ul style="list-style-type: none"> <li>✓ Review the application's permission structure to ensure that permissions granted to calling applications are appropriately scoped and do not grant excessive access.</li> </ul> </li> <li><b>4. Source Code Inspection:</b> <ul style="list-style-type: none"> <li>✓ Conduct a review of client and server-side code to verify that permissions are defined and enforced consistently in both components.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IAA-102]
<b>Requirement description:</b> The application must enforce all authentication and authorization controls on the server side to ensure security, independent of any client-side enforcement.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ 1.3: Verify that security controls are never enforced only on the client side, but on the respective remote endpoints [19].</li> <li>✓ 2.1. Do not rely on client-side security controls. Application controls can be easily tampered by an adversary. Both authentication and authorization controls should be implemented on the server side [16].</li> <li>✓ Insecure Authorization Prevention: 1. Backend systems should independently verify the roles and permissions of the authenticated user. Do not rely on any roles or permission information that comes from the mobile device [3].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Client-side security controls can be tampered with by attackers, leading to unauthorized access and exploitation of application resources. Enforcing authentication and authorization on the server side ensures a robust and independent layer of protection that cannot be easily bypassed.
<b>Number of Children:</b> 0

<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Client-Side Tampering Test:</b> <ul style="list-style-type: none"> <li>✓ Modify client-side application controls to bypass authentication or authorization.</li> <li>✓ Confirm that unauthorized access is blocked by server-side security mechanisms.</li> </ul> </li> <li><b>2. Role and Permission Validation:</b> <ul style="list-style-type: none"> <li>✓ Ensure the backend system independently validates user roles and permissions for all requests.</li> <li>✓ Simulate scenarios where incorrect or manipulated role information is sent from the client and verify that the server denies the request.</li> </ul> </li> <li><b>3. Server-Side Enforcement Audit:</b> <ul style="list-style-type: none"> <li>✓ Review server-side implementation to confirm that all security controls, including authentication and authorization, are fully enforced on remote endpoints.</li> </ul> </li> <li><b>4. Source Code Review:</b> <ul style="list-style-type: none"> <li>✓ Inspect server-side code to verify independent validation of user roles and permissions without reliance on client-side inputs.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IAA-103]
<b>Requirement description:</b> The backend platform must operate with a hardened configuration and apply the latest security patches to the operating system, web server, and other application components to ensure flaw remediation and maintain a secure environment.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ 5.4. Ensure that the back-end platform (server) is running with a hardened configuration with the latest security patches applied to the OS, web server and other application components [16].</li> <li>✓ SI-2 FLAW REMEDIATION</li> </ul> <p><b>Control:</b></p> <ol style="list-style-type: none"> <li>a. Identify, report, and correct system flaws;</li> </ol>

- b. Test software and firmware updates related to flaw remediation for effectiveness and potential side effects before installation;
- c. Install security-relevant software and firmware updates within [Assignment: organization-defined time period] of the release of the updates; and
- d. Incorporate flaw remediation into the organizational configuration management process.

**Discussion:** The need to remediate system flaws applies to all types of software and firmware. Organizations identify systems affected by software flaws, including potential vulnerabilities resulting from those flaws, and report this information to designated organizational personnel with information security and privacy responsibilities. Security-relevant updates include patches, service packs, and malicious code signatures. Organizations also address flaws discovered during assessments, continuous monitoring, incident response activities, and system error handling. By incorporating flaw remediation into configuration management processes, required remediation actions can be tracked and verified [11].

**Priority:** Not described

**Rationale:** Ensuring the backend platform is secure and up to date minimizes vulnerabilities that can be exploited by attackers. Applying the latest patches and maintaining a hardened configuration mitigates risks arising from unpatched flaws and improves the overall security posture of the application.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

**1. Patch Management Test:**

- ✓ Verify that the backend platform has the latest security patches applied to the operating system, web server, and other application components.
- ✓ Validate that a process exists to regularly check for and apply new patches.

**2. Configuration Hardening Audit:**

- ✓ Inspect the backend configuration to ensure adherence to best practices for hardening, such as disabling unused ports, enforcing least privilege, and implementing robust firewall rules.

**3. Vulnerability Scanning:**

- ✓ Perform regular vulnerability scans on the backend to identify and remediate system flaws promptly.
- ✓ Confirm that identified vulnerabilities are addressed in a timely manner as per the organization-defined timeline.

**4. Update Effectiveness Review:**

- ✓ Test applied patches and updates for their effectiveness and verify that they do not introduce regressions or new vulnerabilities.

**5. Configuration Management Process Review:**

✓ Ensure that the flaw remediation process is integrated into the organization's configuration management, with tracking and documentation of applied updates.
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IIA-104]
<b>Requirement description:</b> The application must securely manage user sessions after initial authentication by implementing randomly generated session tokens, enforcing proper session timeouts, and securely storing session data on both client and server sides.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ 2.6. Ensure that the session management is handled securely after the initial authentication, using appropriate secure protocols [16].</li> <li>✓ Apply Secure Session Management: 1. Implement secure session management techniques, such as using randomly generated session tokens, setting proper session timeouts, and securely storing session data on the client and server sides [3].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Secure session management is critical to maintaining the integrity and confidentiality of user interactions with the application. Proper handling of session tokens and timeouts prevents unauthorized access, session hijacking, and other potential threats.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li>1. <b>Session Token Security Test:</b> <ul style="list-style-type: none"> <li>✓ Verify that session tokens are randomly generated, unique, and securely transmitted over encrypted channels (e.g., HTTPS).</li> </ul> </li> <li>2. <b>Timeout Enforcement Test:</b> <ul style="list-style-type: none"> <li>✓ Confirm that user sessions are automatically terminated after a predefined period of inactivity and upon user logout.</li> </ul> </li> <li>3. <b>Client-Side Session Data Handling Review:</b> <ul style="list-style-type: none"> <li>✓ Ensure session data stored on the client side is securely managed, using encrypted storage or other secure mechanisms.</li> </ul> </li> </ol>

<b>4. Server-Side Session Validation Test:</b>
✓ Validate that the server-side securely maintains session data and invalidates expired or logged-out sessions promptly.
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda
José L Fernández-Alemán
Juan Manuel Carrillo-de-Gea
Joaquín Nicolás
08/01/2026

<b>PUID:</b> [SECM-CAT-IAA-105]
<b>Requirement description:</b> The application must notify users of all login activities associated with their accounts and provide a view of devices accessing the account, with the capability to block specific devices.
<b>Source:</b>
✓ 4.10: Verify that the app informs the user of all login activities with his or her account. Users are able view a list of devices used to access the account, and to block specific devices [19].
<b>Priority:</b> Not described
<b>Rationale:</b> Enabling users to monitor account activity and control access improves security by helping users detect unauthorized logins and revoke access to suspicious devices.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<b>1. Login Notification Test:</b>
✓ Verify that users are notified in real-time of all login activities with details such as the device, location, and timestamp.
<b>2. Device Access List Review Test:</b>
✓ Ensure users can view a complete and accurate list of devices that have accessed their account, including the last access timestamp.
<b>3. Device Blocking Test:</b>

<ul style="list-style-type: none"> <li>✓ Confirm that users can block access to their account from specific devices and that blocked devices are immediately unable to access the account.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IIA-106]
<b>Requirement description:</b> The application must display the time and date of the last successful logon and details of any unsuccessful logon attempts since the last successful logon to improve user awareness and account security.
<b>Source:</b> <ul style="list-style-type: none"> <li>✓ V-222437: The application must display the time and date of the users last successful logon. Design and configure the application to display the date and time when the user was last successfully granted access to the application [15].</li> <li>✓ AC-9 PREVIOUS LOGON NOTIFICATION Control: Notify the user, upon successful logon to the system, of the date and time of the last logon [11].</li> <li>✓ 9.4.2 Secure log-on procedures: h) display the following information on completion of a successful log-on:           <ul style="list-style-type: none"> <li>1) date and time of the previous successful log-on;</li> <li>2) details of any unsuccessful log-on attempts since the last successful log-on; [7].</li> </ul> </li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Displaying the last successful logon date and time along with unsuccessful attempts informs users of potential unauthorized access attempts, enabling immediate reporting or corrective action.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b> <ol style="list-style-type: none"> <li><b>1. Successful Logon Notification Test:</b> <ul style="list-style-type: none"> <li>✓ Verify that the application displays the date and time of the last successful logon immediately after a new successful logon.</li> </ul> </li> <li><b>2. Unsuccessful Attempts Notification Test:</b></li> </ol>

<ul style="list-style-type: none"> <li>✓ Ensure the application lists the number and details (timestamps, source) of any unsuccessful logon attempts since the last successful logon.</li> </ul> <p><b>3. Alert Test for Suspicious Activity:</b></p> <ul style="list-style-type: none"> <li>✓ Confirm that users are notified of suspicious activity if a high number of unsuccessful logon attempts are detected before the last successful logon.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IAA-107]
<b>Requirement description:</b> Remote endpoints must verify that connecting clients are using the current version of the mobile application to ensure compatibility and security.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ 1.12: Verify that remote endpoints ensure that connecting clients use the current version of the mobile app [19].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Enforcing the use of the latest mobile app version at remote endpoints mitigates risks associated with outdated app versions, such as known vulnerabilities, compatibility issues, and lack of updated security features.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Version Verification Test:</b> <ul style="list-style-type: none"> <li>✓ Verify that the remote endpoint validates the version of the connecting client during each connection attempt.</li> </ul> </li> <li><b>2. Connection Denial Test:</b> <ul style="list-style-type: none"> <li>✓ Ensure that clients using outdated app versions are denied access with a clear message to update to the latest version.</li> </ul> </li> <li><b>3. Update Guidance Test:</b> <ul style="list-style-type: none"> <li>✓ Verify that users attempting to connect with outdated versions receive actionable guidance to update their application</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer

<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IAA-108]
<b>Requirement description:</b> The application must utilize mutual authentication when endpoint device non-repudiation protections are required by DoD policy or by the data owner.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ 1.12: Verify that remote endpoints ensure that connecting clients use the current version of the mobile app [19].</li> <li>✓ 5) Mutual Authentication: In the design of epidemic situation applications, mutual authentication is the primary and principal safety feature, which ensures that both units (Fog device, IoT device, Cloud center, etc.) in a session of communication can mutually authenticate each other. Thus, the identity theft attacks to specific equipment can be prevented [31].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Mutual authentication ensures that both the client and server verify each other's identities, which is critical for protecting sensitive data and ensuring that endpoint devices cannot deny their involvement in secure transactions. This is essential when non-repudiation protections are mandated by policy or the data owner.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Mutual Authentication Implementation Test:</b> <ul style="list-style-type: none"> <li>✓ Verify that mutual authentication is enforced in accordance with the data protection requirements defined by DoD policy or the data owner.</li> </ul> </li> <li><b>2. Non-Repudiation Test:</b> <ul style="list-style-type: none"> <li>✓ Test that the application logs and retains relevant information to support non-repudiation, ensuring both client and server identities are verified and recorded.</li> </ul> </li> <li><b>3. Security Failure Test:</b> <ul style="list-style-type: none"> <li>✓ Validate that access is denied if either the client or server fails to properly authenticate during a mutual authentication process.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer

<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IAA-109]
<b>Requirement description:</b> The application must authenticate all network-connected endpoint devices and perform acceptable user authentication, such as username/password, before establishing connections with a remote service.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ 4.1: Verify that if the app provides users with access to a remote service, an acceptable form of authentication such as username/password authentication is performed at the remote endpoint [19].</li> <li>✓ V-222533: The application must authenticate all network connected endpoint devices before establishing any connection. Configure the application to authenticate all network connected endpoint devices/service consumers before establishing connections [15].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Authenticating all endpoint devices and users ensures the integrity and security of network connections by preventing unauthorized access to remote services. This is critical for safeguarding sensitive information and maintaining secure communication channels.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Endpoint Device Authentication Test:</b> <ul style="list-style-type: none"> <li>✓ Verify that the application performs device authentication for all network-connected endpoints before initiating a connection.</li> </ul> </li> <li><b>2. User Authentication Test:</b> <ul style="list-style-type: none"> <li>✓ Confirm that the application enforces user authentication at the remote endpoint, using acceptable mechanisms such as username/password, before granting access.</li> </ul> </li> <li><b>3. Connection Denial Test:</b> <ul style="list-style-type: none"> <li>✓ Validate that the application denies connections to devices or users that fail authentication requirements.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described

<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda
José L Fernández-Alemán
Juan Manuel Carrillo-de-Gea
Joaquín Nicolás
08/01/2026

<b>PUID:</b> [SECM-CAT-IAA-110]
<b>Requirement description:</b> The application must ensure that the remote endpoint uses randomly generated access tokens to authenticate client requests, avoiding the transmission of user credentials.
<b>Source:</b>
✓ 4.2: Verify that the remote endpoint uses randomly generated access tokens to authenticate client requests without sending the user's credentials [19].
<b>Priority:</b> Not described
<b>Rationale:</b> Using randomly generated access tokens for client request authentication reduces the risk of exposing user credentials during communication, ensuring enhanced security and mitigating potential threats like credential interception or reuse attacks.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Access Token Generation Test:</b> <ul style="list-style-type: none"> <li>✓ Verify that the remote endpoint generates unique, random access tokens for each client session.</li> </ul> </li> <li><b>2. Credential Transmission Test:</b> <ul style="list-style-type: none"> <li>✓ Confirm that user credentials are not transmitted during client-server communication after initial authentication.</li> </ul> </li> <li><b>3. Token Expiry and Revocation Test:</b> <ul style="list-style-type: none"> <li>✓ Ensure that access tokens have defined expiration periods and can be revoked if necessary.</li> </ul> </li> <li><b>4. Secure Token Storage Test:</b> <ul style="list-style-type: none"> <li>✓ Validate that access tokens are securely stored on the client side using platform-recommended secure storage mechanisms (e.g., keychain or keystore).</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda

José L Fernández-Alemán  
Juan Manuel Carrillo-de-Gea  
Joaquín Nicolás  
08/01/2026

<b>PUID:</b> [SECM-CAT-IIA-111]
<b>Requirement description:</b> The application must ensure that the remote endpoint terminates any existing session when the user logs out.
<b>Source:</b>
✓ 4.3: Verify that the remote endpoint terminates the existing session when the user logs out. [19].
<b>Priority:</b> Not described
<b>Rationale:</b> Terminating existing sessions upon user logout is essential to prevent unauthorized access and ensure that previously established sessions cannot be exploited by attackers. This practice strengthens the overall security of user accounts.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
1. <b>Session Termination Test:</b> ✓ Log out from the application and verify that the remote endpoint invalidates the session ID associated with the user.
2. <b>Access Denial Test Post Logout:</b> ✓ Attempt to access resources using the previous session credentials after logging out. Ensure that all access attempts are denied.
3. <b>Token Revocation Test:</b> ✓ Confirm that access tokens or session tokens issued to the user are revoked upon logout.
4. <b>Simultaneous Session Termination Test:</b> ✓ Ensure that if the user has multiple sessions, all sessions are terminated upon logout, unless specified otherwise by the application design.
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IAA-112]
<b>Requirement description:</b> The application must clear temporary storage and cookies upon session termination to prevent unauthorized access to sensitive data after the user logs out.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ V-222388: The application must clear temporary storage and cookies when the session is terminated. Design and configure the application to clear sensitive data from cookies and local storage when the user logs out of the application [15].</li> <li>✓ The app should not store Cookies [25].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Clearing temporary storage and cookies upon session termination ensures that sensitive user data is not retained in the application, mitigating the risk of unauthorized access due to data residue or session hijacking.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Temporary Storage Clearance Test:</b> <ul style="list-style-type: none"> <li>✓ Initiate a user session, store temporary data, and log out. Verify that all temporary storage is cleared upon session termination.</li> </ul> </li> <li><b>2. Cookie Clearance Test:</b> <ul style="list-style-type: none"> <li>✓ Log in to the application, ensure cookies are created during the session, and verify that cookies are deleted after logout.</li> </ul> </li> <li><b>3. Access Attempt Post Logout:</b> <ul style="list-style-type: none"> <li>✓ Attempt to use stored session data or cookies to reestablish access after logout. Ensure access is denied, confirming data clearance.</li> </ul> </li> <li><b>4. Automation Check:</b> <ul style="list-style-type: none"> <li>✓ Use automated tools to inspect local storage and cookies after session termination to ensure no sensitive data is retained.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IAA-113]
<b>Requirement description:</b> The application must automatically terminate inactive sessions after a predetermined period of inactivity to prevent unauthorized access, particularly in high-risk locations or when handling sensitive data.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ § 164.312 Technical safeguards.             <ul style="list-style-type: none"> <li>(a)                     <ul style="list-style-type: none"> <li>(2) Implementation specifications:                             <ul style="list-style-type: none"> <li>(ii) Automatic logoff (Addressable):                                     <p>Implement electronic procedures that terminate an electronic session after a predetermined time of inactivity [9].</p> </li> </ul> </li> </ul> </li> </ul> </li> <li>✓ Control Enhancements:             <ul style="list-style-type: none"> <li>(5) ACCOUNT MANAGEMENT   INACTIVITY LOGOUT                     <p>Require that users log out when [Assignment: organization-defined time period of expected inactivity or description of when to log out].</p> <p>Discussion: Inactivity logout is behavior- or policy-based and requires users to take physical action to log out when they are expecting inactivity longer than the defined period. Automatic enforcement of inactivity logout is addressed by AC-11 [11].</p> </li> </ul> </li> <li>✓ 4.7: Verify that sessions are terminated at the remote endpoint after a predefined period of inactivity [19].</li> <li>✓ SC-10 NETWORK DISCONNECT             <p>Control: Terminate the network connection associated with a communications session at the end of the session or after [Assignment: organization-defined time period] of inactivity.</p> <p>Discussion: Network disconnect applies to internal and external networks. Terminating network connections associated with specific communications sessions includes de-allocating TCP/IP address or port pairs at the operating system level and de-allocating the networking assignments at the application level if multiple application sessions are using a single operating system-level network connection. Periods of inactivity may be established by organizations and include time periods by type of network access or for specific network accesses [11].</p> </li> <li>✓ 9.4.2 Secure log-on procedures: k) terminate inactive sessions after a defined period of inactivity, especially in high-risk locations such as public or external areas outside the organization's security management or on mobile devices; [7]</li> <li>✓ Session timeout after 30 min maximum in the case of connection to a server [25].</li> <li>✓ Security Requirement 71 – Restricting Access to Unattended Workstations: All PoS systems connected to the EHRi must protect unattended workstations against an unauthorized person using the workstation while the PoS is active, such as with an automatic timeout after a period of inactivity. First, the best approach is to place workstations in a physically secure area in the first place. (está relacionado con cerrar sesión cuando hay inactividad en la app)</li> </ul>
Consideration MC21 – Establish and Enforce Session Timeouts: When designing or acquiring mobile applications intended to access or store PHI, organizations should ensure that the application has the ability to enforce a mandatory session timeout when left unattended [10].
<b>Priority:</b> Not described
<b>Rationale:</b> Enforcing session timeouts ensures that unattended sessions are automatically terminated, reducing the risk of unauthorized access and maintaining compliance with security standards for sensitive systems and environments.
<b>Number of Children:</b> 0

<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b> <ol style="list-style-type: none"> <li><b>1. Inactivity Test:</b> <ul style="list-style-type: none"> <li>✓ Start a session, leave it inactive for the predetermined timeout period, and verify that the session is automatically terminated.</li> </ul> </li> <li><b>2. Remote Endpoint Validation:</b> <ul style="list-style-type: none"> <li>✓ Verify that session termination propagates to the remote endpoint, ensuring complete disconnection of all associated network activities.</li> </ul> </li> <li><b>3. Manual Timeout Configuration Review:</b> <ul style="list-style-type: none"> <li>✓ Confirm that the application allows administrators to define and enforce session timeout durations per organizational policy.</li> </ul> </li> <li><b>4. Session Reauthentication Test:</b> <ul style="list-style-type: none"> <li>✓ Attempt to resume a terminated session and verify that reauthentication is required to regain Access.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IAA-114]
<b>Requirement description:</b> The application must provide all users with the capability to manually terminate their communication session to ensure user-initiated logoff functionality.
<b>Source:</b> <ul style="list-style-type: none"> <li>✓ V-222391: Applications requiring user access authentication must provide a logoff capability for user-initiated communication session. Design and configure the application to provide all users with the capability to manually terminate their application session [15].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Providing users with a manual session termination capability ensures greater control over session security, reduces the risk of unauthorized access, and complies with security best practices for sensitive applications.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0

<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<b>1. Manual Logoff Test:</b>
<ul style="list-style-type: none"><li>✓ Verify that users can manually terminate their session using the logoff feature within the application.</li></ul>
<b>2. Session State Test:</b>
<ul style="list-style-type: none"><li>✓ Confirm that all session-related data, such as tokens and cookies, are invalidated upon logoff.</li></ul>
<b>3. Reauthentication Test:</b>
<ul style="list-style-type: none"><li>✓ Attempt to access the application after logging off and ensure that reauthentication is required.</li></ul>
<b>4. UI Accessibility Test:</b>
<ul style="list-style-type: none"><li>✓ Ensure the logoff functionality is clearly visible and easily accessible to users within the application interface.</li></ul>
<b>5. Session Termination Validation:</b>
<ul style="list-style-type: none"><li>✓ Verify that terminated sessions are reflected both on the client and server sides, ensuring complete disconnection.</li></ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

**PUID:** [SECM-CAT-IAA-115]

**Requirement description:** The application must display an explicit logoff message to users indicating the reliable termination of authenticated communication sessions upon session termination.

**Source:**

- ✓ V-222392: The application must display an explicit logoff message to users indicating the reliable termination of authenticated communications sessions. Design and configure the application to provide an explicit logoff message to users indicating a successful logoff has occurred upon user session termination [15].

**Priority:** Not described

**Rationale:** Displaying an explicit logoff message ensures users are clearly informed of session termination, enhancing transparency and providing reassurance that the session has ended securely.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

**1. Logoff Message Display Test:**

- ✓ Verify that a logoff message is displayed immediately after the user terminates the session.

**2. Message Content Validation:**

- ✓ Confirm that the logoff message explicitly states that the authenticated session has been successfully terminated.

**3. Session Termination Test:**

- ✓ Ensure that all session tokens and session-related data are invalidated when the logoff message is displayed.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda

José L Fernández-Alemán

Juan Manuel Carrillo-de-Gea

Joaquín Nicolás

08/01/2026

**PUID:** [SECM-CAT-IAA-116]

**Requirement description:** The application must enforce two-factor authentication (2FA) for access to personal health information and ensure it is consistently implemented for sensitive operations.

**Source:**

- ✓ 9.4.1 Information access restriction: Health information systems processing personal health information shall authenticate users and should do so by means of authentication involving at least two factors [7].
- ✓ Access to information and application system functions related to the processing personal health information should be isolated from (and separate to) access to information processing infrastructure that is unrelated to the processing of personal health information.
- ✓ 4.8: Verify that a second factor of authentication exists at the remote endpoint and the 2FA requirement is consistently enforced [19].
- ✓ MASVS-AUTH-3: The app secures sensitive operations with additional authentication: Some additional form of authentication is often desirable for sensitive actions inside the app. This can be done in different ways (biometric, pin, MFA code generator, email, deep links, etc) and they all need to be implemented securely [20].

**Priority:** Not described

**Rationale:** Two-factor authentication (2FA) adds an extra layer of security, reducing the risk of unauthorized access to personal health information and ensuring sensitive operations are protected.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

**1. 2FA Mechanism Validation Test:**

- ✓ Confirm that the application enforces two-factor authentication (2FA) during login and before accessing personal health information.

**2. Sensitive Operations Authentication Test:**

- ✓ Verify that sensitive operations (e.g., accessing financial or health records) require a second factor of authentication.

**3. Endpoint Consistency Check:**

- ✓ Ensure that the 2FA requirement is consistently applied across all remote endpoints.

**4. Fallback and Recovery Mechanism Test:**

- ✓ Verify that fallback mechanisms for 2FA (e.g., recovery codes) are implemented securely and do not compromise user accounts.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0**Author and date:**

Carlos M. Mejía-Granda  
 José L Fernández-Alemán  
 Juan Manuel Carrillo-de-Gea  
 Joaquín Nicolás  
 08/01/2026

**PUID:** [SECM-CAT-IAA-117]

**Requirement description:** The application must ensure that the remote endpoint implements an exponential back-off mechanism or temporarily locks the user account after a predefined number of incorrect authentication attempts to mitigate brute-force attacks.

**Source:**

- ✓ 4.5: Verify that the remote endpoint implements an exponential back-off, or temporarily locks the user account, when incorrect authentication credentials are submitted an excessive number of times. [19].

**Priority:** Not described

**Rationale:** Implementing account lockout mechanisms or exponential back-off reduces the likelihood of successful brute-force attacks by introducing delays or temporarily disabling access for repeated failed attempts.

**Number of Children:** 0**Number of parents:** 0**Cycles:** 0**Number Audit:** 0**Child PUIDs:** Not described**Parent PUIDs:** Not described**Exclusion PUIDs:** Not described**Importance:** Not described**Current state:** To be determined**Verification method:** Demonstration/Analysis**Validation criteria:****1. Failed Authentication Attempt Test:**

- ✓ Simulate multiple incorrect authentication attempts and verify that the remote endpoint applies an exponential back-off delay or locks the user account after the defined threshold (e.g., three failed attempts).

**2. Lockout Duration Test:**

- ✓ Confirm that locked accounts are inaccessible during the lockout period and can only be unlocked after the predefined time or through administrative intervention.

**3. Notification Test:**

- ✓ Verify that users are notified of account lockout events via secure communication channels without revealing sensitive details of the account.

**4. Reset Mechanism Test:**

- ✓ Validate that there is a secure mechanism for unlocking user accounts, ensuring proper user identity verification before account reactivation.

**Requested by:** The organization**Responsible:** Developer

<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda
José L Fernández-Alemán
Juan Manuel Carrillo-de-Gea
Joaquín Nicolás
08/01/2026

<b>PUID:</b> [SECM-CAT-IAA-118]
<b>Requirement description:</b> The application must enforce a minimum device-access-security policy, ensuring the device is secured with a passcode or equivalent protection method.
<b>Source:</b>
✓ 2.11: Verify that the app enforces a minimum device-access-security policy, such as requiring the user to set a device passcode [19].
<b>Priority:</b> Not described
<b>Rationale:</b> Requiring a device passcode enhances security by ensuring only authorized users can access the device and its sensitive applications, reducing risks associated with unauthorized access.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Passcode Requirement Test:</b> <ul style="list-style-type: none"> <li>✓ Confirm that the application checks for a device passcode during installation or first run and prompts the user to enable it if not already active.</li> </ul> </li> <li><b>2. Device Compliance Test:</b> <ul style="list-style-type: none"> <li>✓ Verify that the app refuses to operate or restricts sensitive operations on devices without an active passcode or equivalent device-access security.</li> </ul> </li> <li><b>3. Policy Enforcement Test:</b> <ul style="list-style-type: none"> <li>✓ Ensure the application consistently enforces the device-access-security policy, including after device restarts or when reactivating the app.</li> </ul> </li> <li><b>4. Passcode Strength Test:</b> <ul style="list-style-type: none"> <li>✓ Validate that the device passcode meets a minimum complexity requirement, such as a six-character alphanumeric passcode or equivalent.</li> </ul> </li> <li><b>5. Integration Test:</b> <ul style="list-style-type: none"> <li>✓ Verify integration with the device's native security framework to detect and enforce passcode policies without additional user intervention.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer

<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda
José L Fernández-Alemán
Juan Manuel Carrillo-de-Gea
Joaquín Nicolás
08/01/2026

<b>PUID:</b> [SECM-CAT-IAA-119]
<b>Requirement description:</b> The application must verify the X.509 certificate of the remote endpoint during secure channel establishment, ensuring only certificates signed by a valid Certificate Authority (CA) are accepted, and fail securely if validation fails.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ 5.3: Verify that the app verifies the X.509 certificate of the remote endpoint when the secure channel is established. Only certificates signed by a valid CA are accepted [19].</li> <li>✓ iOS Specific Best Practices: Ensure that certificates are valid and fail closed [3].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Verifying the validity of X.509 certificates signed by a trusted CA prevents man-in-the-middle attacks and ensures secure communication with remote endpoints. Failing securely upon validation failure ensures that no insecure connections are established.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis

<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Certificate Validation Test:</b> <ul style="list-style-type: none"> <li>✓ Confirm that the application validates the X.509 certificate chain for the remote endpoint during secure channel establishment, accepting only certificates signed by a trusted CA.</li> </ul> </li> <li><b>2. Invalid Certificate Test:</b> <ul style="list-style-type: none"> <li>✓ Simulate a connection attempt with an invalid, expired, or self-signed certificate, and verify that the application rejects the connection.</li> </ul> </li> <li><b>3. CA Trust Store Test:</b> <ul style="list-style-type: none"> <li>✓ Confirm that the application relies only on the platform's trusted CA store or a custom CA store configured securely within the application.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0

**Author and date:**

Carlos M. Mejía-Granda  
 José L Fernández-Alemán  
 Juan Manuel Carrillo-de-Gea  
 Joaquín Nicolás  
 08/01/2026

**PUID: [SECM-CAT-IAA-120]**

**Requirement description:** The application must enhance the security of API communications by replacing static API keys with OAuth 2.0 or other token-based mechanisms, and secure sensitive operations with additional authentication layers.

**Source:**

- ✓ Protect API Key Communication: Use OAuth 2.0 or other token-based mechanisms instead of static API keys when possible to enhance security [18].
- ✓ MASVS-AUTH-1: The app uses secure authentication and authorization protocols and follows the relevant best practices: Most apps connecting to a remote endpoint require user authentication and also enforce some kind of authorization. While the enforcement of these mechanisms must be on the remote endpoint, the apps also have to ensure that it follows all the relevant best practices to ensure a secure use of the involved protocols [20].
- ✓ MASVS-AUTH-3: The app secures sensitive operations with additional authentication: Some additional form of authentication is often desirable for sensitive actions inside the app. This can be done in different ways (biometric, pin, MFA code generator, email, deep links, etc) and they all need to be implemented securely [20].

**Priority:** Not described

**Rationale:** Static API keys are susceptible to theft and misuse, leading to potential unauthorized access. Token-based mechanisms such as OAuth 2.0 ensure better control and security. Additional authentication for sensitive operations mitigates the risk of unauthorized access.

**Number of Children:** 0**Number of parents:** 0**Cycles:** 0**Number Audit:** 0**Child PUIDs:** Not described**Parent PUIDs:** Not described**Exclusion PUIDs:** Not described**Importance:** Not described**Current state:** To be determined**Verification method:** Demonstration/Analysis**Validation criteria:****1. API Key Replacement Test:**

- ✓ Verify that static API keys are not embedded within the app. Confirm the use of OAuth 2.0 or token-based mechanisms for API authentication.

**2. Token Expiry Test:**

- ✓ Ensure that authentication tokens have a defined expiration period and are revoked upon inactivity or logout.

<b>3. Sensitive Operation Authentication Test:</b>
✓ Verify that additional authentication mechanisms (e.g., MFA, biometric verification) are required for accessing sensitive app functions.
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IAA-121]
<b>Requirement description:</b> The application must implement the OAuth 2.0 authorization framework for API access, restricting API key usage to specific clients, defining minimum access scopes, and enforcing secure authentication protocols.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ OAuth 2.0 Authorization Framework: Where applicable, use OAuth 2.0 to authorize API access, restricting API key usage to specific clients and defining the minimum access scope for each API key. Ensure that each key only provides access necessary for its specific use case [18].</li> <li>✓ MASVS-AUTH-1: The app uses secure authentication and authorization protocols and follows the relevant best practices: Most apps connecting to a remote endpoint require user authentication and also enforce some kind of authorization. While the enforcement of these mechanisms must be on the remote endpoint, the apps also have to ensure that it follows all the relevant best practices to ensure a secure use of the involved protocols [20]</li> <li>✓ 8.5 Secure Authentication</li> </ul> <p><b>Control:</b> Secure authentication technologies and procedures should be implemented based on information access restrictions and the topic-specific policy on access control.</p> <p><b>Purpose:</b> To ensure a user or an entity is securely authenticated, when access to systems, applications and services is granted.</p> <p><b>Guidance:</b> A suitable authentication technique should be chosen to substantiate the claimed identity of a user, software, messages and other entities [6].</p>
<b>Priority:</b> Not described
<b>Rationale:</b> Using OAuth 2.0 for API access ensures secure and fine-grained control over API usage while minimizing exposure to unauthorized access. Limiting API key scope reduces the risk of data breaches.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0

<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Scope Limitation Test:</b> <ul style="list-style-type: none"> <li>✓ Verify that API keys are restricted to minimum required access scopes for their respective use cases.</li> </ul> </li> <li><b>2. OAuth 2.0 Implementation Test:</b> <ul style="list-style-type: none"> <li>✓ Confirm that the application uses OAuth 2.0 authorization for API access, ensuring that only authorized clients can access the API.</li> </ul> </li> <li><b>3. Secure Authentication Test:</b> <ul style="list-style-type: none"> <li>✓ Validate the integration of secure authentication technologies in compliance with access control policies.</li> </ul> </li> <li><b>4. Client-Specific Key Test:</b> <ul style="list-style-type: none"> <li>✓ Ensure that API keys are client-specific and cannot be reused across different applications.</li> </ul> </li> <li><b>5. Token Lifecycle Management Test:</b> <ul style="list-style-type: none"> <li>✓ Verify that OAuth tokens are configured with appropriate expiration times and can be revoked without affecting unrelated clients.</li> </ul> </li> <li><b>6. API Key Audit Test:</b> <ul style="list-style-type: none"> <li>✓ Review API key usage logs to ensure compliance with defined scopes and access controls.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IAA-122]
<b>Requirement description:</b> The application must generate unique API keys for each application instance to ensure efficient identification and isolation of compromised or unauthorized access.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ Protect Unique API Key per Application: Generate unique API keys for each application to ensure the ability to identify and isolate compromised or unauthorized access efficiently [18].</li> </ul>

<b>Priority:</b> Not described
--------------------------------

<b>Rationale:</b> Using unique API keys per application instance enhances the ability to identify and mitigate security breaches by isolating compromised keys without affecting other instances.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Unique Key Generation Test:</b> <ul style="list-style-type: none"> <li>✓ Verify that the system generates unique API keys for each application instance upon registration.</li> </ul> </li> <li><b>2. Key Association Test:</b> <ul style="list-style-type: none"> <li>✓ Confirm that each API key is correctly associated with a single application instance and cannot be reused across multiple instances.</li> </ul> </li> <li><b>3. Compromised Key Isolation Test:</b> <ul style="list-style-type: none"> <li>✓ Simulate the compromise of an API key and validate that it can be revoked without affecting other application instances.</li> </ul> </li> <li><b>4. Key Expiry and Rotation Test:</b> <ul style="list-style-type: none"> <li>✓ Validate that the application enforces periodic expiration and secure rotation of API keys.</li> </ul> </li> <li><b>5. Unauthorized Access Test:</b> <ul style="list-style-type: none"> <li>✓ Attempt to use an API key from a different application instance and confirm that access is denied.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IAA-123]
<b>Requirement description:</b> The application must enforce IP-based and service-level restrictions on API key usage, associating keys with specific mobile applications, certificates, or platforms to prevent unauthorized access and misuse.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ Protect IP Restrictions for API Keys: Where applicable, restrict API key usage to specific IP addresses or ranges to limit unauthorized access from unknown locations [18].</li> </ul>

<ul style="list-style-type: none"> <li>✓ Mobile App Key Usage Limitation: Limit API key usage to specific mobile apps by associating keys with app certificates or bundling them within the app's metadata [18].</li> <li>✓ Service-Level Key Restriction: Ensure the service allows restricting API keys to a particular package or platform (e.g., limit key access by package name and signing key, as with Google Maps API) [18].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Restricting API key usage to defined IP addresses, app certificates, or specific platforms minimizes the risk of unauthorized or malicious access and ensures that API keys are used only by approved applications or services.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. IP Restriction Test:</b> <ul style="list-style-type: none"> <li>✓ Validate that API keys are restricted to specific IP addresses or ranges, denying requests from unauthorized IPs.</li> </ul> </li> <li><b>2. App Certificate Association Test:</b> <ul style="list-style-type: none"> <li>✓ Confirm that API keys can only be used by applications with valid certificates associated with the key.</li> </ul> </li> <li><b>3. Platform Restriction Test:</b> <ul style="list-style-type: none"> <li>✓ Ensure that the API key access is limited to specified platforms or services (e.g., Google Maps API restricted by package name and signing key).</li> </ul> </li> <li><b>4. Unauthorized Usage Test:</b> <ul style="list-style-type: none"> <li>✓ Attempt to use the API key from an unauthorized IP, app certificate, or platform and verify that access is denied.</li> </ul> </li> <li><b>5. Audit Log Verification Test:</b> <ul style="list-style-type: none"> <li>✓ Confirm that all API key access requests and denials are logged, with details on IP addresses, app certificates, and platforms.</li> </ul> </li> <li><b>6. Key Configuration Review Test:</b> <ul style="list-style-type: none"> <li>✓ Validate that the key restriction configurations are correctly set and align with the organization's security policies.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda
José L Fernández-Alemán
Juan Manuel Carrillo-de-Gea
Joaquín Nicolás

08/01/2026

<b>PUID:</b> [SECM-CAT-IAA-124]
<b>Requirement description:</b> The application must prevent the exposure of sensitive information, such as usernames or personal data, in error messages or logs, ensuring that error responses are generic and do not reveal the existence of registered accounts.
<b>Source:</b>
<ul style="list-style-type: none"><li>✓ 2.4. Do not reveal registered usernames and remove any fingerprint of their existence from verbose error messages [16].</li><li>✓ 10.3. Do not reveal sensitive information such as usernames, personal data and others through error messages [16].</li></ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Restricting API key usage to defined IP addresses, app certificates, or specific platforms minimizes the risk of unauthorized or malicious access and ensures that API keys are used only by approved applications or services.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"><li><b>1. Error Message Verification Test:</b><ul style="list-style-type: none"><li>✓ Verify that error messages do not disclose sensitive information, such as whether a username exists in the system.</li></ul></li><li><b>2. Generic Error Response Test:</b><ul style="list-style-type: none"><li>✓ Attempt to log in with invalid credentials and confirm that the error response is generic (e.g., “Invalid credentials”) and does not reveal specific details.</li></ul></li><li><b>3. Log Inspection Test:</b><ul style="list-style-type: none"><li>✓ Review application logs to ensure no sensitive data, such as usernames or personal data, is recorded in verbose error logs.</li></ul></li><li><b>4. Penetration Test:</b><ul style="list-style-type: none"><li>✓ Conduct security testing to verify that no sensitive information can be inferred from error messages during malicious activities.</li></ul></li><li><b>5. Error Handling Review:</b><ul style="list-style-type: none"><li>✓ Analyze error-handling code to confirm that generic responses are implemented consistently across all failure scenarios.</li></ul></li></ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer

<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda
José L Fernández-Alemán
Juan Manuel Carrillo-de-Gea
Joaquín Nicolás
08/01/2026

#### 2.9.3.4 Insufficient Input/Output Validation

<b>PUID:</b> [SECM-CAT-IOV-001]
<b>Requirement description:</b> The application must implement comprehensive input validation mechanisms to safeguard against vulnerabilities and insecure deserialization. All input from external sources, including files, network communications, and inter-process communications, must be validated and sanitized using platform-level security features provided by Android.
<b>Source:</b>
<ul style="list-style-type: none"><li>✓ General Input Validation: The application must implement robust input validation for all data received from external sources, including files, network communications, and inter-process communication (IPC). The application must utilize platform-level security features provided by Android to minimize input validation issues whenever possible [18].</li><li>✓ V-222609: The application must not be subject to input handling vulnerabilities. Follow best practice when accepting user input and verify that all input is validated before the application processes the input. Remediate identified vulnerabilities and obtain documented risk acceptance for those issues that cannot be remediated immediately [15].</li><li>✓ SI-10 INFORMATION INPUT VALIDATION</li><li>✓ Control: Check the validity of the following information inputs: [Assignment: organization-defined information inputs to the system] [11].</li><li>✓ Input Validation: 1. Validate and sanitize user input using strict validation techniques [3].</li><li>✓ Security Requirement 76 – Validating Input Data: The EHRI and all PoS systems connected to the EHRI must include, wherever feasible, measures to safeguard against user error by validating data input to ensure that it is correct and appropriate. The following controls should be considered:<ul style="list-style-type: none"><li>a) Input checks to detect the following errors:<ul style="list-style-type: none"><li>i. out-of-range values;</li><li>ii. invalid characters in data fields;</li><li>iii. missing or incomplete data;</li><li>iv. exceeding upper and lower data volume limits;</li><li>v. unauthorized or inconsistent control data.</li></ul></li><li>b) Procedures for responding to validation errors [10].</li></ul></li></ul>

- ✓ MASVS-CODE-4: The app validates and sanitizes all untrusted inputs: Apps have many data entry points including the UI, IPC, the network, the file system, etc. This incoming data might have been inadvertently modified by untrusted actors and may lead to bypass of critical security checks as well as classical injection attacks such as SQL injection, XSS or insecure deserialization. This control ensures that this data is treated as untrusted input and is properly verified and sanitized before it's used [20]
- ✓ V-222602: The application must protect from Cross-Site Scripting (XSS) vulnerabilities. Verify user input is validated and encode or escape user input to prevent embedded script code from executing. Develop your application using a web template system or a web application development framework that provides auto escaping features rather than building your own escape logic [15].
- ✓ Validate Input and Sanitize Data: 1. Implement input validation and data sanitization techniques to prevent injection attacks and ensure that only valid and expected data is stored. Validate user inputs to mitigate the risk of malicious code injection or unintended data leakage [3].
- ✓ CWE-20 Improper Input Validation: The product receives input or data, but it does not validate or incorrectly validates that the input has the properties that are required to process the data safely and correctly [35].

**Priority:** Hight

**Rationale:** Proper input validation prevents exploitation of vulnerabilities arising from malformed or malicious data, ensuring application integrity, data confidentiality, and compliance with security best practices.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

1. **Input Range Validation:**
  - ✓ **Test:** Input out-of-range values in data fields.
  - ✓ **Expected Result:** System detects and rejects values outside the allowable range.
2. **Invalid Character Test:**
  - ✓ **Test:** Submit inputs containing invalid characters.
  - ✓ **Expected Result:** Application rejects input and provides an appropriate error message.
3. **Data Completeness Test:**
  - ✓ **Test:** Submit incomplete forms.
  - ✓ **Expected Result:** System identifies and flags incomplete submissions.
4. **Untrusted Input Handling:**
  - ✓ **Test:** Submit unexpected or malformed data.

✓ <b>Expected Result:</b> The system rejects or sanitizes such data to ensure application stability.
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IOV-002]
<b>Requirement description:</b> The application must ensure that error messages are generic and do not disclose system details or sensitive data to users. Error messages must only provide information necessary for corrective actions while safeguarding against exploitation by adversaries.
<b>Source:</b>
✓ V-222610: The application must generate error messages that provide information necessary for corrective actions without revealing information that could be exploited by adversaries. Configure the server to not send error messages containing system information or sensitive data to users. Use generic error messages [15].
<b>Priority:</b> High
<b>Rationale:</b> Restricting the information in error messages reduces the risk of exposing system vulnerabilities or sensitive data, minimizing the likelihood of exploitation by attackers.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li>1. <b>Generic Error Message Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Simulate user errors such as invalid input or login failures.</li> <li>✓ Expected Result: The system returns generic error messages (e.g., "Invalid credentials") without exposing sensitive information.</li> </ul> </li> <li>2. <b>System Information Disclosure Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Trigger server-side exceptions or system failures.</li> <li>✓ Expected Result: No system-specific information, such as stack traces or configuration details, is included in the error messages presented to users.</li> </ul> </li> <li>3. <b>Penetration Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Conduct simulated attacks to identify error handling weaknesses.</li> </ul> </li> </ol>

<ul style="list-style-type: none"> <li>✓ Expected Result: Error messages provide no exploitable details to attackers.</li> </ul> <p><b>4. Code Review Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Review the application's error handling implementation.</li> <li>✓ Expected Result: Error messages are implemented in compliance with security guidelines and do not expose sensitive information</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IOV-003]
<b>Requirement description:</b> The application must restrict detailed error messages containing system information or sensitive data to privileged users such as the ISSO, ISSM, or SA. For non-privileged users, error messages must be generic, providing minimal information to avoid exposing system vulnerabilities.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ V-222611: The application must reveal error messages only to the ISSO, ISSM, or SA. Configure the server to only send error messages containing system information or sensitive data to privileged users. Use generic error messages for non-privileged users.</li> </ul>
<b>Priority:</b> Not defined
<b>Rationale:</b> Restricting access to detailed error messages mitigates the risk of exposing system vulnerabilities or sensitive data to unauthorized users, reducing the attack surface for potential adversaries.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Privileged User Error Message Test:</b></li> </ol> <ul style="list-style-type: none"> <li>✓ Test: Simulate errors in the system while logged in as a privileged user (e.g., ISSO, ISSM, or SA).</li> <li>✓ Expected Result: Detailed error messages containing necessary information for diagnostics are displayed only to privileged users.</li> </ul>

- 2. Non-Privileged User Error Message Test:**
  - ✓ Test: Simulate errors in the system while logged in as a non-privileged user.
  - ✓ Expected Result: Generic error messages are displayed, containing no sensitive system information.
- 3. Error Logging Test:**
  - ✓ Test: Examine server logs for error message content.
  - ✓ Expected Result: Detailed error information is securely logged for review by authorized personnel, while user-facing messages remain generic.
- 4. Configuration Review Test:**
  - ✓ Test: Review server configurations for role-based error message handling.
  - ✓ Expected Result: System is configured to deliver detailed error messages only to privileged users.
- 5. Penetration Test:**
  - ✓ Test: Conduct simulated attacks to determine if error messages exposed to non-privileged users reveal sensitive system details.
  - ✓ Expected Result: Non-privileged users are not able to obtain sensitive information from error messages.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda  
 José L Fernández-Alemán  
 Juan Manuel Carrillo-de-Gea  
 Joaquín Nicolás  
 08/01/2026

**PUID:** [SECM-CAT-IOV-004]

**Requirement description:** The application must prevent the population of WebViews loaded from the file URI scheme with user-supplied DOM input to avoid potential security vulnerabilities, such as injection attacks.

**Source:**

- ✓ 12.16. Avoid populating webviews loaded from the file URI scheme with user supplied DOM input [16].

**Priority:** Not defined

**Rationale:** WebViews loaded from file URIs are prone to security risks if populated with user-supplied DOM input. This vulnerability can be exploited to execute malicious scripts, compromise user data, or bypass application security.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<b>1. Input Validation Test:</b> <ul style="list-style-type: none"><li>✓ Test: Attempt to load a WebView from a file URI and populate it with user-supplied DOM input.</li><li>✓ Expected Result: The application prevents user-supplied DOM input from being loaded into the WebView.</li></ul>
<b>2. Error Handling Test:</b> <ul style="list-style-type: none"><li>✓ Test: Verify the application's behavior when attempting to load user-supplied input into a WebView.</li><li>✓ Expected Result: The application throws an error or blocks the operation without exposing vulnerabilities.</li></ul>
<b>3. Code Review Test:</b> <ul style="list-style-type: none"><li>✓ Test: Analyze the implementation to ensure WebViews are not loaded with file URIs containing user-supplied DOM input.</li><li>✓ Expected Result: Code includes checks and safeguards to enforce this requirement.</li></ul>
<b>4. Security Testing:</b> <ul style="list-style-type: none"><li>✓ Test: Perform a penetration test focused on WebView interactions to identify any potential misuse of the file URI scheme.</li><li>✓ Expected Result: No vulnerabilities are found, and the WebView interactions are secure.</li></ul>
<b>5. Configuration Audit Test:</b> <ul style="list-style-type: none"><li>✓ Test: Verify the application's configuration to ensure secure handling of WebView loading processes.</li><li>✓ Expected Result: The configuration explicitly blocks the use of file URIs with user-supplied input.</li></ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IOV-005]
<b>Requirement description:</b> The application must implement policies and procedures to ensure the confidentiality, integrity, and availability of electronic protected health information (ePHI). This includes validating output data to ensure consistency with expected content and preventing unauthorized alterations or destruction of data.
<b>Source:</b>
<ul style="list-style-type: none"><li>✓ (a) A covered entity or business associate must, in accordance with § 164.306:</li></ul>

1. Ensure the confidentiality, integrity, and availability of all electronic protected health information the covered entity or business associate creates, receives, maintains, or transmits.
- i. **Standard: Security management process:**  
Implement policies and procedures to prevent, detect, contain, and correct security violations [9].
- ✓ § 164.312 Technical safeguards.
- (c)
- (1) Standard: integrity.  
Implement policies and procedures to protect electronic protected health information from improper alteration or destruction
- (2)  
Implement electronic mechanisms to corroborate that electronic protected health information has not been altered or destroyed in an unauthorized manner [9].
- ✓ SI-15 INFORMATION OUTPUT FILTERING
- Control: Validate information output from the following software programs and/or applications to ensure that the information is consistent with the expected content: [Assignment: organization-defined software programs and/or applications].
- Discussion: Certain types of attacks, including SQL injections, produce output results that are unexpected or inconsistent with the output results that would be expected from software programs or applications. Information output filtering focuses on detecting extraneous content, preventing such extraneous content from being displayed, and then alerting monitoring tools that anomalous behavior has been discovered [11].
- ✓ Data Integrity Checks: 1. Implement data integrity checks to detect and prevent data corruption or unauthorized modifications [3].

**Priority:** Not defined

**Rationale:** Protecting the integrity of ePHI ensures compliance with legal and regulatory standards, such as HIPAA. It minimizes the risks of data corruption, unauthorized access, and potential harm to individuals relying on accurate health information.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

### 1. Output Validation Test:

- ✓ Test: Simulate SQL injection and other input-based attacks to assess the application's ability to validate and sanitize output data.
- ✓ Expected Result: The application filters extraneous content and outputs consistent, valid information.

## **2. Data Integrity Audit Test:**

- ✓ Test: Introduce unauthorized modifications to ePHI and assess whether the application detects and prevents the alterations.
- ✓ Expected Result: The application successfully detects and mitigates unauthorized changes to ePHI.

## **3. Mechanism Inspection Test:**

- ✓ Test: Verify the presence and functionality of electronic mechanisms that corroborate the integrity of ePHI.
- ✓ Expected Result: Mechanisms actively monitor and validate ePHI integrity in all operations.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda

José L Fernández-Alemán

Juan Manuel Carrillo-de-Gea

Joaquín Nicolás

08/01/2026

**PUID:** [SECM-CAT-IOV-006]

**Requirement description:** The application must not disclose unnecessary information, such as technical details of its architecture, during error events to prevent attackers from exploiting sensitive information.

**Source:**

- ✓ V-222600: The application must not disclose unnecessary information to users. Configure the application to not display technical details about the application architecture on error events [15].

**Priority:** Not defined

**Rationale:** Restricting unnecessary disclosure of technical information prevents attackers from gaining insights into the application's architecture, reducing the risk of targeted exploitation or malicious activity.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<p><b>1. Output Validation Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Simulate SQL injection and other input-based attacks to assess the application's ability to validate and sanitize output data.</li> <li>✓ Expected Result: The application filters extraneous content and outputs consistent, valid information.</li> </ul> <p><b>2. Error Message Review Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Trigger various error events within the application and review the error messages displayed to users.</li> <li>✓ Expected Result: Error messages are generic and do not disclose technical information about the application's architecture or systems.</li> </ul> <p><b>3. Penetration Testing:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Conduct penetration tests to verify that no sensitive technical details are disclosed through error responses.</li> <li>✓ Expected Result: No technical information is revealed during the testing of error scenarios.</li> </ul> <p><b>4. Code Review Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Analyze the error-handling code to ensure that it does not expose application details in user-facing error messages.</li> <li>✓ Expected Result: The code ensures that all error messages shown to users are sanitized and generic.</li> </ul> <p><b>5. Configuration Inspection Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Inspect application configurations to confirm that error responses are configured to provide generic feedback without technical details.</li> <li>✓ Expected Result: The application configurations align with the requirement to suppress unnecessary information in error messages.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IOV-007]
<b>Requirement description:</b> The system must enforce strict control over the data format during information exchange between applications to ensure data integrity, prevent inconsistencies, and safeguard against potential vulnerabilities.
<b>Source:</b>

✓ Data Format Control: The system must enforce control over the data format when sharing information between applications to maintain data integrity and consistency [18].
<b>Priority:</b> Not defined
<b>Rationale:</b> Enforcing data format control ensures that information exchanged between applications is processed correctly and securely, reducing the risk of data corruption, misinterpretation, or exploitation.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Data Format Enforcement Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Test the system's ability to validate and enforce the expected data format during inter-application communication.</li> <li>✓ Expected Result: The system rejects data that does not comply with the defined format and generates appropriate error messages.</li> </ul> </li> <li><b>2. Interoperability Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Simulate data exchange between multiple applications to verify consistent data handling and processing.</li> <li>✓ Expected Result: Data shared between applications remains consistent and follows the enforced format.</li> </ul> </li> <li><b>3. Code Review Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Analyze the code to ensure that strict data format validations are implemented for all incoming and outgoing data exchanges.</li> <li>✓ Expected Result: The code enforces proper data format checks, preventing invalid data processing.</li> </ul> </li> <li><b>4. Injection Attack Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Attempt to inject malformed or malicious data during inter-application communication.</li> <li>✓ Expected Result: The system detects and blocks the injection attempts, maintaining data integrity.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda
José L Fernández-Alemán
Juan Manuel Carrillo-de-Gea
Joaquín Nicolás
08/01/2026

**PUID:** [SECM-CAT-IOV-008]

**Requirement description:** The mobile application must validate the information output from designated software programs and/or applications to ensure it aligns with expected content, preventing extraneous or malicious data from being displayed or transmitted.

**Source:**

- ✓ SRG-APP-000449-MAPP-000100: The mobile app must validate information output from software programs and/or applications defined in SI-15, CCI-0002770 to ensure the information is consistent with the expected content [17].

**Priority:** Not defined

**Rationale:** Validating output data ensures that anomalies, potentially stemming from attacks such as SQL injections or unauthorized data manipulations, are detected and mitigated, preserving system integrity and user trust.

**Number of Children:** 0**Number of parents:** 0**Cycles:** 0**Number Audit:** 0**Child PUIDs:** Not described**Parent PUIDs:** Not described**Exclusion PUIDs:** Not described**Importance:** Not described**Current state:** To be determined**Verification method:** Demonstration/Analysis**Validation criteria:****1. Output Data Validation Test:**

- ✓ Test: Verify that the application checks all output data for conformity with predefined content standards.
- ✓ Expected Result: The application rejects or logs data that does not meet the expected content criteria.

**2. Injection Anomaly Test:**

- ✓ Test: Simulate injection attacks to produce unexpected output.
- ✓ Expected Result: The system detects anomalies in output and prevents improper data from being displayed or transmitted.

**3. Code Analysis Test:**

- ✓ Test: Perform a code review to ensure output validation mechanisms are implemented for all software programs and applications.
- ✓ Expected Result: Output validation logic is present, ensuring consistent enforcement across all components.

**Requested by:** The organization**Responsible:** Developer**Configurable value:** Not described**Version history:** v1.0**Author and date:**

Carlos M. Mejía-Granda

José L Fernández-Alemán

Juan Manuel Carrillo-de-Gea

Joaquín Nicolás

08/01/2026

**PUID:** [SECM-CAT-IOV-009]

**Requirement description:** The application must implement strict validation and sanitization of all user-supplied URLs or similar requests to prevent Server-Side Request Forgery (SSRF) attacks, ensuring that requests are only sent to trusted and expected destinations.

**Source:**

- ✓ CWE-918: Server-Side Request Forgery (SSRF)

The web server receives a URL or similar request from an upstream component and retrieves the contents of this URL, but it does not sufficiently ensure that the request is being sent to the expected destination. By providing URLs to unexpected hosts or ports, attackers can make it appear that the server is sending the request, possibly bypassing access controls such as firewalls that prevent the attackers from accessing the URLs directly. The server can be used as a proxy to conduct port scanning of hosts in internal networks, use other URLs such as that can access documents on the system (using file://), or use other protocols such as gopher:// or tftp://, which may provide greater control over the contents of requests. It can cause XSPA:Cross Site Port Attack [35].

**Priority:** Not defined

**Rationale:** Preventing SSRF attacks protects the server from being exploited as a proxy to bypass access controls, conduct internal port scans, or perform other unauthorized actions, maintaining the integrity and security of the application and underlying systems.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:****1. Input Validation Test:**

- ✓ Test: Submit various URL inputs, including malicious and unexpected URLs.
- ✓ Expected Result: The application rejects or sanitizes unexpected or malformed URLs.

**2. Whitelist Test:**

- ✓ Test: Verify that the application sends requests only to pre-approved, trusted destinations.
- ✓ Expected Result: The application restricts outgoing requests to URLs in the defined whitelist.

**3. Protocol Restriction Test:**

- ✓ Test: Attempt to use unsupported or unsafe protocols such as gopher://, tftp://, or file://.
- ✓ Expected Result: The application rejects URLs using unsafe or unsupported protocols.

**4. Internal Network Access Test:**

- ✓ Test: Attempt to use SSRF to access internal network resources or perform port scans.
- ✓ Expected Result: The application blocks attempts to access internal resources via SSRF.

##### 5. Error Logging and Alerting Test:

- ✓ Test: Review logs and alert mechanisms for SSRF attempts.
- ✓ Expected Result: All SSRF attempts are logged with sufficient details, and alerts are generated for administrative review.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda

José L Fernández-Alemán

Juan Manuel Carrillo-de-Gea

Joaquín Nicolás

08/01/2026

**PUID:** [SECM-CAT-IOV-010]

**Requirement description:** The application must prevent canonical representation vulnerabilities by choosing a suitable canonical form and ensuring all user input is canonicalized before performing any authorization or security checks. Security validations must occur after decoding is completed, and the encoding method used must be verified as valid for its intended representation.

**Source:**

- ✓ V-222605: The application must protect from canonical representation vulnerabilities. A suitable canonical form should be chosen and all user input canonicalized into that form before any authorization decisions are performed. Security checks should be carried out after decoding is completed. Moreover, it is recommended to check that the encoding method chosen is a valid canonical encoding for the symbol it represents [15].

**Priority:** Not defined

**Rationale:** Canonicalization ensures consistency in input validation, preventing attackers from exploiting ambiguities in encoding to bypass security checks or perform unauthorized actions.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

###### 1. Canonicalization Test:

- ✓ Test: Provide inputs with various encodings (e.g., UTF-8, URL-encoded, Base64) and validate their transformation to a standard canonical form.
- ✓ Expected Result: All inputs are properly canonicalized to the selected standard form.

**2. Decoding Validation Test:**

- ✓ Test: Submit encoded data and inspect the decoding process for compliance with valid encoding methods.
- ✓ Expected Result: Only valid encodings are accepted, and decoding errors are properly handled.

**3. Authorization Consistency Test:**

- ✓ Test: Perform authorization checks using non-canonicalized and canonicalized inputs.
- ✓ Expected Result: Authorization decisions are consistent and based on canonicalized inputs.

**4. Injection Attempt Test:**

- ✓ Test: Attempt to exploit vulnerabilities by injecting encoded payloads (e.g., double encoding or mixed encodings).
- ✓ Expected Result: The application detects and blocks malformed or malicious payloads.

**5. Encoding Verification Test:**

- ✓ Test: Verify that the encoding used matches the expected canonical form for the data symbol it represents.
- ✓ Expected Result: Only correctly encoded representations are processed.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda  
 José L Fernández-Alemán  
 Juan Manuel Carrillo-de-Gea  
 Joaquín Nicolás  
 08/01/2026

**PUID:** [SECM-CAT-IOV-011]

**Requirement description:** The application must implement robust output sanitization measures to prevent cross-site scripting (XSS) attacks, ensuring all dynamic content displayed in WebViews, forms, or user interfaces is sanitized. This includes safeguarding against reflected, stored, and DOM-based XSS vulnerabilities by neutralizing untrusted input before it is used in dynamically generated content.

**Source:**

- ✓ Output Sanitization: 1. Properly sanitize output data to prevent cross-site scripting (XSS) attacks [3].
- ✓ Avoid Cross-Site Script Attacks (HTML modifications through malware) [25].
- ✓ CWE-79 Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')

The product does not neutralize or incorrectly neutralizes user-controllable input before it is placed in output that is used as a web page that is served to other users. It is related with XSS.

Cross-site scripting (XSS) vulnerabilities occur when:

1. Untrusted data enters a web application, typically from a web request.
2. The web application dynamically generates a web page that contains this untrusted data.
3. During page generation, the application does not prevent the data from containing content that is executable by a web browser, such as JavaScript, HTML tags, HTML attributes, mouse events, Flash, ActiveX, etc.
4. A victim visits the generated web page through a web browser, which contains malicious script that was injected using the untrusted data.
5. Since the script comes from a web page that was sent by the web server, the victim's web browser executes the malicious script in the context of the web server's domain.
6. This effectively violates the intention of the web browser's same-origin policy, which states that scripts in one domain should not be able to access resources or run code in a different domain.

There are three main kinds of XSS:

- Type 1: Reflected XSS (or Non-Persistent) - The server reads data directly from the HTTP request and reflects it back in the HTTP response. Reflected XSS exploits occur when an attacker causes a victim to supply dangerous content to a vulnerable web application, which is then reflected back to the victim and executed by the web browser. The most common mechanism for delivering malicious content is to include it as a parameter in a URL that is posted publicly or e-mailed directly to the victim. URLs constructed in this manner constitute the core of many phishing schemes, whereby an attacker convinces a victim to visit a URL that refers to a vulnerable site. After the site reflects the attacker's content back to the victim, the content is executed by the victim's browser.
- Type 2: Stored XSS (or Persistent) - The application stores dangerous data in a database, message forum, visitor log, or other trusted data store. At a later time, the dangerous data is subsequently read back into the application and included in dynamic content. From an attacker's perspective, the optimal place to inject malicious content is in an area that is displayed to either many users or particularly interesting users. Interesting users typically have elevated privileges in the application or interact with sensitive data that is valuable to the attacker. If one of these users executes malicious content, the attacker may be able to perform privileged operations on behalf of the user or gain access to sensitive data belonging to the user. For example, the attacker might inject XSS into a log message, which might not be handled properly when an administrator views the logs.
- Type 0: DOM-Based XSS - In DOM-based XSS, the client performs the injection of XSS into the page; in the other types, the server performs the injection. DOM-

<p>based XSS generally involves server-controlled, trusted script that is sent to the client, such as Javascript that performs sanity checks on a form before the user submits it. If the server-supplied script processes user-supplied data and then injects it back into the web page (such as with dynamic HTML), then DOM-based XSS is possible [35].</p>
<b>Priority:</b> Not defined
<b>Rationale:</b> Canonicalization ensures consistency in input validation, preventing attackers from exploiting ambiguities in encoding to bypass security checks or perform unauthorized actions.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<p><b>1. Reflected XSS Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Attempt to inject malicious scripts through input fields or query parameters that are reflected in the mobile app's WebView or dynamic content.</li> <li>✓ Expected Result: The app properly sanitizes input, preventing the execution of injected scripts in the WebView or UI.</li> </ul> <p><b>2. Stored XSS Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Inject scripts into persistent storage locations (e.g., notes, messages) and view the data in other app sessions or users.</li> <li>✓ Expected Result: Sanitized output ensures malicious scripts do not execute when stored data is displayed.</li> </ul> <p><b>3. DOM-based XSS Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Manipulate the Document Object Model (DOM) using scripts embedded in user inputs or URLs.</li> <li>✓ Expected Result: The app validates and neutralizes scripts to prevent client-side XSS execution.</li> </ul> <p><b>4. WebView Security Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Load dynamic content in WebViews and attempt to exploit XSS vulnerabilities using JavaScript or HTML injection.</li> <li>✓ Expected Result: WebView content is properly sanitized and secured against script execution.</li> </ul> <p><b>5. Output Encoding Validation:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Examine all output data for proper encoding (e.g., HTML, JavaScript).</li> <li>✓ Expected Result: Encoded output ensures that special characters are rendered harmless and do not execute as code.</li> </ul> <p><b>6. Penetration Testing:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Conduct comprehensive penetration testing to identify any XSS vulnerabilities.</li> <li>✓ Expected Result: No exploitable XSS vulnerabilities are found in the app.</li> </ul>
<b>Requested by:</b> The organization

<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IOV-012]
<b>Requirement description:</b> The application must implement output encoding techniques to sanitize data before displaying it in user interfaces or transmitting it over communication channels. This ensures that special characters are rendered safely, preventing unintended behavior or security vulnerabilities such as cross-site scripting (XSS).
<b>Source:</b>
✓ Output Sanitization: 1. Properly sanitize output data to prevent cross-site scripting (XSS) attacks [3].
<b>Priority:</b> Not defined
<b>Rationale:</b> Proper output encoding prevents malicious scripts or unauthorized code execution in dynamically generated content, protecting sensitive health data and ensuring application security.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li>1. <b>Encoding Inspection Test:</b> <ul style="list-style-type: none"> <li>✓ <b>Test:</b> Verify that all special characters in dynamic content are encoded before rendering in the UI or WebView.</li> <li>✓ <b>Expected Result:</b> Special characters, such as &lt;, &gt;, and ", are rendered as their encoded equivalents (e.g., &amp;lt;, &amp;gt;, &amp;quot;).</li> </ul> </li> <li>2. <b>HTML Injection Test:</b> <ul style="list-style-type: none"> <li>✓ <b>Test:</b> Attempt to inject HTML tags or scripts into data displayed in the UI or WebView.</li> <li>✓ <b>Expected Result:</b> HTML tags or scripts are displayed as text rather than executed in the WebView or UI.</li> </ul> </li> <li>3. <b>Communication Encoding Test:</b> <ul style="list-style-type: none"> <li>✓ <b>Test:</b> Monitor data transmitted from the app to ensure that output data containing special characters is properly encoded.</li> <li>✓ <b>Expected Result:</b> Data transmitted via API or other communication protocols uses output encoding to safeguard against injection attacks.</li> </ul> </li> </ol>

<p><b>4. Dynamic Content Validation:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Input and display special characters or scripts in dynamic content (e.g., search results, user-generated content).</li> <li>✓ Expected Result: Special characters are safely encoded and rendered without execution.</li> </ul> <p><b>5. Secure WebView Rendering Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Load and render user-supplied or external data in WebViews.</li> <li>✓ Expected Result: All dynamic content is encoded and displayed securely without enabling the execution of untrusted code.</li> </ul> <p><b>6. Penetration Testing:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Conduct penetration testing to identify any vulnerabilities related to output encoding.</li> <li>✓ Expected Result: The application passes all tests with no exploitable encoding vulnerabilities</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<p><b>Author and date:</b>            Carlos M. Mejía-Granda            José L Fernández-Alemán            Juan Manuel Carrillo-de-Gea            Joaquín Nicolás            08/01/2026</p>

<b>PUID:</b> [SECM-CAT-IOV-013]
<b>Requirement description:</b> The mobile healthcare application must be designed to prevent vulnerabilities to XML-oriented attacks by ensuring the use of secure and updated XML parsers and libraries, patching components promptly when vulnerabilities are identified, and implementing input validation for XML data
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ V-222608: The application must not be vulnerable to XML-oriented attacks. Design the application to utilize components that are not vulnerable to XML attacks. Patch the application components when vulnerabilities are discovered [15].</li> </ul>
<b>Priority:</b> Not defined
<b>Rationale:</b> Preventing XML-oriented attacks ensures the confidentiality, integrity, and availability of sensitive healthcare data processed by the application.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis

**Validation criteria:**

**1. XML Parser Security Test:**

- ✓ Verify that the application uses secure, modern, and updated XML parsers.
- ✓ Ensure that vulnerable or deprecated parsers are not used in the application.

**2. External Entity Resolution Test:**

- ✓ Confirm that the application disables external entity resolution (XXE) in its XML parsers to mitigate XXE attacks.

**3. Input Validation Test:**

- ✓ Validate that XML input data is sanitized and verified before processing.
- ✓ Test with malformed and malicious XML payloads to confirm proper rejection.

**4. Patch and Update Test:**

- ✓ Check that the application components, libraries, and parsers are up-to-date with the latest security patches.

**5. Penetration Test:**

- ✓ Conduct security testing to identify any residual vulnerabilities to XML-based attacks.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda  
José L Fernández-Alemán  
Juan Manuel Carrillo-de-Gea  
Joaquín Nicolás  
08/01/2026

**PUID:** [SECM-CAT-IOV-014]

**Requirement description:** The mobile application must mitigate XML-based DoS attacks by implementing robust mechanisms, such as XML filters, parser options, or gateways, to validate recursive payloads, oversized payloads, and prevent XML entity expansion. It must optimize configurations to limit overlong element names and ensure high message throughput while maintaining protection against service disruption.

**Source:**

- ✓ V-222593: XML-based applications must mitigate DoS attacks by using XML filters, parser options, or gateways. Implement:
  - Validation against recursive payloads

<ul style="list-style-type: none"> <li>○ Validation against oversized payloads</li> <li>○ Protection against XML entity expansion</li> <li>○ Validation against overlong element names</li> <li>○ Optimized configuration for maximum message throughput in order to ensure DoS attacks against web services are limited [15].</li> </ul>
<b>Priority:</b> Not defined
<b>Rationale:</b> Preventing XML-based DoS attacks ensures system availability and protects critical healthcare data in mobile applications.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Recursive Payload Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Send an XML payload with deeply nested elements to the application.</li> <li>✓ Expected Result: The application detects and rejects the payload without performance degradation.</li> </ul> </li> <li><b>2. Oversized Payload Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Send an XML payload exceeding the maximum allowed size to the application.</li> <li>✓ Expected Result: The application rejects the oversized payload and logs the event without crashing.</li> </ul> </li> <li><b>3. XML Entity Expansion Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Submit an XML payload containing a large number of entities that reference each other (e.g., a billion laughs attack).</li> <li>✓ Expected Result: The application rejects the payload and prevents resource exhaustion.</li> </ul> </li> <li><b>4. Overlong Element Names Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Submit an XML payload with excessively long element names.</li> <li>✓ Expected Result: The application rejects the payload and logs the attempt as a potential DoS attack.</li> </ul> </li> <li><b>5. Throughput Optimization Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Send a high volume of valid XML requests to the application concurrently.</li> <li>✓ Expected Result: The application processes requests efficiently without bottlenecks or increased latency.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda
José L Fernández-Alemán
Juan Manuel Carrillo-de-Gea
Joaquín Nicolás

08/01/2026

<b>PUID:</b> [SECM-CAT-IOV-015]
<b>Requirement description:</b> The mobile application must perform context-specific validation for all user inputs, particularly in file uploads or database queries, to prevent vulnerabilities such as path traversal and injection attacks. This includes validating and sanitizing file paths to ensure operations remain confined to designated directories.
<b>Source:</b>
<ul style="list-style-type: none"><li>✓ Context-Specific Validation: 1. Perform specific validation based on data context (e.g., file uploads, database queries) to prevent attacks like path traversal or injection [3].</li><li>✓ CWE-22: Improper Limitation of a Pathname to a Restricted Directory ('Path Transversal')<p>The product uses external input to construct a pathname that is intended to identify a file or directory that is located underneath a restricted parent directory, but the product does not properly neutralize special elements within the pathname that can cause the pathname to resolve to a location that is outside of the restricted directory. Many file operations are intended to take place within a restricted directory. By using special elements such as ".." and "/" separators, attackers can escape outside of the restricted location to access files or directories that are elsewhere on the system. One of the most common special elements is the ".." sequence, which in most modern operating systems is interpreted as the parent directory of the current location. This is referred to as relative path traversal. Path traversal also covers the use of absolute pathnames such as "/usr/local/bin", which may also be useful in accessing unexpected files. This is referred to as absolute path traversal. In many programming languages, the injection of a null byte (the 0 or NUL) may allow an attacker to truncate a generated filename to widen the scope of attack. For example, the product may add ".txt" to any pathname, thus limiting the attacker to text files, but a null injection may effectively remove this restriction [35].</p></li></ul>
<b>Priority:</b> Not defined
<b>Rationale:</b> Ensuring robust input validation protects against unauthorized access to sensitive files or directories, maintaining data integrity and system security.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"><li><b>1. Relative Path Traversal Test:</b><ul style="list-style-type: none"><li>✓ Test: Attempt to upload or access a file using "../" sequences in the file path.</li><li>✓ Expected Result: The application rejects the request and logs the attempt as a potential security threat.</li></ul></li><li><b>2. Absolute Path Traversal Test:</b><ul style="list-style-type: none"><li>✓ Test: Submit an absolute file path (e.g., /etc/passwd) in a user input field.</li></ul></li></ol>

- ✓ Expected Result: The application validates the input, restricts access to the allowed directory, and logs the attempt.

### 3. Null Byte Injection Test:

- ✓ Test: Include a null byte (%00) in the file name to bypass intended restrictions (e.g., "filename%00.jpg").
- ✓ Expected Result: The application sanitizes the input, rejects the malformed file name, and logs the activity.

### 4. File Upload Directory Restriction Test:

- ✓ Test: Attempt to upload a file to a directory outside the application's intended storage path.
- ✓ Expected Result: The application confines the file upload to designated directories and prevents directory traversal.

### 5. Database Query Validation Test:

- ✓ Test: Submit specially crafted inputs in queries to verify that database interactions are safe from injection attacks.
- ✓ Expected Result: The application detects and rejects unsafe inputs, logging the attempted violation.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda  
 José L Fernández-Alemán  
 Juan Manuel Carrillo-de-Gea  
 Joaquín Nicolás  
 08/01/2026

**PUID:** [SECM-CAT-IOV-016]

**Requirement description:** The mobile application must validate and sanitize all inputs from external sources and users, including data received through the UI, inter-process communication (IPC) mechanisms such as intents, custom URLs, and network sources, to prevent injection attacks and ensure input integrity.

**Source:**

- ✓ 6.2: Verify that all inputs from external sources and the user are validated and if necessary sanitized. This includes data received via the UI, IPC mechanisms such as intents, custom URLs, and network sources [19]
- ✓ Validate and sanitize user input: Validate and sanitize user input to prevent injection attacks that could expose API keys [18].
- ✓ MASVS-CODE-4: The app validates and sanitizes all untrusted inputs: Apps have many data entry points including the UI, IPC, the network, the file system, etc. This incoming data might have been inadvertently modified by untrusted actors and may lead to bypass of critical security checks as well as classical injection attacks such as SQL injection, XSS or

insecure deserialization. This control ensures that this data is treated as untrusted input and is properly verified and sanitized before it's used [20]
<b>Priority:</b> Not defined
<b>Rationale:</b> Unvalidated inputs can be exploited by attackers to bypass security controls, leading to data leaks, injection attacks, or system compromise. Proper input sanitization ensures robust protection against these threats.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. UI Input Validation Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Submit invalid or potentially malicious input (e.g., SQL commands, script tags) through the application's user interface.</li> <li>✓ Expected Result: The application sanitizes or rejects the input, preventing it from executing unintended operations.</li> </ul> </li> <li><b>2. IPC Mechanism Input Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Send malformed or malicious data via intents or custom URLs to test inter-process communication security.</li> <li>✓ Expected Result: The application detects and safely handles the malformed data without compromising functionality or security.</li> </ul> </li> <li><b>3. Network Data Input Validation Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Intercept network communications to inject untrusted data into the application.</li> <li>✓ Expected Result: The application validates and sanitizes incoming data, maintaining integrity and logging any suspicious activity.</li> </ul> </li> <li><b>4. File System Input Validation Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Attempt to load or process files containing unexpected or malformed data.</li> <li>✓ Expected Result: The application processes only valid, expected file formats and rejects unsafe inputs.</li> </ul> </li> <li><b>5. Injection Attack Simulation:</b> <ul style="list-style-type: none"> <li>✓ Test: Simulate injection attacks (e.g., SQL injection, XSS) through various input vectors.</li> <li>✓ Expected Result: The application resists all injection attempts, logging any detected malicious activities.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda
José L Fernández-Alemán
Juan Manuel Carrillo-de-Gea

Joaquín Nicolás 08/01/2026
-------------------------------

**PUID:** [SECM-CAT-IOV-017]

**Requirement description:** The mobile application must perform rigorous input validation for all data retrieved from external storage, treating it as untrusted input to prevent unauthorized access, data corruption, or injection attacks.

**Source:**

- ✓ Input validation: Perform input validation when handling data from external storage as you would with data from any untrusted source [18].

**Priority:** Not defined

**Rationale:** External storage is prone to tampering and may expose the application to risks such as injection attacks, data corruption, or unauthorized access. Validating input ensures that only safe and expected data is processed.

**Number of Children:** 0**Number of parents:** 0**Cycles:** 0**Number Audit:** 0**Child PUIDs:** Not described**Parent PUIDs:** Not described**Exclusion PUIDs:** Not described**Importance:** Not described**Current state:** To be determined**Verification method:** Demonstration/Analysis**Validation criteria:****1. External Storage Input Validation Test:**

- ✓ Test: Access files from external storage with different formats, including corrupted and maliciously crafted files.
- ✓ Expected Result: The application detects and rejects invalid files while processing only files with safe and expected formats.

**2. Injection Simulation Test:**

- ✓ Test: Inject malicious content into files on external storage (e.g., SQL commands, scripts) and attempt to access them through the application.
- ✓ Expected Result: The application identifies the malicious input, prevents execution, and logs the activity for further analysis.

**3. Boundary Value Test for Input Size:**

- ✓ Test: Provide files with excessively large or zero-byte sizes to test how the application handles boundary conditions.
- ✓ Expected Result: The application processes only files within expected size limits and gracefully handles boundary conditions.

**4. Error Handling Test:**

- ✓ Test: Force the application to access malformed or encrypted files without proper decryption keys.

✓ Expected Result: The application displays appropriate error messages without crashing or leaking sensitive information.
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IOV-018]
<b>Requirement description:</b> The mobile application must restrict access to device storage by implementing a whitelist of allowed file paths when arbitrary file selection is permitted, ensuring that only intended files are accessible. Additionally, comprehensive escape syntax must be applied to mitigate injection risks during file path handling.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ 1.30. If the application allows the arbitrary selection of files from the device storage, consider the use of a white-list to restrict access only to the intended (absolute) file paths [16].</li> <li>✓ 10.2. Define comprehensive escape syntax as appropriate [16].</li> </ul>
<b>Priority:</b> Not defined
<b>Rationale:</b> Allowing arbitrary file access poses significant security risks, including unauthorized data exposure and path traversal attacks. A whitelist ensures that only predefined, secure file paths are accessed. Escape syntax helps mitigate injection risks in file path manipulation.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Whitelist Enforcement Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Attempt to access files outside the predefined whitelist, including paths with ".." (parent directory traversal).</li> <li>✓ Expected Result: The application blocks access to files not explicitly included in the whitelist and logs the attempt.</li> </ul> </li> <li><b>2. Valid File Path Access Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Access files from the allowed whitelist paths.</li> </ul> </li> </ol>

<ul style="list-style-type: none"> <li>✓ Expected Result: The application successfully processes files from valid paths without errors.</li> </ul> <p><b>3. Escape Syntax Injection Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Attempt to inject escape sequences (e.g., null bytes, special characters) into file paths.</li> <li>✓ Expected Result: The application identifies and neutralizes escape sequences, preventing unauthorized file access or command execution.</li> </ul> <p><b>4. Boundary Path Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Provide edge-case file paths, such as very long filenames, symbolic links, or files with special characters.</li> <li>✓ Expected Result: The application handles such cases securely without exposing unintended files or directories.</li> </ul> <p><b>5. Error Handling Test for Unauthorized Files:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Try to access unauthorized files and observe the application's response.</li> <li>✓ Expected Result: The application provides a generic error message without revealing sensitive details about the file system.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IOV-019]
<b>Requirement description:</b> The application must enforce strict input size limitations to prevent excessive or malicious data from being passed to interpreters, ensuring data is within acceptable length parameters.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ 10.6. Limit size of input data passed to interpreters [16].</li> <li>✓ Input Validation: 2. Implement input length restrictions and reject unexpected or malicious data [3].</li> </ul>
<b>Priority:</b> Not defined
<b>Rationale:</b> Limiting input size helps mitigate risks such as buffer overflows, denial-of-service attacks, and potential exploitation through excessive data processing.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described

<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Input Size Validation Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Attempt to input data exceeding the defined size limit.</li> <li>✓ Expected result: The application rejects the oversized input with a user-friendly error message.</li> </ul> </li> <li><b>2. Buffer Overflow Simulation Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Input crafted data designed to overflow buffers.</li> <li>✓ Expected result: The application gracefully handles the input without crashing or exposing sensitive information.</li> </ul> </li> <li><b>3. Malicious Payload Length Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Inject a payload that reaches the size limit to evaluate rejection mechanisms.</li> <li>✓ Expected result: The payload is rejected, and the interpreter processes only validated data.</li> </ul> </li> <li><b>4. Penetration Testing:</b> <ul style="list-style-type: none"> <li>✓ Test: Simulate attacks with oversized or malicious input.</li> <li>✓ Expected result: The application resists the attack and maintains normal operations</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026
<b>PUID:</b> [SECM-CAT-IOV-020]
<b>Requirement description:</b> The application must implement robust data format verification for all file path inputs, avoiding reliance on character blocking or replacement mechanisms, to ensure only valid and intended paths are processed.
<b>Source:</b> <ul style="list-style-type: none"> <li>✓ Data Format Verification: The application should avoid relying solely on blocking specific characters or performing character replacements, as these methods can introduce errors and vulnerabilities [18].</li> </ul>
<b>Priority:</b> Not defined
<b>Rationale:</b> Ensuring proper data format verification reduces the risk of path traversal vulnerabilities and processing errors caused by inadequate input validation techniques.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described

<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Data Format Validation Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Submit file paths with various invalid formats or unexpected characters.</li> <li>✓ Expected result: The application identifies and rejects invalid file paths while processing only valid inputs.</li> </ul> </li> <li><b>2. Character Replacement Bypass Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Attempt to bypass validation by using special or encoded characters in file paths.</li> <li>✓ Expected result: The application rejects manipulated inputs and maintains the integrity of allowed file paths.</li> </ul> </li> <li><b>3. Path Traversal Prevention Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Use crafted input (e.g., ../ sequences) to attempt directory traversal.</li> <li>✓ Expected result: The application blocks all traversal attempts and ensures only legitimate paths are accessed.</li> </ul> </li> <li><b>4. Whitelisted Path Verification Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Access files using explicitly whitelisted paths.</li> <li>✓ Expected result: The application allows access only to valid, predefined paths and prevents unintended access.</li> </ul> </li> <li><b>5. Error Handling Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Provide malformed file path input to the application.</li> <li>✓ Expected result: The application gracefully handles errors and provides informative feedback without exposing sensitive details.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IOV-021]
<b>Requirement description:</b> The application must disable keyboard caching for text inputs that handle sensitive data, such as credentials or personal information, to prevent unintended exposure or unauthorized access.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ 2.4: Verify that the keyboard cache is disabled on text inputs that process sensitive data [19].</li> <li>✓ Avoid that credentials (personal data) be visible in the cache or in the code [25].</li> </ul>
<b>Priority:</b> Not defined
<b>Rationale:</b> Disabling keyboard caching for sensitive input fields mitigates the risk of exposing user credentials or personal data in keyboard prediction caches or memory.
<b>Number of Children:</b> 0

<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Keyboard Cache Disabling Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Inspect text input fields used for credentials or personal data to verify that keyboard caching is disabled.</li> <li>✓ Expected result: Keyboard prediction and caching are disabled for all sensitive input fields.</li> </ul> </li> <li><b>2. Data Caching Validation Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Enter sensitive data into text fields and check if the data appears in keyboard prediction or autocomplete suggestions.</li> <li>✓ Expected result: No sensitive data is retained or suggested by the keyboard.</li> </ul> </li> <li><b>3. Code Inspection Test:</b> <ul style="list-style-type: none"> <li>✓ <b>Test:</b> Review the application codebase to confirm that the <code>autofill</code> and <code>autocomplete</code> attributes are set to "off" for sensitive fields.</li> <li>✓ Expected result: Sensitive input fields explicitly disable keyboard caching and autocomplete functionality.</li> </ul> </li> <li><b>4. User Data Persistence Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Restart the application and confirm that previously entered sensitive data is not accessible or displayed.</li> <li>✓ Expected result: Sensitive data is not persisted in caches or visible upon app restart.</li> </ul> </li> <li><b>5. Penetration Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Perform a security test to ensure no sensitive data leaks occur through keyboard caching mechanisms.</li> <li>✓ Expected result: Sensitive data remains secure and is not retrievable through keyboard-related vulnerabilities.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IOV-022]
---------------------------------

<b>Requirement description:</b> The application must disable auto-correction and auto-suggestion features for input fields that handle sensitive data to prevent unintended disclosure or storage of sensitive information in predictive text systems.
<b>Source:</b>
✓ 1.20. Disable Auto Correction and Autosuggestion for inputs that contain sensitive data [16].
<b>Priority:</b> Not defined
<b>Rationale:</b> Disabling auto-correction and auto-suggestion for sensitive data fields reduces the risk of exposing private information through predictive text or keyboard cache systems.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Auto-Correction Feature Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Inspect input fields for sensitive data, such as passwords or personal identifiers, to verify that auto-correction is disabled.</li> <li>✓ Expected result: Auto-correction is not applied to sensitive input fields.</li> </ul> </li> <li><b>2. Auto-Suggestion Feature Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Enter sample sensitive data (e.g., simulated personal identifiers) into input fields and observe if suggestions or predictions appear.</li> <li>✓ Expected result: Auto-suggestions do not appear for sensitive input fields.</li> </ul> </li> <li><b>3. Code Attribute Validation Test:</b> <ul style="list-style-type: none"> <li>✓ <b>Test:</b> Review the application codebase to ensure input fields for sensitive data include attributes like <code>autocorrect="off"</code> and <code>autocomplete="off"</code>.</li> <li>✓ Expected result: All sensitive input fields have auto-correction and auto-suggestion explicitly disabled in the code.</li> </ul> </li> <li><b>4. Penetration Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Perform a security assessment to attempt retrieval of sensitive input data from keyboard auto-suggestion or correction systems.</li> <li>✓ Expected result: No sensitive data is accessible through keyboard systems.</li> </ul> </li> <li><b>5. Device Keyboard Behavior Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Use the application on devices with third-party keyboards and confirm sensitive fields do not trigger auto-suggestion or auto-correction.</li> <li>✓ Expected result: Sensitive fields remain unaffected by third-party keyboard behaviors.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda
José L Fernández-Alemán

Juan Manuel Carrillo-de-Gea  
 Joaquín Nicolás  
 08/01/2026

**PUID:** [SECM-CAT-IOV-023]

**Requirement description:** The application must restrict the use of third-party keyboards for input fields that handle sensitive data, such as credentials or payment information, and should use a custom keyboard for secure inputs instead.

**Source:**

- ✓ 1.19. Do not allow third party keyboards to be used for inputs that may contain sensitive data (e.g., credentials, credit card information). Prefer a custom keyboard for such inputs instead [16].

**Priority:** Not defined

**Rationale:** Limiting the use of third-party keyboards for sensitive inputs reduces the risk of data leakage or unauthorized access caused by potentially malicious or insecure third-party keyboard apps.

**Number of Children:** 0**Number of parents:** 0**Cycles:** 0**Number Audit:** 0**Child PUIDs:** Not described**Parent PUIDs:** Not described**Exclusion PUIDs:** Not described**Importance:** Not described**Current state:** To be determined**Verification method:** Demonstration/Analysis**Validation criteria:****1. Third-Party Keyboard Restriction Test:**

- ✓ Test: Attempt to use a third-party keyboard for sensitive input fields.
- ✓ Expected result: The sensitive input fields automatically disable third-party keyboards and use the default or custom secure keyboard.

**2. Custom Keyboard Implementation Test:**

- ✓ Test: Check whether sensitive input fields use a custom keyboard specifically designed for secure data entry.
- ✓ Expected result: Custom keyboards are applied to all sensitive data input fields.

**3. Code Attribute Validation Test:**

- ✓ Test: Inspect the codebase to ensure sensitive input fields include attributes or logic to enforce custom keyboard usage.
- ✓ Expected result: Sensitive fields explicitly enforce custom keyboard restrictions and block third-party keyboard usage.

**4. User Experience Test:**

- ✓ Test: Interact with sensitive input fields on various devices to ensure that users experience no interruptions or usability issues with the custom keyboard.
- ✓ Expected result: Custom keyboards function seamlessly and do not hinder user experience.

**5. Security Penetration Test:**

<ul style="list-style-type: none"> <li>✓ Test: Conduct tests to verify that sensitive data cannot be captured by third-party keyboards when interacting with sensitive input fields.</li> <li>✓ Expected result: No sensitive data is accessible by third-party keyboard apps.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IOV-024]
<b>Requirement description:</b> The application must disable clipboard functionalities, including cut, copy, and paste, for text fields containing sensitive data or restrict clipboard access to the application only.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ 2.5: Verify that the clipboard is deactivated on text fields that may contain sensitive data [19].</li> <li>✓ 1.21. Disable cut, copy and paste functionalities for inputs that may contain sensitive data or restrict the pasteboard to be accessible only from this application [16].</li> </ul>
<b>Priority:</b> Not defined
<b>Rationale:</b> Disabling clipboard access for sensitive inputs prevents unauthorized access, accidental exposure, or malicious interception of sensitive data via clipboard sharing or monitoring.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Clipboard Disablement Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Attempt to cut, copy, or paste sensitive data (e.g., passwords or credit card information) from text fields into another application.</li> <li>✓ Expected result: Clipboard functionalities are disabled for sensitive text fields, and no data transfer occurs.</li> </ul> </li> <li><b>2. Restricted Clipboard Scope Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Copy data within the application and attempt to paste it into a third-party application.</li> </ul> </li> </ol>

<ul style="list-style-type: none"> <li>✓ Expected result: Clipboard content is restricted to the application and cannot be accessed externally.</li> </ul> <p><b>3. Functionality Verification Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Verify that non-sensitive fields maintain clipboard functionality as expected.</li> <li>✓ Expected result: Clipboard actions work normally for non-sensitive fields.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IOV-025]
<b>Requirement description:</b> The application must implement input field masking for sensitive data, such as passwords and PINs, ensuring that they are not displayed in clear text during entry or stored as plain text.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ 1.24. Introduce input field masking for inputs that contain sensitive data (e.g., passwords) [16].</li> <li>✓ 9.4.2 Secure log-on procedures: i) not display a password being entered [7];</li> <li>✓ V-222554: The application must not display passwords/PINs as clear text. Configure the application to obfuscate passwords and PINs when they are being entered so they cannot be read. Design the application so obfuscated passwords cannot be copied and then pasted as clear text [15].</li> </ul>
<b>Priority:</b> Not defined
<b>Rationale:</b> Masking sensitive input fields prevents shoulder-surfing attacks and accidental exposure of sensitive information during data entry, enhancing user security.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Input Masking Test:</b></li> </ol> <ul style="list-style-type: none"> <li>✓ Test: Enter data into a password or PIN field and observe the display.</li> </ul>

<ul style="list-style-type: none"> <li>✓ Expected result: Entered characters are masked (e.g., displayed as dots or asterisks) and not shown in clear text.</li> </ul> <p><b>2. Copy and Paste Protection Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Attempt to copy and paste masked content from a password or PIN field into another application.</li> <li>✓ Expected result: Masked content cannot be copied or pasted as clear text.</li> </ul> <p><b>3. Clear Text Prohibition Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Inspect application logs and memory to ensure sensitive input fields are not stored as clear text during or after entry.</li> <li>✓ Expected result: No clear text instances of sensitive inputs are present in logs or memory.</li> </ul> <p><b>4. Dynamic Display Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Toggle masking (if the feature exists) during data entry for user convenience and security.</li> <li>✓ Expected result: Masking can be toggled securely without exposing the entire input unnecessarily.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IOV-026]
<b>Requirement description:</b> The application must validate all input data received through intent filters in intent receivers, ensuring the data conforms to the expected format and preventing unintended manipulation or execution.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ Input Validation in Intent Receivers: Intent filters are not a security mechanism. Always perform input validation within the intent receiver to ensure that the data conforms to the expected format for the intended activity, service, or receiver. This step prevents unintended data manipulation or execution [18].</li> <li>✓ MASVS-PLATFORM-1: The app uses IPC mechanisms securely: Apps typically use platform provided IPC mechanisms to intentionally expose data or functionality. Both installed apps and the user are able to interact with the app in many different ways. This control ensures that all interactions involving IPC mechanisms happen securely [20].</li> </ul>
<b>Priority:</b> Not defined
<b>Rationale:</b> Relying solely on intent filters for security is insufficient; input validation within intent receivers strengthens the app's defense against malicious or malformed intents.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0

<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Input Validation Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Send a crafted intent with unexpected or malformed data to the intent receiver.</li> <li>✓ Expected result: The application detects and rejects the invalid data, preventing further processing.</li> </ul> </li> <li><b>2. Expected Format Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Pass data conforming to the defined format to the intent receiver.</li> <li>✓ Expected result: The receiver processes the data without errors, ensuring the format is properly validated.</li> </ul> </li> <li><b>3. Malicious Intent Handling Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Inject intents with malicious payloads (e.g., unexpected scripts or code).</li> <li>✓ Expected result: The application rejects malicious intents and logs the attempt securely.</li> </ul> </li> <li><b>4. Security Audit Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Review the application's input validation logic within intent receivers.</li> <li>✓ Expected result: The logic confirms robust validation for all expected and unexpected inputs.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

### 2.9.3.5 Insecure Communication

<b>PUID:</b> [SECM-CAT-ICO-001]
<b>Requirement description:</b> The application must avoid reliance on a single insecure communication channel, such as email or SMS, for critical operations like enrollment and account recovery, ensuring multi-factor authentication mechanisms leverage secure channels.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ 5.5: Verify that the app doesn't rely on a single insecure communication channel (email or SMS) for critical operations, such as enrollments and account recovery [19].</li> <li>✓ 2.11. Do only rely on adequately secure channels for multi factor authentication (phone numbers and voice mails can be hijacked, see Section 4-10) [16]</li> </ul>

<b>Priority:</b> Not defined
<b>Rationale:</b> Using insecure channels like SMS or email for sensitive operations increases the risk of interception, hijacking, and unauthorized access, compromising user accounts and sensitive information.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Communication Channel Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Simulate account recovery using SMS or email as the sole communication channel.</li> <li>✓ Expected result: The application requires an additional secure channel (e.g., app-based MFA, push notification, or biometric authentication) before completing the operation.</li> </ul> </li> <li><b>2. Secure Multi-Factor Authentication Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Attempt to enroll or recover an account using insecure communication channels (e.g., SMS or email) without additional authentication.</li> <li>✓ Expected result: The application denies the operation and prompts the user to authenticate via a secure method.</li> </ul> </li> <li><b>3. Channel Hijacking Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Simulate a scenario where an attacker intercepts an SMS or email during account recovery.</li> <li>✓ Expected result: The intercepted data alone is insufficient to gain access, requiring authentication through a secure channel.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-ICO-002]
<b>Requirement description:</b> The application must avoid using SMS, MMS, or notifications to transfer sensitive data, opting instead for secure IP-based protocols such as Firebase Cloud Messaging (FCM) to ensure encrypted and authenticated communication.
<b>Source:</b>

- ✓ General Best Practices: 11. Do not send sensitive data over alternate channels (e.g, SMS, MMS, or notifications) [3].
- ✓ Use of SMS for Data Transfer: The application must not rely on the SMS protocol for transferring sensitive data due to its inherent limitations in encryption and authentication. Firebase Cloud Messaging (FCM) and IP-based networking must be used for sending data messages between the server and the app on user devices [18].

**Priority:** Not defined

**Rationale:** Using insecure channels like SMS or email for sensitive operations increases the risk of interception, hijacking, and unauthorized access, compromising user accounts and sensitive information.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

**1. Data Transmission Protocol Test:**

- ✓ Test: Attempt to transfer sensitive data over SMS or MMS in the application.
- ✓ Expected result: The application prevents sensitive data from being sent over SMS or MMS and redirects such operations to secure protocols like FCM.

**2. Notification Data Security Test:**

- ✓ Test: Inspect notifications sent by the application to verify the absence of sensitive information.
- ✓ Expected result: Notifications exclude sensitive data and adhere to secure messaging practices.

**3. Secure Channel Verification Test:**

- ✓ Test: Perform a data transfer from the app to the server and analyze the communication protocol.
- ✓ Expected result: The application exclusively uses IP-based networking with encryption, such as HTTPS or FCM, to send sensitive data.

**4. Penetration Testing:**

- ✓ Test: Simulate a man-in-the-middle (MITM) attack to intercept data during transmission.
- ✓ Expected result: Sensitive data is encrypted and securely transmitted, with no exposure via SMS or MMS channels.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda

José L Fernández-Alemán

Juan Manuel Carrillo-de-Gea  
Joaquín Nicolás  
08/01/2026

**PUID:** [SECM-CAT-ICO-003]

**Requirement description:** The application must apply an additional layer of encryption to sensitive data before transmitting it through the SSL/TLS channel, ensuring confidentiality even in the event of SSL/TLS vulnerabilities.

**Source:**

- ✓ General Best Practices: 12. If possible, apply a separate layer of encryption to any sensitive data before it is given to the SSL channel. In the event that future vulnerabilities are discovered in the SSL implementation, the encrypted data will provide a secondary defense against confidentiality violation [3].

**Priority:** Not defined

**Rationale:** Encrypting sensitive data independently of the transport layer provides a secondary defense mechanism, safeguarding user data against confidentiality violations caused by potential weaknesses in SSL/TLS implementations.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

**1. Secondary Encryption Implementation Test:**

- ✓ Test: Analyze the application's data transmission flow to verify the use of secondary encryption before sending data over SSL/TLS.
- ✓ Expected result: Sensitive data is encrypted with an additional secure algorithm prior to being transmitted via SSL/TLS.

**2. Decryption Robustness Test:**

- ✓ Test: Attempt to decrypt captured data from SSL/TLS sessions without the secondary encryption key.
- ✓ Expected result: The data remains inaccessible, proving that the secondary encryption is effective.

**3. SSL/TLS Bypass Simulation Test:**

- ✓ Test: Simulate an SSL/TLS vulnerability or downgrade attack to intercept data in transit.
- ✓ Expected result: The intercepted data is still encrypted and cannot be understood without the secondary encryption key.

**4. Algorithm Verification Test:**

- ✓ Test: Inspect the encryption algorithm used for the secondary layer to ensure it meets industry standards (e.g., AES-256).

- ✓ Expected result: The application employs a secure and widely recognized encryption algorithm for the secondary layer.

##### **5. Data Integrity Check:**

- ✓ Test: Validate that the secondary encryption does not alter the integrity of sensitive data during transmission or storage.
- ✓ Expected result: Sensitive data remains intact and unaltered after being encrypted and subsequently decrypted.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda

José L Fernández-Alemán

Juan Manuel Carrillo-de-Gea

Joaquín Nicolás

08/01/2026

**PUID:** [SECM-CAT-ICO-004]

**Requirement description:** The application must prioritize secure and authenticated communication methods, such as Firebase Cloud Messaging (FCM) or encrypted IP-based networking protocols, instead of relying on SMS for app-server interactions.

**Source:**

- ✓ Alternatives to SMS: For communication between the app and a server, the app must prioritize secure, authenticated methods such as FCM or other encrypted IP networking protocols instead of SMS [18].

**Priority:** Not defined

**Rationale:** SMS lacks robust encryption and authentication, making it vulnerable to interception and impersonation. Using secure, IP-based protocols ensures the confidentiality and authenticity of data transmitted between the app and server.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

##### **1. Communication Protocol Analysis Test:**

- ✓ Test: Examine the communication framework used for app-server interactions.
- ✓ Expected result: Communication is established using secure, authenticated methods such as FCM or encrypted IP protocols, without reliance on SMS.

##### **2. Authentication Mechanism Verification Test:**

- ✓ Test: Validate the authentication methods applied in app-server communications.

<ul style="list-style-type: none"> <li>✓ Expected result: Communications are authenticated, ensuring only legitimate devices and servers can exchange data.</li> </ul> <p><b>3. SMS Interception Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Attempt to intercept communications sent via SMS if any fallback is enabled.</li> <li>✓ Expected result: No sensitive data is sent via SMS, even during fallback scenarios.</li> </ul> <p><b>4. Encryption Assessment Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Confirm that all data exchanged over the communication channel is encrypted end-to-end.</li> <li>✓ Expected result: Data in transit is encrypted with robust protocols such as TLS 1.3.</li> </ul> <p><b>5. Alternative Channel Resilience Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Introduce simulated network vulnerabilities and evaluate the app's ability to maintain secure communication without reverting to SMS.</li> <li>✓ Expected result: The app continues to utilize secure alternatives without compromising communication security.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-ICO-005]
<b>Requirement description:</b> The application must treat SMS data as originating from untrusted sources and avoid performing sensitive operations based on unauthenticated SMS data. Safeguards against spoofing and interception must be implemented to mitigate risks associated with SMS as an insecure communication method.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ SMS Data Security: The application must assume that any SMS data received can be from an untrusted source and must not perform sensitive operations based on unauthenticated SMS data. SMS should be treated as a potentially insecure communication method, and the app must implement appropriate safeguards against spoofing and interception. [18].</li> </ul>
<b>Priority:</b> Not defined
<b>Rationale:</b> SMS lacks sufficient security features, such as encryption and robust authentication, making it vulnerable to spoofing, interception, and manipulation by adversaries. Implementing safeguards ensures the integrity and authenticity of actions triggered by SMS data.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined

<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<b>1. Input Source Validation Test:</b> ✓ Test: Analyze how SMS inputs are handled by the application. ✓ Expected result: The application validates SMS data against predefined rules and rejects unauthenticated or malformed data.
<b>2. Spoofing Simulation Test:</b> ✓ Test: Simulate an SMS spoofing attack to attempt triggering sensitive operations in the application. ✓ Expected result: The app does not execute sensitive operations based on unauthenticated SMS data.
<b>3. Operational Authentication Test:</b> ✓ Test: Attempt to use SMS as a sole authentication mechanism. ✓ Expected result: The app rejects SMS-based operations unless they are verified via an additional secure method.
<b>4. Intercepted SMS Test:</b> ✓ Test: Assess the application's behavior when SMS data is intercepted and modified. ✓ Expected result: The app flags tampered SMS data and avoids performing any operations based on it.
<b>5. Code Review for SMS Handling:</b> ✓ Test: Inspect the application's source code to identify safeguards against reliance on unauthenticated SMS data. ✓ Expected result: Secure coding practices are implemented to treat SMS data as untrusted and to prevent misuse.
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejia-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-ICO-006]
<b>Requirement description:</b> The application must assume that the network layer is unsecure and susceptible to eavesdropping. Implement end-to-end encryption and other secure communication practices to safeguard sensitive data transmitted over the network.
<b>Source:</b> ✓ General Best Practices: 1. Assume that the network layer is not secure and is susceptible to eavesdropping [3].
<b>Priority:</b> Not defined
<b>Rationale:</b> Network communications may be intercepted by malicious actors, leading to unauthorized access or data breaches. End-to-end encryption ensures confidentiality and integrity, even when the network is compromised.

<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Network Interception Simulation Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Simulate a man-in-the-middle (MITM) attack during data transmission between the app and the server.</li> <li>✓ Expected result: The app uses encrypted communication protocols, preventing eavesdropping or tampering with transmitted data.</li> </ul> </li> <li><b>2. Encryption Protocol Validation Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Verify the implementation of TLS (minimum v1.2) or equivalent encryption for all network communications.</li> <li>✓ Expected result: Data in transit is encrypted, and the server's certificate is validated.</li> </ul> </li> <li><b>3. End-to-End Encryption Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Confirm that the application encrypts sensitive data at the application layer before transmission.</li> <li>✓ Expected result: Sensitive data remains encrypted throughout transmission and is decrypted only on the intended recipient's device.</li> </ul> </li> <li><b>4. Traffic Inspection Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Inspect network traffic using tools like Wireshark or Burp Suite to detect any plaintext data transmission.</li> <li>✓ Expected result: No sensitive data is visible in plaintext during transmission.</li> </ul> </li> <li><b>5. Code Review for Secure Communication Practices:</b> <ul style="list-style-type: none"> <li>✓ Test: Analyze the app's source code for the implementation of secure communication libraries and proper encryption methods.</li> <li>✓ Expected result: Secure communication libraries are used, and no sensitive data is transmitted without encryption.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-ICO-007]
---------------------------------

<b>Requirement description:</b> The application must restrict the exposure of sensitive information via broadcast intents, especially when handling SMS messages. Broadcast messages must be secured,
---

limited to authorized applications, and sensitive data must not be transmitted using insecure methods such as SMS or MMS.

**Source:**

- ✓ Use of Broadcast Intents: Since SMS messages on Android devices are transmitted as broadcast intents, the app must not expose sensitive information through these intents. Any SMS message that the app handles must be secured and must not be broadcast or accessed by other apps unless they have the necessary permissions (e.g., READ\_SMS) [18].
- ✓ 1.18. Restrict broadcast messages (e.g., Android Broadcast Intents) to authorized applications and audit the application's broadcast messages for sensitive content [16].
- ✓ 12.8. Restrict the third-party applications whose broadcast messages will be accepted by the application [16]
- ✓ 4.10. SMS and MMS should not be used to send sensitive data (e.g., two-factor authentication tokens) to or from mobile end-points as SMS and MMS can be intercepted [16].

**Priority:** Not defined

**Rationale:** Broadcast intents, SMS, and MMS are inherently vulnerable to interception and unauthorized access. Ensuring secure handling of such messages reduces the risk of data leaks and unauthorized access to sensitive information.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:****1. Broadcast Intent Permission Test:**

- ✓ Test: Verify that all broadcast intents involving sensitive data are protected using permissions such as READ\_SMS.
- ✓ Expected result: Only authorized apps with the specified permissions can access broadcast intents.

**2. Sensitive Data Inspection Test:**

- ✓ Test: Analyze all broadcast messages for the presence of sensitive data.
- ✓ Expected result: Sensitive data is never included in broadcast intents.

**3. Third-Party Access Restriction Test:**

- ✓ Test: Attempt to access broadcast messages from unauthorized third-party applications.
- ✓ Expected result: Unauthorized apps are denied access to the app's broadcast messages.

**4. Secure Messaging Test:**

- ✓ Test: Validate that sensitive information such as two-factor authentication tokens is transmitted using secure channels rather than SMS or MMS.
- ✓ Expected result: Sensitive data is transmitted securely using encrypted IP-based messaging protocols.

<b>5. Code Review for Broadcast Intent Handling:</b> ✓ <b>Test:</b> Inspect the source code to confirm the use of <code>setPackage()</code> or similar methods to target specific applications in broadcast intents. ✓ <b>Expected result:</b> Broadcast intents are explicitly restricted to intended recipients, and sensitive data is adequately protected.
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-ICO-008]
<b>Requirement description:</b> The mobile application must ensure the confidentiality and integrity of electronically transmitted data, including electronic protected health information (ePHI), by implementing industry-standard encryption protocols (e.g., TLS/SSL) and integrity controls during data transmission.
<b>Source:</b> ✓ § 164.312 Technical safeguards. (e) (1) Standard: Transmission security. Implement technical security measures to guard against unauthorized access to electronic protected health information that is being transmitted over an electronic communications network. (2) Implementation specifications: (i) Integrity controls (Addressable). Implement security measures to ensure that electronically transmitted electronic protected health information is not improperly modified without detection until disposed of. (ii) Encryption (Addressable). Implement a mechanism to encrypt electronic protected health information whenever deemed appropriate [9]. ✓ 4.7. Communications and Operations Management  Security Requirement 30 – Encrypting PHI During Transmission: The EHRI and PoS systems connected to the EHRI must apply industry-standard cryptographic algorithms and protocols during transmission of PHI to maintain the confidentiality and integrity of this data whenever it is transmitted outside the physical security perimeters that protects information processing facilities supporting EHRI servers, applications or data [10].  ✓ SC-8 TRANSMISSION CONFIDENTIALITY AND INTEGRITY

Control: Protect the [Selection (one or more): confidentiality; integrity] of transmitted information.

Discussion: Protecting the confidentiality and integrity of transmitted information applies to internal and external networks as well as any system components that can transmit information, including servers, notebook computers, desktop computers, mobile devices, printers, copiers, scanners, facsimile machines, and radios. Unprotected communication paths are exposed to the possibility of interception and modification. Protecting the confidentiality and integrity of information can be accomplished by physical or logical means. Physical protection can be achieved by using protected distribution systems. A protected distribution system is a wireline or fiber-optics telecommunications system that includes terminals and adequate electromagnetic, acoustical, electrical, and physical controls to permit its use for the unencrypted transmission of classified information. Logical protection can be achieved by employing encryption techniques [11].

- ✓ V-222598: The application must maintain the confidentiality and integrity of information during preparation for transmission. Configure all of the application systems to require TLS encryption [15].
- ✓ SRG-APP-000439-MAPP-000100: The mobile app must protect the confidentiality and integrity of transmitted information [17].
- ✓ MASVS-NETWORK-1: The app secures all network traffic according to the current best practices: Ensuring data privacy and integrity of any data in transit is critical for any app that communicates over the network. This is typically done by encrypting data and authenticating the remote endpoint, as TLS does. However, there are many ways for a developer to disable the platform secure defaults, or bypass them completely by using low-level APIs or third-party libraries. This control ensures that the app is in fact setting up secure connections in any situation [20].
- ✓ 4.9.3. Cryptographic Controls.

Consideration MC22 – Ensure Communications Channel Encryption: Organizations should ensure that all mobile communications channels that transmit or receive confidential information are encrypted [10].

- ✓ Test ID 15: The confidentiality and integrity of EHR information is protected while in transit (SC-8) by using a cryptographic mechanism [37].
- ✓ Secure Data Transmission: Utilize secure communication protocols (e.g., HTTPS, SSL/TLS) to protect data during transmission between the mobile application and backend servers. Avoid sending sensitive data over unsecured channels [3].
- ✓ iOS Specific Best Practices: 2. When using CFNetwork, consider using the Secure Transport API to designate trusted client certificates. In almost all situations, NSSocketSecurityLevelTLSv1 should be used for higher standard cipher strength [3].

**Priority:** Not defined

**Rationale:** Transmitting sensitive health information over electronic communication networks requires robust security to protect against unauthorized access, interception, or modification, ensuring compliance with legal and regulatory standards for ePHI protection.

<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<p><b>1. Secure Protocol Verification Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Verify that all network communications between the mobile app and backend servers use TLS 1.2 or higher.</li> <li>✓ Expected result: All communications are encrypted with a secure protocol, and unencrypted channels are rejected.</li> </ul> <p><b>2. Integrity Validation Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Validate that integrity checks (e.g., message authentication codes) are implemented for transmitted data.</li> <li>✓ Expected result: Data modifications during transit are detected and rejected by the system.</li> </ul> <p><b>3. Encryption Algorithm Verification Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Ensure that the encryption algorithms used (e.g., AES-256 for data encryption) meet current industry standards.</li> <li>✓ Expected result: Encryption mechanisms comply with industry best practices for security and performance.</li> </ul> <p><b>4. Simulated Interception Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Perform penetration testing to simulate data interception attempts during transmission.</li> <li>✓ Expected result: Attempts to intercept or modify transmitted data are unsuccessful, and alerts are generated where applicable.</li> </ul> <p><b>5. Configuration Inspection Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Inspect the app's network configuration to ensure the use of secure libraries and APIs (e.g., Secure Transport for iOS or strong cipher suites in TLS).</li> <li>✓ Expected result: Secure configurations are in place, and insecure protocols or cipher suites are disabled.</li> </ul> <p><b>6. Endpoint Authentication Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Verify that the app authenticates the remote server using trusted certificates and rejects connections to untrusted endpoints.</li> <li>✓ Expected result: Only trusted endpoints are connected, and certificate validation errors are logged and reported.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda
José L Fernández-Alemán

Juan Manuel Carrillo-de-Gea  
 Joaquín Nicolás  
 08/01/2026

**PUID:** [SECM-CAT-ICO-009]

**Requirement description:** The mobile application must use hashed values derived from secure hash functions (e.g., SHA-256 or bcrypt) as primary keys or identifiers to avoid directly storing or transmitting personal information.

**Source:**

- ✓ Data Handling Practices: The application must use hashed values (e.g., hash of an email address) as primary keys to avoid storing or transmitting personal information [18].
- ✓ Use Strong Hash Functions: 1. Choose widely recognized and cryptographically secure hash functions like SHA-256 or bcrypt. These algorithms are designed to resist attacks and provide a high level of security [3].

**Priority:** Not defined

**Rationale:** Using hashed values as primary keys or identifiers minimizes the exposure of personal information and enhances data privacy and security. Secure hash functions resist attacks, ensuring data integrity and confidentiality.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:****1. Hash Function Verification Test:**

- ✓ Test: Verify that the application uses cryptographically secure hash functions (e.g., SHA-256 or bcrypt) for generating hashed primary keys.
- ✓ Expected result: All hashed identifiers are generated using secure and approved hash functions.

**2. Personal Information Obfuscation Test:**

- ✓ Test: Inspect application data structures to ensure personal information (e.g., email addresses) is obfuscated using hash functions before storage or transmission.
- ✓ Expected result: No plain-text personal information is stored or transmitted; only hashed values are utilized.

**3. Hash Collision Test:**

- ✓ Test: Simulate high-volume data to check for hash collisions in primary keys.
- ✓ Expected result: No hash collisions occur, ensuring the uniqueness of primary keys.

**4. Data Transmission Test:**

- ✓ Test: Analyze network traffic during application operations to confirm that no plain-text personal information is transmitted.

- ✓ Expected result: Only hashed values or anonymized identifiers are transmitted over the network.

#### 5. Penetration Test for Reversibility:

- ✓ Test: Attempt to reverse-engineer hashed identifiers to retrieve original personal information.
- ✓ Expected result: The hash function implementation ensures irreversibility, and no personal information is recoverable.

#### 6. Storage Configuration Test:

- ✓ Test: Verify that hashed primary keys are securely stored and protected with additional safeguards (e.g., encryption at rest).
- ✓ Expected result: Hashed values are stored securely, and access to sensitive data is appropriately restricted.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda  
José L Fernández-Alemán  
Juan Manuel Carrillo-de-Gea  
Joaquín Nicolás  
08/01/2026

**PUID:** [SECM-CAT-ICO-010]

**Requirement description:** The mobile application must ensure confidentiality and integrity of information during reception by enforcing TLS encryption for all data exchanges.

**Source:**

- ✓ V-222599: The application must maintain the confidentiality and integrity of information during reception. Configure all of the application systems to require TLS encryption [15].
- ✓ Data transmission via a secure channel [26].

**Priority:** Not defined

**Rationale:** TLS encryption ensures that sensitive information remains confidential and intact during data transmission, protecting it from interception, tampering, or unauthorized access. This is critical for maintaining trust and security in health-related mobile applications.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

1. **TLS Configuration Test:**

- ✓ Test: Verify that all communication channels in the application enforce TLS 1.2 or higher for secure data transmission during information reception.

- ✓ Expected result: All endpoints and communication protocols use TLS 1.2 or higher with no fallback to insecure protocols.

**2. Certificate Validation Test:**

- ✓ Test: Check the application's ability to validate the server certificate during the handshake process.

- ✓ Expected result: Only valid, trusted certificates are accepted, and connections are terminated for invalid or expired certificates.

**3. Data Integrity Test:**

- ✓ Test: Simulate data tampering during transmission and inspect whether the application detects and rejects modified data.

- ✓ Expected result: Any tampered data during reception is rejected, and the user is notified appropriately.

**4. Encryption Protocol Inspection:**

- ✓ Test: Use network traffic analysis tools to confirm that data received by the application is encrypted using TLS.

- ✓ Expected result: All data packets are encrypted, and no sensitive information is transmitted in plaintext.

**5. Fallback Prevention Test:**

- ✓ Test: Attempt to force the application to use insecure protocols (e.g., SSL 3.0 or TLS 1.0) during data reception.

- ✓ Expected result: The application rejects the connection and enforces the use of TLS 1.2 or higher.

**6. Penetration Testing:**

- ✓ Test: Conduct penetration testing to identify vulnerabilities in the encryption and data reception process.

- ✓ Expected result: No exploitable vulnerabilities are found in the TLS configuration or data handling mechanisms.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda

José L Fernández-Alemán

Juan Manuel Carrillo-de-Gea

Joaquín Nicolás

08/01/2026

**PUID:** [SECM-CAT-ICO-011]

<b>Requirement description:</b> The mobile application must sign critical message elements in SOAP messages requiring integrity, including Message ID, Service Request, Timestamp, SAML Assertion, and Message elements.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ V-222398: Design and configure the application to sign the following message elements for SOAP messages requiring integrity:           <ul style="list-style-type: none"> <li>- Message ID</li> <li>- Service Request</li> <li>- Timestamp</li> <li>- SAML Assertion</li> <li>- Message elements [15].</li> </ul> </li> </ul>
<b>Priority:</b> Not defined
<b>Rationale:</b> Signing critical elements ensures the integrity and authenticity of SOAP messages during transmission. This is essential to prevent unauthorized modifications, replay attacks, and ensure trust in message exchange, especially in sensitive health-related data transactions.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Message Signing Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Verify that the application signs the specified SOAP message elements (Message ID, Service Request, Timestamp, SAML Assertion, and Message elements) before transmission.</li> <li>✓ Expected result: All specified message elements are cryptographically signed with the correct keys.</li> </ul> </li> <li><b>2. Signature Validation Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Ensure the receiving endpoint validates the signatures for the signed elements of the SOAP message.</li> <li>✓ Expected result: Messages with invalid or missing signatures are rejected.</li> </ul> </li> <li><b>3. Tamper Detection Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Modify one or more signed elements of a SOAP message during transmission and verify the application's response.</li> <li>✓ Expected result: The application detects tampering and rejects the message with an appropriate error log.</li> </ul> </li> <li><b>4. Timestamp Integrity Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Confirm that the signed Timestamp element prevents replay attacks by enforcing expiration policies.</li> <li>✓ Expected result: Expired or modified timestamps are detected, and messages are rejected.</li> </ul> </li> <li><b>5. Key Usage Validation Test:</b></li> </ol>

- ✓ Test: Validate that the application uses appropriate cryptographic keys for signing SOAP message elements.
- ✓ Expected result: Keys used for signing are valid, trusted, and adhere to organizational cryptographic standards.

#### 6. Interoperability Test:

- ✓ Test: Test SOAP message signing and verification between the mobile application and third-party systems.
- ✓ Expected result: All signed messages are successfully verified by compliant third-party systems, ensuring interoperability.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda  
 José L Fernández-Alemán  
 Juan Manuel Carrillo-de-Gea  
 Joaquín Nicolás  
 08/01/2026

**PUID:** [SECM-CAT-ICO-012]

**Requirement description:** The mobile application must ensure the confidentiality and integrity of transmitted information by requiring TLS encryption for all communications. Cryptographic mechanisms such as HMAC-SHA1, HMAC-SHA-256, HMAC-SHA-512, or GCM mode must be implemented to prevent unauthorized disclosure and detect tampering during data transmission.

**Source:**

- ✓ V-222596: The application must protect the confidentiality and integrity of transmitted information. Configure all of the application systems to require TLS encryption in accordance with data protection requirements [15].
- ✓ V-222597: The application must implement cryptographic mechanisms to prevent unauthorized disclosure of information and/or detect changes to information during transmission unless otherwise protected by alternative physical safeguards, such as, at a minimum, a Protected Distribution System (PDS). Configure the application to use cryptographic protections to prevent unauthorized disclosure of application data based upon the application architecture [15].
- ✓ Data transmission via a secure channel [26].
- ✓ Integrity Implementation: Ensure data integrity by pairing encryption modes with one of the following:
  - HMAC-SHA1
  - HMAC-SHA-256
  - HMAC-SHA-512
  - GCM mode [18].

- ✓ MASVS-NETWORK-1: The app secures all network traffic according to the current best practices: Ensuring data privacy and integrity of any data in transit is critical for any app that communicates over the network. This is typically done by encrypting data and authenticating the remote endpoint, as TLS does. However, there are many ways for a developer to disable the platform secure defaults, or bypass them completely by using low-level APIs or third-party libraries. This control ensures that the app is in fact setting up secure connections in any situation [20].
- ✓ Explicit encryption on data stored on my mobile device and the data transmitted to a remote server [26].

**Priority:** Not defined

**Rationale:** Securing network traffic with TLS encryption and strong cryptographic mechanisms ensures data protection during transmission, safeguarding against unauthorized access, tampering, or replay attacks. This is critical for maintaining the confidentiality and integrity of sensitive healthcare data.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

### 1. TLS Encryption Verification Test:

- ✓ Test: Analyze the network traffic to confirm that all transmitted data is encrypted using TLS.
- ✓ Expected result: All network communications are encrypted using TLS 1.2 or higher.

### 2. Cryptographic Mechanism Test:

- ✓ Test: Validate the use of HMAC-SHA1, HMAC-SHA-256, HMAC-SHA-512, or GCM mode to ensure data integrity during transmission.
- ✓ Expected result: Cryptographic mechanisms are correctly applied to prevent unauthorized data tampering.

### 3. Integrity Verification Test:

- ✓ Test: Modify transmitted data in transit and validate the system's ability to detect unauthorized changes.
- ✓ Expected result: Any tampering with the data is detected, and the transmission is rejected.

**4. Certificate Validation Test:**

- ✓ Test: Verify that the application validates TLS certificates, including checking for expiration and revocation.
- ✓ Expected result: Only valid and trusted certificates are accepted.

**5. Fallback Prevention Test:**

- ✓ Test: Attempt to establish connections using deprecated or insecure encryption protocols such as TLS 1.0 or SSL.
- ✓ Expected result: The application rejects connections using insecure protocols.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda  
José L Fernández-Alemán  
Juan Manuel Carrillo-de-Gea  
Joaquín Nicolás  
08/01/2026

**PUID:** [SECM-CAT-ICO-013]

**Requirement description:** The mobile application must validate all data downloaded over insecure protocols, such as HTTP. Input validation must be enforced on all responses, particularly when processing data in WebView components or handling responses to HTTP intents, to ensure untrusted data does not compromise application security.

**Source:**

- ✓ Validation of Downloaded Data: Data downloaded over insecure protocols such as HTTP must not be trusted by default. Input validation must be enforced for all responses, especially when handling data in components like WebView or processing responses to HTTP intents [18]

**Priority:** Not defined

**Rationale:** Validating downloaded data from insecure sources ensures that malicious or manipulated data cannot compromise the integrity, functionality, or security of the application. This is critical for protecting sensitive information and preventing unauthorized access or code execution.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<b>1. HTTP Response Validation Test:</b> <ul style="list-style-type: none"><li>✓ Test: Simulate data downloads over HTTP and verify that the application performs input validation on the received data.</li><li>✓ Expected result: Data is validated, and any unexpected or malformed data is rejected or sanitized.</li></ul>
<b>2. WebView Data Handling Test:</b> <ul style="list-style-type: none"><li>✓ Test: Load untrusted or malformed data into WebView and observe application behavior.</li><li>✓ Expected result: The WebView component rejects or properly escapes unsafe content without executing scripts or rendering malicious elements.</li></ul>
<b>3. Injection Attack Simulation Test:</b> <ul style="list-style-type: none"><li>✓ Test: Inject malicious data into HTTP responses and validate that the application detects and blocks the attack.</li><li>✓ Expected result: The application identifies and neutralizes injection attempts, ensuring no compromise of security.</li></ul>
<b>4. Secure Protocol Usage Test:</b> <ul style="list-style-type: none"><li>✓ Test: Attempt to force the application to process data downloaded over insecure protocols without validation.</li><li>✓ Expected result: The application enforces validation or prevents processing untrusted data.</li></ul>
<b>5. Error Handling Test:</b> <ul style="list-style-type: none"><li>✓ Test: Validate the application's response when receiving corrupted or unexpected data over HTTP.</li><li>✓ Expected result: The application gracefully handles errors, rejecting malicious data while maintaining functionality.</li></ul>
<b>6. Penetration Testing:</b> <ul style="list-style-type: none"><li>✓ Test: Conduct penetration testing focused on manipulating data downloaded over HTTP to identify vulnerabilities.</li><li>✓ Expected result: No vulnerabilities are exploitable through manipulated or unvalidated data responses.</li></ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

**PUID:** [SECM-CAT-ICO-014]

<b>Requirement description:</b> The mobile application must configure all session cookies to include the <code>HTTPOnly</code> flag to prevent client-side scripts from accessing session cookies, mitigating the risk of cross-site scripting (XSS) attacks.
<b>Source:</b>
✓ V-222575: The application must set the <code>HTTPOnly</code> flag on session cookies. Configure the application to set the <code>HTTPOnly</code> flag on session cookies [15].
<b>Priority:</b> Not defined
<b>Rationale:</b> Setting the <code>HTTPOnly</code> flag on session cookies ensures that sensitive cookies are inaccessible to malicious client-side scripts, reducing the potential for XSS attacks to compromise session data.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Cookie Configuration Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Inspect session cookies set by the application using developer tools or browser extensions.</li> <li>✓ Expected result: All session cookies include the <code>HTTPOnly</code> flag in their attributes.</li> </ul> </li> <li><b>2. Client-Side Access Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Attempt to access session cookies using JavaScript in a simulated XSS attack.</li> <li>✓ Expected result: Access to session cookies is blocked, and their contents remain secure.</li> </ul> </li> <li><b>3. Cross-Site Scripting Simulation Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Inject a malicious script that tries to steal session cookies through JavaScript.</li> <li>✓ Expected result: The script fails to access the cookies due to the <code>HTTPOnly</code> flag.</li> </ul> </li> <li><b>4. Error Handling and Fallback Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Verify the application's behavior when setting <code>HTTPOnly</code> cookies in environments with restrictive policies.</li> <li>✓ Expected result: The application falls back to secure handling mechanisms or informs the user of the limitation without compromising session data.</li> </ul> </li> <li><b>5. Compliance Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Conduct a compliance review of cookie policies against security standards (e.g., OWASP).</li> <li>✓ Expected result: Cookie configuration aligns with best practices, including <code>HTTPOnly</code> for session cookies.</li> </ul> </li> <li><b>6. Penetration Testing:</b> <ul style="list-style-type: none"> <li>✓ Test: Perform penetration testing to simulate attempts to access or manipulate session cookies.</li> <li>✓ Expected result: Cookies remain protected and inaccessible through unauthorized means.</li> </ul> </li> <li><b>7. Log Inspection Test:</b></li> </ol>

<ul style="list-style-type: none"> <li>✓ Test: Review server logs to ensure that session cookies are flagged appropriately and securely transmitted.</li> <li>✓ <b>Expected result:</b> All logs indicate proper configuration of session cookies with the <code>HTTPOnly</code> attribute.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-ICO-015]
<b>Requirement description:</b> The mobile application must configure all session cookies to include the <code>Secure</code> flag, ensuring that cookies are only transmitted over encrypted HTTPS connections to prevent their exposure in plaintext communication.
<b>Source:</b> <ul style="list-style-type: none"> <li>✓ V-222576: The application must set the secure flag on session cookies. Configure the application to ensure the secure flag is set on session cookies [15].</li> </ul>
<b>Priority:</b> Not defined
<b>Rationale:</b> Setting the <code>Secure</code> flag on session cookies ensures that they are transmitted exclusively over secure channels, such as HTTPS, mitigating the risk of interception by attackers through network sniffing or man-in-the-middle (MITM) attacks.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b> <ol style="list-style-type: none"> <li>1. <b>Cookie Configuration Test:</b> <ul style="list-style-type: none"> <li>✓ <b>Test:</b> Inspect session cookies using developer tools or browser extensions to verify the inclusion of the <code>Secure</code> flag.</li> <li>✓ <b>Expected result:</b> All session cookies include the <code>Secure</code> flag in their attributes.</li> </ul> </li> <li>2. <b>Insecure Channel Transmission Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Attempt to transmit session cookies over an HTTP connection.</li> <li>✓ Expected result: Session cookies are not transmitted over insecure HTTP channels.</li> </ul> </li> <li>3. <b>Encryption Enforcement Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Ensure the application enforces HTTPS connections for all cookie transmission.</li> </ul> </li> </ol>

<ul style="list-style-type: none"> <li>✓ Expected result: Cookies are transmitted only over encrypted HTTPS connections, and HTTP connections are blocked or redirected.</li> </ul>
<b>4. Cross-Browser Test:</b>
<ul style="list-style-type: none"> <li>✓ <b>Test:</b> Verify that session cookies with the <code>Secure</code> flag function correctly across different browsers and platforms.</li> <li>✓ <b>Expected result:</b> The <code>Secure</code> flag is enforced consistently across all tested environments.</li> </ul>
<b>5. Penetration Testing:</b>
<ul style="list-style-type: none"> <li>✓ <b>Test:</b> Perform penetration tests to identify vulnerabilities in cookie handling, specifically focusing on secure flag enforcement.</li> <li>✓ <b>Expected result:</b> Session cookies are not accessible or exposed during testing.</li> </ul>
<b>6. Server Configuration Test:</b>
<ul style="list-style-type: none"> <li>✓ <b>Test:</b> Review the server-side configuration to ensure all session cookies are automatically set with the <code>Secure</code> flag.</li> <li>✓ <b>Expected result:</b> The server enforces the <code>Secure</code> flag for all session cookies by default.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-ICO-016]
<b>Requirement description:</b> The mobile application must only use certificates signed by a trusted Certificate Authority (CA) and ensure strict certificate validation, including checking for self-signed, expired, revoked, or misconfigured certificates. Connections with endpoints offering invalid certificates must not be allowed.
<b>Source:</b> <ul style="list-style-type: none"> <li>✓ General Best Practices: 5. Use certificates signed by a trusted CA provider [3].</li> <li>✓ 5.4: Verify that the app either uses its own certificate store, or pins the endpoint certificate or public key, and subsequently does not establish connections with endpoints that offer a different certificate or key, even if signed by a trusted CA [19].</li> <li>✓ 4.7. Use certificates signed by trusted CA providers. Do not allow self-signed certificates and do not disable or ignore certificate chain validation [16].</li> <li>✓ Encrypt the code. Apps must be encoded and signed by trusted sources [25].</li> </ul>

- ✓ General Best Practices: 6. Never allow bad certificates (self-signed, expired, untrusted root, revoked, wrong host..) [3].
- ✓ 1.3. When storing sensitive data on the device, use a file encryption API provided by the OS or other trusted source. Some platforms (e.g., iOS and Android) provide file encryption API's which use a secret key protected by the device unlock code and deletable on remote wipe. If this is available, it should be used as it increases the security of the encryption without creating extra burden on the end-user. It also makes stored data safer in the case of loss or theft. However, it should be borne in mind that even when protected by the device unlock key, if data is stored on the device, its security is dependent on the security of the device unlock code if remote deletion of the key is for any reason not possible [16].
- ✓ iOS Specific Best Practices: 3. After development, ensure all NSURL calls (or wrappers of NSURL) do not allow self-signed or invalid certificates such as the NSURL class method setAllowsAnyHTTPSCertificate [3].

**Priority:** Not defined

**Rationale:** Using certificates from trusted CAs and enforcing strict validation ensures the integrity and security of communication channels. It prevents attackers from intercepting or tampering with data through techniques like SSL stripping, certificate forgery, or man-in-the-middle (MITM) attacks.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:****1. Certificate Validation Test:**

- ✓ Test: Attempt to connect to a server with a valid certificate and verify that the connection is successful.
- ✓ Expected result: The application accepts connections only from endpoints with valid certificates signed by a trusted CA.

**2. Self-Signed Certificate Rejection Test:**

- ✓ Test: Attempt to connect to a server using a self-signed certificate.
- ✓ Expected result: The connection is rejected, and an appropriate error message is displayed.

**3. Expired Certificate Test:**

- ✓ Test: Use an expired certificate to establish a connection with the application.
- ✓ Expected result: The application rejects the connection and logs an error indicating certificate expiration.

**4. Revoked Certificate Test:**

- ✓ Test: Attempt to connect to a server with a revoked certificate.

- ✓ Expected result: The application validates certificate revocation through mechanisms like OCSP or CRL and rejects the connection.

#### 5. Hostname Mismatch Test:

- ✓ Test: Try to connect to a server with a certificate issued to a different hostname.
- ✓ Expected result: The application identifies the hostname mismatch and rejects the connection.

#### 6. Pinned Certificate Test:

- ✓ Test: Implement certificate pinning for specific endpoints and attempt to connect with an unpinned certificate.
- ✓ Expected result: The application blocks connections to endpoints using certificates that do not match the pinned certificate or key.

#### 7. Platform-Specific Test:

- ✓ Test: For iOS, ensure that NSURL methods do not allow self-signed or invalid certificates. For Android, verify that trust managers validate the full certificate chain.
- ✓ Expected result: The application enforces certificate validation consistently across platforms.

#### 8. Certificate Source Validation:

- ✓ Test: Review the application's certificate store and ensure that only trusted CA certificates are used.
- ✓ Expected result: The application does not allow certificates from untrusted or unauthorized sources.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda  
José L Fernández-Alemán  
Juan Manuel Carrillo-de-Gea  
Joaquín Nicolás  
08/01/2026

**PUID:** [SECM-CAT-ICO-017]

**Requirement description:** The application must prevent the exposure of sensitive data through Inter-Process Communication (IPC) mechanisms by ensuring strict access controls and verifying that permission-protected data is not accessible to unauthorized clients. Sensitive functionality must not be exported via IPC unless adequately protected.

**Source:**

- ✓ 2.6: Verify that no sensitive data is exposed via IPC mechanisms [19].
- ✓ Prevention of Permission Leakage: The application must not expose permission-protected data through IPC to clients that do not have the required permissions. Strict access control measures must be implemented to prevent accidental data leakage [18].
- ✓ 6.4: Verify that the app does not export sensitive functionality through IPC facilities, unless these mechanisms are properly protected [19].

<ul style="list-style-type: none"> <li>✓ MASVS-STORAGE-2: The app prevents leakage of sensitive data: Prevent the unintentional storage or exposure of sensitive data in publicly accessible locations. This control addresses potential leaks that can be avoided through proactive measures by the developer [20].</li> </ul>
<b>Priority:</b> Not defined
<b>Rationale:</b> Unprotected IPC mechanisms can expose sensitive data or functionality to unauthorized clients, leading to potential data leakage or misuse of application features. Enforcing strict access control and preventing unintended IPC exports ensures the confidentiality and security of sensitive data.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Unauthorized IPC Data Access Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Attempt to access sensitive data via IPC from a client application without the required permissions.</li> <li>✓ Expected result: Access is denied, and no sensitive data is leaked.</li> </ul> </li> <li><b>2. Permission Verification Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Verify that only clients with the correct permissions can access IPC-protected data or functionality.</li> <li>✓ Expected result: Data and functionality are accessible only to authorized clients with valid permissions.</li> </ul> </li> <li><b>3. Sensitive Functionality Export Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Review exported IPC facilities (e.g., services, activities, content providers) for sensitive functionality. Attempt to invoke these mechanisms from unauthorized clients.</li> <li>✓ Expected result: Sensitive functionality is not accessible unless explicitly authorized and protected.</li> </ul> </li> <li><b>4. IPC Data Leakage Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Monitor data exchanged through IPC channels (e.g., intents, binders) to identify any sensitive information being inadvertently exposed.</li> <li>✓ Expected result: No sensitive data is transmitted via IPC channels unless explicitly required and secured.</li> </ul> </li> <li><b>5. Penetration Testing for IPC Exploits:</b> <ul style="list-style-type: none"> <li>✓ Test: Conduct penetration tests to identify vulnerabilities in IPC mechanisms, such as intent spoofing or unauthorized access.</li> <li>✓ Expected result: All identified IPC vulnerabilities are mitigated, and no sensitive data is exposed.</li> </ul> </li> <li><b>6. Code Review for Exported Components:</b> <ul style="list-style-type: none"> <li>✓ Test: Review the application's codebase to ensure that all exported components (e.g., services, receivers) are explicitly marked and properly secured.</li> </ul> </li> </ol>

- ✓ Expected result: Only components explicitly marked for export and protected with permissions are accessible via IPC.

#### 7. Platform-Specific IPC Protection Test:

- ✓ **Test:** For Android, validate that exported components are secured with `android:exported="false"` where appropriate and that permissions are defined in the manifest.
- ✓ Expected result: Exported components are properly secured, and IPC mechanisms are only accessible with the required permissions.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda  
José L Fernández-Alemán  
Juan Manuel Carrillo-de-Gea  
Joaquín Nicolás  
08/01/2026

**PUID:** [SECM-CAT-ICO-018]

**Requirement description:** The application must secure Inter-Process Communication (IPC) mechanisms by implementing permissions using the `<permission>` element for sensitive components like `ContentProvider`. For IPC between apps by the same developer, signature protection must be used. Explicit intents must be utilized over implicit ones for asynchronous IPC to prevent data exposure.

**Source:**

- ✓ Secure Inter-Process Communication (IPC): When exposing security-sensitive IPC components like a `ContentProvider`, the app must use the `<permission>` element to protect it. Signature protection must be implemented for IPC communication between apps developed by the same developer. [18]
- ✓ MASVS-STORAGE-2: The app prevents leakage of sensitive data: Prevent the unintentional storage or exposure of sensitive data in publicly accessible locations. This control addresses potential leaks that can be avoided through proactive measures by the developer [20].
- ✓ Explicit Intents for Security: For asynchronous IPC between activities, services, and broadcast receivers, always prefer explicit intents over implicit ones for security purposes. [18]
- ✓ IPC and Data Exposure: The application must ensure that it does not inadvertently expose user data to other applications through overly permissive IPC mechanisms, world-writable files, or unsecured network sockets [18].

**Priority:** Not defined

**Rationale:** Insecure IPC mechanisms can unintentionally expose sensitive data or functionality to unauthorized applications, leading to potential data breaches. Implementing strong permissions and using explicit intents ensures that only authorized entities access sensitive IPC components, protecting user data and application integrity.

<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<p><b>1. Permission Enforcement Test:</b></p> <ul style="list-style-type: none"> <li>✓ <b>Test:</b> Verify that all security-sensitive IPC components, such as ContentProvider, require permissions defined using the &lt;permission&gt; element.</li> <li>✓ <b>Expected result:</b> Access to sensitive components is restricted to authorized applications with the correct permissions.</li> </ul>
<p><b>2. Signature Protection Test:</b></p> <ul style="list-style-type: none"> <li>✓ <b>Test:</b> Verify that IPC communication between apps from the same developer is secured using signature-level permissions.</li> <li>✓ <b>Expected result:</b> Only apps signed with the same certificate can communicate via IPC.</li> </ul>
<p><b>3. Explicit Intent Verification Test:</b></p> <ul style="list-style-type: none"> <li>✓ <b>Test:</b> Inspect all IPC mechanisms (activities, services, broadcast receivers) to confirm that explicit intents are used.</li> <li>✓ <b>Expected result:</b> All intents for IPC are explicit, specifying the target component by name.</li> </ul>
<p><b>4. IPC Data Exposure Test:</b></p> <ul style="list-style-type: none"> <li>✓ <b>Test:</b> Attempt to access sensitive user data through IPC mechanisms from unauthorized apps.</li> <li>✓ <b>Expected result:</b> Sensitive data is not exposed through IPC mechanisms to unauthorized applications.</li> </ul>
<p><b>5. World-Writable and Network Socket Review Test:</b></p> <ul style="list-style-type: none"> <li>✓ <b>Test:</b> Check for the presence of world-writable files or unsecured network sockets used by the app.</li> <li>✓ <b>Expected result:</b> No sensitive data is exposed via world-writable files or unsecured network sockets.</li> </ul>
<p><b>6. Code Review for ContentProvider Protection:</b></p> <ul style="list-style-type: none"> <li>✓ <b>Test:</b> Review the app's codebase to ensure that all ContentProvider components are protected by permissions and not left unguarded.</li> <li>✓ <b>Expected result:</b> All ContentProvider components require appropriate permissions and are secure.</li> </ul>
<p><b>7. Penetration Testing for IPC Exploits:</b></p> <ul style="list-style-type: none"> <li>✓ <b>Test:</b> Conduct penetration tests to identify potential IPC-related vulnerabilities, such as intent spoofing or unauthorized access to ContentProvider.</li> <li>✓ <b>Expected result:</b> No vulnerabilities are found, and sensitive IPC components are secure.</li> </ul>
<p><b>8. Dynamic Testing for Intent Handling:</b></p>

<ul style="list-style-type: none"> <li>✓ Test: Send both explicit and implicit intents to activities, services, or broadcast receivers and observe their behavior.</li> <li>✓ Expected result: Only explicit intents are accepted for sensitive IPC operations, and unauthorized implicit intents are rejected.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-ICO-019]
<b>Requirement description:</b> The application must ensure that all external connections to third-party entities, such as analytics providers or social networks, use SSL/TLS for secure communication when running routines through the browser or WebView. Mixed SSL sessions must be avoided to protect the user's session ID and data integrity.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ Use General Best Practices: 3. Account for outside entities like third-party analytics companies, social networks, etc. by using their SSL versions when an application runs a routine via the browser/webkit. Avoid mixed SSL sessions as they may expose the user's session ID [3].</li> </ul>
<b>Priority:</b> Not defined
<b>Rationale:</b> Mixed SSL sessions expose users to potential man-in-the-middle attacks and session hijacking, compromising user data. By enforcing SSL/TLS for all external connections, the application ensures a secure environment for data exchange with third-party services.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Third-Party SSL Enforcement Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Attempt to establish connections to third-party analytics or social networks without SSL/TLS.</li> <li>✓ Expected result: All connections must fail if SSL/TLS is not enforced.</li> </ul> </li> <li><b>2. WebView Security Validation Test:</b></li> </ol>

- ✓ Test: Inspect WebView configurations to ensure that only HTTPS URLs are allowed for third-party services.
- ✓ Expected result: WebView enforces HTTPS, and attempts to load HTTP content are blocked.

**3. Mixed Content Protection Test:**

- ✓ Test: Load a webpage containing mixed SSL content (both HTTP and HTTPS) in WebView.
- ✓ Expected result: The application blocks or warns about mixed content loading, ensuring no session IDs or data are exposed.

**4. Dynamic Network Traffic Analysis Test:**

- ✓ Test: Use a network proxy to monitor the application's outgoing traffic to third-party entities during browser/WebView interactions.
- ✓ Expected result: All traffic is encrypted using SSL/TLS, and no unencrypted requests are made.

**5. Session ID Security Test:**

- ✓ Test: Simulate session hijacking attempts by intercepting session IDs during third-party interactions.
- ✓ Expected result: Session IDs are protected through secure SSL/TLS connections, and no session data is exposed.

**6. Configuration Review Test:**

- ✓ Test: Review the application's codebase and configuration files for secure connection enforcement with third-party APIs.
- ✓ Expected result: SSL/TLS is enforced programmatically for all third-party connections.

**7. Penetration Test for Mixed Content Vulnerabilities:**

- ✓ Test: Perform penetration testing focusing on mixed content vulnerabilities in WebView and browser interactions.
- ✓ Expected result: The application is not vulnerable to mixed SSL content exploitation.

**8. End-to-End Encryption Verification Test:**

- ✓ Test: Verify that the data transmitted between the application and third-party services is encrypted end-to-end.
- ✓ Expected result: Encryption is maintained throughout the transmission, with no plaintext data visible.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda

José L Fernández-Alemán

Juan Manuel Carrillo-de-Gea

Joaquín Nicolás

08/01/2026

**PUID:** [SECM-CAT-ICO-020]

<b>Requirement description:</b> The application must ensure that session IDs are not exposed to unauthorized entities and must implement measures to protect session IDs from interception, manipulation, or unauthorized reuse.
<b>Source:</b>
✓ V-222577: The application must not expose session IDs. Configure the application to protect session IDs from interception or from manipulation [15].
<b>Priority:</b> Not defined
<b>Rationale:</b> Exposing session IDs compromises session integrity and allows attackers to impersonate users, leading to unauthorized access. By securing session IDs, the application mitigates risks of session hijacking and ensures user session confidentiality and integrity.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Session ID Transmission Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Monitor all network traffic during user authentication and session creation.</li> <li>✓ Expected result: Session IDs are transmitted only over secure channels (e.g., HTTPS) and are not visible in plaintext.</li> </ul> </li> <li><b>2. Session ID Storage Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Inspect local storage (e.g., SharedPreferences or keychain) and application memory for session ID storage practices.</li> <li>✓ Expected result: Session IDs are stored securely, using encryption or secure OS-level storage mechanisms, and are not stored in plaintext.</li> </ul> </li> <li><b>3. Session Cookie Security Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Verify that the <code>HTTPOnly</code> and <code>Secure</code> flags are set on session cookies.</li> <li>✓ Expected result: Session cookies have the <code>HTTPOnly</code> and <code>Secure</code> flags enabled to prevent client-side scripting attacks and transmission over unsecure connections.</li> </ul> </li> <li><b>4. Session ID Exposure Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Attempt to retrieve session IDs through browser caches, logs, or URL parameters.</li> <li>✓ Expected result: Session IDs are not exposed in logs, caches, or URLs.</li> </ul> </li> <li><b>5. Session ID Validation Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Test the application's behavior when a manipulated session ID is submitted.</li> <li>✓ Expected result: The application invalidates manipulated session IDs and logs the attempt.</li> </ul> </li> <li><b>6. Replay Attack Simulation Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Attempt to reuse a captured session ID on another device or network.</li> <li>✓ Expected result: The application detects and blocks reused session IDs, invalidating the session.</li> </ul> </li> <li><b>7. Idle Timeout Enforcement Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Leave a session idle beyond the timeout period and observe the behavior upon resuming.</li> </ul> </li> </ol>

<ul style="list-style-type: none"> <li>✓ Expected result: The session is invalidated, requiring re-authentication.</li> </ul>
<b>8. Session Termination Verification Test:</b>
<ul style="list-style-type: none"> <li>✓ Test: Log out of the application and monitor whether the session ID is invalidated.</li> <li>✓ Expected result: The session ID is destroyed upon logout and cannot be reused.</li> </ul>
<b>9. Session Integrity Penetration Test:</b>
<ul style="list-style-type: none"> <li>✓ Test: Conduct penetration testing to simulate session hijacking attacks.</li> <li>✓ Expected result: The application is resilient against session hijacking, and session integrity is maintained.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda
José L Fernández-Alemán
Juan Manuel Carrillo-de-Gea
Joaquín Nicolás
08/01/2026

<b>PUID:</b> [SECM-CAT-ICO-021]
<b>Requirement description:</b> The application must enforce the use of secure communication protocols, such as HTTPS, for all network communication involving sensitive data. Connections over unsecured protocols, such as HTTP, must be blocked, and HTTPS must be used by default, especially on unsecured networks like public Wi-Fi.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ Use of Secure Protocols: All network communication handling sensitive data must use secure protocols, such as HttpsURLConnection, instead of HTTP. HTTPS must be used wherever supported by the server, particularly when the mobile device connects to unsecured networks like public Wi-Fi [18].</li> <li>✓ Test ID 4: Connection to the EHR system is permitted only through specific secure protocols [37].</li> </ul>
<b>Priority:</b> Not defined
<b>Rationale:</b> Using secure protocols ensures the confidentiality and integrity of sensitive data during transmission. This mitigates risks of eavesdropping, man-in-the-middle attacks, and data tampering on unsecured or public networks.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
1. Protocol Validation Test:

<ul style="list-style-type: none"> <li>✓ Test: Attempt to establish a connection to the backend server over HTTP.</li> <li>✓ Expected result: The application rejects connections over HTTP and enforces HTTPS.</li> </ul>
<b>2. Protocol Enforcement Test:</b>
<ul style="list-style-type: none"> <li>✓ Test: Intercept and analyze traffic between the app and the backend using a proxy tool.</li> <li>✓ Expected result: All traffic uses secure HTTPS protocols, and attempts to downgrade to HTTP fail.</li> </ul>
<b>3. Configuration Review Test:</b>
<ul style="list-style-type: none"> <li>✓ <b>Test:</b> Review the application's network configuration to ensure that <code>HttpsURLConnection</code> is used for all communication.</li> <li>✓ <b>Expected result:</b> No instances of unsecured <code>HttpURLConnection</code> or non-HTTPS endpoints are present in the application code.</li> </ul>
<b>4. Server Compatibility Test:</b>
<ul style="list-style-type: none"> <li>✓ Test: Attempt to connect to the backend server using the latest secure TLS protocol versions.</li> <li>✓ Expected result: The application successfully establishes connections with servers using secure TLS versions (e.g., TLS 1.2 or later).</li> </ul>
<b>5. Downgrade Attack Simulation Test:</b>
<ul style="list-style-type: none"> <li>✓ Test: Simulate a downgrade attack by forcing the server to use older, less secure protocols.</li> <li>✓ Expected result: The application blocks the connection and logs an appropriate error.</li> </ul>
<b>6. Backend Verification Test:</b>
<ul style="list-style-type: none"> <li>✓ Test: Inspect server configurations to verify support for secure protocols and enforcement of HTTPS connections.</li> <li>✓ Expected result: The server enforces HTTPS and rejects any insecure connections.</li> </ul>
<b>7. Penetration Testing:</b>
<ul style="list-style-type: none"> <li>✓ Test: Conduct penetration tests focusing on network security to identify potential vulnerabilities in protocol usage.</li> <li>✓ Expected result: The application demonstrates resilience against protocol-related attacks, with all sensitive data securely encrypted.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-ICO-022]
<b>Requirement description:</b> The application must secure all network communication with SSL/TLS protocols to ensure the confidentiality and integrity of transmitted data, particularly for sensitive operations such as API calls, authentication, and data exchanges. Mutual authentication must be utilized when handling non-releasable or highly sensitive data.
<b>Source:</b>

- ✓ Use SSL/HTTPS: Always use HTTPS communication to encrypt your API requests [18].
- ✓ Data can be encrypted end to end using HTTPS protocols to allow only secure apps from devices to communicate with the web service layer [32].
- ✓ V-222534: Service-Oriented Applications handling non-releasable data must authenticate endpoint devices via mutual SSL/TLS. Configure the application to utilize mutual authentication when the application is processing non-releasable data [15].
- ✓ V-222396: The application must implement DoD-approved encryption to protect the confidentiality of remote access sessions. Design and configure applications to use TLS encryption to protect the confidentiality of remote access sessions [15].
- ✓ V-222397: The application must implement cryptographic mechanisms to protect the integrity of remote access sessions. Design and configure applications to use TLS encryption to protect the integrity of remote access sessions [15].
- ✓ Encrypt communication: Use HTTPS and similar technologies to ensure the data your app sends over a network is protected [18].
- ✓ SR3: Mutual Authentication and Trust A valid patient and authorized medical professional must have unique identities and must be able to authenticate each other for a trustful session. The mobile devices may have malware that can risk the health card and reader applications. The two devices must prove their trustful states to each other before the exchange of any data [30].
- ✓ All information passed through the m-health system must been done via encrypted messages [22].
- ✓ Protect API Key Communication: Secure all communication between the app and the service using HTTPS to prevent API keys from being exposed during transit [18].
- ✓ General Best Practices: 2. Apply SSL/TLS to transport channels that the mobile app will use to transmit data to a backend API or web service [3].

**Priority:** Not defined

**Rationale:** Encrypting communication using HTTPS (SSL/TLS) ensures data protection against eavesdropping and tampering during transmission. This is essential for safeguarding sensitive healthcare data and API keys, particularly when transmitted over unsecured networks.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

<p><b>1. SSL/TLS Verification Test</b></p> <ul style="list-style-type: none"> <li>✓ Test: Attempt to establish a connection using an HTTP endpoint instead of HTTPS.</li> <li>✓ Expected Result: The application rejects any non-HTTPS connection.</li> </ul> <p><b>2. Certificate Validation Test</b></p> <ul style="list-style-type: none"> <li>✓ Test: Inspect the handling of SSL/TLS certificates, including rejection of expired, self-signed, or untrusted certificates.</li> <li>✓ Expected Result: The application accepts only certificates from trusted Certificate Authorities (CAs).</li> </ul> <p><b>3. Mutual Authentication Test</b></p> <ul style="list-style-type: none"> <li>✓ Test: Establish a connection between a client and server requiring mutual authentication.</li> <li>✓ Expected Result: Both endpoints validate certificates, and the connection succeeds only with proper authentication.</li> </ul> <p><b>4. Network Traffic Inspection Test</b></p> <ul style="list-style-type: none"> <li>✓ Test: Capture network traffic between the app and server using a packet analyzer to verify encryption.</li> <li>✓ Expected Result: All transmitted data is encrypted and unintelligible.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<p><b>Author and date:</b></p> <p>Carlos M. Mejía-Granda      José L Fernández-Alemán      Juan Manuel Carrillo-de-Gea      Joaquín Nicolás      08/01/2026</p>

<b>PUID:</b> [SECM-CAT-ICO-023]
<b>Requirement description:</b> The application must implement short-lived access tokens for API calls and operations to minimize the exposure of sensitive credentials. Authentication tokens must be required for all subsequent requests, especially those involving privileged access or sensitive data modifications.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ Use Short-Lived Access Tokens: Implement short-lived tokens for operations and API calls to limit the exposure of sensitive credentials [18].</li> <li>✓ 2.7. Require authentication credentials or tokens to be passed with any subsequent request (especially those granting privileged access or modification) [16].</li> </ul>
<b>Priority:</b> Not defined
<b>Rationale:</b> Short-lived tokens reduce the risk of unauthorized access by limiting the timeframe in which stolen or intercepted credentials can be misused. Requiring tokens for each request enhances security by enforcing session validity and authentication for every action.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described

<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<p><b>1. Token Expiry Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Attempt to reuse an expired token for API calls or operations.</li> <li>✓ Expected Result: The application rejects expired tokens and requires re-authentication.</li> </ul> <p><b>2. Token Scope Verification:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Verify that tokens only provide access to operations and resources within their defined scope.</li> <li>✓ Expected Result: Tokens are restricted to their intended permissions and cannot perform unauthorized actions.</li> </ul> <p><b>3. Request Authentication Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Inspect API calls to confirm that authentication tokens are required for all requests, including privileged operations.</li> <li>✓ Expected Result: All requests are authenticated, and unauthenticated requests are denied.</li> </ul> <p><b>4. Token Transmission Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Monitor network traffic to verify that tokens are transmitted securely over encrypted channels.</li> <li>✓ Expected Result: Tokens are securely transmitted using HTTPS, with no exposure in plaintext.</li> </ul> <p><b>5. Token Revocation Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Attempt to use a revoked token for API access.</li> <li>✓ Expected Result: The application blocks requests made with revoked tokens.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-ICO-024]
<b>Requirement description:</b> The application must implement encrypted socket-level communication using HTTPS URIs or the SSLSocket class for secure file retrieval and data transmission. Additional hostname verification precautions must be applied when using SSLSocket to prevent man-in-the-middle (MITM) attacks. All network traffic must follow best practices for securing data privacy and integrity during transit.
<b>Source:</b> <ul style="list-style-type: none"> <li>✓ Secure Communications: For secure file retrieval from known network locations, use HTTPS URIs. When a secure tunnel is required, use HttpsURLConnection or SSLSocket.</li> </ul>

<p>Be aware that SSLSocket doesn't perform hostname verification, and additional precautions are necessary [18].</p> <ul style="list-style-type: none"> <li>✓ Encrypted Communication: Applications must implement encrypted socket-level communication using the SSLSocket class for secure data transmission. Secure networking is mandatory for all applications that communicate over the network to mitigate risks from unsecured wireless networks [18].</li> <li>✓ MASVS-NETWORK-1: The app secures all network traffic according to the current best practices: Ensuring data privacy and integrity of any data in transit is critical for any app that communicates over the network. This is typically done by encrypting data and authenticating the remote endpoint, as TLS does. However, there are many ways for a developer to disable the platform secure defaults, or bypass them completely by using low-level APIs or third-party libraries. This control ensures that the app is in fact setting up secure connections in any situation [20].</li> </ul>
<b>Priority:</b> Not defined
<b>Rationale:</b> Enforcing encrypted communication ensures data confidentiality and integrity while mitigating risks associated with unsecured networks, such as public Wi-Fi. Using HTTPS and SSLSocket with appropriate security measures minimizes the likelihood of MITM attacks and unauthorized data interception.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. HTTPS Connection Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Establish a connection to the server and verify that only HTTPS is used for data transmission.</li> <li>✓ Expected Result: The application rejects insecure HTTP connections and uses HTTPS exclusively.</li> </ul> </li> <li><b>2. Socket Encryption Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Verify that the SSLSocket class is used with proper configurations, including hostname verification.</li> <li>✓ Expected Result: The application enforces hostname verification and does not allow connections with unverified hosts.</li> </ul> </li> <li><b>3. MITM Attack Simulation Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Simulate a man-in-the-middle attack to intercept data during transmission.</li> <li>✓ Expected Result: The application detects and blocks the attack, preventing data leakage.</li> </ul> </li> <li><b>4. Protocol Inspection Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Analyze network traffic to confirm that all communication is encrypted using modern TLS versions.</li> </ul> </li> </ol>

<ul style="list-style-type: none"> <li>✓ Expected Result: Only TLS 1.2 or higher protocols are used, and weak protocols are rejected.</li> </ul>
<b>5. Hostname Validation Test:</b>
<ul style="list-style-type: none"> <li>✓ Test: Attempt to connect to a server with a mismatched certificate hostname.</li> <li>✓ Expected Result: The application refuses to establish a connection, ensuring the hostname is validated.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda
José L Fernández-Alemán
Juan Manuel Carrillo-de-Gea
Joaquín Nicolás
08/01/2026

<b>PUID:</b> [SECM-CAT-ICO-025]
<b>Requirement description:</b> The application must utilize only DoD-approved Public Key Infrastructure (PKI) certificate authorities (CAs) for the verification of protected session establishments. Configure the application to ensure all DoD-signed certificates are validated using DoD-approved PKI CAs exclusively.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ V-222584: The application must only allow the use of DoD-approved certificate authorities for verification of the establishment of protected sessions. Configure the application to utilize DoD-approved PKI established CAs when verifying DoD-signed certificates [15].</li> </ul>
<b>Priority:</b> Not defined
<b>Rationale:</b> Enforcing the use of DoD-approved CAs ensures the authenticity and trustworthiness of certificates, reducing the risk of man-in-the-middle (MITM) attacks and unauthorized access. This aligns with the stringent security standards required for handling sensitive or classified information in a defense context.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. CA Verification Test:</b></li> </ol>
<ul style="list-style-type: none"> <li>✓ Test: Validate that the application only accepts certificates issued by DoD-approved CAs during secure session establishment.</li> <li>✓ Expected Result: The application rejects certificates issued by non-DoD-approved CAs.</li> </ul>
<ol style="list-style-type: none"> <li><b>2. Certificate Validation Test:</b></li> </ol>

<ul style="list-style-type: none"> <li>✓ Test: Attempt to establish a connection using a certificate signed by a non-approved CA.</li> <li>✓ Expected Result: The connection is denied, and an error is logged.</li> </ul> <p><b>3. Revocation Check Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Ensure the application performs Online Certificate Status Protocol (OCSP) or Certificate Revocation List (CRL) checks for all DoD-approved certificates.</li> <li>✓ Expected Result: The application identifies and rejects revoked certificates during the verification process.</li> </ul> <p><b>4. PKI Configuration Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Review the application's PKI configuration to confirm it points only to DoD-approved CA repositories.</li> <li>✓ Expected Result: The configuration exclusively lists DoD-approved PKI CAs without exceptions.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<p><b>Author and date:</b></p> <p>Carlos M. Mejía-Granda  José L Fernández-Alemán  Juan Manuel Carrillo-de-Gea  Joaquín Nicolás  08/01/2026</p>

<b>PUID:</b> [SECM-CAT-ICO-026]
<p><b>Requirement description:</b> The application must enforce secure communication by utilizing strong, industry-standard cipher suites with appropriate key lengths and transport layer encryption protocols, such as HTTPS or TLS, for all network traffic. Implement strict certificate validation and ensure consistent use of secure channels across the app.</p>
<p><b>Source:</b></p> <ul style="list-style-type: none"> <li>✓ General Best Practices: 4. Use strong, industry standard cipher suites with appropriate key lengths [3].</li> <li>✓ 5.1: Verify that data is encrypted on the network using TLS. The secure channel is used consistently throughout the app [19].</li> <li>✓ Secure Communication Any communication or data handling performed in native code must be secured using encryption protocols (e.g., HTTPS, TLS) to prevent potential exploits or data tampering [18].</li> <li>✓ 4.2. Applications should enforce the use of an end-to-end secure channel (such as TLS) when sending sensitive information over any network (e.g., using Strict Transport Security - STS). This includes passing user credentials and other authentication equivalents [16].</li> <li>✓ MASVS-NETWORK-1: The app secures all network traffic according to the current best practices: Ensuring data privacy and integrity of any data in transit is critical for any app that communicates over the network. This is typically done by encrypting data and</li> </ul>

authenticating the remote endpoint, as TLS does. However, there are many ways for a developer to disable the platform secure defaults, or bypass them completely by using low-level APIs or third-party libraries. This control ensures that the app is in fact setting up secure connections in any situation [20].

- ✓ Employ Secure Transport Layer: 1. Use secure transport layer protocols, such as HTTPS (HTTP Secure), for transmitting encrypted data over networks. Implement proper certificate validation and ensure secure communication channels between the mobile app and backend systems [3].

**Priority:** Not defined

**Rationale:** Ensuring secure communication protects sensitive data from interception, tampering, or unauthorized access. This is especially critical for applications that transmit personal or healthcare-related information over networks, including public Wi-Fi.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

#### Validation criteria:

##### 1. Cipher Suite Compliance Test:

- ✓ Test: Verify that the application uses strong, industry-standard cipher suites with appropriate key lengths (e.g., AES-256).
- ✓ Expected Result: The application exclusively uses approved cipher suites and blocks insecure ones (e.g., RC4 or MD5).

##### 2. TLS Consistency Test:

- ✓ Test: Validate that all network communication utilizes TLS (e.g., via packet capture during data transmission).
- ✓ Expected Result: No data is transmitted over unencrypted channels, and HTTPS/TLS is enforced throughout the application.

##### 3. Certificate Validation Test:

- ✓ Test: Check that the application performs proper certificate validation, rejecting self-signed, expired, or untrusted certificates.
- ✓ Expected Result: Connections using invalid certificates are denied, and appropriate error messages are logged.

##### 4. Strict Transport Security (STS) Test:

- ✓ Test: Confirm the implementation of HTTP Strict Transport Security (HSTS) to enforce HTTPS connections.
- ✓ Expected Result: HSTS is correctly configured, and all HTTP requests are automatically upgraded to HTTPS.

##### 5. Secure Communication Logging Test:

- ✓ Test: Analyze application logs to verify that connection details do not expose sensitive information and that encryption status is logged appropriately.

<ul style="list-style-type: none"> <li>✓ Expected Result: Logs confirm the use of secure protocols without exposing sensitive data.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<p><b>PUID:</b> [SECM-CAT-ICO-027]</p> <p><b>Requirement description:</b> The application must implement FIPS 140-2-validated cryptographic modules for encryption, key exchange, digital signatures, random number generation, and hash functionality. Secure communication channels, such as IPSec or TLS, must protect all stored, collected, and exchanged data with encryption mechanisms aligned with current best practices.</p> <p><b>Source:</b></p> <ul style="list-style-type: none"> <li>✓ Security: 23. The app has encryption mechanisms for storing, collecting and exchanging information [29].</li> <li>✓ 5.2: Verify that the TLS settings are in line with current best practices, as far as they are supported by the mobile operating system [19].</li> <li>✓ 4.3.5 Transmission Security: The app has encryption mechanisms for storing, collecting, and exchanging information. It has password management mechanisms.</li> <li>✓ All communication among a typical patient, doctor, health IT administrator, and the electronic health record system is protected via end-to-end encryption by using IPSec, TLS, or similar technology. Federal agencies should verify that all components using Extensible Authentication Protocol (EAP) Transport Layer Security (TLS) are Federal Information Processing Standard (FIPS) 140-2 validated. In our implementation, because we used such a varied set of products, not all of the products were FIPS 140-2 validated [37].</li> <li>✓ MASVS-NETWORK-1: The app secures all network traffic according to the current best practices: Ensuring data privacy and integrity of any data in transit is critical for any app that communicates over the network. This is typically done by encrypting data and authenticating the remote endpoint, as TLS does. However, there are many ways for a developer to disable the platform secure defaults, or bypass them completely by using low-level APIs or third-party libraries. This control ensures that the app is in fact setting up secure connections in any situation [20].</li> <li>✓ Security: The app has encryption mechanisms for storing, collecting, and exchanging information. It has password management mechanisms [21].</li> <li>✓ 4.6. Enforce secure TLS versions. Safely abort the connection, if this is not possible [16].</li> </ul>
--

- ✓ V-222583: The application must use the Federal Information Processing Standard (FIPS) 140-2-validated cryptographic modules and random number generator if the application implements encryption, key exchange, digital signature, and hash functionality. Configure the application to use FIPS 140-2-validated cryptographic modules when the application implements encryption, key exchange, digital signatures, random number generators, and hash functionality [15].
- ✓ V-222641: The application must use encryption to implement key exchange and authenticate endpoints prior to establishing a communication channel for key exchange. Use encryption for key exchange [15].

**Priority:** Not defined

**Rationale:** The use of validated cryptographic modules and secure communication protocols ensures the confidentiality, integrity, and authenticity of sensitive information, including personal health data, in compliance with regulatory and security standards.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

**1. Cryptographic Module Validation Test:**

- ✓ Test: Verify that the cryptographic modules used in the application meet FIPS 140-2 standards for encryption and related functionalities.
- ✓ Expected Result: All cryptographic modules are FIPS 140-2 validated and correctly implemented.

**2. TLS Version Enforcement Test:**

- ✓ Test: Confirm that the application enforces the use of secure TLS versions (e.g., TLS 1.2 or higher) for all communications.
- ✓ Expected Result: Connections using older or insecure TLS versions are rejected.

**3. Encryption Key Exchange Test:**

- ✓ Test: Verify that key exchange mechanisms, such as Diffie-Hellman or RSA, use encryption to authenticate endpoints before communication channels are established.
- ✓ Expected Result: Key exchange processes securely authenticate endpoints and prevent unauthorized access.

**4. End-to-End Encryption Test:**

- ✓ Test: Inspect the application's communication flows to ensure all data exchanges between patients, healthcare providers, and backend systems are encrypted.
- ✓ Expected Result: Data exchanged across all channels is encrypted and protected from interception or tampering.

**5. Message Integrity Verification Test:**

- ✓ Test: Validate that the application uses digital signatures to maintain message integrity and prevent unauthorized modifications.

<ul style="list-style-type: none"> <li>✓ Expected Result: Messages are digitally signed, and any modifications during transit are detected and flagged.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-ICO-028]
<b>Requirement description:</b> The application must avoid binding to the non-specific IP address INADDR_ANY. Instead, it must bind only to specific and secure IP addresses to restrict unauthorized access and prevent unintended exposure to all network interfaces.
<b>Source:</b> <ul style="list-style-type: none"> <li>✓ Prohibit Use of INADDR_ANY: The application must avoid binding to the non-specific IP address INADDR_ANY, as this allows the app to receive requests from any IP address. Binding must be limited to specific and secure addresses to prevent unintended access [18].</li> </ul>
<b>Priority:</b> Not defined
<b>Rationale:</b> Binding to INADDR_ANY increases the risk of unauthorized access as it permits connections from any network interface. Restricting bindings to specific IP addresses enhances the application's security by limiting exposure and ensuring controlled network communication.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b> <ol style="list-style-type: none"> <li><b>1. Binding Address Verification Test:</b> <ul style="list-style-type: none"> <li>✓ <b>Test:</b> Analyze the application's network configuration to confirm that it does not bind to INADDR_ANY.</li> <li>✓ <b>Expected Result:</b> The application binds only to specific and secure IP addresses.</li> </ul> </li> <li><b>2. Access Control Test:</b> <ul style="list-style-type: none"> <li>✓ <b>Test:</b> Simulate unauthorized access attempts from untrusted IP addresses to ensure binding restrictions are enforced.</li> <li>✓ <b>Expected Result:</b> Connections from untrusted IP addresses are rejected.</li> </ul> </li> <li><b>3. Configuration Review Test:</b> <ul style="list-style-type: none"> <li>✓ <b>Test:</b> Inspect the application's code and configuration files for hardcoded or default use of INADDR_ANY.</li> </ul> </li> </ol>

<ul style="list-style-type: none"> <li>✓ <b>Expected Result:</b> No occurrences of <code>INADDR_ANY</code> are found in the codebase or configuration.</li> </ul> <p><b>4. Network Interface Monitoring Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Monitor the application's active network bindings during runtime to verify that only intended network interfaces are used.</li> <li>✓ Expected Result: The application binds only to the specified and secured IP addresses.</li> </ul> <p><b>5. Penetration Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Conduct a penetration test to identify potential vulnerabilities caused by improper binding practices.</li> <li>✓ Expected Result: No vulnerabilities related to insecure IP address bindings are detected.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-ICO-029]
<b>Requirement description:</b> The application must not override the <code>onReceivedSslError</code> method to bypass SSL certificate validation, ensuring that invalid SSL certificates are not accepted.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ Android Specific Best Practices: 3. Avoid overriding <code>onReceivedSslError</code> to allow invalid SSL certificates [3].</li> </ul>
<b>Priority:</b> Not defined
<b>Rationale:</b> Overriding <code>onReceivedSslError</code> to accept invalid certificates exposes the application to man-in-the-middle (MITM) attacks and compromises the security of sensitive data in transit. Strict SSL certificate validation ensures secure communication channels and protects against interception and tampering.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<p><b>1. Code Inspection Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Inspect the application's source code to verify that <code>onReceivedSslError</code> is not overridden to bypass certificate validation.</li> </ul>

<ul style="list-style-type: none"> <li>✓ Expected Result: The onReceivedSslError method is either not overridden or properly configured to reject invalid SSL certificates.</li> </ul> <p><b>2. SSL Certificate Validation Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Simulate connections using invalid or self-signed SSL certificates.</li> <li>✓ Expected Result: The application rejects connections using invalid SSL certificates.</li> </ul> <p><b>3. Network Traffic Analysis Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Monitor network traffic to ensure encrypted data is transmitted only through valid SSL connections.</li> <li>✓ Expected Result: All communication uses valid and verified SSL certificates without bypassing validation.</li> </ul> <p><b>4. Dynamic Security Testing:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Use tools to simulate MITM attacks and verify that the application does not accept invalid SSL certificates.</li> <li>✓ Expected Result: The application detects and blocks MITM attacks effectively.</li> </ul> <p><b>• Configuration Audit Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Review SSL/TLS configurations and settings to ensure best practices for certificate validation are applied.</li> <li>✓ Expected Result: The application enforces strict SSL validation and does not bypass errors.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<p><b>Author and date:</b></p> <p>Carlos M. Mejía-Granda      José L Fernández-Alemán      Juan Manuel Carrillo-de-Gea      Joaquín Nicolás      08/01/2026</p>

<p><b>PUID:</b> [SECM-CAT-ICO-030]</p> <p><b>Requirement description:</b> The application must establish secure TLS connections only after verifying the identity of the endpoint server using trusted certificates in the key chain and enforcing SSL chain verification.</p> <p><b>Source:</b></p> <ul style="list-style-type: none"> <li>✓ 4.3. For sensitive data, to reduce the risk of man-in-middle attacks (like SSL proxy, SSL strip), a secure connection should only be established after verifying the identity of the remote-end-point (server). This can be achieved by ensuring that TLS is only established with end-points having the trusted certificates in the key chain [16].</li> <li>✓ General Best Practices: 8. Always require SSL chain verification [3].</li> </ul>
--

<ul style="list-style-type: none"> <li>✓ General Best Practices: 9. Only establish a secure connection after verifying the identity of the endpoint server using trusted certificates in the key chain [3].</li> </ul>
<b>Priority:</b> Not defined
<b>Rationale:</b> Verifying the identity of the endpoint server through trusted certificates mitigates the risk of man-in-the-middle (MITM) attacks, such as SSL proxy and SSL strip, ensuring secure and trusted communication channels for sensitive data.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Certificate Validation Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Attempt to connect to servers with invalid, expired, or self-signed certificates.</li> <li>✓ Expected Result: The application rejects connections to servers without trusted certificates in the key chain.</li> </ul> </li> <li><b>2. SSL Chain Verification Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Verify that the application enforces SSL chain validation by attempting connections through a compromised certificate chain.</li> <li>✓ Expected Result: The application identifies and blocks certificate chains that are not fully trusted.</li> </ul> </li> <li><b>3. Endpoint Identity Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Simulate connections to endpoints with spoofed server certificates.</li> <li>✓ Expected Result: The application verifies the identity of the endpoint and rejects spoofed certificates.</li> </ul> </li> <li><b>4. Static Code Analysis Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Review the source code to ensure TLS initialization validates the server's certificate against the trusted key chain.</li> <li>✓ Expected Result: The code implements strict server certificate validation logic.</li> </ul> </li> <li><b>5. Dynamic Security Testing:</b> <ul style="list-style-type: none"> <li>✓ Test: Perform MITM attacks using SSL stripping and proxy tools.</li> <li>✓ Expected Result: The application detects and prevents connection establishment under MITM scenarios.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda
José L Fernández-Alemán
Juan Manuel Carrillo-de-Gea
Joaquín Nicolás
08/01/2026

<b>PUID:</b> [SECM-CAT-ICO-031]
<b>Requirement description:</b> The application must leverage platform-specific mechanisms, such as App Transport Security (ATS) in iOS and clear text traffic opt-out in Android, to enforce additional security requirements for HTTP-based networking requests.
<b>Source:</b>
<ul style="list-style-type: none"><li>✓ 4.3. For sensitive data, to reduce the risk of man-in-middle attacks (like SSL proxy, SSL strip), a secure connection should only be established after verifying the identity of the remote-end-point (server). This can be achieved by ensuring that TLS is only established with end-points having the trusted certificates in the key chain [16].</li><li>✓ General Best Practices: 8. Always require SSL chain verification [3].</li><li>✓ General Best Practices: 9. Only establish a secure connection after verifying the identity of the endpoint server using trusted certificates in the key chain [3].</li></ul>
<b>Priority:</b> Not defined
<b>Rationale:</b> Enforcing platform-specific security requirements ensures secure communication by default, reducing risks associated with clear text traffic and misconfigured networking requests, particularly for sensitive data transmissions.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"><li><b>1. Platform Security Configuration Test:</b><ul style="list-style-type: none"><li>✓ Test: Verify that ATS is enabled for iOS applications and clear text traffic is disabled in Android applications.</li><li>✓ Expected Result: iOS apps enforce ATS, and Android apps block clear text traffic unless explicitly allowed for specific trusted endpoints.</li></ul></li><li><b>2. Traffic Analysis Test:</b><ul style="list-style-type: none"><li>✓ Test: Monitor network traffic to confirm that all HTTP-based requests are encrypted and adhere to platform-specific security configurations.</li><li>✓ Expected Result: All HTTP requests use secure protocols, such as HTTPS, and no clear text traffic is detected.</li></ul></li><li><b>3. Code Review Test:</b><ul style="list-style-type: none"><li>✓ <b>Test:</b> Review the application's networking configuration files (e.g., <code>Info.plist</code> for iOS, <code>NetworkSecurityConfig</code> for Android).</li><li>✓ <b>Expected Result:</b> Security configurations explicitly enforce secure transport policies, such as ATS and clear text traffic restrictions.</li></ul></li><li><b>4. Exception Handling Test:</b></li></ol>

- ✓ Test: Attempt to connect to endpoints using clear text traffic or non-compliant configurations.
- ✓ Expected Result: The application denies such connections and logs appropriate error messages.

## 5. Dynamic Security Testing:

- ✓ Test: Use security tools to test compliance with platform-specific networking security standards.
- ✓ Expected Result: The application passes all security compliance checks for HTTP-based networking on its platform.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda  
José L Fernández-Alemán  
Juan Manuel Carrillo-de-Gea  
Joaquín Nicolás  
08/01/2026

**PUID:** [SECM-CAT-ICO-032]

**Requirement description:** The application must alert users through the user interface (UI) if the peer certificate does not match the expected certificate, providing clear warnings and the option to abort further interaction.

**Source:**

- ✓ 4.9. Design the user interface in a way that warns the user if the peer certificate does not match the expected certificate and provide the ability to abort any further interaction [16].
- ✓ General Best Practices: 10. Alert users through the UI if the mobile app detects an invalid certificate [3].

**Priority:** Not defined

**Rationale:** Informing users about invalid certificates protects against man-in-the-middle (MITM) attacks and unauthorized endpoint connections, ensuring secure communication and user trust in the application.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

### 1. Certificate Validation Test:

- ✓ Test: Simulate a connection to a server with an invalid or mismatched certificate.

<ul style="list-style-type: none"> <li>✓ Expected Result: The application displays a clear warning message to the user, indicating the certificate issue.</li> </ul> <p><b>2. Abort Interaction Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Attempt to proceed after encountering a certificate warning.</li> <li>✓ Expected Result: The application allows the user to abort further interactions with the endpoint.</li> </ul> <p><b>3. UI Design Review Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Evaluate the UI for clear and comprehensible certificate warning messages.</li> <li>✓ Expected Result: The warning message is user-friendly, detailing the certificate issue and the risks involved.</li> </ul> <p><b>4. Certificate Handling Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Connect to a trusted server with a valid certificate.</li> <li>✓ Expected Result: No warnings are displayed, and the connection proceeds seamlessly.</li> </ul> <p><b>5. Security Penetration Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Use penetration testing tools to simulate certificate manipulation and observe application behavior.</li> <li>✓ Expected Result: The application detects the issue, alerts the user, and offers an option to abort the connection</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<p><b>Author and date:</b></p> <p>Carlos M. Mejía-Granda      José L Fernández-Alemán      Juan Manuel Carrillo-de-Gea      Joaquín Nicolás      08/01/2026</p>

#### 2.9.3.6 Inadequate Privacy Controls

<b>PUID:</b> [SECM-CAT-IPC-001]
<p><b>Requirement description:</b> The application must assign and enforce the use of unique user identifiers (IDs) to track and manage user activity, ensuring each user can be uniquely identified in combination with additional attributes such as jurisdictional or organizational identifiers. These IDs must also prevent session fixation and unauthorized reuse of session identifiers.</p>
<p><b>Source:</b></p> <ul style="list-style-type: none"> <li>✓ SEC. 262. ADMINISTRATIVE SIMPLIFICATION STANDARDS FOR INFORMATION TRANSACTIONS AND DATA ELEMENTS.</li> <li>“SEC. 1173. (a) STANDARDS TO ENABLE ELECTRONIC EXCHANGE.</li> <li>“(b) UNIQUE HEALTH IDENTIFIERS. —</li> <li>“(1) IN GENERAL.—The Secretary shall adopt standards providing for a standard unique health identifier for each individual, employer, health plan, and health care provider for use in the health care system.</li> </ul>

“(2) USE OF IDENTIFIERS. —The standards adopted under paragraph (1) shall specify the purposes for which a unique health identifier may be used [38].

- ✓ 1.2. Does the app capture a unique username or “fixed device identifier” used as a user identifier (for both patient and health care provider)? [33].
- ✓ The authentication must be done with a unique ID and a password only known by the user [23].
- ✓ § 164.312 Technical safeguards.
  - (a)
    - (2) Implementation specifications:
      - (i) Unique user identification (Required).  
Assign a unique name and/or number for identifying and tracking user identity [9].

- ✓ 4.2.8.1. User Registration: Security Requirement 54 – Assigning Identifiers to Users: All organizations connecting to the EHRI must ensure that users of PoS systems that connect to the EHRI are assigned an identifier (User ID) that, in combination with other identifiers (e.g. facility identifiers, jurisdictional identifiers), can uniquely identify the user within the EHRI. PoS systems must support the unique identification of users [10].

- ✓ 4.9.3. Cryptographic Controls

Consideration MC17 – Uniquely Identify the Individual Using a Mobile Device: Establish approved mechanisms that will allow the organization to uniquely identify the user of a shared mobile device, prior to granting access to confidential information [10].

- ✓ V-222579: Applications must use system-generated session identifiers that protect against session fixation. Design the application to generate new session IDs with unique values when authenticating user sessions [15].

- ✓ IA-4 IDENTIFIER MANAGEMENT

Control: Manage system identifiers by:

- a. Receiving authorization from [Assignment: organization-defined personnel or roles] to assign an individual, group, role, service, or device identifier;
  - b. Selecting an identifier that identifies an individual, group, role, service, or device;
  - c. Assigning the identifier to the intended individual, group, role, service, or device [11];
- ✓ SESSION AUTHENTICITY | UNIQUE SYSTEM-GENERATED SESSION IDENTIFIERS

Generate a unique session identifier for each session with [Assignment: organization-defined randomness requirements] and recognize only session identifiers that are system-generated.

<p>Discussion: Generating unique session identifiers curtails the ability of adversaries to reuse previously valid session IDs. Employing the concept of randomness in the generation of unique session identifiers protects against brute-force attacks to determine future session identifiers [11].</p>
<b>Priority:</b> Not defined
<b>Rationale:</b> Assigning unique user identifiers ensures secure and precise tracking of individual user activity, compliance with regulatory requirements like HIPAA, and mitigates risks associated with session hijacking or fixation attacks. This control supports accountability and enhances user management within health systems
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Unique ID Generation Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Confirm that the application generates unique user identifiers that cannot be duplicated across the system.</li> <li>✓ Expected result: Each user ID must be globally unique and immutable once assigned.</li> </ul> </li> <li><b>2. Session ID Verification Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Verify that a new session ID is generated upon user login or session establishment.</li> <li>✓ Expected result: Session IDs are unique and adhere to randomness and unpredictability requirements.</li> </ul> </li> <li><b>3. Session Fixation Prevention Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Attempt to reuse an old or invalid session ID to authenticate.</li> <li>✓ Expected result: The system must reject reused or invalid session IDs and issue new ones upon re-authentication.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IPC-002]
---------------------------------

**Requirement description:** The application must individually authenticate users with unique credentials before granting access to group accounts or resources associated with group authenticators.

**Source:**

- ✓ V-222529: The application must ensure users are authenticated with an individual authenticator prior to using a group authenticator. Design and configure the application to individually authenticate group account members prior to allowing access [15].
- ✓ SR1: Confidentiality The health folder must be encrypted and be accessible to authorized users [30].

**Priority:** Not defined

**Rationale:** Individual authentication ensures accountability by tracking user actions within group accounts and prevents unauthorized users from gaining access through shared credentials. This practice strengthens security by enforcing identity verification prior to granting group-level privileges.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

**1. Individual Authentication Test:**

- ✓ Test: Attempt to access group resources without prior individual authentication.
- ✓ Expected result: Access is denied, and the system requires individual authentication before granting group-level permissions.

**2. Access Control Verification Test:**

- ✓ Test: Authenticate a user individually and validate that the group access permissions are correctly applied.
- ✓ Expected result: Group permissions are granted only after successful individual authentication.

**3. Multiple User Authentication Test:**

- ✓ Test: Simulate multiple users attempting to access a group account simultaneously without individual authentication.
- ✓ Expected result: All access attempts are denied until each user individually authenticates.

**4. Session Continuity Test:**

- ✓ Test: Verify that individual authentication is required for a user resuming access to group accounts after a session timeout.
- ✓ Expected result: Reauthentication is enforced before granting access to group accounts.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda  
 José L Fernández-Alemán  
 Juan Manuel Carrillo-de-Gea  
 Joaquín Nicolás  
 08/01/2026

**PUID: [SECM-CAT-IPC-003]**

**Requirement description:** The application must use non-persistent and app-specific identifiers instead of device hardware identifiers (e.g., IMEI, UDID) to ensure privacy and prevent cross-app tracking.

**Source:**

- ✓ 1.13. Use non-persistent identifiers which are not shared with other apps wherever possible (e.g., do not use the device unique hardware identifiers such as IMEI or UDID as an identifier) [16].
- ✓ IA-4 IDENTIFIER MANAGEMENT

Control: Manage system identifiers by:

- d. Preventing reuse of identifiers for [Assignment: organization-defined time period] [11].

**Priority:** Not defined

**Rationale:** Using app-specific identifiers protects user privacy by avoiding exposure of sensitive hardware-based identifiers. Non-persistent identifiers minimize the risk of tracking or misuse of unique device data, aligning with best practices for secure identifier management.

**Number of Children:** 0**Number of parents:** 0**Cycles:** 0**Number Audit:** 0**Child PUIDs:** Not described**Parent PUIDs:** Not described**Exclusion PUIDs:** Not described**Importance:** Not described**Current state:** To be determined**Verification method:** Demonstration/Analysis**Validation criteria:**

1. **Identifier Generation Test:**
  - ✓ Test: Generate an identifier upon app installation and verify its uniqueness.
  - ✓ Expected result: Identifier is unique and not based on hardware data like IMEI or UDID.
2. **Cross-App Validation Test:**
  - ✓ Test: Install two different apps and attempt to match their identifiers.
  - ✓ Expected result: Each app has a unique, non-persistent identifier that is not shared between apps.
3. **Identifier Persistence Test:**
  - ✓ Test: Uninstall and reinstall the application and check if the identifier remains the same.

<ul style="list-style-type: none"> <li>✓ Expected result: The identifier changes upon reinstallation, confirming non-persistence.</li> </ul> <p><b>4. Tracking Prevention Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Attempt to track user activity across multiple apps using the identifier.</li> <li>✓ Expected result: The identifier is unique to the app and prevents cross-app tracking.</li> </ul> <p><b>5. Audit Verification Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Audit application logs to ensure that no hardware-based identifiers (e.g., IMEI, UDID) are used or stored.</li> <li>✓ Expected result: Application logs confirm no reliance on persistent or hardware-based identifiers.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IPC-004]
<b>Requirement description:</b> The application must assess the necessity of collecting personally identifiable information (PII), such as name, address, gender, and age, and minimize the collection of sensitive data wherever possible.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ Reducing PII collection: 1. The system should allow for the need to collect all PII processed, such as name, address, gender, and age, to be assessed in order to minimize the amount of sensitive data [3].</li> </ul>
<b>Priority:</b> Not defined
<b>Rationale:</b> Minimizing the collection of PII reduces the risk of data breaches and unauthorized access to sensitive information. This aligns with privacy principles and regulations, ensuring only the necessary data is processed for intended purposes.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<p><b>1. Data Necessity Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Review the data collection process and identify the types of PII being collected.</li> <li>✓ Expected result: Only essential PII required for the application's functionality is collected.</li> </ul>

<p><b>2. Data Minimization Assessment:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Verify that optional PII fields are marked as non-mandatory in forms and user interfaces.</li> <li>✓ Expected result: Users can complete processes without providing unnecessary sensitive data.</li> </ul> <p><b>3. PII Collection Audit Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Audit the system's PII collection mechanisms and data storage logs.</li> <li>✓ Expected result: Logs confirm that only necessary PII is collected and stored.</li> </ul> <p><b>4. Privacy Impact Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Conduct a privacy impact assessment to evaluate the justification for each type of PII collected.</li> <li>✓ Expected result: Documentation demonstrates a clear purpose for every collected PII type.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<p><b>Author and date:</b></p> <p>Carlos M. Mejía-Granda  José L Fernández-Alemán  Juan Manuel Carrillo-de-Gea  Joaquín Nicolás  08/01/2026</p>

<b>PUID:</b> [SECM-CAT-IPC-005]
<b>Requirement description:</b> The application must associate and retain organization-defined security attributes with information during storage and processing. These attributes must include organization-defined security values and must persist across storage and processing activities.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ V-222393: The application must associate organization-defined types of security attributes having organization-defined security attribute values with information in storage. Design and configure the application to assign data marking and ensure the marking is retained when the data is stored [15].</li> <li>✓ V-222394: The application must associate organization-defined types of security attributes having organization-defined security attribute values with information in process. Design and configure the application to retain the data marking when processing data [15].</li> </ul>
<b>Priority:</b> Not defined
<b>Rationale:</b> Associating and retaining security attributes with data ensures that sensitive information is properly classified and protected throughout its lifecycle. This supports compliance with organizational data protection policies and prevents unauthorized access or processing.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described

<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Storage Attribute Association Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Verify that security attributes are correctly assigned to data when it is stored in the application database or file system.</li> <li>✓ Expected result: Security attributes, such as classification levels, are consistently and correctly associated with all stored data.</li> </ul> </li> <li><b>2. Attribute Persistence Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Simulate data retrieval and confirm that security attributes persist after data is stored and retrieved.</li> <li>✓ Expected result: Security attributes remain intact and unchanged after data retrieval.</li> </ul> </li> <li><b>3. Data Marking Validation Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Inspect stored and processed data to ensure that organizationally defined data markings are applied and visible.</li> <li>✓ Expected result: Data markings are applied and comply with organizational standards.</li> </ul> </li> <li><b>4. Compliance Assessment Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Evaluate the application's security attribute handling against organizational data governance policies.</li> <li>✓ Expected result: The application meets all organizational and regulatory requirements for security attribute assignment and retention.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IPC-006]
<b>Requirement description:</b> The mobile application must clear or overwrite memory blocks used to process sensitive data, such as Personally Identifiable Information (PII), location data, or authentication credentials. Additionally, sensitive personal data must not be retained beyond its required use, and temporary storage must adhere to strict access controls.
<b>Source:</b> <ul style="list-style-type: none"> <li>✓ SRG-APP-000516-MAPP-000067: The mobile app must clear or overwrite memory blocks used to process potentially sensitive data. Sensitive data may include PII, a user's location, or authentication credentials [17].</li> </ul>

- ✓ 1.7. Do not store historical location data or other sensitive information on the device beyond the period required by the application. Assume that shared storage is untrusted - information may easily leak in unexpected ways through any shared storage. In particular:
  - Be aware of caches and temporary storage as a possible leakage channel, when shared with other apps.
  - Be aware of shared storage such as address book, media gallery, audio files, as a possible leakage channel. For example, storing images with location metadata in the media-gallery allows that information to be shared in unintended ways.
  - Do not store temporary cached data in a world readable directory [16]
- ✓ 1.8. For sensitive personal data, deletion should be scheduled according to a maximum retention period, (to prevent e.g. data remaining in caches indefinitely). [16]

**Priority:** Not defined

**Rationale:** Proper management and timely deletion of sensitive data mitigate risks of unintended exposure through shared or temporary storage. Clearing memory and enforcing retention limits enhance user privacy and align with security best practices.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To\_be\_determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

**1. Memory Clearing Test:**

- ✓ Test: Simulate operations involving sensitive data (e.g., authentication, PII processing) and verify that memory blocks used during these operations are cleared immediately after use.
- ✓ Expected result: Memory is cleared or overwritten, ensuring no sensitive data remains accessible post-operation.

**2. Retention Policy Compliance Test:**

- ✓ Test: Verify that sensitive data stored on the device adheres to a defined retention period and is deleted automatically when no longer required.
- ✓ Expected result: Sensitive data is deleted in accordance with the configured retention policy.

**3. Temporary Storage Test:**

- ✓ Test: Confirm that sensitive data is not stored in temporary locations or world-readable directories (e.g., shared storage, caches).
- ✓ Expected result: Sensitive data is only stored in secure, private storage locations.

**4. Cache Management Test:**

- ✓ Test: Inspect the app's caching mechanism to ensure that cached sensitive data is cleared at appropriate intervals.
- ✓ Expected result: No sensitive data remains in the cache beyond the designated period.

<b>5. Shared Storage Test:</b>
✓ Test: Attempt to access sensitive data from shared storage directories (e.g., media gallery, address book) by unauthorized applications.
✓ Expected result: Sensitive data is not accessible through shared storage.
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IPC-007]
<b>Requirement description:</b> The system must replace critical Personally Identifiable Information (PII) with less sensitive data when possible, such as replacing detailed location information with approximate location. The application must also implement processes to securely handle, classify, retain, and dispose of data based on sensitivity.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ Replacing critical PII: 1. The system should be able to replace critical PII with less sensitive data, such as replacing detailed location with approximate location [3].</li> <li>✓ CIS Critical Security Control 3: Data Protection: Develop processes and technical controls to identify, classify, securely handle, retain, and dispose of data [14]</li> </ul>
<b>Priority:</b> Not defined
<b>Rationale:</b> Replacing detailed sensitive data with less specific alternatives reduces privacy risks while maintaining functional requirements. Proper data handling ensures compliance with data protection standards and minimizes exposure to breaches or misuse.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Data Substitution Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Replace detailed location data with approximate location within the app and observe the output.</li> <li>✓ Expected result: Detailed location data is replaced with approximate data during processing and storage.</li> </ul> </li> <li><b>2. PII Classification Test:</b></li> </ol>

<ul style="list-style-type: none"> <li>✓ Test: Assess the system's ability to classify data types based on sensitivity levels.</li> <li>✓ Expected result: Data is classified correctly, enabling appropriate handling and protection measures.</li> </ul>
<b>3. Data Retention Test:</b>
<ul style="list-style-type: none"> <li>✓ Test: Verify that data classified as sensitive is securely retained only for the defined retention period.</li> <li>✓ Expected result: Sensitive data is deleted after the retention period in compliance with policies.</li> </ul>
<b>4. Data Disposal Test:</b>
<ul style="list-style-type: none"> <li>✓ Test: Confirm that the system securely deletes or disposes of sensitive data when it is no longer required.</li> <li>✓ Expected result: Sensitive data is irretrievably deleted without leaving residual information.</li> </ul>
<b>5. Data Handling Test:</b>
<ul style="list-style-type: none"> <li>✓ Test: Review the application's processes for securely handling data during transmission, processing, and storage.</li> <li>✓ Expected result: Data is consistently handled securely and adheres to classification guidelines.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IPC-008]
<b>Requirement description:</b> The system must provide an option to reduce the frequency of Personally Identifiable Information (PII) updates, such as reducing the update interval for user location from every minute to every hour. The application must ensure data is securely identified, classified, handled, and disposed of following organizational policies.
<b>Source:</b> <ul style="list-style-type: none"> <li>✓ PII update frequency: 1. The system should provide the option to reduce the PII update frequency, such as updating the user's location every hour instead of every minute [3].</li> <li>✓ CIS Critical Security Control 3: Data Protection: Develop processes and technical controls to identify, classify, securely handle, retain, and dispose of data [14]</li> </ul>
<b>Priority:</b> Not defined
<b>Rationale:</b> Reducing the frequency of PII updates minimizes the amount of sensitive data collected, processed, and transmitted, lowering the risk of exposure in case of a breach or unauthorized access while still meeting operational requirements.

<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<p><b>1. Update Frequency Configuration Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Adjust the PII update frequency in the system's settings (e.g., change location updates to hourly).</li> <li>✓ Expected result: The system successfully reduces the update frequency without impacting core functionality.</li> </ul> <p><b>2. PII Collection Volume Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Monitor the amount of PII collected before and after reducing the update frequency.</li> <li>✓ Expected result: The volume of PII collected is reduced in line with the configured update interval.</li> </ul> <p><b>3. Data Classification Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Verify the system's ability to identify and classify PII, ensuring that reduced update frequency does not impact proper classification.</li> <li>✓ Expected result: PII is still accurately identified and classified, regardless of update frequency.</li> </ul> <p><b>4. Data Disposal Compliance Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Verify that outdated PII is securely disposed of according to the defined retention period and policies.</li> <li>✓ Expected result: No residual sensitive data remains after disposal processes are triggered.</li> </ul> <p><b>5. Privacy Risk Assessment Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Conduct a privacy impact analysis before and after implementing reduced PII update frequency.</li> <li>✓ Expected result: The system demonstrates a measurable reduction in privacy risks due to decreased PII update intervals.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IPC-009]
---------------------------------

**Requirement description:** The system must implement mechanisms to automatically delete Personally Identifiable Information (PII) after a defined retention period, such as retaining only data from the last week.

**Source:**

- ✓ Deleting PII based on retention time: 1. The system should have a mechanism to delete PII after a certain expiration period, such as retaining only data from the last week. [3].
- ✓ CIS Critical Security Control 3: Data Protection: Develop processes and technical controls to identify, classify, securely handle, retain, and dispose of data [14]

**Priority:** Not defined

**Rationale:** Automating the deletion of PII after a retention period minimizes the risk of unauthorized access to outdated or unnecessary sensitive data, aligns with data protection regulations, and reduces storage overhead.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

**1. Retention Time Configuration Test:**

- ✓ Test: Configure the system to retain PII for a specific period (e.g., one week).
- ✓ Expected result: The system allows defining and updating the retention period without requiring code changes.

**2. Automatic Deletion Test:**

- ✓ Test: Verify that PII older than the defined retention period is automatically deleted or archived.
- ✓ Expected result: Data exceeding the retention period is securely deleted and no longer accessible.

**3. Data Integrity Test:**

- ✓ Test: Confirm that data within the retention period remains intact and accessible.
- ✓ Expected result: PII within the retention period is available and unaffected by the deletion process.

**4. Audit Log Verification Test:**

- ✓ Test: Review system logs to ensure deletion events are recorded, including timestamps and data types removed.
- ✓ Expected result: All deletion actions are logged for audit and compliance purposes.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda

José L Fernández-Alemán

Juan Manuel Carrillo-de-Gea  
Joaquín Nicolás  
08/01/2026

**PUID:** [SECM-CAT-IPC-010]

**Requirement description:** The system must allow users to manage access to their personal information at any time, including granting or revoking access to specific data.

**Source:**

- ✓ The users should be able to allow or forbid access to their information at any moment [23].

**Priority:** Not defined

**Rationale:** Providing users with control over their data access promotes user autonomy, ensures compliance with data protection regulations, and enhances trust in the application by allowing dynamic consent management.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

**1. Access Management Interface Test:**

- ✓ Test: Verify that the application provides a user-friendly interface for managing data access permissions.
- ✓ Expected result: Users can view, modify, grant, or revoke access to specific personal information at any time.

**2. Real-Time Update Test:**

- ✓ Test: Ensure that changes to access permissions are updated in real time across the system.
- ✓ Expected result: Access revocation or granting is applied immediately without delays.

**3. Permission Audit Test:**

- ✓ Test: Verify that the system maintains an accurate record of granted or revoked permissions for audit purposes.
- ✓ Expected result: A clear log of permission changes is stored securely and is accessible to authorized users.

**4. Revocation Confirmation Test:**

- ✓ Test: Confirm that users receive confirmation notifications after revoking access to their data.
- ✓ Expected result: The system provides confirmation messages that access has been revoked.

**5. Data Accessibility Test:**

- ✓ Test: Ensure that revoked data access is no longer available to previously authorized entities.

✓ Expected result: Once access is revoked, no unauthorized entity can retrieve or use the data.
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IPC-011]
<b>Requirement description:</b> The system must retain a complete history of PHI (Protected Health Information), including its former content, and track details of who accessed, modified, or entered the data, along with timestamps. This functionality must be available in both EHRI systems and PoS systems connected to EHRI.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ 4.7. Communications and Operations Management</li> </ul> <p>Security Requirement 38 – Preserving the History of PHI in the EHRI: The EHRI must be capable of displaying the former content of a record at any point in the past, as well the associated details of who entered, accessed or modified the data, and at what time</p> <p>Security Requirement 39 – Preserving the History of PHI in PoS Systems: All PoS systems connected to the EHRI should be capable of displaying the former content of a record at any point in the past, as well as the associated details of who entered, accessed or modified the data, and at what time [10].</p> <ul style="list-style-type: none"> <li>✓ SR13: Audit Logs All events of reading and writing to the health folder must be recorded in the cloud along with a backup of the transaction for reference in the case of improper access [30].</li> </ul>
<b>Priority:</b> Not defined
<b>Rationale:</b> Preserving a historical record of PHI ensures compliance with legal and regulatory standards, enhances data traceability, and provides accountability for modifications or access to sensitive health data.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>

**1. Historical Data Retrieval Test:**

- ✓ Test: Verify that the system allows retrieval of PHI content at any specified point in the past.
- ✓ Expected result: The system displays accurate historical data along with its associated metadata.

**2. Access Log Verification Test:**

- ✓ Test: Check that the system logs every instance of data access, including user identity, timestamp, and action performed.
- ✓ Expected result: Logs accurately detail access and modification activities for PHI records.

**3. Modification Traceability Test:**

- ✓ Test: Confirm that the system provides a clear audit trail of modifications, showing previous and updated values for PHI.
- ✓ Expected result: All changes to PHI are fully traceable, with detailed records of who made the changes and when.

**4. PoS System Synchronization Test:**

- ✓ Test: Verify that connected PoS systems can display the same historical data and logs as the EHRi.
- ✓ Expected result: PoS systems synchronize correctly with EHRi and display consistent data and audit logs.

**5. Data Integrity Test:**

- ✓ Test: Ensure that historical PHI content and associated logs cannot be altered or deleted by unauthorized users.
- ✓ Expected result: The system enforces robust data integrity measures, preventing tampering or unauthorized changes.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda  
 José L Fernández-Alemán  
 Juan Manuel Carrillo-de-Gea  
 Joaquín Nicolás  
 08/01/2026

**PUID:** [SECM-CAT-IPC-012]

**Requirement description:** The system must implement procedures to collect, review, and retain audit logs of all significant events, including access reports and security incident tracking, ensuring compliance with regulatory requirements and supporting investigation and recovery processes.

**Source:**

- ✓ (a) A covered entity or business associate must, in accordance with § 164.306:
  - (1) Ensure the confidentiality, integrity, and availability of all electronic protected health information
  - (ii) Implementation specifications:
  - (D) Information system activity review

<p>Implement procedures to regularly review records of information system activity, such as audit logs, access reports, and security incident tracking reports [9].</p> <ul style="list-style-type: none"> <li>✓ CIS Critical Security Control 8: Audit Log Management: Collect, alert, review, and retain audit logs of events that could help detect, understand, or recover from an attack [14].</li> <li>✓ All health care providers' data access activities are logged and can be audited [26].</li> <li>✓ 5.28 Collection of evidence: The organization should establish and implement procedures for the identification, collection, acquisition and preservation of evidence related to information security events (LOGS, auditing) [6].</li> </ul>
<b>Priority:</b> Not defined
<b>Rationale:</b> Regular review and retention of audit logs provide critical evidence for detecting, understanding, and recovering from security incidents. This practice ensures the confidentiality, integrity, and availability of sensitive health information and helps maintain regulatory compliance.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<p><b>1. Audit Log Collection Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Verify that the system collects audit logs for all significant events, including user access, data modifications, and security incidents.</li> <li>✓ Expected result: Logs are consistently generated and stored for all relevant activities.</li> </ul> <p><b>2. Log Review Functionality Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Ensure that administrators can access and review audit logs through the system interface.</li> <li>✓ Expected result: The system provides an easy-to-use interface for reviewing detailed audit logs.</li> </ul> <p><b>3. Alert System Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Test the system's ability to generate alerts for suspicious or abnormal activities detected in audit logs.</li> <li>✓ Expected result: The system triggers alerts for predefined anomalies and logs these events.</li> </ul> <p><b>4. Retention Policy Enforcement Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Confirm that the system retains audit logs for the duration defined in the organizational retention policy.</li> <li>✓ Expected result: Logs are securely stored and retrievable for the required retention period.</li> </ul> <p><b>5. Evidence Preservation Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Verify that the system preserves logs in an unalterable format for use in investigations or legal proceedings.</li> </ul>

<ul style="list-style-type: none"> <li>✓ Expected result: Audit logs are immutable and accessible for evidence collection during security event investigations.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda
José L Fernández-Alemán
Juan Manuel Carrillo-de-Gea
Joaquín Nicolás
08/01/2026

<b>PUID:</b> [SECM-CAT-IPC-013]
<b>Requirement description:</b> The application must implement logging mechanisms to audit the execution of privileged functions. Each log entry must include, at a minimum, the specific action taken, the date, and the time of the event.
<b>Source:</b> <ul style="list-style-type: none"> <li>✓ V-222431: The application must audit the execution of privileged functions. Configure the application to write log entries when privileged functions are executed. At a minimum, ensure the specific action taken, date and time of event are recorded [15].</li> <li>✓ Test ID 7: Application and system components contain a mechanism to allow the auditing of privileged functions [37].</li> </ul>
<b>Priority:</b> Not defined
<b>Rationale:</b> Auditing privileged functions ensures accountability and facilitates the detection of unauthorized or inappropriate use of elevated privileges. This supports compliance with security policies and enables effective monitoring of system activity.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b> <ol style="list-style-type: none"> <li><b>1. Privileged Function Audit Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Execute a series of privileged functions and verify that corresponding log entries are generated.</li> <li>✓ Expected result: Each privileged action is logged with the specific action, date, and time recorded accurately.</li> </ul> </li> <li><b>2. Log Content Verification Test:</b></li> </ol>

<ul style="list-style-type: none"> <li>✓ Test: Review the log entries to confirm they contain sufficient details, including the identity of the user executing the function.</li> <li>✓ Expected result: Log entries include user identity, action performed, date, and time.</li> </ul>
<b>3. Log Accessibility Test:</b>
<ul style="list-style-type: none"> <li>✓ Test: Ensure that authorized administrators can access and review logs of privileged function execution.</li> <li>✓ Expected result: Logs are accessible through a secure interface and only to authorized personnel.</li> </ul>
<b>4. Unauthorized Access Attempt Test:</b>
<ul style="list-style-type: none"> <li>✓ Test: Attempt to execute privileged functions without the necessary permissions and verify the logging of such attempts.</li> <li>✓ Expected result: Unauthorized attempts to execute privileged functions are logged with sufficient details for investigation.</li> </ul>
<b>5. Retention Policy Test:</b>
<ul style="list-style-type: none"> <li>✓ Test: Verify that logs of privileged function execution are retained according to the organizational data retention policy.</li> <li>✓ Expected result: Logs are securely stored and retrievable for the defined retention period.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IPC-014]
<b>Requirement description:</b> The mobile application must ensure that error messages are only transmitted to authorized destinations, including audit logs, the Mobile Device Management (MDM) system, or the device display. Error messages must not be sent to unauthorized entities.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ SRG-APP-000267-MAPP-000060: The mobile app must not transmit error messages to any entity other than authorized audit logs, the MDM, or the device display [17].</li> <li>✓ SR2: Integrity An intruder must not be able to alter the health information [30].</li> </ul>
<b>Priority:</b> Not defined
<b>Rationale:</b> Restricting the transmission of error messages to authorized entities reduces the risk of exposing sensitive application details that could be exploited by attackers. This helps maintain the confidentiality and integrity of the application's operational data.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described

<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<b>1. Authorized Destination Test:</b> <ul style="list-style-type: none"><li>✓ Test: Simulate application errors and monitor where error messages are transmitted.</li><li>✓ Expected result: Error messages are logged in audit logs, sent to the MDM system, or displayed on the device. No unauthorized destinations receive error messages.</li></ul>
<b>2. Unauthorized Transmission Test:</b> <ul style="list-style-type: none"><li>✓ Test: Attempt to intercept error messages by introducing unauthorized monitoring tools.</li><li>✓ Expected result: No error messages are transmitted to unauthorized entities.</li></ul>
<b>3. Error Message Content Verification Test:</b> <ul style="list-style-type: none"><li>✓ Test: Review the content of error messages sent to audit logs, MDM, and the device display.</li><li>✓ Expected result: Messages contain sufficient information for troubleshooting without revealing sensitive internal details.</li></ul>
<b>4. Log Audit Test:</b> <ul style="list-style-type: none"><li>✓ Test: Verify that error messages transmitted to audit logs comply with logging standards and organizational policies.</li><li>✓ Expected result: Error messages in logs are appropriately categorized and retained securely.</li></ul>
<b>5. MDM Integration Test:</b> <ul style="list-style-type: none"><li>✓ Test: Ensure that error messages sent to the MDM system are appropriately handled and stored according to organizational policies.</li><li>✓ Expected result: Error messages are securely transmitted to the MDM and stored as per retention requirements.</li></ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IPC-015]
<b>Requirement description:</b> The application must ensure that server logs adequately capture information about established connections, including parsing HTTP headers like X-Forwarded-For in cases involving multiple intermediate proxies.
<b>Source:</b>

<ul style="list-style-type: none"> <li>✓ 4.12. Ensure adequate logs on the server are retained about established connections. In the case of multiple intermediate proxies, make sure that HTTP headers are parsed correctly (e.g., X-Forwarded-For) [16].</li> </ul>
<b>Priority:</b> Not defined
<b>Rationale:</b> Comprehensive logging of connection details, including correct interpretation of HTTP headers in proxied environments, enables accurate tracing of client requests and enhances the organization's ability to investigate and respond to security incidents.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Server Log Retention Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Verify that the server logs capture all relevant information about established connections, such as IP addresses and timestamps.</li> <li>✓ Expected result: Server logs include complete connection details for all client interactions.</li> </ul> </li> <li><b>2. Proxy Header Parsing Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Simulate a connection routed through multiple proxies and examine whether the X-Forwarded-For header is correctly parsed.</li> <li>✓ Expected result: The application accurately logs the original client IP address and intermediary proxies' details.</li> </ul> </li> <li><b>3. Log Completeness Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Review logs to ensure they contain sufficient data to trace requests back to the originating client.</li> <li>✓ Expected result: Logs include client IP, proxy chain, connection timestamps, and session identifiers, if applicable.</li> </ul> </li> <li><b>4. Log Retention Policy Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Ensure logs are retained in accordance with organizational retention policies.</li> <li>✓ Expected result: Logs are securely stored and available for the defined retention period.</li> </ul> </li> <li><b>5. Log Security Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Verify that server logs are protected against unauthorized access or tampering.</li> <li>✓ Expected result: Logs are securely stored with access controls and integrity protection mechanisms in place.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda
José L Fernández-Alemán
Juan Manuel Carrillo-de-Gea

Joaquín Nicolás 08/01/2026
-------------------------------

<b>PUID:</b> [SECM-CAT-IPC-016]
<b>Requirement description:</b> The mobile health application must retain adequate logs on the back-end to support incident detection, response, and forensic analysis, ensuring compliance with applicable data protection laws and regulations. These logs should record security-relevant events while maintaining the privacy of sensitive health data.
<b>Source:</b>
✓ 5.5. Ensure adequate logs are retained on the back-end in order to detect and respond to incidents and perform forensics (within the limits of data protection law) [16].
<b>Priority:</b> Not defined
<b>Rationale:</b> Retaining adequate logs enables timely detection of security incidents and supports forensic analysis to identify potential breaches or misuse. This is critical in mobile health applications, where protecting sensitive personal health information (PHI) is paramount.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Log Retention Verification Test</b> <ul style="list-style-type: none"> <li>✓ Test: Confirm that the application backend retains logs of all security-relevant events, such as logins, access attempts, and data modifications.</li> <li>✓ Expected Result: Logs must include timestamps, user identifiers, and actions performed without storing sensitive data like plaintext PHI.</li> </ul> </li> <li><b>2. Incident Detection Test</b> <ul style="list-style-type: none"> <li>✓ Test: Simulate a security incident and verify that the application logs capture sufficient details to detect and respond to the incident.</li> <li>✓ Expected Result: Logs should provide a clear timeline and relevant information for incident response.</li> </ul> </li> <li><b>3. Forensic Capability Test</b> <ul style="list-style-type: none"> <li>✓ Test: Analyze the stored logs to confirm they support forensic investigations while respecting privacy constraints.</li> <li>✓ Expected Result: Logs should enable accurate reconstruction of security events without exposing sensitive health data.</li> </ul> </li> <li><b>4. Access Control Test</b> <ul style="list-style-type: none"> <li>✓ Test: Verify that access to logs is restricted to authorized personnel only.</li> <li>✓ Expected Result: Only authorized administrators can access or modify log files, and access attempts are logged.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer

<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda
José L Fernández-Alemán
Juan Manuel Carrillo-de-Gea
Joaquín Nicolás
08/01/2026

<b>PUID:</b> [SECM-CAT-IPC-017]
<b>Requirement description:</b> Mobile health applications must audit and log all non-local maintenance and diagnostic sessions, capturing organization-defined auditable events. Additionally, cryptographic mechanisms must be implemented to protect the integrity of communications during such sessions, ensuring that remote maintenance is secure and traceable.
<b>Source:</b>
<ul style="list-style-type: none"><li>✓ V-222561: Applications used for non-local maintenance sessions must audit non-local maintenance and diagnostic sessions for organization-defined auditable events. Configure the application to log when application maintenance functionality is executed remotely [15].</li><li>✓ V-222562: Applications used for non-local maintenance sessions must implement cryptographic mechanisms to protect the integrity of non-local maintenance and diagnostic communications. Configure the application to encrypt remote application maintenance sessions [15].</li></ul>

<b>Priority:</b> Not defined
<b>Rationale:</b> Non-local maintenance sessions are potential vectors for unauthorized access and data breaches. Proper auditing ensures accountability, while encryption safeguards the integrity of sensitive communication, reducing the risk of malicious interception or alteration during remote sessions.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis

<b>Validation criteria:</b>
<b>1. Audit Log Verification Test</b>
<ul style="list-style-type: none"><li>✓ Test: Verify that the application logs all remote maintenance sessions, including event details such as timestamps, user identifiers, and actions performed.</li><li>✓ Expected Result: Logs should accurately capture all organization-defined auditable events for non-local maintenance sessions.</li></ul>
<b>2. Cryptographic Integrity Test</b>

- ✓ Test: Establish a remote maintenance session and verify that all communications are encrypted using approved cryptographic protocols (e.g., TLS).
- ✓ Expected Result: Communication integrity is protected, and data cannot be intercepted or altered during transmission.

### 3. Configuration Review Test

- ✓ Test: Inspect the application configuration to ensure that encryption mechanisms are enabled for remote maintenance sessions.
- ✓ Expected Result: The application must enforce encryption and reject connections that do not meet security standards.

### 4. Unauthorized Access Test

- ✓ Test: Attempt to initiate a non-local maintenance session without proper credentials or encryption.
- ✓ Expected Result: The application should deny access and log the unauthorized attempt.

### 5. Audit Review Capability Test

- ✓ Test: Verify that audit logs for remote maintenance sessions can be reviewed by authorized personnel and are stored securely.
- ✓ Expected Result: Logs are accessible only to authorized users, tamper-proof, and retained for a predefined period as per policy.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda

José L Fernández-Alemán

Juan Manuel Carrillo-de-Gea

Joaquín Nicolás

08/01/2026

**PUID:** [SECM-CAT-IPC-018]

**Requirement description:** Mobile health applications must generate audit records for the creation of session IDs, ensuring that each session initiation event is logged with relevant details such as timestamp, session identifier, and associated user account.

**Source:**

- ✓ V-222441: The application must provide audit record generation capability for the creation of session IDs. Enable session ID creation event auditing [15].

**Priority:** Not defined

**Rationale:** Auditing session ID creation enhances security by enabling the detection of unauthorized or suspicious session initiation activities. This is critical for protecting sensitive health data and ensuring accountability in user access.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<p><b>1. Session ID Audit Logging Test</b></p> <ul style="list-style-type: none"> <li>✓ Test: Initiate a user session and verify that the creation of the session ID is logged in the audit records.</li> <li>✓ Expected Result: The log entry includes a unique session ID, timestamp, and the associated user account information.</li> </ul> <p><b>2. Timestamp Accuracy Test</b></p> <ul style="list-style-type: none"> <li>✓ Test: Confirm that the timestamp for the session ID creation in the log matches the actual session initiation time.</li> <li>✓ Expected Result: The timestamp is accurate to within a defined threshold of system clock synchronization.</li> </ul> <p><b>3. Log Security Test</b></p> <ul style="list-style-type: none"> <li>✓ Test: Ensure that the audit logs containing session ID creation events are protected from unauthorized access or modification.</li> <li>✓ Expected Result: Audit logs are stored securely and accessible only to authorized personnel.</li> </ul> <p><b>4. Audit Review Test</b></p> <ul style="list-style-type: none"> <li>✓ Test: Validate that the application provides an interface or mechanism to review session ID creation events.</li> <li>✓ Expected Result: Authorized personnel can retrieve and analyze audit records related to session ID creation.</li> </ul> <p><b>5. Unauthorized Session Test</b></p> <ul style="list-style-type: none"> <li>✓ Test: Attempt to create a session ID using unauthorized methods or accounts and verify the logging behavior.</li> <li>✓ Expected Result: The unauthorized attempt is logged with appropriate details, and no valid session ID is created.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IPC-019]
<b>Requirement description:</b> Mobile health applications must generate audit records for the destruction of session IDs, capturing relevant details such as the session ID, timestamp, and associated user account to enhance traceability and security.
<b>Source:</b>

✓ V-222442: The application must provide audit record generation capability for the destruction of session IDs. Enable session ID destruction event auditing [15].
<b>Priority:</b> Not defined
<b>Rationale:</b> Auditing the destruction of session IDs ensures traceability and accountability, allowing administrators to detect abnormal termination of sessions that could indicate potential security issues. This is vital for protecting sensitive health information and maintaining compliance with data security standards.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Session ID Destruction Logging Test</b> <ul style="list-style-type: none"> <li>✓ Test: Terminate a user session and verify that the session ID destruction event is logged in the audit records.</li> <li>✓ Expected Result: The log entry includes the destroyed session ID, timestamp, and associated user account information.</li> </ul> </li> <li><b>2. Timestamp Accuracy Test</b> <ul style="list-style-type: none"> <li>✓ Test: Confirm that the timestamp for the session ID destruction in the log matches the actual session termination time.</li> <li>✓ Expected Result: The timestamp is accurate to within a defined threshold of system clock synchronization.</li> </ul> </li> <li><b>3. Log Integrity Test</b> <ul style="list-style-type: none"> <li>✓ Test: Ensure that audit logs for session ID destruction are protected from unauthorized access or tampering.</li> <li>✓ Expected Result: The logs are securely stored and accessible only to authorized personnel.</li> </ul> </li> <li><b>4. Audit Retrieval Test</b> <ul style="list-style-type: none"> <li>✓ Test: Validate that authorized users can retrieve and review audit logs for session ID destruction.</li> <li>✓ Expected Result: The system provides detailed and retrievable logs for session ID destruction events.</li> </ul> </li> <li><b>5. Unauthorized Session Destruction Test</b> <ul style="list-style-type: none"> <li>✓ Test: Attempt to destroy a session ID using unauthorized methods and verify the logging behavior.</li> <li>✓ Expected Result: The unauthorized attempt is logged with appropriate details, and no valid session ID destruction occurs.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>

Carlos M. Mejía-Granda  
 José L Fernández-Alemán  
 Juan Manuel Carrillo-de-Gea  
 Joaquín Nicolás  
 08/01/2026

**PUID:** [SECM-CAT-IPC-020]

**Requirement description:** Mobile health applications must generate audit records for the renewal of session IDs, capturing details such as session ID, timestamp, user account, and changes in privileges or permissions, to ensure transparency and traceability.

**Source:**

- ✓ V-222443: The application must provide audit record generation capability for the renewal of session IDs. Design or reconfigure the application to log session renewal events on those application events that provide changes in the users privileges or permissions to the application [15].

**Priority:** Not defined

**Rationale:** Logging session ID renewals provides a clear audit trail of user privilege or permission changes, which is critical for detecting unauthorized access or privilege escalation attempts. This supports compliance with health data security standards and enhances accountability.

**Number of Children:** 0**Number of parents:** 0**Cycles:** 0**Number Audit:** 0**Child PUIDs:** Not described**Parent PUIDs:** Not described**Exclusion PUIDs:** Not described**Importance:** Not described**Current state:** To be determined**Verification method:** Demonstration/Analysis**Validation criteria:**

- 1. Session ID Renewal Logging Test**
  - ✓ Test: Trigger a session ID renewal due to a change in user privileges or permissions and verify that the event is logged.
  - ✓ Expected Result: The audit log captures the session ID, timestamp, associated user account, and the nature of the privilege or permission change.
- 2. Timestamp Accuracy Test**
  - ✓ Test: Confirm the timestamp of the session ID renewal log matches the actual event timing.
  - ✓ Expected Result: The timestamp is accurate and synchronized with the system clock.
- 3. Privilege Change Detail Test**
  - ✓ Test: Verify that the log includes detailed information about the changes in privileges or permissions during the session ID renewal.
  - ✓ Expected Result: The log entry specifies the old and new privileges or permissions.
- 4. Log Integrity Test**
  - ✓ Test: Ensure that session ID renewal logs are stored securely and protected from unauthorized access or modification.

- ✓ Expected Result: Logs are encrypted and access-controlled to authorized personnel only.

## 5. Audit Retrieval Test

- ✓ Test: Validate that authorized personnel can retrieve and review session ID renewal logs, including filtering by user account or event type.
- ✓ Expected Result: The system provides accessible and detailed logs for session ID renewal events.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda  
José L Fernández-Alemán  
Juan Manuel Carrillo-de-Gea  
Joaquín Nicolás  
08/01/2026

**PUID:** [SECM-CAT-IPC-021]

**Requirement description:** Mobile health applications must enable audit record generation for HTTP headers, including User-Agent, Referer, GET, and POST, capturing relevant data to support forensic analysis and incident detection.

**Source:**

- ✓ V-222447: The application must provide audit record generation capability for HTTP headers including User-Agent, Referer, GET, and POST. Configure the web application and/or the web server to log HTTP headers [15].

**Priority:** Not defined

**Rationale:** Auditing HTTP headers provides valuable insights into application usage patterns, potential misuse, or attack vectors. This ensures compliance with security standards for healthcare applications and aids in identifying malicious activities.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

### 1. HTTP Header Logging Test

- ✓ **Test:** Send HTTP requests with User-Agent, Referer, GET, and POST headers to the application and verify that the headers are logged.
- ✓ Expected Result: Logs capture all specified headers along with their associated values.

### 2. Timestamp and Source Accuracy Test

- ✓ **Test:** Confirm that each HTTP header log entry includes accurate timestamps and the IP address of the request source.

<ul style="list-style-type: none"> <li>✓ Expected Result: Log entries contain precise timestamps and source IP information.</li> </ul> <p><b>3. Sensitive Data Masking Test</b></p> <ul style="list-style-type: none"> <li>✓ Test: Verify that sensitive data in HTTP headers, such as authentication tokens or personal identifiers, are appropriately masked or obfuscated in the logs.</li> <li>✓ Expected Result: Sensitive information is not stored in plain text in logs.</li> </ul> <p><b>4. Log Retention and Accessibility Test</b></p> <ul style="list-style-type: none"> <li>✓ Test: Ensure HTTP header logs are stored securely and are accessible only to authorized personnel for the defined retention period.</li> <li>✓ Expected Result: Logs are encrypted, access-controlled, and meet organizational retention policies.</li> </ul> <p><b>5. Log Search and Filtering Test</b></p> <ul style="list-style-type: none"> <li>✓ Test: Validate that administrators can search and filter HTTP header logs by criteria such as date, IP address, or specific header types.</li> <li>✓ Expected Result: Logs are searchable and provide relevant results efficiently.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IPC-022]
<b>Requirement description:</b> Mobile health applications must generate audit records for session timeout events, capturing details such as user ID, session duration, and timestamp to ensure traceability and compliance with security standards.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ V-222445: The application must provide audit record generation capability for session timeouts. Configure the application to record session timeout events in the logs [15].</li> </ul>
<b>Priority:</b> Not defined
<b>Rationale:</b> Auditing session timeouts helps identify potential misuse, such as unauthorized access due to prolonged inactive sessions. This is crucial for detecting anomalies and maintaining secure user access to sensitive healthcare data.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis

**Validation criteria:**

- 1. Session Timeout Logging Test**
  - ✓ Test: Initiate a user session, allow it to time out due to inactivity, and verify the audit log entry.
  - ✓ Expected Result: The log records the session timeout event, including the user ID, session start and end times, and timeout reason.
- 2. Timestamp Accuracy Test**
  - ✓ Test: Compare the recorded timestamp of the session timeout event with the system time.
  - ✓ Expected Result: The timestamp in the log matches the actual timeout event time.
- 3. Session Data Completeness Test**
  - ✓ Test: Validate that log entries include relevant session details, such as user ID, session duration, and the IP address of the device used.
  - ✓ Expected Result: All session-related details are present in the audit logs.
- 4. Log Access and Retention Test**
  - ✓ Test: Verify that only authorized personnel can access session timeout logs and that they are retained according to organizational policies.
  - ✓ Expected Result: Logs are encrypted, access-controlled, and retained securely as per retention policy.
- 5. Session Anomaly Detection Test**
  - ✓ Test: Analyze session timeout logs to confirm the system flags anomalies, such as repeated session timeouts from a single user or unusual session durations.
  - ✓ Expected Result: Anomalous patterns are identified and logged for further review.

**Requested by:** The organization**Responsible:** Developer**Configurable value:** Not described**Version history:** v1.0**Author and date:**

Carlos M. Mejía-Granda  
 José L Fernández-Alemán  
 Juan Manuel Carrillo-de-Gea  
 Joaquín Nicolás  
 08/01/2026

**PUID:** [SECM-CAT-IPC-023]

**Requirement description:** Mobile health applications must retain audit trails for at least one year for non-SAMI data and five years for applications processing SAMI (Sensitive Application Management Information) data to ensure traceability, compliance, and forensic analysis capabilities.

**Source:**

- ✓ V-222621: The ISSO must ensure application audit trails are retained for at least 1 year for applications without SAMI data, and 5 years for applications including SAMI data. Retain application audit log files for one year and five years for SAMI data [15].

**Priority:** Not defined

<b>Rationale:</b> Retention of audit trails is critical for ensuring accountability, supporting incident investigations, and meeting regulatory compliance requirements specific to healthcare data management.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Retention Policy Test</b> <ul style="list-style-type: none"> <li>✓ Test: Verify the system retains audit logs for at least one year for non-SAMI data and five years for SAMI data.</li> <li>✓ Expected Result: Audit logs comply with the retention periods defined by the application type.</li> </ul> </li> <li><b>2. Log Accessibility Test</b> <ul style="list-style-type: none"> <li>✓ Test: Verify that authorized personnel can access audit logs during the retention period.</li> <li>✓ Expected Result: Logs are accessible within the specified retention timeframe while maintaining appropriate access controls.</li> </ul> </li> <li><b>3. Storage Encryption Test</b> <ul style="list-style-type: none"> <li>✓ Test: Verify that retained audit logs are encrypted during storage.</li> <li>✓ Expected Result: Logs are securely stored using encryption standards compliant with healthcare regulations.</li> </ul> </li> <li><b>4. Retention Expiry Test</b> <ul style="list-style-type: none"> <li>✓ Test: Verify that logs exceeding the defined retention period are securely deleted.</li> <li>✓ Expected Result: Logs are purged automatically or manually according to policy, ensuring no data remains after the retention period.</li> </ul> </li> <li><b>5. Retention Policy Configuration Test</b> <ul style="list-style-type: none"> <li>✓ Test: Validate that the application's retention policies are configurable to meet regulatory and organizational requirements.</li> <li>✓ Expected Result: Retention periods can be adjusted, and changes are logged for audit purposes.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda
José L Fernández-Alemán
Juan Manuel Carrillo-de-Gea
Joaquín Nicolás
08/01/2026

<b>PUID:</b> [SECM-CAT-IPC-024]
<b>Requirement description:</b> Mobile health applications must ensure that no sensitive data, including personally identifiable information (PII) or sensitive health information, is written to application logs. Logs must only capture essential data necessary for troubleshooting while maintaining confidentiality and security.
<b>Source:</b> <ul style="list-style-type: none"><li>✓ 2.2: Verify that no sensitive data is written to application logs [19].</li><li>✓ MASVS-STORAGE-2: The app prevents leakage of sensitive data: Prevent the unintentional storage or exposure of sensitive data in publicly accessible locations. This control addresses potential leaks that can be avoided through proactive measures by the developer [20].</li><li>✓ Sensitive information should never be logged [28].</li><li>✓ V-222444: The application must not write sensitive data into the application logs. Design or reconfigure the application to not write sensitive data to the logs [15].</li><li>✓ 1.1. Classify data storage according to sensitivity and apply controls accordingly (e.g. passwords, personal data, location, error logs, etc.). Process, store and use data according to its classification [16].</li><li>✓ SI-11 ERROR HANDLING</li></ul>
<b>Control:</b> <ol style="list-style-type: none"><li>a. Generate error messages that provide information necessary for corrective actions without revealing information that could be exploited; and</li><li>b. Reveal error messages only to [Assignment: organization-defined personnel or roles]</li></ol> <b>Discussion:</b> Organizations consider the structure and content of error messages. The extent to which systems can handle error conditions is guided and informed by organizational policy and operational requirements. Exploitable information includes stack traces and implementation details; erroneous logon attempts with passwords mistakenly entered as the username; mission or business information that can be derived from, if not stated explicitly by, the information recorded; and personally identifiable information, such as account numbers, social security numbers, and credit card numbers. Error messages may also provide a covert channel for transmitting information [11].
<b>Priority:</b> Not defined
<b>Rationale:</b> Preventing sensitive data from being logged minimizes the risk of unauthorized access and inadvertent exposure, ensuring compliance with healthcare regulations and safeguarding user privacy.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>

<p><b>1. Log Content Review Test</b></p> <ul style="list-style-type: none"> <li>✓ Test: Analyze application logs to verify no sensitive data, such as passwords, PII, or health information, is recorded.</li> <li>✓ Expected Result: Logs do not contain sensitive data under any circumstances.</li> </ul> <p><b>2. Sensitive Data Handling Test</b></p> <ul style="list-style-type: none"> <li>✓ Test: Validate that sensitive data is obfuscated or excluded from logging during runtime events and error handling.</li> <li>✓ Expected Result: Sensitive data is either not logged or replaced with placeholder values.</li> </ul> <p><b>3. Log Configuration Test</b></p> <ul style="list-style-type: none"> <li>✓ Test: Confirm that the logging configuration explicitly prevents sensitive data from being captured.</li> <li>✓ Expected Result: Logging settings are appropriately configured to avoid storing sensitive information.</li> </ul> <p><b>4. Error Logging Test</b></p> <ul style="list-style-type: none"> <li>✓ Test: Trigger error scenarios to verify that error logs do not include stack traces, implementation details, or sensitive data.</li> <li>✓ Expected Result: Logs provide generic error information necessary for debugging without exposing sensitive details.</li> </ul> <p><b>5. Security Audit Test</b></p> <ul style="list-style-type: none"> <li>✓ Test: Perform a security audit of the application's logging mechanism to ensure compliance with data handling policies.</li> <li>✓ Expected Result: The audit confirms that sensitive data is not captured in logs and that logs comply with security and privacy standards.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<p><b>Author and date:</b></p> <p>Carlos M. Mejía-Granda  José L Fernández-Alemán  Juan Manuel Carrillo-de-Gea  Joaquín Nicolás  08/01/2026</p>

<p><b>PUID:</b> [SECM-CAT-IPC-025]</p> <p><b>Requirement description:</b> Mobile health applications must ensure that sensitive data, including database administration credentials and health-related information, is removed or sanitized from production database exports and backups. Backups must be excluded from device-level and cloud synchronization mechanisms to prevent unauthorized access.</p> <p><b>Source:</b></p> <ul style="list-style-type: none"> <li>✓ V-222666: Production database exports must have database administration credentials and sensitive data removed before releasing the export. Remove sensitive data from production database exports [15].</li> <li>✓ 2.8: Verify that no sensitive data is included in backups [19].</li> </ul>
---

- ✓ Disable backup mode (Android): 1. By disabling backup mode on Android devices, you prevent the inclusion of app data in the device's backup, ensuring that sensitive data from the app is not stored in the device backup. 2. Limit application attack surface by only exporting activities, content providers and services that are necessary to be exported [3].
- ✓ MASVS-STORAGE-2: The app prevents leakage of sensitive data: Prevent the unintentional storage or exposure of sensitive data in publicly accessible locations. This control addresses potential leaks that can be avoided through proactive measures by the developer [20].
- ✓ 1.1. Classify data storage according to sensitivity and apply controls accordingly (e.g. passwords, personal data, location, error logs, etc.). Process, store and use data according to its classification [16].
- ✓ 1.29. Exclude sensitive application files from device backups and cloud synchronization services. If this option is not available in the in use platform (e.g., Android), exclude the whole application from device backups [16].

**Priority:** Not defined

**Rationale:** Removing sensitive data from database exports and backups reduces the risk of data breaches, unauthorized access, and misuse of confidential information. This ensures compliance with healthcare data protection regulations and safeguards user privacy.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

#### 1. Database Export Review Test

- ✓ Test: Export a production database and verify that sensitive data, such as user credentials and PII, is removed or anonymized.
- ✓ Expected Result: The exported database does not contain sensitive information, and fields are sanitized or anonymized as per policy.

#### 2. Backup Configuration Test

- ✓ Test: Check that backup and synchronization features are disabled for sensitive application files.
- ✓ Expected Result: Sensitive data is excluded from device backups and cloud synchronization.

#### 3. Restore Simulation Test

- ✓ Test: Simulate restoring a backup to ensure no sensitive data is retrievable.
- ✓ Expected Result: Restored data complies with the sensitive data exclusion policy.

#### 4. Audit of Data Classification Policies

- ✓ Test: Review the data classification and handling policies for backup and export processes.

- ✓ Expected Result: Policies ensure that sensitive data is identified and excluded during exports and backups.

## 5. Security Audit Test

- ✓ Test: Conduct a security audit of backup and export mechanisms to ensure compliance with data sanitization and exclusion policies.
- ✓ Expected Result: The audit confirms that sensitive data is appropriately handled and excluded from backups and exports.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda

José L Fernández-Alemán

Juan Manuel Carrillo-de-Gea

Joaquín Nicolás

08/01/2026

**PUID:** [SECM-CAT-IPC-026]

**Requirement description:** Mobile health applications must disable debugging features and limit logging in production environments to prevent inadvertent exposure of sensitive information. Personally identifiable information (PII) must not be logged, and logging levels must be configurable using debug flags and custom logging classes.

**Source:**

- ✓ Logging Practices: The application must avoid logging personally identifiable information (PII) and limit log usage in production environments. Logging must be managed using debug flags and custom Log classes to easily configure logging levels and prevent inadvertent data exposure [18].
- ✓ Disable application logging and debug messages in production releases. All exceptions should be handled securely [16].
- ✓ Disable Debugging: 1. Disable debugging features in the production version of the app [3].

**Priority:** Not defined

**Rationale:** Limiting logging and disabling debugging in production environments ensures sensitive data is not exposed inadvertently and reduces the risk of attackers exploiting debug information. Proper log management enhances security and complies with privacy regulations for healthcare applications.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

- 1. Debugging Features Test**
  - ✓ Test: Verify that debugging features, such as verbose logging and debug messages, are disabled in the production version of the app.
  - ✓ Expected Result: Debugging options are not available or active in the production build.
- 2. Log Content Review Test**
  - ✓ Test: Generate application logs during normal operations and review for any inclusion of PII or sensitive information.
  - ✓ Expected Result: Logs do not contain PII, credentials, or other sensitive data.
- 3. Custom Logging Mechanism Test**
  - ✓ Test: Inspect the implementation of custom logging classes and debug flags to verify that logging levels can be configured securely.
  - ✓ Expected Result: Logging levels can be controlled, and sensitive logs are restricted to non-production environments.
- 4. Production Log Management Test**
  - ✓ Test: Test the app in a production-like environment to ensure logs are minimal and do not contain sensitive details.
  - ✓ Expected Result: Logging in production environments is limited to essential operational information.
- 5. Security Audit of Logging Practices**
  - ✓ Test: Conduct a security audit to review logging mechanisms and debug settings.
  - ✓ Expected Result: Audit confirms compliance with log management and debugging feature restrictions

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda  
 José L Fernández-Alemán  
 Juan Manuel Carrillo-de-Gea  
 Joaquín Nicolás  
 08/01/2026

**PUID:** [SECM-CAT-IPC-027]

**Requirement description:** The application must prevent the unintentional storage or exposure of sensitive data, ensuring that any sensitive data (including keys) is encrypted and stored in a tamper-proof, platform-supported area if required.

**Source:**

- ✓ MASVS-STORAGE-2: The app prevents leakage of sensitive data: Prevent the unintentional storage or exposure of sensitive data in publicly accessible locations. This control addresses potential leaks that can be avoided through proactive measures by the developer [20].

✓ 1.5. Do not store/cache sensitive data (including keys) unless they are encrypted and if possible stored in a platform supported tamper-proof area [5].
<b>Priority:</b> Not defined
<b>Rationale:</b> Unintentional storage or exposure of sensitive data in publicly accessible locations increases the risk of data breaches. Encryption and tamper-proof storage minimize the potential for unauthorized access to sensitive information, enhancing security.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<p><b>1. Storage Location Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Verify that sensitive data is not stored in publicly accessible directories or shared storage.</li> <li>✓ Expected result: Sensitive data is stored only in secure, private directories with restricted access.</li> </ul> <p><b>2. Encryption Validation Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Check that all stored sensitive data, including cryptographic keys, is encrypted using industry-standard encryption algorithms.</li> <li>✓ Expected result: Sensitive data at rest is encrypted and inaccessible without decryption.</li> </ul> <p><b>3. Tamper-Proof Storage Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Confirm that sensitive data is stored in a platform-supported, tamper-proof storage area (e.g., Keychain on iOS, Keystore on Android).</li> <li>✓ Expected result: Sensitive data resides in a tamper-proof storage location supported by the platform.</li> </ul> <p><b>4. Data Leakage Prevention Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Attempt to access sensitive data through unintended mechanisms, such as debugging or file browsing tools.</li> <li>✓ Expected result: Sensitive data is inaccessible through unauthorized methods.</li> </ul> <p><b>5. Cache Storage Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Verify that sensitive data is not cached or temporarily stored in a manner accessible to unauthorized users.</li> <li>✓ Expected result: Sensitive data is not cached in unprotected or publicly accessible locations.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda
José L Fernández-Alemán
Juan Manuel Carrillo-de-Gea

Joaquín Nicolás 08/01/2026
-------------------------------

**PUID: [SECM-CAT-IPC-028]**

**Requirement description:** The mobile application must implement security controls to prevent unauthorized and unintended information transfer via shared system resources. Data classification mechanisms should be utilized to manage sensitive information, ensuring that personal or confidential data is not exposed or shared without explicit justification and user consent.

**Source:**

- ✓ SRG-APP-000516-MAPP-000075: The mobile app must not record or forward sensor data unless explicitly authorized to do so [17].

**Priority:** Not defined

**Rationale:** Protecting sensitive data in mobile health applications is critical to prevent unauthorized access, minimize security risks, and comply with privacy regulations. Proper classification and control measures reduce the likelihood of data leakage or misuse.

**Number of Children:** 0**Number of parents:** 0**Cycles:** 0**Number Audit:** 0**Child PUIDs:** Not described**Parent PUIDs:** Not described**Exclusion PUIDs:** Not described**Importance:** Not described**Current state:** To be determined**Verification method:** Demonstration/Analysis**Validation criteria:****1. Data Sharing Test:**

- ✓ Test: Attempt to access or transfer sensitive information via shared system resources without proper authorization.
- ✓ Expected Result: Access is denied, and unauthorized transfer is blocked.

**2. Data Classification Review:**

- ✓ Test: Verify the application's data classification mechanism for sensitive information (e.g., passwords, health data).
- ✓ Expected Result: Data is appropriately classified, and protective controls are applied based on the classification.

**3. Third-Party Component Test:**

- ✓ Test: Evaluate third-party components (e.g., analytics or advertising libraries) for data access and sharing behaviors.
- ✓ Expected Result: No unauthorized sharing of sensitive or personal information occurs.

**4. Boundary Enforcement Test:**

- ✓ Test: Simulate scenarios where shared system resources are accessed without proper boundaries.
- ✓ Expected Result: The application enforces boundaries and prevents unintended data transfer.

**5. Audit and Logging Test:**

- ✓ Test: Review logs for unauthorized attempts to access or transfer sensitive data.

✓ Expected Result: Unauthorized attempts are logged, flagged, and blocked by the application.
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IPC-029]
<b>Requirement description:</b> The mobile application must apply data classification practices and enforce strict access boundaries to safeguard sensitive information. It must ensure that personal, confidential, or health-related data is only processed, stored, or shared when explicitly required and authorized. The application should actively prevent data leakage through shared system resources by validating access, restricting exposure, and minimizing unnecessary interactions with third-party components.
<b>Source:</b>
<ul style="list-style-type: none"><li>✓ V-222592: Applications must prevent unauthorized and unintended information transfer via shared system resources. Configure or design the application to utilize a security control that will implement a boundary that will prevent unauthorized and unintended information transfer via shared system resources [15].</li><li>✓ Cautious Data Exposure: The application must evaluate the necessity of providing personal information to third-party components, such as advertising services. If the purpose is unclear, personal information must not be shared. The application should reduce access to personal information to minimize potential security risks [18].</li><li>✓ MASVS-STORAGE-2: The app prevents leakage of sensitive data: Prevent the unintentional storage or exposure of sensitive data in publicly accessible locations. This control addresses potential leaks that can be avoided through proactive measures by the developer [20].</li><li>✓ 5.12 Classification of information: Information should be classified according to the information security needs of the organization based on confidentiality, integrity, availability and relevant interested party requirements [6].</li><li>✓ 1.1. Classify data storage according to sensitivity and apply controls accordingly (e.g. passwords, personal data, location, error logs, etc.). Process, store and use data according to its classification [16].</li></ul>
<b>Priority:</b> Not defined
<b>Rationale:</b> Protecting sensitive data in mobile health applications is critical to prevent unauthorized access, minimize security risks, and comply with privacy regulations. Proper classification and control measures reduce the likelihood of data leakage or misuse.

<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Data Sharing Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Attempt to access or transfer sensitive information via shared system resources without proper authorization.</li> <li>✓ Expected Result: Access is denied, and unauthorized transfer is blocked.</li> </ul> </li> <li><b>2. Data Classification Review:</b> <ul style="list-style-type: none"> <li>✓ Test: Verify the application's data classification mechanism for sensitive information (e.g., passwords, health data).</li> <li>✓ Expected Result: Data is appropriately classified, and protective controls are applied based on the classification.</li> </ul> </li> <li><b>3. Third-Party Component Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Evaluate third-party components (e.g., analytics or advertising libraries) for data access and sharing behaviors.</li> <li>✓ Expected Result: No unauthorized sharing of sensitive or personal information occurs.</li> </ul> </li> <li><b>4. Boundary Enforcement Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Simulate scenarios where shared system resources are accessed without proper boundaries.</li> <li>✓ Expected Result: The application enforces boundaries and prevents unintended data transfer.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IPC-030]
<b>Requirement description:</b> The mobile health application must identify and clearly document all sensitive data elements and their associated protection requirements. This includes maintaining a Security Classification Guide for classified or sensitive data and implementing measures to prevent data leakage in publicly accessible locations.
<b>Source:</b>

- ✓ 1.5. Verify that data considered sensitive in the context of the mobile app is clearly identified [19].
- ✓ MASVS-STORAGE-2: The app prevents leakage of sensitive data: Prevent the unintentional storage or exposure of sensitive data in publicly accessible locations. This control addresses potential leaks that can be avoided through proactive measures by the developer [20].
- ✓ V-222423: Application data protection requirements must be identified and documented. Identify and document the application data elements and the data protection requirements [15].
- ✓ V-222664: If the application contains classified data, a Security Classification Guide must exist containing data elements and their classification. Create and maintain a security classification guide [15].

**Priority:** Not defined

**Rationale:** Clearly identifying and classifying sensitive data ensures that appropriate security measures are implemented to protect user information and comply with privacy regulations, reducing the risk of data exposure or misuse.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

**1. Data Identification Test:**

- ✓ Test: Verify that all sensitive data elements (e.g., health records, PII) are identified and classified within the application documentation.
- ✓ Expected Result: All sensitive data is identified and appropriately classified based on protection requirements.

**2. Security Classification Guide Review:**

- ✓ Test: Review the existence and completeness of the Security Classification Guide for sensitive or classified data.
- ✓ Expected Result: The guide includes detailed classifications and protection requirements for all data elements.

**3. Public Data Exposure Test:**

- ✓ Test: Attempt to access sensitive data from publicly accessible storage locations (e.g., shared directories).
- ✓ Expected Result: Sensitive data is not stored or exposed in publicly accessible locations.

**4. Logging and Monitoring Test:**

- ✓ Test: Simulate unauthorized access attempts to classified or sensitive data.
- ✓ Expected Result: All access attempts are logged, and unauthorized attempts are flagged.

**5. Developer Training and Documentation Test:**

<ul style="list-style-type: none"> <li>✓ Test: Evaluate whether development teams are provided with documentation and guidelines on identifying and classifying sensitive data.</li> <li>✓ Expected Result: Documentation is complete and training has been provided to ensure adherence to data protection protocols.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IPC-031]
<b>Requirement description:</b> The mobile health application must have the capability to mark sensitive or classified output when required, ensuring that sensitive data is not unintentionally exposed or shared with unauthorized third parties. The application must minimize sensitive data access, adhere to the principle of minimal disclosure, and securely manage sensitive data in the UI and storage.
<b>Source:</b> <ul style="list-style-type: none"> <li>✓ V-222643: The application must have the capability to mark sensitive/classified output when required. Enable the application to adequately mark sensitive/classified output [15].</li> <li>✓ 2.3: Verify that no sensitive data is shared with third parties unless it is a necessary part of the architecture [19].</li> <li>✓ Do not send data to third parties, or, failing that, inform user [25].</li> <li>✓ Minimization of Sensitive Data Access: The application must minimize the use of APIs that access sensitive or personal information. If user data access is necessary, the application should avoid storing or transmitting this data whenever possible, opting instead for non-reversible forms such as hashes [18].</li> <li>✓ MASVS-STORAGE-2: The app prevents leakage of sensitive data: Prevent the unintentional storage or exposure of sensitive data in publicly accessible locations. This control addresses potential leaks that can be avoided through proactive measures by the developer [20].</li> <li>✓ MASVS-PLATFORM-3: The app uses the user interface securely: Sensitive data has to be displayed in the UI in many situations (e.g. passwords, credit card details, OTP codes in notifications). This control ensures that this data doesn't end up being unintentionally leaked due to platform mechanisms such as auto-generated screenshots or accidentally disclosed via e.g. shoulder surfing or sharing the device with another person [20].</li> </ul>

<ul style="list-style-type: none"> <li>✓ 1.1. Classify data storage according to sensitivity and apply controls accordingly (e.g. passwords, personal data, location, error logs, etc.). Process, store and use data according to its classification [16].</li> <li>✓ 1.12. Apply the principle of minimal disclosure - only collect and disclose data which is required for business use of the application [16].</li> </ul>
<b>Priority:</b> Not defined
<b>Rationale:</b> Marking sensitive or classified data ensures proper handling and reduces the risk of accidental exposure. Minimizing sensitive data usage and adhering to secure UI practices protects against data leaks and unauthorized access, aligning with privacy and security regulations.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Output Marking Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Verify that all sensitive or classified output is appropriately marked during storage or export.</li> <li>✓ Expected Result: Sensitive data is clearly marked with appropriate classifications, such as "Confidential" or "Restricted."</li> </ul> </li> <li><b>2. Third-Party Data Sharing Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Analyze whether sensitive data is shared with third-party APIs or services without explicit necessity or user consent.</li> <li>✓ Expected Result: Sensitive data is only shared with third parties when necessary and with user consent, accompanied by clear notifications.</li> </ul> </li> <li><b>3. Data Minimization Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Review the application's API usage to ensure that sensitive data access is minimized and limited to essential functionality.</li> <li>✓ Expected Result: APIs accessing sensitive data are only used where strictly necessary, with mechanisms to avoid data storage or transmission.</li> </ul> </li> <li><b>4. UI Protection Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Check the user interface for secure handling of sensitive data, such as masking passwords and preventing sensitive data from being captured in screenshots.</li> <li>✓ Expected Result: Sensitive data in the UI is protected from accidental exposure through platform mechanisms like masking or disabling screenshot functionality.</li> </ul> </li> <li><b>5. Data Storage Classification Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Evaluate whether the application applies proper controls based on data sensitivity classifications in storage.</li> <li>✓ Expected Result: Sensitive data is stored securely, with controls appropriate to its classification, such as encryption or access restrictions.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer

<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda
José L Fernández-Alemán
Juan Manuel Carrillo-de-Gea
Joaquín Nicolás
08/01/2026

<b>PUID:</b> [SECM-CAT-IPC-032]
<b>Requirement description:</b> The mobile health application must encrypt all multimedia content, including audio, video, and file-sharing data, to ensure secure storage and transfer in telehealth systems.
<b>Source:</b>
✓ Secure mobile telehealth systems must also allow for efficient storage and transfer of multimedia content. That is, they must encrypt all of the data to allow audio, video, and file sharing [24].
<b>Priority:</b> Not defined
<b>Rationale:</b> Encrypting multimedia content safeguards sensitive patient data and communication from unauthorized access, ensuring compliance with privacy and security standards for telehealth systems.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<b>1. Multimedia Encryption Test:</b>
✓ Test: Verify that all audio, video, and file-sharing data are encrypted during storage and transmission.
✓ Expected Result: Multimedia content is encrypted using industry-standard algorithms (e.g., AES-256) and cannot be accessed without proper decryption.
<b>2. Secure Transfer Protocol Test:</b>
✓ Test: Check whether multimedia data transfer uses secure protocols such as HTTPS or TLS.
✓ Expected Result: Multimedia data is transmitted securely, without vulnerabilities to interception or eavesdropping.
<b>3. File Integrity Verification Test:</b>
✓ Test: Validate that shared multimedia files retain integrity during transfer and cannot be tampered with.
✓ Expected Result: Multimedia files are validated for integrity post-transfer, ensuring no unauthorized modifications occurred.

<p><b>4. Data Storage Security Test:</b></p> <ul style="list-style-type: none"><li>✓ Test: Evaluate the application's multimedia storage mechanism for compliance with encryption requirements.</li><li>✓ Expected Result: Multimedia data stored on the device or server is encrypted and protected from unauthorized access.</li></ul> <p><b>5. End-to-End Encryption Test:</b></p> <ul style="list-style-type: none"><li>✓ Test: Confirm that multimedia content in telehealth sessions is protected using end-to-end encryption.</li><li>✓ Expected Result: Only the intended sender and receiver can access the encrypted multimedia data during telehealth communication.</li></ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IPC-033]
<b>Requirement description:</b> The mobile health application must prevent the unintentional exposure of sensitive data (e.g., passwords, credit card details) through the user interface or screenshots. Mechanisms such as disabling screen capture or notifying users of security implications must be implemented.
<b>Source:</b> <ul style="list-style-type: none"><li>✓ 2.7: Verify that no sensitive data, such as passwords and credit card numbers, is exposed through the user interface or leaks to screenshots [19].</li><li>✓ 1.22. Disable screen capture for interfaces that contain sensitive data. If the platform does not support this option (e.g., iOS), notify the user about potential security implications of storing a screenshot in unprotected storage [16].</li><li>✓ MASVS-STORAGE-2: The app prevents leakage of sensitive data: Prevent the unintentional storage or exposure of sensitive data in publicly accessible locations. This control addresses potential leaks that can be avoided through proactive measures by the developer [20].</li><li>✓ MASVS-PLATFORM-3: The app uses the user interface securely: Sensitive data has to be displayed in the UI in many situations (e.g. passwords, credit card details, OTP codes in notifications). This control ensures that this data doesn't end up being unintentionally leaked due to platform mechanisms such as auto-generated screenshots or accidentally disclosed via e.g. shoulder surfing or sharing the device with another person [20].</li></ul>
<b>Priority:</b> Not defined

<b>Rationale:</b> Preventing sensitive data exposure through the user interface or screenshots reduces the risk of unintended data leakage, ensuring compliance with privacy requirements and safeguarding user information.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<p><b>1. Screen Capture Prevention Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Verify that screen capture is disabled on interfaces displaying sensitive information.</li> <li>✓ Expected Result: The application prevents screen captures for sensitive screens or notifies users of potential risks if platform constraints exist.</li> </ul>
<p><b>2. UI Data Obfuscation Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Validate that sensitive data displayed in the UI (e.g., passwords) is obfuscated or masked.</li> <li>✓ Expected Result: Sensitive data is displayed in a secure manner, such as masked characters (e.g., "•••") for passwords.</li> </ul>
<p><b>3. Screenshot Leakage Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Check if sensitive information appears in auto-generated screenshots (e.g., task switcher thumbnails).</li> <li>✓ Expected Result: Screens containing sensitive data are excluded from task switcher thumbnails or display generic content.</li> </ul>
<p><b>4. Secure Notification Display Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Evaluate if sensitive data (e.g., OTP codes) appears in notification previews.</li> <li>✓ Expected Result: Sensitive data is hidden or minimized in notification previews unless explicitly enabled by the user.</li> </ul>
<p><b>5. Penetration Testing:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Conduct penetration tests to attempt data extraction from screenshots or UI components.</li> <li>✓ Expected Result: No sensitive data is retrievable through screenshots or UI leaks during the tests.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda
José L Fernández-Alemán
Juan Manuel Carrillo-de-Gea
Joaquín Nicolás
08/01/2026

<b>PUID:</b> [SECM-CAT-IPC-034]
<b>Requirement description:</b> The mobile application must not use URL-embedded session IDs to ensure the security of session management. Session ID information must be securely transmitted via cookies or other secure mechanisms. Sensitive functionality must not be exposed via custom URL schemes unless robust protection measures are implemented.
<b>Source:</b> <ul style="list-style-type: none"><li>✓ V-222581: Applications must not use URL embedded session IDs. Configure the application to transmit session ID information via cookies [15].</li><li>✓ 6.3: Verify that the app does not export sensitive functionality via custom URL schemes, unless these mechanisms are properly protected [19].</li><li>✓ 1.1. Classify data storage according to sensitivity and apply controls accordingly (e.g. passwords, personal data, location, error logs, etc.). Process, store and use data according to its classification [16].</li></ul>
<b>Priority:</b> Not defined
<b>Rationale:</b> URL-embedded session IDs can lead to session hijacking if the URLs are exposed in logs, browser history, or shared inadvertently. Transmitting session data via cookies ensures confidentiality and integrity, reducing the risk of unauthorized access. Protecting custom URL schemes prevents potential exploitation by malicious actors.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b> <ol style="list-style-type: none"><li><b>1. Session Transmission Validation Test:</b><ul style="list-style-type: none"><li>✓ Test: Confirm that session IDs are not transmitted via URLs during app operations.</li><li>✓ Expected Result: Session IDs are exclusively transmitted via secure cookies or equivalent mechanisms.</li></ul></li><li><b>2. URL Scheme Security Test:</b><ul style="list-style-type: none"><li>✓ Test: Attempt to exploit sensitive functionality via custom URL schemes without proper authentication.</li><li>✓ Expected Result: Sensitive functionalities are inaccessible without authentication or other protection mechanisms.</li></ul></li><li><b>3. Data Storage Classification Review:</b><ul style="list-style-type: none"><li>✓ Test: Validate the classification of session ID storage based on sensitivity and ensure adherence to security policies.</li><li>✓ Expected Result: Session IDs are stored and processed securely according to their classification.</li></ul></li><li><b>4. Penetration Test for Session Hijacking:</b></li></ol>

- ✓ Test: Conduct a penetration test to simulate session hijacking scenarios via URL sharing or interception.
- ✓ Expected Result: The application resists hijacking attempts, and session data remains secure.

##### 5. Cookie Flag Verification Test:

- ✓ Test: Verify that secure and HTTPOnly flags are set on cookies transmitting session data.
- ✓ Expected Result: All cookies carrying session data have secure and HTTPOnly flags enabled.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda  
José L Fernández-Alemán  
Juan Manuel Carrillo-de-Gea  
Joaquín Nicolás  
08/01/2026

**PUID:** [SECM-CAT-IPC-035]

**Requirement description:** The mobile application must ensure that, when using JavaScript code running in the context of a file scheme URL, all unused platform-supported attributes are disabled. Specifically, the application must restrict access to content from other file scheme URLs and content from any origin unless explicitly required and securely implemented.

**Source:**

- ✓ 12.3. In the case that the application uses JavaScript code running in the context of a file scheme URL, it is recommended to disable any unused platform supported attributes, such as accessing content from other file scheme URL and content from any origin [16].

**Priority:** Not defined

**Rationale:** Allowing unrestricted access to file scheme URLs or content from any origin increases the risk of unauthorized data access and exploitation of application vulnerabilities. Restricting such attributes minimizes the attack surface and enhances the security posture of the application.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

##### 1. JavaScript Access Control Test:

- ✓ Test: Attempt to access content from another file scheme URL within the application's JavaScript context.

<ul style="list-style-type: none"> <li>✓ Expected Result: Access to other file scheme URLs is blocked unless explicitly required and securely configured.</li> </ul> <p><b>2. Content Origin Restriction Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Attempt to load content from an untrusted origin using JavaScript in the context of a file scheme URL.</li> <li>✓ Expected Result: Loading content from any origin is restricted unless explicitly allowed by secure configurations.</li> </ul> <p><b>3. Configuration Review Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Examine the application's configuration to ensure unused attributes related to file scheme URLs are disabled.</li> <li>✓ Expected Result: All unused platform-supported attributes are explicitly disabled in the application's settings.</li> </ul> <p><b>4. Penetration Test for File Scheme URLs:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Conduct a penetration test targeting vulnerabilities related to file scheme URL handling.</li> <li>✓ Expected Result: The application resists unauthorized access or exploitation through file scheme URLs.</li> </ul> <p><b>5. Security Code Review:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Perform a security review of the JavaScript implementation and its interaction with file scheme URLs.</li> <li>✓ Expected Result: The JavaScript code adheres to secure practices, and unnecessary attributes are disabled.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IPC-036]
<b>Requirement description:</b> The mobile application must remove sensitive data from views when transitioning to the background. If the platform retains a screenshot of the visible screen in local storage during backgrounding (e.g., iOS), the application must either disable backgrounding or apply a blurred screen overlay to protect sensitive data.
<b>Source:</b> <ul style="list-style-type: none"> <li>✓ 2.9: Verify that the app removes sensitive data from views when backgrounded [19].</li> <li>✓ 1.23. Disable backgrounding or use a blurry screen when the application transitions to the background in platforms that maintain a screenshot of the visible screen in the local storage (e.g., iOS) [16].</li> <li>✓ Allow deactivation of running of the application in the background [25].</li> </ul>
<b>Priority:</b> Not defined

<b>Rationale:</b> Sensitive data displayed on the screen when an application transitions to the background may be captured in screenshots or cached by the operating system. This increases the risk of unauthorized access and data leakage, particularly on shared or compromised devices.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Background Transition Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Transition the application to the background while sensitive data is displayed.</li> <li>✓ Expected Result: Sensitive data is removed, obscured, or replaced with a blurred overlay when the application is backgrounded.</li> </ul> </li> <li><b>2. Screenshot Storage Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Inspect the operating system's local storage to verify if sensitive data is visible in screenshots captured during background transitions.</li> <li>✓ Expected Result: Screenshots of the application display blurred or generic content instead of sensitive data.</li> </ul> </li> <li><b>3. Configuration Review Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Verify the application's configuration for handling background transitions and disabling backgrounding, if necessary.</li> <li>✓ Expected Result: Backgrounding is disabled or properly configured to obscure sensitive data.</li> </ul> </li> <li><b>4. User Deactivation Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Attempt to deactivate the application's background running feature through user settings or administrative configurations.</li> <li>✓ Expected Result: The application allows users or administrators to disable backgrounding if required.</li> </ul> </li> <li><b>5. Penetration Test for Data Leakage:</b> <ul style="list-style-type: none"> <li>✓ Test: Perform a penetration test focusing on capturing sensitive data through screenshots or background caching mechanisms.</li> <li>✓ Expected Result: No sensitive data is exposed through screenshots or other cached views during background transitions.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IPC-037]
<b>Requirement description:</b> The mobile application must exclude sensitive user interface elements from being accessed by custom accessibility applications, particularly on platforms like Android that support such permissions. Sensitive data must not be exposed through accessibility features.
<b>Source:</b>
✓ 12.15. In platforms that support custom applications with accessibility permissions (e.g., Android), exclude sensitive user interface elements from being accessed by accessibility applications [16].
<b>Priority:</b> Not defined
<b>Rationale:</b> Accessibility services can potentially be exploited by malicious applications to extract sensitive information from user interface elements. Protecting sensitive data ensures compliance with privacy and security requirements, especially in healthcare applications where confidentiality is paramount.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Accessibility Exclusion Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Use an accessibility service to attempt access to sensitive UI elements in the application.</li> <li>✓ Expected Result: Sensitive UI elements are inaccessible to accessibility services.</li> </ul> </li> <li><b>2. UI Configuration Review Test:</b> <ul style="list-style-type: none"> <li>✓ <b>Test:</b> Verify the code and configuration for sensitive UI elements to ensure proper use of attributes like <code>importantForAccessibility="no"</code>.</li> <li>✓ Expected Result: Sensitive elements are properly marked to prevent exposure through accessibility features.</li> </ul> </li> <li><b>3. Permission Abuse Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Simulate a malicious application with accessibility permissions attempting to extract sensitive data from the application.</li> <li>✓ Expected Result: The malicious application cannot access sensitive information.</li> </ul> </li> <li><b>4. Platform-Specific Compliance Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Ensure the application follows platform-specific best practices to restrict sensitive data access via accessibility features (e.g., Android's security guidelines for accessibility services).</li> <li>✓ Expected Result: Application complies with platform-specific security requirements.</li> </ul> </li> <li><b>5. Dynamic Accessibility Audit:</b> <ul style="list-style-type: none"> <li>✓ Test: Conduct a dynamic audit of the application's accessibility features during runtime to identify any unintended exposure of sensitive elements.</li> <li>✓ Expected Result: No sensitive UI elements are exposed to accessibility services during runtime.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization

<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IPC-038]
<b>Requirement description:</b> The mobile application must ensure that passwords and cryptographic keys are not visible in any cache, application logs, or debug information to prevent unintended exposure of sensitive data.
<b>Source:</b>
✓ 3.8. Ensure passwords and keys are not visible in cache or logs [16].
<b>Priority:</b> Not defined
<b>Rationale:</b> Storing sensitive information such as passwords and keys in logs or caches can lead to severe security risks if the data is exposed to unauthorized entities. This is particularly critical in healthcare applications where confidentiality is a priority.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Log Inspection Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Review application logs during authentication and cryptographic operations to verify that passwords and keys are not logged.</li> <li>✓ Expected Result: No passwords or keys are present in the application logs.</li> </ul> </li> <li><b>2. Cache Review Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Inspect the device cache and temporary storage for any residual sensitive data during and after application use.</li> <li>✓ Expected Result: No passwords or cryptographic keys are found in the cache or temporary storage.</li> </ul> </li> <li><b>3. Code Analysis Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Perform a static code analysis to ensure no debug statements or logs contain sensitive information like passwords or keys.</li> <li>✓ Expected Result: Sensitive information is explicitly excluded from logging or debugging outputs.</li> </ul> </li> <li><b>4. Dynamic Debugging Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Use debugging tools during runtime to verify that passwords and keys are not displayed in debug logs or console outputs.</li> </ul> </li> </ol>

<ul style="list-style-type: none"> <li>✓ Expected Result: No sensitive data appears in runtime debugging outputs.</li> </ul> <p><b>5. Penetration Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Simulate an attack scenario to retrieve sensitive data from logs or cache through unauthorized access.</li> <li>✓ Expected Result: Sensitive information remains inaccessible during penetration testing.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IPC-039]
<b>Requirement description:</b> The mobile application must require user re-authentication when transitioning from background to foreground if it contains sensitive data. Alternatively, the application must verify that the device is secured with a PIN, pattern, or password.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ 2.17. For applications that contain sensitive data, it is also recommended to request for user re-authentication when the application state changes to background or verify that the device is secured with PIN, pattern or password [16].</li> </ul>
<b>Priority:</b> Not defined
<b>Rationale:</b> Re-authentication or device security verification ensures that unauthorized users cannot access sensitive data when the application is backgrounded and subsequently reopened. This mitigates the risk of unauthorized access due to unattended devices or session hijacking, especially in healthcare scenarios.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Re-authentication Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Transition the application to the background and back to the foreground. Verify that the user is prompted to re-authenticate.</li> <li>✓ Expected Result: The application requires re-authentication before granting access to sensitive data.</li> </ul> </li> <li><b>2. Device Security Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Attempt to access the application after locking and unlocking the device. Verify the application checks for device security measures like PIN, pattern, or password.</li> </ul> </li> </ol>

<ul style="list-style-type: none"> <li>✓ Expected Result: The application ensures the device is secured before resuming access.</li> </ul> <p><b>3. State Change Handling Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Observe the application's behavior during state transitions (backgrounding and foregrounding).</li> <li>✓ Expected Result: Sensitive data is protected or hidden, and access is contingent on security measures.</li> </ul> <p><b>4. Penetration Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Simulate unauthorized access during application state transitions.</li> <li>✓ Expected Result: Sensitive data remains inaccessible without re-authentication or device security.</li> </ul> <p><b>5. Configuration Review Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Review the application configuration and code to ensure security measures are implemented for state transitions.</li> <li>✓ Expected Result: The application includes robust mechanisms to enforce re-authentication or device security checks.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<p><b>Author and date:</b></p> <p>Carlos M. Mejía-Granda      José L Fernández-Alemán      Juan Manuel Carrillo-de-Gea      Joaquín Nicolás      08/01/2026</p>

<b>PUID:</b> [SECM-CAT-IPC-040]
<b>Requirement description:</b> The mobile application must enforce re-authentication for users and devices under organization-defined circumstances, such as privilege escalation, role changes, or after a defined time period. Dual authorization must be enforced for specific privileged commands or critical actions as required by organizational policies.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ V-222520: The application must require users to reauthenticate when organization-defined circumstances or situations require reauthentication. Configure the application to require reauthentication before user privilege is escalated and user roles are changed [15].</li> <li>✓ V-222521: The application must require devices to reauthenticate when organization-defined circumstances or situations requiring reauthentication. Configure the application to require reauthentication periodically [15].</li> <li>✓ IA-11 RE-AUTHENTICATION</li> </ul> <p>Control: Require users to re-authenticate when [Assignment: organization-defined circumstances or situations requiring re-authentication].</p> <p>Discussion: In addition to the re-authentication requirements associated with device locks, organizations may require re-authentication of individuals in certain situations, including when roles, authenticators or credentials change, when security categories of systems</p>

<p>change, when the execution of privileged functions occurs, after a fixed time period, or periodically [11].</p> <p>✓ Control Enhancements:</p> <p>(2) ACCESS ENFORCEMENT   DUAL AUTHORIZATION</p> <p>Enforce dual authorization for [Assignment: organization-defined privileged commands and/or other organization-defined actions].</p> <p>Discussion: Dual authorization, also known as two-person control, reduces risk related to insider threats. Dual authorization mechanisms require the approval of two authorized individuals to execute. To reduce the risk of collusion, organizations consider rotating dual authorization duties. Organizations consider the risk associated with implementing dual authorization mechanisms when immediate responses are necessary to ensure public and environmental safety [11].</p>
<b>Priority:</b> Not defined
<b>Rationale:</b> Requiring re-authentication ensures the security of sensitive operations and prevents unauthorized privilege escalation or role changes. Dual authorization adds an additional layer of protection for critical operations, reducing the risk of insider threats and unauthorized actions in healthcare applications.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<p><b>1. Privilege Escalation Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Attempt to escalate privileges or change user roles without re-authenticating.</li> <li>✓ Expected Result: The application prompts the user for re-authentication before completing the operation.</li> </ul> <p><b>2. Periodic Re-authentication Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Allow a session to remain active beyond the defined time period without user activity.</li> <li>✓ Expected Result: The application requires re-authentication to continue using the session.</li> </ul> <p><b>3. Dual Authorization Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Initiate an action requiring dual authorization. Verify that the approval of two authorized users is required to proceed.</li> <li>✓ Expected Result: The action cannot be completed without dual authorization.</li> </ul> <p><b>4. Role Change Re-authentication Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Change the user role within the application.</li> <li>✓ Expected Result: The application prompts for re-authentication before applying the role change.</li> </ul>

<b>5. Penetration Test:</b>
✓ Test: Simulate attempts to bypass re-authentication during privilege escalation or critical actions.
✓ Expected Result: Re-authentication mechanisms and dual authorization successfully block unauthorized access.
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IPC-041]
<b>Requirement description:</b> The mobile application must inspect custom notifications for sensitive content and provide users with the ability to disable notifications or hide sensitive information within notification content.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ V-222520: The application must require users to reauthenticate when organization-defined circumstances or situations require reauthentication. Configure the application to require reauthentication before user privilege is escalated and user roles are changed [15].</li> <li>✓ V-222521: The application must require devices to reauthenticate when organization-defined circumstances or situations requiring reauthentication. Configure the application to require reauthentication periodically [15].</li> <li>✓ IA-11 RE-AUTHENTICATION</li> </ul> <p>Control: Require users to re-authenticate when [Assignment: organization-defined circumstances or situations requiring re-authentication].</p> <p>Discussion: In addition to the re-authentication requirements associated with device locks, organizations may require re-authentication of individuals in certain situations, including when roles, authenticators or credentials change, when security categories of systems change, when the execution of privileged functions occurs, after a fixed time period, or periodically [11].</p> <ul style="list-style-type: none"> <li>✓ Control Enhancements: <ul style="list-style-type: none"> <li>(2) ACCESS ENFORCEMENT   DUAL AUTHORIZATION</li> </ul> <p>Enforce dual authorization for [Assignment: organization-defined privileged commands and/or other organization-defined actions].</p> <p>Discussion: Dual authorization, also known as two-person control, reduces risk related to insider threats. Dual authorization mechanisms require the approval of two authorized</p> </li> </ul>

individuals to execute. To reduce the risk of collusion, organizations consider rotating dual authorization duties. Organizations consider the risk associated with implementing dual authorization mechanisms when immediate responses are necessary to ensure public and environmental safety [11].
<b>Priority:</b> Not defined
<b>Rationale:</b> Notifications may unintentionally expose sensitive information to unauthorized individuals. Allowing users to manage notification settings enhances privacy and reduces the risk of sensitive data leakage in healthcare applications.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Notification Content Inspection Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Send a notification containing sensitive content (e.g., patient information or medical data).</li> <li>✓ Expected Result: The application prevents sensitive data from being displayed directly in the notification.</li> </ul> </li> <li><b>2. Notification Disablement Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Provide the user the option to disable notifications in the application settings.</li> <li>✓ Expected Result: Notifications cease to appear after the user disables them.</li> </ul> </li> <li><b>3. Sensitive Content Hiding Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Configure the application to display notifications with sensitive information hidden (e.g., "New message received").</li> <li>✓ Expected Result: Notifications do not reveal sensitive content when this setting is enabled.</li> </ul> </li> <li><b>4. User Control Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Validate that the user can toggle notification settings (disable notifications or hide content) from the app's settings interface.</li> <li>✓ Expected Result: The user successfully updates notification preferences, and the changes take effect immediately.</li> </ul> </li> <li><b>5. Penetration Test for Notifications:</b> <ul style="list-style-type: none"> <li>✓ Test: Simulate an attempt to extract sensitive content from notifications through unauthorized access.</li> <li>✓ Expected Result: Sensitive content remains inaccessible or obfuscated in notification messages.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda

José L Fernández-Alemán  
Juan Manuel Carrillo-de-Gea  
Joaquín Nicolás  
08/01/2026

**PUID:** [SECM-CAT-IPC-042]

**Requirement description:** The mobile health application must inform users about the confidentiality of personal health information (PHI) accessed or processed by the system. This includes displaying confidentiality notices at login, labeling hardcopy outputs containing PHI as confidential, and providing clear privacy policies in compliance with applicable privacy regulations.

**Source:**

- ✓ 8.2.2 Labelling of Information: All health information systems processing personal health information should inform users of the confidentiality of personal health information accessible from the system (e.g. at start-up or log-in) and should label hardcopy output as confidential when it contains personal health information [7].
- ✓ 2.12: Verify that the app educates the user about the types of personally identifiable information processed, as well as security best practices the user should follow in using the app [19].
- ✓ Compliance with Privacy Regulations: If personal information is accessed, the application must provide a clear privacy policy detailing the use and storage of that data, in accordance with jurisdictional requirements [18].

**Priority:** Not defined

**Rationale:** Ensuring that users are aware of the sensitive nature of PHI reinforces the importance of secure handling and compliance with jurisdictional privacy laws. Educating users and appropriately labeling data reduces the risk of unintentional disclosure or misuse.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:****1. Confidentiality Notice Test:**

- ✓ Test: Launch the application and observe whether a confidentiality notice about PHI is displayed at login or startup.
- ✓ Expected Result: The application displays a notice informing users of the confidentiality of the PHI being processed.

**2. Hardcopy Label Test:**

- ✓ Test: Generate a hardcopy output containing PHI through the application.

<ul style="list-style-type: none"><li>✓ Expected Result: The output includes a clear label marking it as "Confidential" or similar.</li></ul> <p><b>3. Privacy Policy Availability Test:</b></p> <ul style="list-style-type: none"><li>✓ Test: Access the application's privacy policy through the settings or login screen.</li><li>✓ Expected Result: A detailed privacy policy is displayed, explaining the collection, use, and storage of PHI in compliance with regulations.</li></ul> <p><b>4. User Education Test:</b></p> <ul style="list-style-type: none"><li>✓ Test: Review the application for educational prompts or materials informing users of security best practices and types of personally identifiable information being processed.</li><li>✓ Expected Result: The application includes clear and accessible user education materials.</li></ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

#### 2.9.3.7 Insufficient Binary Protections

<b>PUID:</b> [SECM-CAT-IBP-001]
<b>Requirement description:</b> The mobile health application installation package must be digitally signed using FIPS 186-3 approved methods or equivalent. The app must ensure only signed and provisioned packages are installed, preventing unsigned patches or service packs. Additionally, a cryptographic hash value must be provided for administrator verification before installation.
<b>Source:</b> <ul style="list-style-type: none"><li>✓ SRG-APP-000516-MAPP-000078: Unless the MOS manages app signing, the mobile app installation package must be digitally signed in accordance with FIPS 186-3 approved methods [17].</li><li>✓ 7.1: Verify that the app is signed and provisioned with valid certificate [19].</li><li>✓ The application must have the capability to prevent the installation of patches, service packs, or application components without verification the software component has been digitally signed using a certificate that is recognized and approved by the organization.</li><li>✓ Design and configure the application to have the capability to prevent unsigned patches and packages from being installed.</li><li>✓ Provide a cryptographic hash value that can be verified by a system administrator prior to installation [15].</li></ul>
<b>Priority:</b> Not defined
<b>Rationale:</b> Digital signing ensures the integrity and authenticity of the application installation package, preventing unauthorized or malicious modifications. This measure is critical in safeguarding the app from tampered or rogue installations, maintaining trustworthiness in health applications that handle sensitive data.

<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<b>1. Digital Signature Verification Test</b>
✓ <b>Test:</b> Verify the app installation package is digitally signed using a certificate compliant with FIPS 186-3 standards or equivalent.
✓ <b>Expected result:</b> Installation is only permitted for signed packages, and unsigned packages are rejected.
<b>2. Provisioning Certificate Test</b>
✓ <b>Test:</b> Check that the app provisioning process validates certificates as trusted.
✓ <b>Expected result:</b> Only certificates issued by approved authorities are accepted during app provisioning.
<b>3. Patch Installation Test</b>
✓ <b>Test:</b> Attempt to install unsigned patches or service packs.
✓ <b>Expected result:</b> The app rejects installation of unsigned or unverified components.
<b>4. Hash Verification Test</b>
✓ <b>Test:</b> Verify the cryptographic hash value provided matches the app package content.
✓ <b>Expected result:</b> The system administrator successfully validates the hash prior to installation.
<b>5. Tampered Package Test</b>
✓ <b>Test:</b> Use a tampered app package to initiate installation.
✓ <b>Expected result:</b> The app prevents installation and flags the package as invalid.
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IBP-002]
<b>Requirement description:</b> The mobile health application must implement mechanisms to detect and handle interactions from modified or tampered app binaries. The app must verify its code integrity at runtime and react to integrity violations by preventing unauthorized modifications and safeguarding data.
<b>Source:</b>

<ul style="list-style-type: none"> <li>✓ Protection Against Tampered Versions: The system must ensure that interactions from modified or tampered app binaries are identified and handled securely to protect app functionality and data [18].</li> <li>✓ React in case of a code integrity violation preventing it being modified [25].</li> </ul>
<b>Priority:</b> Not defined
<b>Rationale:</b> Detecting and addressing tampered app binaries is essential to maintaining the security and integrity of health applications. Tampering can lead to unauthorized data access, compromised functionality, and breaches of sensitive health information.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Code Integrity Check Test</b> <ul style="list-style-type: none"> <li>✓ <b>Test:</b> Validate that the application performs runtime integrity checks of its binaries.</li> <li>✓ <b>Expected result:</b> The app detects tampering attempts and halts execution securely.</li> </ul> </li> <li><b>2. Reaction to Integrity Violation Test</b> <ul style="list-style-type: none"> <li>✓ <b>Test:</b> Simulate a scenario where app binaries are modified and executed.</li> <li>✓ <b>Expected result:</b> The app prevents access to sensitive data and terminates execution, logging the integrity violation.</li> </ul> </li> <li><b>3. Tampered Interaction Test</b> <ul style="list-style-type: none"> <li>✓ <b>Test:</b> Attempt to use the app with a modified binary to interact with its backend services.</li> <li>✓ <b>Expected result:</b> The app denies requests and flags interactions from tampered binaries.</li> </ul> </li> <li><b>4. Secure Data Handling Test</b> <ul style="list-style-type: none"> <li>✓ <b>Test:</b> Ensure sensitive data is not exposed or compromised during a detected code integrity violation.</li> <li>✓ <b>Expected result:</b> Sensitive information remains secure, and unauthorized access is blocked.</li> </ul> </li> <li><b>5. Logging and Notification Test</b> <ul style="list-style-type: none"> <li>✓ <b>Test:</b> Verify that tampering attempts are logged and optionally notify administrators.</li> <li>✓ <b>Expected result:</b> Tampering events are recorded in a secure log with actionable details.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda
José L Fernández-Alemán
Juan Manuel Carrillo-de-Gea
Joaquín Nicolás
08/01/2026

<b>PUID:</b> [SECM-CAT-IBP-003]
<b>Requirement description:</b> The mobile health application must ensure that all debugging symbols and verbose logging or debugging messages are removed from production builds. Native binaries should not include debugging symbols, and the app must not generate verbose error messages that expose sensitive information.
<b>Source:</b>
<ul style="list-style-type: none"><li>✓ 7.4: Verify that debugging code has been removed, and the app does not log verbose errors or debugging messages [19].</li><li>✓ 7.3: Verify that debugging symbols have been removed from native binaries [19].</li></ul>
<b>Priority:</b> Not defined
<b>Rationale:</b> Removing debugging symbols and verbose logs is critical to prevent attackers from gaining insights into the application's internal workings, which could be exploited to compromise sensitive health data or application functionality.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"><li><b>1. Debugging Symbol Removal Test</b><ul style="list-style-type: none"><li>✓ Test: Analyze the production build binaries for the presence of debugging symbols.</li><li>✓ Expected result: No debugging symbols are present in native binaries.</li></ul></li><li><b>2. Verbose Logging Test</b><ul style="list-style-type: none"><li>✓ Test: Execute the app and monitor log outputs for any verbose error or debugging messages.</li><li>✓ Expected result: No verbose logs or debugging messages are generated during app execution.</li></ul></li><li><b>3. Error Message Inspection Test</b><ul style="list-style-type: none"><li>✓ Test: Trigger errors and exceptions in the app and observe the error messages displayed.</li><li>✓ Expected result: Error messages are generic and do not disclose sensitive application details.</li></ul></li><li><b>4. Build Configuration Verification Test</b><ul style="list-style-type: none"><li>✓ Test: Review the build configuration to confirm that debugging options are disabled in production builds.</li><li>✓ Expected result: Debugging options, including verbose logging, are disabled in production builds.</li></ul></li><li><b>5. Static Analysis Test</b><ul style="list-style-type: none"><li>✓ Test: Perform static analysis on the application code to detect remnants of debugging-related functionality.</li><li>✓ Expected result: No debugging-related functionality or logging code remains in the production version.</li></ul></li></ol>

<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda
José L Fernández-Alemán
Juan Manuel Carrillo-de-Gea
Joaquín Nicolás
08/01/2026

<b>PUID:</b> [SECM-CAT-IBP-004]
<b>Requirement description:</b> The mobile health application must be built in release mode for production deployment. The build settings must ensure that the app is non-debuggable and optimized for security, with all debug-related functionalities disabled.
<b>Source:</b>
✓ 7.2: Verify that the app has been built in release mode, with settings appropriate for a release build (e.g. non-debuggable) [19].
<b>Priority:</b> Not defined
<b>Rationale:</b> Building the app in release mode ensures that unnecessary debugging functionalities are removed, reducing the risk of attackers exploiting debug features or extracting sensitive information.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis

**Validation criteria:**

- 1. Build Mode Verification Test**
  - ✓ Test: Inspect the application manifest and build configuration files to confirm that the android:debuggable attribute is set to false.
  - ✓ Expected result: The app is non-debuggable in the production build.
- 2. Release Mode Flag Test**
  - ✓ Test: Analyze the build output to ensure that the release mode flag is enabled in the production build configuration.
  - ✓ Expected result: Release mode settings are enabled, and debugging settings are absent.
- 3. Runtime Behavior Test**
  - ✓ Test: Attempt to attach a debugger to the running application.
  - ✓ Expected result: The debugger cannot attach, confirming the app is in non-debuggable mode.
- 4. Static Analysis Test**
  - ✓ Test: Perform a static analysis of the application binary to detect any residual debugging traces.
  - ✓ Expected result: No debugging traces are present in the application binary.
- 5. Signature Verification Test**
  - ✓ Test: Verify that the app is signed with a release key suitable for production use.
  - ✓ Expected result: The app is signed with the release key, confirming it is built for deployment.

**Requested by:** The organization**Responsible:** Developer**Configurable value:** Not described**Version history:** v1.0**Author and date:**

Carlos M. Mejía-Granda  
 José L Fernández-Alemán  
 Juan Manuel Carrillo-de-Gea  
 Joaquín Nicolás  
 08/01/2026

**PUID:** [SECM-CAT-IBP-005]

**Requirement description:** The mobile health application must implement robust error handling mechanisms to prevent error handling vulnerabilities. Proper return codes and exception handling must be implemented throughout the application to ensure secure and predictable behavior during runtime anomalies.

**Source:**

- ✓ V-222656: The application must not be subject to error handling vulnerabilities. Ensure proper return code and exception handling is implemented throughout the application [15].
- ✓ 7.5: Verify that the app catches and handles possible exceptions [19].

**Priority:** Not described

**Rationale:** Proper error handling ensures that exceptions and errors do not expose sensitive data or compromise the application's security. It mitigates risks such as unhandled exceptions leading to crashes, information leakage, or undefined application behavior.

**Number of Children:** 0

<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Exception Handling Test</b> <ul style="list-style-type: none"> <li>✓ Test: Simulate various runtime exceptions (e.g., null pointer access, network timeouts) and observe the application's behavior.</li> <li>✓ Expected result: The application catches all exceptions and displays generic error messages without exposing sensitive details.</li> </ul> </li> <li><b>2. Return Code Verification Test</b> <ul style="list-style-type: none"> <li>✓ Test: Trigger multiple API calls and inspect the return codes for consistency and proper error handling.</li> <li>✓ Expected result: All API responses include appropriate and documented return codes for errors.</li> </ul> </li> <li><b>3. Log Inspection Test</b> <ul style="list-style-type: none"> <li>✓ Test: Examine application logs for traces of unhandled exceptions or verbose stack traces.</li> <li>✓ Expected result: Logs do not expose sensitive information or full stack traces.</li> </ul> </li> <li><b>4. Crash Simulation Test</b> <ul style="list-style-type: none"> <li>✓ Test: Perform fuzz testing to feed unexpected inputs into the application and observe its behavior.</li> <li>✓ Expected result: The application gracefully handles unexpected inputs and does not crash or expose sensitive information.</li> </ul> </li> <li><b>5. Code Review Test</b> <ul style="list-style-type: none"> <li>✓ <b>Test:</b> Conduct a code review focusing on the implementation of try-catch blocks and error-handling routines.</li> <li>✓ <b>Expected result:</b> Error-handling routines are consistently implemented across all critical functions, and all exceptions are handled securely.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IBP-006]
---------------------------------

**Requirement description:** The mobile health application must prioritize the use of the Android SDK over native code to enhance security, portability, and maintainability. Where native code is necessary, memory corruption vulnerabilities such as buffer overflows must be mitigated through memory protection techniques like data execution prevention (DEP) and address space layout randomization (ASLR). Static analysis tools must be employed during the development process to detect and remediate memory management issues.

**Source:**

- ✓ Prefer Android SDK over Native Code: The application must prioritize using the Android SDK for development. Native code should be avoided unless absolutely necessary due to its complexity, lack of portability, and susceptibility to memory-corruption errors like buffer overflows [18]
- ✓ 12.13. Protect from memory corruptions in applications that are developed using a programming language which supports explicit memory management (e.g., Objective-C, C, C++). Perform static analysis for memory management vulnerabilities in the development process [16].
- ✓ SI-16 MEMORY PROTECTION
- ✓ Control: Implement the following controls to protect the system memory from unauthorized code execution: [Assignment: organization-defined controls].
- ✓ Discussion: Some adversaries launch attacks with the intent of executing code in non-executable regions of memory or in memory locations that are prohibited. Controls employed to protect memory include data execution prevention and address space layout randomization. Data execution prevention controls can either be hardware-enforced or software-enforced with hardware enforcement providing the greater strength of mechanism [11].

**Priority:** Not described

**Rationale:** The use of native code introduces increased risks of memory-corruption vulnerabilities and reduces portability across devices. Leveraging the Android SDK simplifies development, enhances security, and minimizes the attack surface. Memory protection mechanisms and rigorous static analysis further safeguard the application from unauthorized code execution.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

1. **Code Implementation Test**

- ✓ Test: Verify that native code usage is minimized, and the majority of the application relies on the Android SDK.
- ✓ Expected result: The application uses native code only in essential areas where SDK functionality is insufficient.

<p><b>2. Static Analysis Test</b></p> <ul style="list-style-type: none"> <li>✓ Test: Perform static analysis on the application's codebase to identify memory management vulnerabilities.</li> <li>✓ Expected result: No critical memory management issues, such as buffer overflows or dangling pointers, are detected.</li> </ul> <p><b>3. Memory Protection Test</b></p> <ul style="list-style-type: none"> <li>✓ Test: Confirm that memory protection mechanisms like DEP and ASLR are enabled during runtime.</li> <li>✓ Expected result: The application runtime enforces DEP and ASLR effectively.</li> </ul> <p><b>4. Dynamic Testing for Buffer Overflows</b></p> <ul style="list-style-type: none"> <li>✓ Test: Conduct dynamic testing with fuzzing tools to identify potential buffer overflow vulnerabilities.</li> <li>✓ Expected result: The application handles invalid or malicious inputs gracefully without memory corruption.</li> </ul> <p><b>5. Code Review Test</b></p> <ul style="list-style-type: none"> <li>✓ Test: Review the source code to ensure proper implementation of memory management routines and limited native code usage.</li> <li>✓ Expected result: All critical sections involving native code are secure and follow best practices for memory management.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<p><b>Author and date:</b></p> <p>Carlos M. Mejía-Granda  José L Fernández-Alemán  Juan Manuel Carrillo-de-Gea  Joaquín Nicolás  08/01/2026</p>

<b>PUID:</b> [SECM-CAT-IBP-007]
<p><b>Requirement description:</b> The mobile health application must ensure that all permissions are explicitly declared and managed according to the Android permission model, even when utilizing native code. Developers must adhere to Linux security practices, such as the Secure Programming Guidelines, to mitigate risks associated with memory handling, file operations, and processing external inputs in a native code environment.</p>
<p><b>Source:</b></p> <ul style="list-style-type: none"> <li>✓ Application Permissions: Native code must comply with the Android permission model. All permissions must be explicitly declared and managed within the app, even if the application primarily uses native code.</li> <li>✓ Linux Development Best Practices: Developers using native code must be familiar with Linux security practices, including Secure Programming Guidelines (e.g., Secure</li> </ul>

Programming HOWTO). These practices help mitigate risks when handling memory, files, and external inputs in a native code environment. [18]
<b>Priority:</b> Not described
<b>Rationale:</b> Explicitly managing permissions within the Android framework ensures transparency and prevents unauthorized access to sensitive data or device features. Adhering to secure programming practices in native code environments reduces vulnerabilities, such as improper memory management and file handling, which could compromise application security and patient data integrity.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Permission Declaration Test</b> <ul style="list-style-type: none"> <li>✓ <b>Test:</b> Verify that all permissions required by the application are explicitly declared in the <code>AndroidManifest.xml</code> file and are consistent with the app's functionalities.</li> <li>✓ <b>Expected result:</b> Permissions are correctly declared and aligned with the app's intended operations.</li> </ul> </li> <li><b>2. Permission Management Test</b> <ul style="list-style-type: none"> <li>✓ Test: Ensure that runtime permissions are requested and granted only when needed, following Android's runtime permission model.</li> <li>✓ Expected result: The app requests and manages permissions appropriately at runtime, avoiding unnecessary or unused permissions.</li> </ul> </li> <li><b>3. Linux Security Practice Implementation Test</b> <ul style="list-style-type: none"> <li>✓ Test: Evaluate the implementation of Linux security best practices in sections of the code that use native code (e.g., memory allocation, file handling).</li> <li>✓ Expected result: Native code complies with secure programming guidelines, such as preventing buffer overflows and securing file access.</li> </ul> </li> <li><b>4. Static Code Analysis Test</b> <ul style="list-style-type: none"> <li>✓ Test: Run static analysis tools to identify permission mismanagement and vulnerabilities in native code handling sensitive operations.</li> <li>✓ Expected result: No critical permission mismanagement or native code vulnerabilities are detected.</li> </ul> </li> <li><b>5. Dynamic Testing for Input Validation</b> <ul style="list-style-type: none"> <li>✓ Test: Conduct tests to ensure that all external inputs handled by native code are validated and sanitized.</li> <li>✓ Expected result: The application securely processes all external inputs without exposing vulnerabilities.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0

**Author and date:**

Carlos M. Mejía-Granda  
 José L Fernández-Alemán  
 Juan Manuel Carrillo-de-Gea  
 Joaquín Nicolás  
 08/01/2026

**PUID:** [SECM-CAT-IBP-008]

**Requirement description:** The mobile health application must ensure that the final build does not include source code, unreferenced code, or subroutines that are never invoked during operation. Exceptions are allowed only for software components and libraries from approved third-party products explicitly required for the application's functionality.

**Source:**

- ✓ SRG-APP-000141-MAPP-000031: The mobile app must not include source code, unreferenced code or subroutines that are never invoked during operation, except for software components and libraries from approved third-party products [17].

**Priority:** Not described

**Rationale:** Eliminating unused and unreferenced code reduces the attack surface, improves application performance, and minimizes the risk of vulnerabilities arising from outdated or unnecessary components. This ensures a secure, lean, and efficient application suitable for handling sensitive health data.

**Number of Children:** 0**Number of parents:** 0**Cycles:** 0**Number Audit:** 0**Child PUIDs:** Not described**Parent PUIDs:** Not described**Exclusion PUIDs:** Not described**Importance:** Not described**Current state:** To be determined**Verification method:** Demonstration/Analysis**Validation criteria:****1. Static Code Review Test**

- ✓ Test: Conduct a static code review to identify and remove unused or unreferenced code and subroutines in the application source.
- ✓ Expected result: No unused or unreferenced code or subroutines remain in the final build, except for required third-party components.

**2. Third-Party Library Audit Test**

- ✓ Test: Verify that all third-party libraries included in the final build are approved and necessary for the application's functionality.
- ✓ Expected result: All included libraries are vetted, relevant, and compliant with organizational policies.

**3. Dynamic Code Coverage Analysis Test**

- ✓ Test: Use code coverage tools during runtime testing to ensure only referenced and invoked code is executed.

<ul style="list-style-type: none"> <li>✓ Expected result: All code included in the final build is executed during dynamic testing, confirming its relevance.</li> </ul> <p><b>4. Build Process Inspection Test</b></p> <ul style="list-style-type: none"> <li>✓ Test: Validate the build process to ensure unused or unnecessary source code is excluded from the production build.</li> <li>✓ Expected result: The production build contains only necessary, actively used code.</li> </ul> <p><b>5. Penetration Testing for Legacy Code Test</b></p> <ul style="list-style-type: none"> <li>✓ Test: Conduct penetration tests to identify vulnerabilities stemming from any remaining legacy or unused code.</li> <li>✓ Expected result: No vulnerabilities arising from unused or legacy code are detected.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<p><b>Author and date:</b></p> <p>Carlos M. Mejía-Granda      José L Fernández-Alemán      Juan Manuel Carrillo-de-Gea      Joaquín Nicolás      08/01/2026</p>

<b>PUID:</b> [SECM-CAT-IBP-009]
<b>Requirement description:</b> The mobile health application must not load code from outside the APK package to prevent vulnerabilities such as code injection or tampering. Dynamically loaded code must comply with platform-specific and regulatory restrictions, ensuring strict version control, testing, and security verification.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ Avoid External Code Loading: The application must not load code from outside the APK package to prevent security vulnerabilities such as code injection and tampering. Loading external code increases the complexity of testing, version management, and security verification [18].</li> <li>✓ Compliance Considerations: Applications must adhere to any regulatory or platform-specific restrictions regarding dynamically loaded code. Environments that require strict application behavior verification might prohibit dynamic code loading altogether [18].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Avoiding external code loading minimizes the risk of malicious code execution, unauthorized modifications, and compliance violations. It ensures the application operates predictably, maintaining the integrity of its functionality and safeguarding sensitive health data.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described

<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Static Analysis for External Code Loading Test</b> <ul style="list-style-type: none"> <li>✓ <b>Test:</b> Perform a static analysis to detect and flag any instances of code loading from outside the APK package (e.g., using DexClassLoader or similar methods).</li> <li>✓ <b>Expected result:</b> No instances of external code loading are found in the application.</li> </ul> </li> <li><b>2. Runtime Behavior Inspection Test</b> <ul style="list-style-type: none"> <li>✓ <b>Test:</b> Monitor the application during runtime to ensure it does not dynamically load external code.</li> <li>✓ <b>Expected result:</b> The application executes without invoking external or dynamically loaded code.</li> </ul> </li> <li><b>3. Code Injection Simulation Test</b> <ul style="list-style-type: none"> <li>✓ <b>Test:</b> Attempt to load unauthorized or external code into the application during testing to identify potential vulnerabilities.</li> <li>✓ <b>Expected result:</b> The application prevents all attempts to load external or unauthorized code.</li> </ul> </li> <li><b>4. Penetration Testing for Code Tampering Test</b> <ul style="list-style-type: none"> <li>✓ <b>Test:</b> Conduct penetration tests to verify the application is resistant to tampering attempts aimed at exploiting dynamic code loading mechanisms.</li> <li>✓ <b>Expected result:</b> No tampering or unauthorized code execution is possible.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejia-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IBP-010]
<b>Requirement description:</b> The mobile health application must mitigate SQL injection vulnerabilities by using parameterized queries and prepared statements for all database operations. User input must be validated and sanitized to prevent the injection of unauthorized SQL commands. Concatenation of user data into SQL queries is strictly prohibited.
<b>Source:</b> <ul style="list-style-type: none"> <li>✓ V-222607: The application must not be vulnerable to SQL Injection. Modify the application and remove SQL injection vulnerabilities [15].</li> <li>✓ Preventing SQL Injection: All queries, updates, and deletions must be performed using parameterized query methods (query(), update(), delete()) to avoid SQL injection risks. Concatenation of user data into SQL statements is prohibited to prevent injection vulnerabilities [18].</li> </ul>

- ✓ Avoid simple exploits such as SQL injection [25].

Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')

The product constructs all or part of an SQL command using externally-influenced input from an upstream component, but it does not neutralize or incorrectly neutralizes special elements that could modify the intended SQL command when it is sent to a downstream component. It is related to SQL Injection [35].

- ✓ 12.12. Mitigate SQL injections, local file inclusion, JavaScript injections, XML injections. When dealing with dynamic queries (e.g., SQL queries with untrusted inputs) or Content-Providers ensure you are using parameterized queries. Always validate user provided inputs that will be used for file accessing purposes or as part of a dynamic code execution. Use a vetted framework for XML operations [16].
- ✓ Protection Against SQL Injection: The application must use parameterized queries for all database operations to prevent SQL injection attacks. Permissions for database access must be limited to read-only or write-only as appropriate, to minimize potential harm from SQL injection vulnerabilities [35].
- ✓ MASVS-CODE-4: The app validates and sanitizes all untrusted inputs: Apps have many data entry points including the UI, IPC, the network, the file system, etc. This incoming data might have been inadvertently modified by untrusted actors and may lead to bypass of critical security checks as well as classical injection attacks such as SQL injection, XSS or insecure deserialization. This control ensures that this data is treated as untrusted input and is properly verified and sanitized before it's used [20]
- ✓ Secure Coding Practices: 1. Follow secure coding practices, such as using parameterized queries and prepared statements to prevent SQL injection [3].

**Priority:** Not described

**Rationale:** Preventing SQL injection is critical to protecting sensitive health data stored in application databases. SQL injection exploits can lead to unauthorized data access, corruption, or deletion, undermining the application's integrity and confidentiality.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

1. **Static Code Analysis for SQL Injection Test**

- ✓ Test: Perform a static code review to identify instances of string concatenation in SQL queries.

<ul style="list-style-type: none"> <li>✓ Expected result: All database operations use parameterized queries or prepared statements without string concatenation.</li> </ul>
<b>2. Dynamic SQL Injection Simulation Test</b>
<ul style="list-style-type: none"> <li>✓ Test: Simulate SQL injection attacks on user input fields and API endpoints to assess the application's resistance.</li> <li>✓ Expected result: The application does not allow unauthorized access, data manipulation, or unintended query execution.</li> </ul>
<b>3. Database Permissions Audit Test</b>
<ul style="list-style-type: none"> <li>✓ Test: Verify that database user accounts used by the application have the least privileges necessary (e.g., read-only or write-only as appropriate).</li> <li>✓ Expected result: Database accounts follow the principle of least privilege to minimize the impact of potential SQL injection attacks.</li> </ul>
<b>4. Parameterized Query Functionality Test</b>
<ul style="list-style-type: none"> <li>✓ Test: Inspect and validate the implementation of parameterized queries in all database operations.</li> <li>✓ Expected result: Parameterized queries are implemented for all database interactions, and no vulnerabilities are present.</li> </ul>
<b>5. Secure Input Validation Test</b>
<ul style="list-style-type: none"> <li>✓ Test: Validate user input handling to ensure all inputs are sanitized and conform to expected formats before being processed in queries.</li> <li>✓ Expected result: Input validation prevents malicious or malformed inputs from being included in SQL statements.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IBP-011]
<b>Requirement description:</b> The mobile health application must mitigate command injection vulnerabilities by validating and sanitizing all user inputs before executing any system or OS commands. Input filtering must escape or neutralize special characters that could be exploited to alter the intended behavior of commands.
<b>Source:</b> <ul style="list-style-type: none"> <li>✓ V-222604: The application must protect from command injection. Modify the application so as to escape/sanitize special character input or configure the system to protect against command injection attacks based on application architecture [15].</li> <li>✓ CWE-78 Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')</li> </ul>

✓ The product constructs all or part of an OS command using externally-influenced input from an upstream component, but it does not neutralize or incorrectly neutralizes special elements that could modify the intended OS command when it is sent to a downstream component. It is related to command injection in OS. From a weakness standpoint, these variants represent distinct programmer errors. In the first variant, the programmer clearly intends that input from untrusted parties will be part of the arguments in the command to be executed. In the second variant, the programmer does not intend for the command to be accessible to any untrusted party, but the programmer probably has not accounted for alternate ways in which malicious attackers can provide input [35]

- ✓ 10.1. Filter user data passed to interpreters.[16].
- ✓ CWE-94: Improper Control of Generation of Code ('Code Injection')

The product constructs all or part of a code segment using externally-influenced input from an upstream component, but it does not neutralize or incorrectly neutralizes special elements that could modify the syntax or behavior of the intended code segment [35]

- ✓ V-222606: The application must validate all input. Design and configure the application to validate input prior to executing commands [15].
- ✓ CWE-77: Improper Neutralization of Special Elements used in a Command ('Command Injection')

The product constructs all or part of a command using externally-influenced input from an upstream component, but it does not neutralize or incorrectly neutralizes special elements that could modify the intended command when it is sent to a downstream component.

Command injection vulnerabilities typically occur when:

1. Data enters the application from an untrusted source.
2. The data is part of a string that is executed as a command by the application.
3. By executing the command, the application gives an attacker a privilege or capability that the attacker would not otherwise have.

Many protocols and products have their own custom command language. While OS or shell command strings are frequently discovered and targeted, developers may not realize that these other command languages might also be vulnerable to attacks. Command injection is a common problem with wrapper programs [35]

<b>Priority:</b> Not described
--------------------------------

**Rationale:** Command injection vulnerabilities allow attackers to execute unauthorized commands within the system, leading to potential data breaches, system compromise, or unauthorized access to sensitive resources. Proper validation and sanitization of inputs are critical to ensuring the application's security.

<b>Number of Children:</b> 0
------------------------------

<b>Number of parents:</b> 0
-----------------------------

<b>Cycles:</b> 0
------------------

<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
1. <b>Static Code Review Test</b>
✓ Test: Analyze the codebase to identify any instances where external input is used to construct OS commands.
✓ Expected result: No command execution relies on unvalidated or unsanitized user input.
2. <b>Command Injection Simulation Test</b>
✓ Test: Perform penetration testing to simulate command injection attacks using various payloads.
✓ Expected result: The application rejects malicious inputs and prevents unauthorized command execution.
3. <b>Input Filtering Verification Test</b>
✓ Test: Validate that all input fields sanitize special characters before processing or executing commands.
✓ Expected result: Special characters are escaped or neutralized, preventing unintended command modification.
4. <b>Dynamic Runtime Behavior Test</b>
✓ Test: Monitor the application's runtime behavior to ensure that commands are executed as intended and not influenced by external inputs.
✓ Expected result: The application executes commands securely without deviation from expected behavior.
5. <b>Interpreters and Code Injection Test</b>
✓ Test: Verify that the application filters user inputs passed to command interpreters or scripting engines.
✓ Expected result: The application properly validates and sanitizes inputs to prevent command or code injection vulnerabilities.
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

**PUID:** [SECM-CAT-IBP-012]

**Requirement description:** The mobile health application must implement validation and error-handling mechanisms to prevent null pointer dereference vulnerabilities, ensuring that all pointers are checked for null values before dereferencing.

**Source:**

- ✓ CWE-476: NULL Pointer Dereference: A NULL pointer dereference occurs when the application dereferences a pointer that it expects to be valid, but is NULL, typically causing a crash or exit. [35]

**Priority:** Not described

**Rationale:** Null pointer dereference vulnerabilities can cause application crashes, denial of service, or unexpected behavior, impacting the reliability and security of the application. Proper validation of pointers helps maintain system stability and prevents exploitation.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

**1. Static Code Analysis Test**

- ✓ Test: Use static analysis tools to scan the codebase for potential null pointer dereference issues.
- ✓ Expected result: The codebase includes proper null checks for all pointers before dereferencing.

**2. Code Review Test**

- ✓ Test: Conduct a manual review of the code to ensure that null pointer checks are implemented in critical sections.
- ✓ Expected result: No instances of pointers being dereferenced without prior null validation.

**3. Dynamic Testing for Null Values**

- ✓ Test: Simulate scenarios where pointers could potentially be null during runtime, such as uninitialized variables or failed resource allocation.
- ✓ Expected result: The application handles null pointers gracefully without crashing or exhibiting undefined behavior.

**4. Error Handling Verification Test**

- ✓ Test: Validate that the application logs appropriate errors or takes corrective action when a null pointer is encountered.
- ✓ Expected result: The application avoids crashes and logs null pointer-related issues for debugging purposes.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda

José L Fernández-Alemán  
 Juan Manuel Carrillo-de-Gea  
 Joaquín Nicolás  
 08/01/2026

**PUID:** [SECM-CAT-IBP-013]

**Requirement description:** The mobile health application must implement replay-resistant authentication mechanisms for both privileged and non-privileged accounts to prevent unauthorized access through replay attacks.

**Source:**

- ✓ V-222530: The application must implement replay-resistant authentication mechanisms for network access to privileged accounts. Design and configure the application to utilize replay-resistant mechanisms when authenticating privileged accounts [15].
- ✓ V-222531: The application must implement replay-resistant authentication mechanisms for network access to non-privileged accounts. Design and configure the application to utilize replay-resistant mechanisms when authenticating non-privileged accounts [15].

**Priority:** Not described

**Rationale:** Replay attacks can compromise account security by capturing and reusing authentication credentials. Replay-resistant mechanisms ensure secure authentication by using unique, time-sensitive tokens or nonces, safeguarding sensitive user data and application resources.

**Number of Children:** 0**Number of parents:** 0**Cycles:** 0**Number Audit:** 0**Child PUIDs:** Not described**Parent PUIDs:** Not described**Exclusion PUIDs:** Not described**Importance:** Not described**Current state:** To be determined**Verification method:** Demonstration/Analysis**Validation criteria:****1. Token Replay Resistance Test**

- ✓ Test: Attempt to reuse authentication tokens or credentials captured during a valid session.
- ✓ Expected result: The application rejects reused tokens and prevents authentication.

**2. Timestamp Validation Test**

- ✓ Test: Validate that the application uses time-sensitive tokens, such as those with a short expiration period, during authentication.
- ✓ Expected result: Tokens outside their valid time frame are denied, ensuring replay resistance.

**3. Session Hijacking Simulation Test**

- ✓ Test: Simulate a session hijacking attempt by intercepting and reusing session credentials.
- ✓ Expected result: The application detects and blocks the replay attempt, maintaining session integrity.

**4. Nonce Implementation Verification Test**

- ✓ Test: Verify that unique nonces are used during authentication to prevent repeated use of the same authentication data.

<ul style="list-style-type: none"> <li>✓ Expected result: Each authentication request has a unique nonce, and duplicate nonces are rejected.</li> </ul>
<b>5. Multi-Factor Replay Attack Test</b>
<ul style="list-style-type: none"> <li>✓ Test: Test the application's multi-factor authentication process for susceptibility to replay attacks.</li> <li>✓ Expected result: Multi-factor authentication mechanisms are resistant to replay attacks, ensuring secure account access.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IBP-014]
<b>Requirement description:</b> The mobile health application must implement rate-limiting mechanisms for authentication and authorization requests to prevent brute-force attacks. This includes account lockouts, CAPTCHAs, or additional user verification measures after multiple failed login attempts.
<b>Source:</b> <ul style="list-style-type: none"> <li>✓ Limit Authentication Rates: Implement rate-limiting for authentication and authorization requests to prevent brute-force attacks, balancing security with app usability [18].</li> <li>✓ 9.4.2 Secure log-on procedures: e) protect against brute force log-on attempts; [7]</li> <li>✓ 2.5. Introduce a bruteforce protection mechanism for the authentication controls (e.g., password change/reset). Consider enforcing account lockout for a specific duration, extended questions about the user, notifying the user through another channel and completely automated public Turing tests (captcha) in case of multiple failed attempts [16].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Brute-force attacks target authentication systems to gain unauthorized access by systematically trying combinations of credentials. Rate-limiting mechanisms and lockout policies mitigate these risks, enhancing application security while maintaining usability.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
1. Rate-Limiting Test

<ul style="list-style-type: none"> <li>✓ Test: Simulate multiple consecutive failed login attempts within a short time frame.</li> <li>✓ Expected result: The application enforces rate-limiting, rejecting additional requests after the threshold is exceeded.</li> </ul>
<b>2. Account Lockout Policy Test</b>
<ul style="list-style-type: none"> <li>✓ Test: Attempt more than the allowed number of failed logins on a user account.</li> <li>✓ Expected result: The account is temporarily locked, and the user is notified through a secure channel.</li> </ul>
<b>3. CAPTCHA Enforcement Test</b>
<ul style="list-style-type: none"> <li>✓ Test: Simulate multiple failed login attempts to trigger CAPTCHA or similar verification.</li> <li>✓ Expected result: The application requires a CAPTCHA to proceed with further login attempts.</li> </ul>
<b>4. Brute-Force Protection Notification Test</b>
<ul style="list-style-type: none"> <li>✓ Test: Monitor notifications sent to users when their account experiences multiple failed login attempts.</li> <li>✓ Expected result: The user receives timely and clear notifications of suspicious activity.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IBP-015]
<b>Requirement description:</b> The mobile health application must disable interaction events when the app interface is obscured by another layer to prevent tapjacking attacks. This ensures that users do not unknowingly interact with hidden views or unauthorized overlays.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ 12.4. Prevent interaction events when the application is obscured by another interface in the presentation layer in order to mitigate tapjacking attacks. By disabling the application interaction events, the possibility of a user interacting with a hidden view is eliminated. [16].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Tapjacking attacks trick users into interacting with hidden or malicious UI elements, potentially compromising sensitive data or app functionality. Disabling interactions when the app is obscured eliminates this risk, enhancing user security.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>

<p><b>1. Obscured View Test</b></p> <ul style="list-style-type: none"> <li>✓ Test: Overlay the application interface with another UI layer and attempt to interact with the app.</li> <li>✓ Expected result: The application disables interaction events when its UI is obscured.</li> </ul> <p><b>2. Tapjacking Simulation Test</b></p> <ul style="list-style-type: none"> <li>✓ Test: Use a test environment to simulate a tapjacking attack scenario.</li> <li>✓ Expected result: The app prevents unauthorized interactions with underlying UI components.</li> </ul> <p><b>3. User Notification Test</b></p> <ul style="list-style-type: none"> <li>✓ Test: Test if the app notifies the user when interaction is blocked due to obscuring layers.</li> <li>✓ Expected result: A clear and concise notification is displayed to the user indicating blocked interaction.</li> </ul> <p><b>4. Performance Validation Test</b></p> <ul style="list-style-type: none"> <li>✓ Test: Ensure the app functions normally without interruptions when no obscuring layers are present.</li> <li>✓ Expected result: Normal interactions with the app remain unaffected.</li> </ul> <p><b>5. Accessibility Impact Test</b></p> <ul style="list-style-type: none"> <li>✓ Test: Evaluate the impact of disabling interaction events on accessibility features.</li> <li>✓ Expected result: Accessibility functionalities are maintained without introducing usability challenges.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IBP-016]
<b>Requirement description:</b> The mobile health application must implement secure coding practices to prevent vulnerabilities such as buffer overflows, including the use of secure functions and careful handling of native code. Functions and code paths vulnerable to overflow attacks must be identified and replaced with secure alternatives.
<b>Source:</b> <ul style="list-style-type: none"> <li>✓ V-222612: The application must not be vulnerable to overflow attacks [15].</li> <li>✓ SRG-APP-000516-MAPP-000069: The mobile app must not call functions vulnerable to buffer overflows [17].</li> <li>✓ Use secure functions to prevent buffer overflow [25].</li> <li>✓ Handling Native Code: When using native code, the application must ensure that all data handling is performed carefully to prevent vulnerabilities such as buffer overflows, use-after-free errors, and off-by-one errors [18].</li> </ul>
<b>Priority:</b> Not described

<b>Rationale:</b> Buffer overflow vulnerabilities can lead to unauthorized access, application crashes, or arbitrary code execution, posing a significant security risk in mobile health applications. Proactively mitigating these vulnerabilities ensures data protection and application stability.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Static Code Analysis Test</b> <ul style="list-style-type: none"> <li>✓ Test: Perform a static code analysis to identify and verify the absence of unsafe functions (e.g., <code>strcpy</code>, <code>sprintf</code>) commonly associated with buffer overflows.</li> <li>✓ Expected result: No insecure functions or patterns related to buffer overflows are detected.</li> </ul> </li> <li><b>2. Dynamic Testing for Overflows</b> <ul style="list-style-type: none"> <li>✓ Test: Simulate scenarios involving large or malformed inputs to identify potential buffer overflows during runtime.</li> <li>✓ Expected result: The application gracefully handles all inputs without crashes or memory corruption.</li> </ul> </li> <li><b>3. Memory Management Test</b> <ul style="list-style-type: none"> <li>✓ Test: Evaluate the app's handling of memory allocation and deallocation to ensure no use-after-free or off-by-one errors occur.</li> <li>✓ Expected result: Memory is managed securely, preventing overflow vulnerabilities.</li> </ul> </li> <li><b>4. Function Replacement Verification</b> <ul style="list-style-type: none"> <li>✓ Test: Verify that all vulnerable functions have been replaced with secure alternatives (e.g., <code>strncpy</code>, <code>snprintf</code>).</li> <li>✓ Expected result: All potentially insecure functions are replaced or mitigated.</li> </ul> </li> <li><b>5. Third-Party Library Assessment</b> <ul style="list-style-type: none"> <li>✓ Test: Analyze all third-party libraries used in the app to ensure they do not introduce overflow vulnerabilities.</li> <li>✓ Expected result: All libraries are vetted for security and patched to prevent buffer overflows.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IBP-017]
<b>Requirement description:</b> The mobile health application must ensure that all calculations and operations involving integers are securely implemented to prevent integer overflows or

wraparounds. This includes validating input ranges and employing safe arithmetic operations to mitigate risks.

**Source:**

- ✓ CWE-190: Integer Overflow or Wraparound

The product performs a calculation that can produce an integer overflow or wraparound, when the logic assumes that the resulting value will always be larger than the original value.

This can introduce other weaknesses when the calculation is used for resource management or execution control [35].

**Priority:** Not described

**Rationale:** Integer overflows can lead to critical vulnerabilities, including incorrect resource allocation, logic errors, and potential execution of unauthorized operations. Secure handling of integer operations ensures application integrity and reliability.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:****1. Static Code Analysis for Integer Vulnerabilities**

- ✓ Test: Perform a static code analysis to identify potential integer overflow or wraparound vulnerabilities in the codebase.
- ✓ Expected result: No vulnerabilities related to integer operations are detected.

**2. Boundary Condition Testing**

- ✓ Test: Test integer inputs at their upper and lower boundaries to evaluate the app's behavior during extreme values.
- ✓ Expected result: The application handles boundary inputs securely without errors or unexpected behaviors.

**3. Safe Arithmetic Operation Validation**

- ✓ Test: Verify that all arithmetic operations involving integers use safe methods or are wrapped in overflow detection mechanisms.
- ✓ Expected result: All integer operations are securely implemented, with checks for overflow or wraparound.

**4. Dynamic Testing with Malformed Inputs**

- ✓ Test: Introduce intentionally malformed inputs to stress-test the app's integer handling mechanisms.
- ✓ Expected result: The application maintains stable and secure operation under stress conditions.

**5. Third-Party Library Audit**

- ✓ Test: Assess all third-party libraries used in the app to ensure they do not introduce integer overflow vulnerabilities.
- ✓ Expected result: All third-party libraries are verified as safe from integer overflow risks.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda  
 José L Fernández-Alemán  
 Juan Manuel Carrillo-de-Gea  
 Joaquín Nicolás  
 08/01/2026

**PUID: [SECM-CAT-IBP-018]**

**Requirement description:** The mobile health application must prevent vulnerabilities arising from race conditions by implementing proper synchronization mechanisms. All shared resources accessed by concurrent processes or threads must be protected to ensure consistency and integrity during execution.

**Source:**

- ✓ V-222567: The application must not be vulnerable to race conditions. Be aware of potential timing issues related to application programming calls when designing and building the application. Validate those variable values do not change while a switch event is occurring [15].
- ✓ CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition'): The product contains a code sequence that can run concurrently with other code, and the code sequence requires temporary, exclusive access to a shared resource, but a timing window exists in which the shared resource can be modified by another code sequence that is operating concurrently [35].

**Priority:** Not described

**Rationale:** Race conditions can lead to unpredictable behavior, security vulnerabilities, and data corruption by allowing concurrent processes to modify shared resources without proper synchronization. Secure programming practices mitigate these risks and ensure stable application performance.

**Number of Children:** 0**Number of parents:** 0**Cycles:** 0**Number Audit:** 0**Child PUIDs:** Not described**Parent PUIDs:** Not described**Exclusion PUIDs:** Not described**Importance:** Not described**Current state:** To be determined**Verification method:** Demonstration/Analysis**Validation criteria:****1. Code Review for Concurrency Issues**

- ✓ Test: Conduct a code review to identify potential race conditions, focusing on shared resource access and multithreaded operations.
- ✓ Expected result: No unprotected shared resource access or race condition vulnerabilities are detected in the codebase.

**2. Static Analysis for Thread-Safety**

- ✓ Test: Use static analysis tools to scan the application for improper synchronization or potential race conditions.

<ul style="list-style-type: none"> <li>✓ Expected result: No concurrency-related vulnerabilities are flagged by the analysis tools.</li> </ul> <p><b>3. Concurrency Stress Testing</b></p> <ul style="list-style-type: none"> <li>✓ Test: Simulate high levels of concurrent access to shared resources in a test environment to observe application behavior.</li> <li>✓ Expected result: The application operates securely and maintains data integrity under concurrent access.</li> </ul> <p><b>4. Dynamic Validation of Resource Locks</b></p> <ul style="list-style-type: none"> <li>✓ Test: Verify the implementation of locking mechanisms, such as mutexes or semaphores, during runtime.</li> <li>✓ Expected result: Shared resources are properly locked and protected, preventing simultaneous modification by multiple threads.</li> </ul> <p><b>5. Switch Event Timing Tests</b></p> <ul style="list-style-type: none"> <li>✓ Test: Monitor the application during switch events (e.g., context switching or thread execution) to ensure variable values remain consistent.</li> <li>✓ Expected result: No timing-related inconsistencies or data corruption occur during switch events.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IBP-019]
<b>Requirement description:</b> The mobile health application must implement strict bounds checking for all memory operations to prevent out-of-bounds reads and writes. This includes validating buffer sizes, performing pointer arithmetic safely, and avoiding memory access outside the allocated boundaries. Proper error handling and memory management practices must also be enforced to prevent memory corruption vulnerabilities.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ CWE-787 Out-of-bounds Write</li> </ul> <p>The product writes data past the end, or before the beginning, of the intended buffer. Typically, this can result in corruption of data, a crash, or code execution. The product may modify an index or perform pointer arithmetic that references a memory location that is outside of the boundaries of the buffer. A subsequent write operation then produces undefined or unexpected results.</p> <p><b>Memory Corruption:</b>            Often used to describe the consequences of writing to memory outside the bounds of a buffer, or to memory that is invalid, when the root cause is something other than a sequential copy of excessive data from a fixed starting location. This may include issues such as incorrect pointer arithmetic, accessing invalid pointers due to incomplete initialization or memory release, etc [35].</p> <ul style="list-style-type: none"> <li>✓ CWE-125: Out-of-bounds Read</li> </ul>

The product reads data past the end, or before the beginning, of the intended buffer.

Typically, this can allow attackers to read sensitive information from other memory locations or cause a crash. A crash can occur when the code reads a variable amount of data and assumes that a sentinel exists to stop the read operation, such as a NUL in a string. The expected sentinel might not be located in the out-of-bounds memory, causing excessive data to be read, leading to a segmentation fault or a buffer overflow. The product may modify an index or perform pointer arithmetic that references a memory location that is outside of the boundaries of the buffer. A subsequent read operation then produces undefined or unexpected results [35].

- ✓ CWE-119: Improper Restriction of Operations within the Bounds of a Memory Buffer

The product performs operations on a memory buffer, but it can read from or write to a memory location that is outside of the intended boundary of the buffer. As a result, an attacker may be able to execute arbitrary code, alter the intended control flow, read sensitive information, or cause the system to crash [35].

**Priority:** Not described

**Rationale:** Out-of-bounds memory operations, such as reads or writes, can lead to data corruption, crashes, unauthorized code execution, or leaks of sensitive information. Adhering to secure memory management practices ensures application stability, security, and user data integrity.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

#### 1. Static Analysis for Bounds Violations

- ✓ Test: Use static analysis tools to scan the application code for out-of-bounds memory access vulnerabilities, including CWE-787, CWE-125, and CWE-119 issues.
- ✓ Expected result: No instances of out-of-bounds reads or writes are detected by the analysis tools.

#### 2. Code Review for Memory Management

- ✓ Test: Conduct a thorough code review focusing on buffer size calculations, pointer arithmetic, and array indexing to identify potential vulnerabilities.
- ✓ Expected result: All memory operations are safely bounded and validated.

#### 3. Dynamic Analysis and Fuzz Testing

- ✓ Test: Perform fuzz testing by providing unexpected and oversized inputs to identify runtime memory handling issues.
- ✓ Expected result: The application handles oversized or malformed inputs gracefully without crashing or corrupting memory.

#### 4. Boundary Check Validation

- ✓ Test: Execute specific tests to validate that the application properly checks memory boundaries during buffer allocation, read, and write operations.
- ✓ Expected result: The application strictly adheres to buffer boundaries, preventing overflows or underflows.

#### 5. Memory Corruption Testing

<ul style="list-style-type: none"> <li>✓ Test: Simulate scenarios of incomplete pointer initialization or double-free errors to validate memory safety.</li> <li>✓ Expected result: The application correctly handles pointer initialization and memory release, avoiding crashes or undefined behavior</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IBP-020]
<b>Requirement description:</b> The mobile health application must leverage Android's security features, such as Address Space Layout Randomization (ASLR) and Data Execution Prevention (DEP), to protect against memory corruption vulnerabilities. Input validation and safe memory management practices must also be implemented to mitigate issues such as use-after-free errors and other memory handling flaws.
<b>Source:</b> <ul style="list-style-type: none"> <li>✓ Handling Native Code: The application must utilize Android's security technologies, including Address Space Layout Randomization (ASLR) and Data Execution Prevention (DEP), while still addressing the underlying input validation issues [18].</li> <li>✓ CWE-416 Use After Free: Avoid referencing memory after it has been freed can cause a program to crash, use unexpected values, or execute code.</li> </ul> <p>The use of previously-freed memory can have any number of adverse consequences, ranging from the corruption of valid data to the execution of arbitrary code, depending on the instantiation and timing of the flaw. The simplest way data corruption may occur involves the system's reuse of the freed memory. Use-after-free errors have two common and sometimes overlapping causes:</p> <ul style="list-style-type: none"> <li>○ Error conditions and other exceptional circumstances.</li> <li>○ Confusion over which part of the program is responsible for freeing the memory.</li> </ul> <p>In this scenario, the memory in question is allocated to another pointer validly at some point after it has been freed. The original pointer to the freed memory is used again and points to somewhere within the new allocation. As the data is changed, it corrupts the validly used memory; this induces undefined behavior in the process.</p> <p>If the newly allocated data happens to hold a class, in C++ for example, various function pointers may be scattered within the heap data. If one of these function pointers is overwritten with an address to valid shellcode, execution of arbitrary code can be achieved [35].</p>
<b>Priority:</b> Not described
<b>Rationale:</b> Memory corruption vulnerabilities, including use-after-free errors, can lead to crashes, unauthorized code execution, or unexpected application behavior. Leveraging Android's built-in security mechanisms and adhering to secure coding practices reduces the risk of exploitation and enhances application robustness.
<b>Number of Children:</b> 0

<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b> <ol style="list-style-type: none"> <li><b>1. Static Code Analysis for Memory Management</b> <ul style="list-style-type: none"> <li>✓ Test: Use static analysis tools to scan for potential memory management issues, such as use-after-free errors and improper memory handling.</li> <li>✓ Expected result: No memory management flaws, such as CWE-416, are detected.</li> </ul> </li> <li><b>2. Dynamic Memory Safety Testing</b> <ul style="list-style-type: none"> <li>✓ Test: Conduct dynamic testing with tools like AddressSanitizer to identify runtime memory issues, including invalid pointer references and improper memory deallocation.</li> <li>✓ Expected result: The application does not exhibit use-after-free or memory corruption vulnerabilities during execution.</li> </ul> </li> <li><b>3. Validation of ASLR and DEP Configuration</b> <ul style="list-style-type: none"> <li>✓ Test: Verify that the application utilizes Android's ASLR and DEP features by inspecting the binary with security configuration tools.</li> <li>✓ Expected result: ASLR and DEP are correctly enabled and enforced.</li> </ul> </li> <li><b>4. Input Validation Testing</b> <ul style="list-style-type: none"> <li>✓ Test: Provide malformed and oversized inputs to validate that input validation mechanisms prevent memory corruption vulnerabilities.</li> <li>✓ Expected result: The application correctly sanitizes and validates inputs, preventing memory handling issues.</li> </ul> </li> <li><b>5. Code Review for Responsibility and Ownership</b> <ul style="list-style-type: none"> <li>✓ Test: Conduct a code review to ensure clarity in memory ownership and deallocation responsibility to avoid confusion leading to use-after-free errors.</li> <li>✓ Expected result: Memory allocation and deallocation responsibilities are well-documented and implemented securely.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejia-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IBP-021]
<b>Requirement description:</b> The mobile health application must adhere to secure memory management best practices when using native code. This includes preventing vulnerabilities such as buffer overflows, use-after-free errors, and memory leaks through careful coding, testing, and adherence to platform-specific security guidelines.
<b>Source:</b>

✓ Memory Management Best Practices: When native code is necessary, developers must follow best practices for secure memory management to prevent common vulnerabilities such as buffer overflows, use-after-free errors, and memory leaks [18]
<b>Priority:</b> Not described
<b>Rationale:</b> Improper memory management can lead to severe security vulnerabilities, including crashes, unauthorized code execution, or data corruption. Following secure memory management practices ensures application stability and protects sensitive health information.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Static Analysis for Memory Vulnerabilities</b> <ul style="list-style-type: none"> <li>✓ Test: Use static analysis tools to scan for common memory vulnerabilities, such as buffer overflows and memory leaks.</li> <li>✓ Expected result: No vulnerabilities related to memory management are detected in the code.</li> </ul> </li> <li><b>2. Dynamic Testing for Runtime Memory Safety</b> <ul style="list-style-type: none"> <li>✓ Test: Execute the application using dynamic analysis tools, such as Valgrind or AddressSanitizer, to identify runtime memory issues, including use-after-free errors.</li> <li>✓ Expected result: The application does not exhibit runtime memory handling flaws.</li> </ul> </li> <li><b>3. Review of Memory Allocation and Deallocation</b> <ul style="list-style-type: none"> <li>✓ Test: Conduct a manual code review to ensure proper memory allocation and deallocation practices, avoiding dangling pointers or invalid memory access.</li> <li>✓ Expected result: All memory is allocated and deallocated securely, with no untracked or reused pointers.</li> </ul> </li> <li><b>4. Stress Testing Under Load</b> <ul style="list-style-type: none"> <li>✓ Test: Perform stress tests under high memory usage scenarios to check for potential memory leaks or improper memory handling.</li> <li>✓ Expected result: The application maintains stable memory usage and does not crash or leak memory under load.</li> </ul> </li> <li><b>5. Compliance with Platform-Specific Guidelines</b> <ul style="list-style-type: none"> <li>✓ Test: Validate the application against Android's best practices for native code and memory management as outlined in official documentation.</li> <li>✓ Expected result: The application adheres to all recommended platform-specific security guidelines for memory handling.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda
José L Fernández-Alemán
Juan Manuel Carrillo-de-Gea
Joaquín Nicolás

08/01/2026

<b>PUID:</b> [SECM-CAT-IBP-022]
<b>Requirement description:</b> The mobile health application must implement safeguards to prevent vulnerabilities related to integer arithmetic, such as integer overflows, underflows, or wraparounds. These issues can lead to incorrect calculations, resource mismanagement, or security vulnerabilities.
<b>Source:</b>
✓ Memory Management Best Practices: When native code is necessary, developers must follow best practices for secure memory management to prevent common vulnerabilities such as buffer overflows, use-after-free errors, and memory leaks [18]
<b>Priority:</b> Not described
<b>Rationale:</b> Integer arithmetic vulnerabilities can cause critical security and functional issues in mobile health applications. These include data corruption, crashes, and potential exploits, compromising the application's reliability and user safety.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<b>1. Static Analysis for Arithmetic Errors</b>
✓ Test: Use static analysis tools to identify potential integer arithmetic vulnerabilities, including overflows and underflows, in the application code.
✓ Expected result: No integer arithmetic vulnerabilities are detected.
<b>2. Boundary Condition Testing</b>
✓ Test: Test the application with maximum and minimum values for all integer operations, such as resource allocation, counters, and input fields.
✓ Expected result: The application handles edge cases gracefully without arithmetic errors.
<b>3. Code Review for Safe Arithmetic Practices</b>
✓ Test: Conduct a manual code review to verify the use of safe arithmetic functions and appropriate data types for calculations.
✓ Expected result: Code uses safe arithmetic operations and proper typecasting where necessary to prevent overflows or wraparounds.
<b>4. Dynamic Testing for Runtime Behavior</b>
✓ Test: Use runtime monitoring tools to simulate extreme conditions and observe integer arithmetic operations during execution.
✓ Expected result: The application operates correctly under stress without runtime arithmetic issues.
<b>5. Compliance with Secure Coding Standards</b>
✓ Test: Ensure adherence to secure coding standards, such as CERT C or OWASP recommendations for integer handling.

<ul style="list-style-type: none"> <li>✓ Expected result: The application complies with secure coding standards, avoiding integer arithmetic vulnerabilities.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IBP-023]
<b>Requirement description:</b> The mobile health application must avoid using API calls that enable bridging between dynamic code (e.g., JavaScript) and native code (e.g., Objective-C or Kotlin). This prevents potential security vulnerabilities where injection attacks in the dynamic code could lead to unauthorized execution of native code.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ 12.2. Avoid using API calls that provide bridging of dynamic code (e.g., JavaScript) with native code (e.g., Objective-C) since an injection in the dynamic code will lead to native code execution [16].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Bridging dynamic and native code increases the risk of security vulnerabilities, as malicious code injected into the dynamic layer can exploit the native execution environment. For health applications, this can compromise sensitive data and critical functionality, violating regulatory compliance and user trust.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Code Review for Bridging API Usage</b> <ul style="list-style-type: none"> <li>✓ Test: Perform a manual code review to identify and verify the absence of API calls that enable bridging between dynamic code and native code.</li> <li>✓ Expected result: No instances of bridging APIs, such as evaluateJavaScript, are present in the application code.</li> </ul> </li> <li><b>2. Static Analysis for Code Injection Vulnerabilities</b> <ul style="list-style-type: none"> <li>✓ Test: Use static analysis tools to detect potential injection vulnerabilities in the dynamic code that could lead to native code execution.</li> <li>✓ Expected result: No vulnerabilities indicating possible code injection are reported.</li> </ul> </li> <li><b>3. Dynamic Testing with Malicious Payloads</b></li> </ol>

- ✓ Test: Simulate malicious inputs in dynamic code (e.g., JavaScript) to observe if any unauthorized execution in the native layer occurs.
- ✓ Expected result: The application securely handles all dynamic code without affecting the native layer.

#### 4. Review of JavaScript and Native Code Interactions

- ✓ Test: Analyze all interactions between JavaScript and native code to ensure they are secure, controlled, and explicitly required for the application's functionality.
- ✓ Expected result: All interactions are justified, secured, and do not allow injection-based exploitation.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda  
 José L Fernández-Alemán  
 Juan Manuel Carrillo-de-Gea  
 Joaquín Nicolás  
 08/01/2026

**PUID:** [SECM-CAT-IBP-024]

**Requirement description:** The mobile health application must ensure that any dynamically loaded code inherits the same security permissions as the application and maintains user trust by adhering to the application's defined security policies. All dynamically loaded components must be secure, validated, and aligned with the permissions granted to the application.

**Source:**

- ✓ Permissions and Trust Model: If dynamically loaded code is required, it must be noted that such code will run with the same security permissions as the application itself. The user's trust in the application applies to any code executed within it, including dynamically loaded components. [18]

**Priority:** Not described

**Rationale:** Dynamically loaded code operates under the same security context as the main application, which means any vulnerabilities or malicious behavior in such code could exploit the application's permissions. For mobile health applications, this can lead to unauthorized access to sensitive data, breach of user trust, and regulatory violations.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

##### 1. Dynamic Code Review

- ✓ Test: Inspect the application for instances of dynamically loaded code and verify that all loaded components are secure, signed, and validated before execution.

<ul style="list-style-type: none"> <li>✓ Expected result: All dynamically loaded code is signed and verified as trusted by the application.</li> </ul>
<b>2. Security Policy Compliance</b>
<ul style="list-style-type: none"> <li>✓ Test: Confirm that all dynamically loaded components adhere to the application's security policies and permissions.</li> <li>✓ Expected result: Dynamically loaded components do not exceed or bypass the application's granted permissions.</li> </ul>
<b>3. Behavioral Testing Under Granted Permissions</b>
<ul style="list-style-type: none"> <li>✓ Test: Execute scenarios where dynamically loaded code interacts with sensitive components (e.g., APIs, files, or system features) to ensure that it operates strictly within the allowed permissions.</li> <li>✓ Expected result: Dynamically loaded code behaves as expected and does not attempt unauthorized actions.</li> </ul>
<b>4. User Trust Verification</b>
<ul style="list-style-type: none"> <li>✓ Test: Verify that the app clearly communicates the use of dynamically loaded components to the user in the privacy policy or terms of service, maintaining transparency.</li> <li>✓ Expected result: Documentation and user agreements explicitly mention the use and security of dynamically loaded code.</li> </ul>
<b>5. Malware Analysis and Penetration Testing</b>
<ul style="list-style-type: none"> <li>✓ Test: Conduct penetration testing to simulate attacks on dynamically loaded code, ensuring it does not introduce vulnerabilities.</li> <li>✓ Expected result: No exploitable vulnerabilities are identified in the handling or execution of dynamically loaded components.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IBP-025]
<b>Requirement description:</b> The mobile health application must ensure that dynamically executed code is loaded exclusively from secure, verified sources. Code from insecure sources, such as unencrypted network transmissions or world-writable locations like external storage, must not be loaded to mitigate risks of unauthorized access or malicious code execution.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ Source of Loaded Code: The application must only load dynamically executed code from secure, verified sources. It must not load code from insecure sources like unencrypted network transmissions or world-writable locations (e.g., external storage). [18]</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Loading code from insecure sources exposes mobile health applications to risks like unauthorized modifications, malicious injections, and potential exploitation. For healthcare

applications, such vulnerabilities can compromise sensitive health data and breach user trust, making it essential to restrict dynamically loaded code to verified and secure origins.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Secure Code Source Verification</b> <ul style="list-style-type: none"> <li>✓ Test: Analyze the application's codebase to identify the origins of dynamically loaded code and ensure they are secure and validated.</li> <li>✓ Expected result: All dynamically loaded code is sourced from encrypted, authenticated, and verified locations.</li> </ul> </li> <li><b>2. World-Writable Location Access Check</b> <ul style="list-style-type: none"> <li>✓ Test: Inspect the application to ensure it does not attempt to load executable code from world-writable locations (e.g., external storage).</li> <li>✓ Expected result: No executable code is sourced from world-writable or insecure locations.</li> </ul> </li> <li><b>3. Network Transmission Security</b> <ul style="list-style-type: none"> <li>✓ Test: Validate that any code loaded from a network location is transmitted over secure channels such as HTTPS and authenticated before execution.</li> <li>✓ Expected result: All dynamically loaded code transmitted over a network uses encrypted, secure communication protocols like HTTPS.</li> </ul> </li> <li><b>4. Penetration Testing for Insecure Loading</b> <ul style="list-style-type: none"> <li>✓ Test: Perform penetration testing to simulate attempts to inject malicious code into the application from insecure sources.</li> <li>✓ Expected result: The application does not execute code from untrusted or malicious sources.</li> </ul> </li> <li><b>5. Code Signature Validation</b> <ul style="list-style-type: none"> <li>✓ Test: Verify that dynamically loaded code components are signed and the signatures are validated before execution.</li> <li>✓ Expected result: Only signed and validated code is executed within the application.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejia-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IBP-026]
---------------------------------

**Requirement description:** The mobile health application must ensure that any dynamically loaded code is transmitted over secure, encrypted channels (e.g., HTTPS) and stored in tamper-proof locations to prevent unauthorized access or modification by other applications or users.

**Source:**

- ✓ Secure Transmission and Storage: If code must be downloaded, ensure that it is transmitted over secure, encrypted channels (e.g., HTTPS) and stored in locations that are protected from tampering by other applications or users. [18].

**Priority:** Not described

**Rationale:** Insecure transmission and storage of dynamically loaded code can expose mobile health applications to risks such as tampering, unauthorized access, or the introduction of malicious code. Ensuring secure channels and protected storage locations mitigates these risks, safeguarding sensitive health data and maintaining application integrity.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

**1. Secure Transmission Validation**

- ✓ Test: Inspect the application to ensure that all dynamically loaded code is transmitted over secure channels like HTTPS.
- ✓ Expected result: All code downloads use encrypted protocols (e.g., HTTPS), and no unencrypted transmission occurs.

**2. Tamper-Proof Storage Check**

- ✓ Test: Validate that downloaded code is stored in secure locations, such as app-specific directories, with appropriate file permissions.
- ✓ Expected result: Dynamically loaded code is stored in locations inaccessible to unauthorized applications or users.

**3. Integrity Verification Mechanism**

- ✓ Test: Verify that the application checks the integrity of dynamically loaded code using mechanisms like checksums or digital signatures.
- ✓ Expected result: Code integrity is validated before execution, ensuring it matches the expected secure version.

**4. Penetration Testing for Code Tampering**

- ✓ Test: Perform penetration testing to attempt modification of stored dynamically loaded code.
- ✓ Expected result: The application detects and prevents the execution of tampered or unauthorized code.

**5. Code Source Authentication**

- ✓ Test: Confirm that dynamically loaded code is sourced from authenticated and trusted servers.
- ✓ Expected result: All dynamically loaded code is downloaded only from verified and trusted endpoints.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0**Author and date:**

Carlos M. Mejía-Granda  
 José L Fernández-Alemán  
 Juan Manuel Carrillo-de-Gea  
 Joaquín Nicolás  
 08/01/2026

**PUID:** [SECM-CAT-IBP-027]

**Requirement description:** The mobile health application must package dynamically loaded modules directly within the APK whenever feasible. These modules should be implemented as native libraries or classes loaded using DexClassLoader to prevent tampering or alteration by external applications.

**Source:**

- ✓ Internal Code Modules: Whenever possible, dynamically loaded modules must be packaged directly within the APK to prevent tampering. Modules should be either native libraries or classes loaded using DexClassLoader, ensuring they cannot be altered by external applications. [18].

**Priority:** Not described

**Rationale:** Dynamically loaded modules sourced externally increase the risk of tampering and unauthorized modification, which could compromise sensitive health data and application functionality. Packaging these modules within the APK ensures their integrity and reduces the risk of external interference.

**Number of Children:** 0**Number of parents:** 0**Cycles:** 0**Number Audit:** 0**Child PUIDs:** Not described**Parent PUIDs:** Not described**Exclusion PUIDs:** Not described**Importance:** Not described**Current state:** To be determined**Verification method:** Demonstration/Analysis**Validation criteria:****1. APK Packaging Inspection**

- ✓ **Test:** Inspect the APK to confirm that dynamically loaded modules are included within the package.
- ✓ **Expected result:** All dynamically loaded modules are contained within the APK file, and no external dependencies are required post-installation.

**2. Module Loading Mechanism Verification**

- ✓ **Test:** Verify that DexClassLoader or equivalent secure mechanisms are used to load dynamically included modules.
- ✓ **Expected result:** Modules are securely loaded, and the application does not utilize insecure loading mechanisms.

**3. Tampering Protection Test**

- ✓ **Test:** Attempt to modify dynamically loaded modules within the APK and observe application behavior.

- ✓ Expected result: The application detects tampering and prevents the execution of altered modules.

#### 4. Code Integrity Validation

- ✓ Test: Confirm that integrity checks (e.g., digital signatures) are applied to verify the authenticity of dynamically loaded modules.
- ✓ Expected result: The application validates module integrity before execution and rejects any corrupted or altered modules.

#### 5. Source Code Review

- ✓ Test: Conduct a review of the source code to ensure all dynamically loaded modules are appropriately declared and managed.
- ✓ Expected result: No dynamically loaded modules are sourced externally without explicit and secure handling mechanisms.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda  
José L Fernández-Alemán  
Juan Manuel Carrillo-de-Gea  
Joaquín Nicolás  
08/01/2026

**PUID:** [SECM-CAT-IBP-028]

**Requirement description:** The mobile health application must ensure that no sensitive data, such as location or other private information, is unintentionally transferred between the mobile device and web-server backends or other external interfaces. This includes data embedded within file metadata.

**Source:**

- ✓ 5.1. Carry out a specific check of your code for sensitive data unintentionally transferred between the mobile device and web-server back-ends and other external interfaces - (e.g., is location or other information transferred within file metadata) [16]

**Priority:** Not described

**Rationale:** Unintentional transfer of sensitive data can expose users to privacy risks, including unauthorized access to location information or personal data stored within file metadata. Ensuring secure data transmission protects patient confidentiality and complies with healthcare data protection standards.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:****1. Metadata Transmission Check**

- ✓ Test: Upload a sample file to the backend and inspect the transmitted data for sensitive metadata such as location or user details.
- ✓ Expected result: Metadata containing sensitive information is removed before transmission.

**2. Code Review for Data Sanitization**

- ✓ Test: Review the application code to ensure all file metadata is sanitized or stripped before transmission.
- ✓ Expected result: Proper sanitization processes are implemented to remove metadata before transmission.

**3. Transmission Testing for External Interfaces**

- ✓ Test: Interact with external interfaces (e.g., third-party APIs) and monitor for any unintended data leakage.
- ✓ Expected result: No sensitive data is unintentionally sent to external interfaces.

**4. Dynamic Security Testing**

- ✓ Test: Perform penetration testing to simulate malicious interception of transmitted data to ensure sensitive metadata is not exposed.
- ✓ Expected result: Sensitive data remains protected, and metadata is not transmitted over insecure channels.

**Requested by:** The organization**Responsible:** Developer**Configurable value:** Not described**Version history:** v1.0**Author and date:**

Carlos M. Mejía-Granda  
 José L Fernández-Alemán  
 Juan Manuel Carrillo-de-Gea  
 Joaquín Nicolás  
 08/01/2026

**PUID:** [SECM-CAT-IBP-029]

**Requirement description:** The mobile health application must implement robust anti-static analysis mechanisms, including code obfuscation, to impede reverse engineering. This includes using obfuscation tools, compiling natively when feasible, and ensuring that obfuscation balances complexity with operational correctness of dependent features.

**Source:**

- ✓ MASVS-RESILIENCE-3: The app implements anti-static analysis mechanisms: Understanding the internals of an app is typically the first step towards tampering with it (either dynamically, or statically). This control tries to impede comprehension by making it as difficult as possible to figure out how an app works using static analysis [20].
- ✓ Avoid reverse engineering of the application, "obfuscating" the app code [25].
- ✓ Binary code obfuscation: 1. The system must allow the application's binary code to be obfuscated, making it incomprehensible to prevent reverse engineering. Obfuscation tools

should be used, and when possible, applications should be compiled natively or use interpreters or nested virtual machines [3].
✓ Balance in code obfuscation: 1. Obfuscation must strike a balance between code complexity and robustness against reverse engineering, ensuring that libraries and features dependent on certain strings or symbols in the code operate correctly [3].
<b>Priority:</b> Not described
<b>Rationale:</b> Preventing reverse engineering enhances application resilience by making it difficult for attackers to understand its internal workings. This protects sensitive logic, API keys, and intellectual property, while maintaining the functionality of libraries and dependencies critical to the app's operation.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Code Obfuscation Verification</b> <ul style="list-style-type: none"> <li>✓ Test: Analyze the application's compiled binary to ensure it is obfuscated using industry-standard tools (e.g., ProGuard, DexGuard).</li> <li>✓ Expected result: Code is obfuscated, making it challenging to decompile into a human-readable format.</li> </ul> </li> <li><b>2. Static Analysis Prevention Test</b> <ul style="list-style-type: none"> <li>✓ Test: Perform static analysis using reverse engineering tools (e.g., JADX, IDA Pro) to evaluate the difficulty of extracting meaningful code structures.</li> <li>✓ Expected result: Static analysis is significantly hindered, with no clear extraction of sensitive logic or functionality.</li> </ul> </li> <li><b>3. Dependency Compatibility Check</b> <ul style="list-style-type: none"> <li>✓ Test: Verify that code obfuscation does not interfere with the functionality of libraries or features reliant on specific symbols or strings.</li> <li>✓ Expected result: All dependent features and libraries operate correctly without errors.</li> </ul> </li> <li><b>4. Tampering Resilience Assessment</b> <ul style="list-style-type: none"> <li>✓ Test: Attempt to modify or inject malicious code into the obfuscated binary.</li> <li>✓ Expected result: Tampering attempts fail due to the complexity introduced by obfuscation.</li> </ul> </li> <li><b>5. Performance Evaluation Post-Obfuscation</b> <ul style="list-style-type: none"> <li>✓ Test: Test the app's performance after applying obfuscation to ensure minimal impact on speed and functionality.</li> <li>✓ Expected result: Obfuscation does not degrade app performance or cause functional issues.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described

**Version history:** v1.0**Author and date:**

Carlos M. Mejía-Granda  
 José L Fernández-Alemán  
 Juan Manuel Carrillo-de-Gea  
 Joaquín Nicolás  
 08/01/2026

**PUID:** [SECM-CAT-IBP-030]

**Requirement description:** The mobile health application must ensure that memory in unmanaged code is securely allocated, used, and freed to prevent vulnerabilities such as memory leaks, use-after-free errors, and buffer overflows.

**Source:**

- ✓ 7.7: Verify that in unmanaged code, memory is allocated, freed and used securely [19].

**Priority:** Not described

**Rationale:** Proper memory management is essential in unmanaged code to ensure application stability and security. Incorrect memory handling can lead to critical vulnerabilities such as crashes, data corruption, or exploitation by attackers, compromising the application's reliability and user data integrity.

**Number of Children:** 0**Number of parents:** 0**Cycles:** 0**Number Audit:** 0**Child PUIDs:** Not described**Parent PUIDs:** Not described**Exclusion PUIDs:** Not described**Importance:** Not described**Current state:** To be determined**Verification method:** Demonstration/Analysis**Validation criteria:****1. Secure Memory Allocation Check**

- ✓ Test: Verify that all memory allocations in unmanaged code are performed securely using platform-recommended APIs.
- ✓ Expected result: Memory is allocated without exposing the application to overflow or underflow vulnerabilities.

**2. Memory Freeing Test**

- ✓ Test: Confirm that all allocated memory is properly freed when no longer needed.
- ✓ Expected result: Memory leaks are avoided by ensuring all memory is released during app execution or upon termination.

**3. Use-After-Free Validation**

- ✓ Test: Use automated tools to detect potential use-after-free errors during runtime.
- ✓ Expected result: No instances of memory being accessed after it has been freed.

**4. Buffer Overflow Protection**

- ✓ Test: Perform static and dynamic analysis to detect and prevent buffer overflows in unmanaged code sections.
- ✓ Expected result: No buffer overflows are detected, and bounds checking is consistently applied.

**5. Performance Testing Post-Validation**

- ✓ Test: Evaluate the application's performance to ensure secure memory handling does not introduce latency or excessive resource consumption.

✓ Expected result: Application maintains optimal performance with secure memory management practices.
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IBP-031]
<b>Requirement description:</b> The mobile health application must integrate with the Play Integrity API to ensure that interactions and server requests originate from a legitimate, untampered app binary running on a genuine Android device.
<b>Source:</b>
✓ Play Integrity API Integration: The system must integrate with the Play Integrity API to verify that interactions and server requests are originating from a legitimate app binary running on a genuine Android device [18].
<b>Priority:</b> Not described
<b>Rationale:</b> Integration with the Play Integrity API enhances application security by verifying the authenticity of the app and device, protecting against tampering, fraudulent activity, and interactions from non-legitimate sources. This is critical for safeguarding sensitive health information and ensuring compliance with security standards.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. API Integration Validation</b> <ul style="list-style-type: none"> <li>✓ Test: Verify that the Play Integrity API is properly integrated into the application.</li> <li>✓ Expected result: The API returns expected responses during legitimate interactions and blocks tampered or non-compliant requests.</li> </ul> </li> <li><b>2. Server Request Verification</b> <ul style="list-style-type: none"> <li>✓ Test: Test server-side logic to validate the tokens returned by the Play Integrity API.</li> <li>✓ Expected result: Server accepts requests only from legitimate app binaries on authentic devices.</li> </ul> </li> <li><b>3. Tampered Binary Testing</b> <ul style="list-style-type: none"> <li>✓ Test: Attempt to interact with the app using a tampered or repackaged binary.</li> <li>✓ Expected result: The app and server detect the tampering and block further interactions.</li> </ul> </li> <li><b>4. Device Authenticity Check</b></li> </ol>

- ✓ Test: Simulate interactions from a rooted or emulated device.
- ✓ Expected result: The app detects the insecure environment and denies access to sensitive functionalities.

### 5. End-to-End Integration Test

- ✓ Test: Perform an end-to-end test to validate those interactions between the app, Play Integrity API, and server function seamlessly.
- ✓ Expected result: Legitimate app binaries and devices successfully complete requests, while unauthorized entities are blocked.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda  
José L Fernández-Alemán  
Juan Manuel Carrillo-de-Gea  
Joaquín Nicolás  
08/01/2026

**PUID:** [SECM-CAT-IBP-032]

**Requirement description:** The mobile health application must initialize all parameter values upon startup to ensure predictable behavior, prevent unintentional data leakage, and avoid reliance on uninitialized or potentially unsafe variables.

**Source:**

- ✓ SRG-APP-000516-MAPP-000073: The mobile app must initialize all parameter values on startup. [17].

**Priority:** Not described

**Rationale:** Proper initialization of parameter values at startup enhances the application's security posture by mitigating risks associated with undefined behaviors, memory corruption, or unintended interactions. This practice is critical in applications handling sensitive health data to maintain data integrity and operational reliability.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

#### 1. Initialization Check

- ✓ Test: Inspect the application code to verify that all parameters are initialized during the app startup sequence.
- ✓ Expected result: No uninitialized variables are found in the application's codebase.

#### 2. Dynamic Execution Validation

- ✓ Test: Run the application in a dynamic analysis environment and monitor for usage of uninitialized parameters.

<ul style="list-style-type: none"> <li>✓ Expected result: All parameters show defined and expected values during runtime.</li> </ul>
<b>3. Startup Behavior Test</b>
<ul style="list-style-type: none"> <li>✓ Test: Simulate a range of startup scenarios, including low memory conditions, and verify that parameters are properly initialized.</li> <li>✓ Expected result: Application initializes all parameters and behaves predictably without crashing or showing undefined behavior.</li> </ul>
<b>4. Security Testing</b>
<ul style="list-style-type: none"> <li>✓ Test: Perform penetration testing to determine if uninitialized parameters can be manipulated to cause unexpected behavior.</li> <li>✓ Expected result: No vulnerabilities related to uninitialized parameters are exploitable.</li> </ul>
<b>5. Code Quality Review</b>
<ul style="list-style-type: none"> <li>✓ Test: Perform a static code analysis using tools to detect uninitialized variables in the application's source code.</li> <li>✓ Expected result: The analysis confirms that all parameter values are explicitly initialized during app startup.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IBP-033]
<b>Requirement description:</b> The mobile health app must verify integrity at startup to detect tampering or redistribution, alert stakeholders, and mitigate risks, such as removing unauthorized copies. Annual testing of initialization, shutdown, and abort scenarios must ensure system security.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ Protection against redistribution with malicious code: 1. The system should perform integrity checks during application startup to detect redistribution or malicious modifications of the binary. If a violation is found, it must automatically notify the appropriate parties to remove unauthorized copies of the app from app stores [3].</li> <li>✓ V-222647: Test procedures must be created and at least annually executed to ensure system initialization, shutdown, and aborts are configured to verify the system remains in a secure state [15].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Integrity checks are critical to prevent unauthorized redistribution or tampering with the app binary, which could compromise sensitive health data or system functionality. Regular testing of initialization and shutdown processes ensures the system remains secure during critical states.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described

<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Startup Integrity Validation</b> <ul style="list-style-type: none"> <li>✓ Test: Simulate an application startup with modified binaries and observe the integrity check process.</li> <li>✓ Expected result: The application detects the tampered binary and notifies relevant parties without executing unauthorized code.</li> </ul> </li> <li><b>2. Notification System Test</b> <ul style="list-style-type: none"> <li>✓ Test: Validate the application's ability to automatically send notifications upon detecting malicious modifications.</li> <li>✓ Expected result: Notifications are correctly triggered and delivered to predefined stakeholders (e.g., administrators, app store teams).</li> </ul> </li> <li><b>3. Dynamic Analysis Test</b> <ul style="list-style-type: none"> <li>✓ Test: Conduct dynamic analysis to verify the integrity check mechanism under various attack scenarios (e.g., tampered or injected binaries).</li> <li>✓ Expected result: The application successfully identifies and handles all integrity violations without compromising functionality.</li> </ul> </li> <li><b>4. App Store Integration Check</b> <ul style="list-style-type: none"> <li>✓ Test: Simulate notification workflows to app stores for removing unauthorized app copies.</li> <li>✓ Expected result: Notifications are correctly formatted and received by app stores, ensuring prompt removal of compromised copies.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IBP-034]
<b>Requirement description:</b> The mobile health application system must evaluate application binaries to identify critical content or high popularity that may necessitate binary protection. For applications requiring such protection, a thorough threat modeling analysis must be conducted to identify relevant risks and their potential financial impact.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ Critical content evaluation in binaries: 1. The system must assess whether the binary of each application contains critical content or if its popularity requires binary protection. If so, a</li> </ul>

threat modeling analysis should be performed to identify the most relevant risks and their potential financial impact [3].
<b>Priority:</b> Not described
<b>Rationale:</b> Critical content or highly popular applications are prime targets for attacks such as reverse engineering or tampering. Assessing these risks and implementing appropriate protections can mitigate potential financial and security impacts.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Binary Content Evaluation Test</b> <ul style="list-style-type: none"> <li>✓ Test: Analyze application binaries to determine if they contain critical content or are highly popular.</li> <li>✓ Expected result: The evaluation identifies whether the binary requires protection based on predefined criteria.</li> </ul> </li> <li><b>2. Threat Modeling Analysis</b> <ul style="list-style-type: none"> <li>✓ Test: Perform a threat modeling analysis for binaries flagged as requiring protection.</li> <li>✓ Expected result: The analysis identifies key risks and outlines their potential financial impacts.</li> </ul> </li> <li><b>3. Protection Plan Verification</b> <ul style="list-style-type: none"> <li>✓ Test: Review the implemented protection measures for identified binaries, such as obfuscation, anti-tampering, or encryption.</li> <li>✓ Expected result: Protections align with the risks identified in the threat modeling analysis.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

### 2.9.3.8 Security Misconfiguration

<b>PUID:</b> [SECM-CAT-SMC-001]
---------------------------------

**Requirement description:** The system must enforce secure default configurations for assets and software, minimizing permissions and restricting sensitive access.

**Source:**

- ✓ CIS Critical Security Control 4: Secure Configuration of Enterprise Assets and Software: Establish and maintain the secure configuration of enterprise assets (end-user devices, including portable and mobile; network devices; non-computing/IoT devices; and servers) and software (operating systems and applications) [14]
- ✓ 8.9 Configuration management: Configurations, including security configurations, of hardware, software, services and networks should be established, documented, implemented, monitored and reviewed [6].
- ✓ CWE-276: Incorrect Default Permissions
- ✓ The product performs an authorization check when an actor attempts to access a resource or perform an action, but it does not correctly perform the check. This allows attackers to bypass intended access restrictions [35].
- ✓ Secure default configurations: 1. Ensure that default settings and configurations are properly secured and do not expose sensitive information or provide unnecessary permissions [3].
- ✓ Test ID 20: A mobile device's configuration goes out of compliance while logged in [37].

**Priority:** Not described

**Rationale:** Secure configurations reduce risks of unauthorized access and exploitation due to default settings or misconfigurations.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

**1. Configuration Review Test:**

- ✓ Test: Compare default configurations against security standards.
- ✓ Expected result: Default settings meet secure baseline requirements.

**2. Permission Validation Test:**

- ✓ Test: Check permissions for resources.
- ✓ Expected result: Permissions are restricted appropriately, with no unintended access.

**3. Change Monitoring Test:**

- ✓ Test: Test monitoring and alerts for configuration changes.
- ✓ Expected result: Changes are detected and logged.

**4. Configuration Drift Analysis:**

- ✓ Test: Simulate unauthorized configuration changes.
- ✓ Expected result: Drift is detected and prompts remediation.

**5. Compliance Verification Test:**

- ✓ Test: Check configurations against compliance benchmarks.

✓ Expected result: System meets compliance requirements.
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-SMC-002]
<b>Requirement description:</b> The system must ensure all security controls are centrally implemented and configuration settings are documented and monitored.
<b>Source:</b> <ul style="list-style-type: none"> <li>✓ 1.9: Verify that all security controls have a centralized implementation [19].</li> <li>✓ CM-6 CONFIGURATION SETTINGS</li> </ul> <p>Control:</p> <ul style="list-style-type: none"> <li>a. Establish and document configuration settings for components employed within the system that reflect the most restrictive mode consistent with operational requirements using [Assignment: organization-defined common secure configurations];</li> <li>d. Monitor and control changes to the configuration settings in accordance with organizational policies and procedures.</li> </ul> <p>Discussion: Configuration settings are the parameters that can be changed in the hardware, software, or firmware components of the system that affect the security and privacy posture or functionality of the system. Information technology products for which configuration settings can be defined include mainframe computers, servers, workstations, operating systems, mobile devices, input/output devices, protocols, and applications. Parameters that impact the security posture of systems include registry settings; account, file, or directory permission settings; and settings for functions, protocols, ports, services, and remote connections [11].</p>
<b>Priority:</b> Not described
<b>Rationale:</b> Centralized security control implementation enhances consistency, reduces errors, and ensures compliance with organizational policies.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined

<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li>1. <b>Centralized Control Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Verify that all security controls are managed through a centralized mechanism.</li> <li>✓ Expected result: Security controls are consistently implemented and managed centrally.</li> </ul> </li> <li>2. <b>Configuration Review Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Review and document all configuration settings for security impact.</li> <li>✓ Expected result: Configuration settings are secure and reflect restrictive operational requirements.</li> </ul> </li> <li>3. <b>Change Control Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Simulate configuration changes and validate the monitoring process.</li> <li>✓ Expected result: Changes are logged, monitored, and appropriately controlled.</li> </ul> </li> <li>4. <b>Policy Compliance Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Verify compliance of configuration settings with security and privacy policies.</li> <li>✓ Expected result: All settings align with organizational policies.</li> </ul> </li> <li>5. <b>Security Impact Analysis Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Assess the impact of configuration settings on system security.</li> <li>✓ Expected result: All settings maintain the desired security posture.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-SMC-003]
<b>Requirement description:</b> The system must implement protections against Denial-of-Service (DoS) attacks, including rate limiting, throttling, and QoS technologies.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ V-222667: Protections against DoS attacks must be implemented. Implement mitigations from the threat model for DOS attacks [15].</li> <li>✓ 5.7. Employ rate limiting and throttling on a per-user/IP basis (if user identification is available) to reduce the risk from denial of service (DoS) attack [16].</li> <li>✓ SC-5 DENIAL-OF-SERVICE PROTECTION</li> <li>✓ Control:           <ul style="list-style-type: none"> <li>✓ a. [Selection: Protect against; Limit] the effects of the following types of denial-of-service events: [Assignment: organization-defined types of denial-of-service events]; and</li> <li>✓ b. Employ the following controls to achieve the denial-of-service objective: [Assignment: organization-defined controls by type of denial-of-service event].</li> </ul> </li> </ul>

- ✓ Discussion: Denial-of-service events may occur due to a variety of internal and external causes, such as an attack by an adversary or a lack of planning to support organizational needs with respect to capacity and bandwidth. Such attacks can occur across a wide range of network protocols (e.g., IPv4, IPv6). A variety of technologies are available to limit or eliminate the origination and effects of denial-of-service events. For example, boundary protection devices can filter certain types of packets to protect system components on internal networks from being directly affected by or the source of denial-of-service attacks. Employing increased network capacity and bandwidth combined with service redundancy also reduces the susceptibility to denial-of-service events [11].
- ✓ V-222594: The application must restrict the ability to launch Denial of Service (DoS) attacks against itself or other information systems. Design and deploy the application to utilize controls that will prevent the application from being affected by DoS attacks or being used to attack other systems. This includes but is not limited to utilizing throttling techniques for application traffic such as QoS or implementing logic controls within the application code itself that prevents application use those results in network or system capabilities being exceeded [15].

**Priority:** Not described

**Rationale:** Mitigating DoS attacks ensures service availability and prevents unauthorized disruption to system operations.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

1. **Rate Limiting Test:**
  - ✓ Test: Simulate excessive requests from a single user/IP and observe system behavior.
  - ✓ Expected result: System throttles requests appropriately, preventing overload.
2. **QoS Verification Test:**
  - ✓ Test: Verify that critical application traffic is prioritized using QoS technologies.
  - ✓ Expected result: Critical services maintain availability even under stress conditions.
3. **Throttling Mechanism Test:**
  - ✓ Test: Confirm that user and IP-specific throttling mechanisms are operational.
  - ✓ Expected result: Requests beyond defined thresholds are delayed or blocked.
4. **Resource Monitoring Test:**
  - ✓ Test: Monitor system resource usage under high traffic scenarios.
  - ✓ Expected result: Resource usage remains within acceptable limits, avoiding service degradation.
5. **Penetration Test:**
  - ✓ Test: Conduct a simulated DoS attack and analyze the system's response.
  - ✓ Expected result: System resists attack without service interruption.

**Requested by:** The organization

**Responsible:** Developer

<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda
José L Fernández-Alemán
Juan Manuel Carrillo-de-Gea
Joaquín Nicolás
08/01/2026

<b>PUID:</b> [SECM-CAT-SMC-004]
<b>Requirement description:</b> The application must default to a secure state in the event of initialization, shutdown, or abort failures, ensuring no unintended access or exposure.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ V-222585: The application must fail to a secure state if system initialization fails, shutdown fails, or aborts fail. Fix any vulnerability found when the application is an insecure state (initialization, shutdown and aborts) [15].</li> <li>✓ SRG-APP-000225-MAPP-000047: The mobile app must fail to an initial state when the application unexpectedly terminates, unless it maintains a secure state at all times [17].</li> <li>✓ 7.6: Verify that error handling logic in security controls denies access by default [19].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Mitigating DoS attacks ensures service availability and prevents unauthorized disruption to system operations.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Initialization Failure Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Simulate a system initialization failure.</li> <li>✓ Expected result: System halts securely, denying unauthorized access.</li> </ul> </li> <li><b>2. Shutdown Failure Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Simulate an unexpected shutdown during operations.</li> <li>✓ Expected result: Sensitive data is securely cleared, and system transitions to a protected state.</li> </ul> </li> <li><b>3. Abort Scenario Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Force an application abort under various conditions.</li> <li>✓ Expected result: System denies access to all resources and prevents data leakage.</li> </ul> </li> <li><b>4. Error Handling Review Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Examine error handling logic to ensure access is denied by default in failure scenarios.</li> <li>✓ Expected result: All security controls enforce a deny-by-default policy.</li> </ul> </li> </ol>

<b>5. Penetration Test:</b>
✓ Test: Simulate attempts to exploit failure states (e.g., during initialization or shutdown).
✓ Expected result: System resists attacks and remains secure.
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda
José L Fernández-Alemán
Juan Manuel Carrillo-de-Gea
Joaquín Nicolás
08/01/2026

<b>PUID:</b> [SECM-CAT-SMC-005]
<b>Requirement description:</b> The application must preserve critical information during a system failure to facilitate root cause analysis and ensure minimal disruption to operations upon recovery.
<b>Source:</b>
✓ V-222586: In the event of a system failure, applications must preserve any information necessary to determine cause of failure and any information necessary to return to operations with least disruption to mission processes. Create operational configuration documentation that identifies information needed for the application to return back into service or specify no such data is required, and retain data required to determine root cause of application failures [15].
<b>Priority:</b> Not described
<b>Rationale:</b> Retaining key data during failures enables effective debugging and restoration, supporting mission continuity and reliability.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>Failure Event Logging Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Simulate system failure and verify that logs capture the root cause and relevant operational details.</li> <li>✓ Expected result: Logs contain detailed and actionable failure information without exposing sensitive data.</li> </ul> </li> <li><b>Recovery Simulation Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Restart the application after failure using retained operational configuration data.</li> <li>✓ Expected result: The application resumes with minimal disruption and no data loss.</li> </ul> </li> <li><b>Data Retention Review:</b></li> </ol>

<ul style="list-style-type: none"> <li>✓ Test: Verify that only necessary failure-related information is preserved and securely stored.</li> <li>✓ Expected result: Information retention complies with security and privacy requirements.</li> </ul>
<b>4. Configuration Documentation Validation:</b>
<ul style="list-style-type: none"> <li>✓ Test: Review operational configuration documentation for completeness and accuracy.</li> <li>✓ Expected result: Documentation specifies all required data for system recovery or states no data is required.</li> </ul>
<b>5. Security and Compliance Test:</b>
<ul style="list-style-type: none"> <li>✓ Test: Verify that retained data during failures adheres to data protection and compliance standards.</li> <li>✓ Expected result: Failure-handling mechanisms align with organizational security policies.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-SMC-006]
<b>Requirement description:</b> The application must shut down or implement compensating controls upon audit system failure, ensuring continued compliance or secure operation.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ V-222586: In the event of a system failure, applications must preserve any information necessary to determine cause of failure and any information necessary to return to operations with least disruption to mission processes. Create operational configuration documentation that identifies information needed for the application to return back into service or specify no such data is required, and retain data required to determine root cause of application failures [15].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Shutting down or compensating for audit failures prevents the application from operating in a state where logging and accountability are compromised, thereby maintaining compliance and security.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>

<p><b>1. Audit Failure Shutdown Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Simulate an audit system failure and verify if the application ceases processing securely.</li> <li>✓ Expected result: The application stops processing activities upon audit system failure.</li> </ul> <p><b>2. Compensating Controls Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Configure the application to continue logging with compensating controls after simulating audit failure.</li> <li>✓ Expected result: Alternative logging mechanisms capture audit data without service interruption.</li> </ul> <p><b>3. Operational Continuity Assessment:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Analyze the application behavior in cases where availability overrides the need for shutdown.</li> <li>✓ Expected result: Availability is maintained with sufficient compensating controls.</li> </ul> <p><b>4. Audit Logging Integrity Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Verify integrity and completeness of log data generated during compensating control operation.</li> <li>✓ Expected result: Logs are secure, complete, and free of tampering.</li> </ul> <p><b>5. Security and Compliance Verification:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Ensure that the failure-handling mechanisms comply with regulatory and organizational audit requirements.</li> <li>✓ Expected result: The application adheres to compliance standards even during audit system failures.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<p><b>Author and date:</b></p> <p>Carlos M. Mejía-Granda  José L Fernández-Alemán  Juan Manuel Carrillo-de-Gea  Joaquín Nicolás  08/01/2026</p>

<b>PUID:</b> [SECM-CAT-SMC-007]
<b>Requirement description:</b> The application must incorporate redundancy mechanisms to ensure high availability in critical environments, mitigating single points of failure.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ V-222595: The web service design must include redundancy mechanisms when used with high-availability systems. Build the application to address issues that are found in a redundant environment and utilize redundancy mechanisms to provide high availability [15].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Redundancy mechanisms enable applications to maintain operational continuity during component failures, ensuring the reliability and availability of high-availability systems.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described

<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<p><b>1. Redundancy Mechanism Implementation Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Verify the application's redundancy mechanisms in a simulated high-availability environment.</li> <li>✓ Expected result: The application continues to function seamlessly without downtime during component failures.</li> </ul> <p><b>2. Failover Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Simulate a primary system failure and evaluate the failover process.</li> <li>✓ Expected result: The application automatically switches to a backup system with minimal impact on functionality.</li> </ul> <p><b>3. Load Balancing Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Implement load balancing across redundant components and observe system performance under varying loads.</li> <li>✓ Expected result: The application distributes workload evenly, ensuring stable performance.</li> </ul> <p><b>4. Recovery Time Objective (RTO) Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Measure the time taken to recover operations after a failure in a redundant environment.</li> <li>✓ Expected result: The application meets the predefined RTO as specified in the operational requirements.</li> </ul> <p><b>5. Redundancy Integrity Validation:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Verify that redundancy mechanisms maintain data consistency and system integrity during failover.</li> <li>✓ Expected result: No data loss or corruption occurs during redundancy transitions.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-SMC-008]
<b>Requirement description:</b> The application must not modify or request operating system parameters unless essential for its functionality, and privileged access must be restricted and managed to prevent integrity issues or privilege escalation.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ SRG-APP-000033-MAPP-000010: The mobile app must not modify, request, or assign values for operating system parameters unless necessary to perform application functions. A mobile app that operates with the privileges of its host OS is vulnerable to integrity issues and escalated privileges that would affect the entire platform and device. [17].</li> </ul>

<ul style="list-style-type: none"> <li>✓ 8.2 Privileged access rights           <p>Control: The allocation and use of privileged access rights should be restricted and managed. Purpose: To ensure only authorized users, software components and services are provided with privileged access rights [6].</p> </li> <li>✓ CWE-269: Improper Privilege Management:  The product contains a code sequence that can run concurrently with other code, and the code sequence requires temporary, exclusive access to a shared resource, but a timing window exists in which the shared resource can be modified by another code sequence that is operating concurrently.  ✓ The product does not properly assign, modify, track, or check privileges for an actor, creating an unintended sphere of control for that actor [35].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Restricting unnecessary modifications to operating system parameters and managing privileged access prevents potential integrity violations, unauthorized control, and security vulnerabilities across the host device.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li>1. <b>OS Parameter Modification Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Attempt to modify OS parameters not explicitly required for application functionality.</li> <li>✓ Expected result: The application denies unauthorized modification attempts.</li> </ul> </li> <li>2. <b>Privilege Restriction Validation:</b> <ul style="list-style-type: none"> <li>✓ Test: Verify that only authorized components or users have privileged access.</li> <li>✓ Expected result: Privilege assignment is restricted and appropriately managed.</li> </ul> </li> <li>3. <b>Privilege Escalation Simulation:</b> <ul style="list-style-type: none"> <li>✓ Test: Simulate scenarios where privilege escalation is attempted.</li> <li>✓ Expected result: The application prevents all unauthorized privilege escalation attempts.</li> </ul> </li> <li>4. <b>Code Review for OS Interaction:</b> <ul style="list-style-type: none"> <li>✓ Test: Perform a static analysis of code for interactions with OS parameters.</li> <li>✓ Expected result: Only essential and justified OS parameter modifications are present in the codebase.</li> </ul> </li> <li>5. <b>Security Audit of Privileged Operations:</b> <ul style="list-style-type: none"> <li>✓ Test: Conduct an audit to ensure privileged operations adhere to least privilege principles.</li> <li>✓ Expected result: No excessive or unnecessary privileged operations are identified.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described

<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda
José L Fernández-Alemán
Juan Manuel Carrillo-de-Gea
Joaquín Nicolás
08/01/2026

<b>PUID:</b> [SECM-CAT-SMC-009]
<b>Requirement description:</b> The application must enforce execution policies to prevent unauthorized program execution and restrict privilege levels to the minimum necessary, in compliance with organizational policies and terms of software usage.
<b>Source:</b>
<ul style="list-style-type: none"><li>✓ V-222516: The application must prevent program execution in accordance with organization-defined policies regarding software program usage and restrictions, and/or rules authorizing the terms and conditions of software program usage. Restrict application execution in accordance with the policy, terms, and conditions specified [15].</li><li>✓ SRG-APP-000033-MAPP-000011: The mobile app must not execute as a privileged operating system process unless necessary to perform any app functions [17].</li><li>✓ SRG-APP-000342-MAPP-000100: The mobile app must prevent organization-defined software from executing at higher privilege levels than users executing the software [17].</li><li>✓ Security: 25. It states terms and conditions of cloud services [29].</li><li>✓ Test ID 6: The system will not allow users greater access than their assigned role permits [37].</li><li>✓ 2.19. Ensure that the app runs with user privileges (unprivileged) on the end user device (does not require a rooted or a jailbroken device). Verify that it does not request more access authorizations to system resources and rights in the execution environment than the absolutely necessary (least privilege principle) [16].</li><li>✓ Security: The cloud services used have the relevant security measures. It states the terms and conditions of cloud services user [21].</li></ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Restricting program execution and privilege levels ensures compliance with organizational security policies, minimizes attack surfaces, and prevents unauthorized access to system resources.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described

<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<p><b>1. Privilege Verification Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Verify that the application executes only with unprivileged user rights and does not require elevated system privileges.</li> <li>✓ Expected result: The application runs without requesting excessive or unnecessary system permissions.</li> </ul> <p><b>2. Policy Compliance Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Attempt to execute unauthorized or restricted programs.</li> <li>✓ Expected result: The application denies execution of programs not allowed by organizational policies.</li> </ul> <p><b>3. Access Control Validation:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Attempt to escalate privileges beyond those assigned by user roles.</li> <li>✓ Expected result: The system prevents privilege escalation and maintains access restrictions.</li> </ul> <p><b>4. Cloud Services Configuration Review:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Verify that terms and conditions for cloud services used by the application explicitly restrict unauthorized execution and access.</li> <li>✓ Expected result: Cloud service policies align with organizational security requirements.</li> </ul> <p><b>5. Static Analysis of Privilege Requests:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Perform static code analysis to identify and review all system privilege requests by the application.</li> <li>✓ Expected result: Only essential privilege requests are found, adhering to the least privilege principle.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-SMC-010]
<b>Requirement description:</b> The mobile application must not lock or restrict file permissions in a way that prevents the operating system or authorized backup tools from copying application files.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ SRG-APP-000516-MAPP-000034: The mobile app must not lock or set permissions on application files in a manner such that the operating system or an approved backup application cannot copy the files [17].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Ensuring backup tools and the operating system can access application files is essential for data recovery, operational continuity, and compliance with organizational backup policies.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0

<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Backup Compatibility Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Use an approved backup application to attempt copying all application files.</li> <li>✓ Expected result: The backup application successfully copies application files without encountering permission issues.</li> </ul> </li> <li><b>2. File Access Permissions Review:</b> <ul style="list-style-type: none"> <li>✓ Test: Inspect file permissions to verify they allow access to the operating system and authorized applications.</li> <li>✓ Expected result: Permissions are appropriately set to enable authorized access.</li> </ul> </li> <li><b>3. System Backup Integration Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Perform a full system backup and ensure application data is included in the backup.</li> <li>✓ Expected result: The system backup contains all relevant application files.</li> </ul> </li> <li><b>4. File Lock Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Attempt to lock application files to restrict access by the operating system or backup tools.</li> <li>✓ Expected result: The application does not allow locking that prevents authorized access.</li> </ul> </li> <li><b>5. Restore Verification Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Restore application files from a backup and confirm their functionality.</li> <li>✓ Expected result: Application files restore successfully without data corruption or loss.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-SMC-011]
<b>Requirement description:</b> The mobile application must not alter the file permissions of any files outside its scope of operation, ensuring it modifies only files dedicated to its own functionality.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ SRG-APP-000381-MAPP-000010: The mobile app must not change the file permissions of any files other than those dedicated to its own operation [17].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Restricting file permission changes to the app's operational files minimizes the risk of accidental or malicious modifications to system files, maintaining overall device integrity and security.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0

<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. File Permission Modification Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Attempt to monitor the app while it changes file permissions outside its dedicated files.</li> <li>✓ Expected result: The application does not modify file permissions outside its operational scope.</li> </ul> </li> <li><b>2. System Integrity Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Perform a system scan after app installation to detect unauthorized changes to system file permissions.</li> <li>✓ Expected result: No unauthorized file permissions are modified.</li> </ul> </li> <li><b>3. File Scope Verification Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Analyze the app's code and configuration to confirm it targets only its operational files for permission changes.</li> <li>✓ Expected result: The app's logic explicitly restricts permission changes to its files.</li> </ul> </li> <li><b>4. Cross-Application Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Install the app alongside other applications and monitor for permission changes affecting unrelated app files.</li> <li>✓ Expected result: No file permissions are altered for files belonging to other apps.</li> </ul> </li> <li><b>5. Security Audit Review:</b> <ul style="list-style-type: none"> <li>✓ Test: Conduct a security audit to validate compliance with file permission modification policies.</li> <li>✓ Expected result: The app adheres strictly to file modification policies and passes the audit without issues.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-SMC-012]
<b>Requirement description:</b> The application must prioritize the use of existing system-defined permissions to meet security requirements before defining new permissions.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ Use Existing Permissions: Before creating a new permission, the application must first verify if existing system-defined permissions can be used to satisfy the security needs [18].</li> </ul>

<b>Rationale:</b> Leveraging system-defined permissions reduces complexity, avoids redundancy, and ensures alignment with platform security standards.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Permission Reuse Analysis Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Analyze the app's permissions and verify whether existing system-defined permissions meet the app's security needs.</li> <li>✓ Expected result: All necessary permissions are reused from system-defined permissions when applicable.</li> </ul> </li> <li><b>2. New Permission Justification Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Examine the app's documentation to ensure all newly created permissions are justified by unique security needs.</li> <li>✓ Expected result: New permissions are introduced only when no system-defined permissions suffice.</li> </ul> </li> <li><b>3. Redundancy Avoidance Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Compare newly defined permissions against system-defined ones for potential overlap.</li> <li>✓ Expected result: No redundant permissions overlap with system-defined permissions.</li> </ul> </li> <li><b>4. Platform Compatibility Review:</b> <ul style="list-style-type: none"> <li>✓ Test: Verify that the app's permission strategy aligns with the platform's guidelines and best practices.</li> <li>✓ Expected result: The app adheres to platform standards and avoids unnecessary custom permissions.</li> </ul> </li> <li><b>5. Permission Impact Assessment:</b> <ul style="list-style-type: none"> <li>✓ Test: Assess the impact of any new permissions on user security and system resources.</li> <li>✓ Expected result: New permissions, if any, have minimal impact and are essential for app functionality.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-SMC-013]
---------------------------------

**Requirement description:** The application must adhere to the least privilege principle, requiring only the minimum set of permissions necessary for functionality while avoiding unnecessary or excessive permissions.

**Source:**

- ✓ 4.3.1. access control

least privilege access control for maximum security: a user of a system has enough rights to conduct authorized actions within a system. All other permissions are denied by default [37].

- ✓ 6.1: Verify that the app only requires the minimum set of permissions necessary [19].
- ✓ Least privilege principle: 1. Request only the permissions necessary for the proper functioning of the application [3].
- ✓ No Unnecessary Permissions: If a permission is not required for the app to function, it must not be declared. A thorough evaluation must be conducted to determine the need for each permission before it is included [18].
- ✓ Minimize Permission Creation: The application must define the smallest set of new permissions necessary to meet security requirements. System-defined permissions must be used wherever possible, avoiding the creation of new permissions unless absolutely required [18].
- ✓ Limit Scope of Permissions: Request only the narrowest permission scope necessary for each task, avoiding broad access that isn't required [18].
- ✓ V-222430: The application must execute without excessive account permissions. Configure the application accounts with minimalist privileges. Do not allow the application to operate with admin credentials [15].

**Priority:** Not described

**Rationale:** Leveraging system-defined permissions reduces complexity, avoids redundancy, and ensures alignment with platform security standards.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

**1. Permission Necessity Evaluation:**

- ✓ Test: Analyze all declared permissions in the app and validate their necessity for app functionality.
- ✓ Expected result: Every declared permission is essential and justified by a specific app feature or requirement.

**2. Excessive Permissions Check:**

<ul style="list-style-type: none"> <li>✓ Test: Verify the app does not request administrative or broad permissions unless absolutely required.</li> <li>✓ Expected result: The app operates without excessive or admin-level permissions.</li> </ul>
<b>3. System-Defined Permission Usage:</b>
<ul style="list-style-type: none"> <li>✓ Test: Compare the app's permissions against system-defined permissions to confirm reuse where applicable.</li> <li>✓ Expected result: No new permissions are created if equivalent system-defined permissions exist.</li> </ul>
<b>4. Scope Limitation Test:</b>
<ul style="list-style-type: none"> <li>✓ Test: Inspect whether permissions requested have the narrowest scope necessary (e.g., access to specific files instead of entire storage).</li> <li>✓ Expected result: Permissions are narrowly scoped to specific actions or data.</li> </ul>
<b>5. Impact Assessment on Denied Permissions:</b>
<ul style="list-style-type: none"> <li>✓ Test: Test the app's functionality by denying unnecessary permissions.</li> <li>✓ Expected result: The app continues to function correctly without optional or excessive permissions.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-SMC-014]
<b>Requirement description:</b> The application must request only the minimum number of permissions strictly necessary for its core functionality, avoiding any unnecessary or non-essential permissions.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ Minimize Permission Requests: The application must minimize the number of permissions it requests, only asking for permissions that are strictly necessary for its functionality. Permissions that are not essential to the core operation of the app must not be requested [18].</li> <li>✓ No Unnecessary Permissions: If a permission is not required for the app to function, it must not be declared. A thorough evaluation must be conducted to determine the need for each permission before it is included [18].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Minimizing permission requests reduces security vulnerabilities, preserves user privacy, and ensures compliance with platform security guidelines.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described

<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<b>1. Permission Necessity Review:</b>
✓ Test: Evaluate all permissions declared in the app's manifest or equivalent configuration.
✓ Expected result: Each requested permission directly supports an essential app function.
<b>2. Core Functionality Dependency Test:</b>
✓ Test: Test the app by running all core functionalities without optional permissions.
✓ Expected result: Core functionalities operate without requiring non-essential permissions.
<b>3. Excess Permission Detection:</b>
✓ Test: Use automated tools or static analysis to detect permissions that are not utilized by the app.
✓ Expected result: No permissions are declared that are not explicitly required by the app's features.
<b>4. User Prompt Analysis:</b>
✓ Test: Verify that users are prompted only for permissions needed at the time of use.
✓ Expected result: Permissions are requested contextually and only when required by a feature.
<b>5. Comparison to Platform Guidelines:</b>
✓ Test: Check the app's permissions against platform security guidelines and best practices.
✓ Expected result: The app complies with platform recommendations for permission minimization.
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-SMC-015]
<b>Requirement description:</b> The application must notify System Administrators and Information System Security Officers (ISSOs) of account lifecycle events, including creation, modification, enabling, disabling, and removal.
<b>Source:</b>
✓ V-222417: The application must notify System Administrators and Information System Security Officers when accounts are created. Configure the application to notify the system administrator and the ISSO when application accounts are created [15].

<ul style="list-style-type: none"> <li>✓ V-222418: The application must notify System Administrators and Information System Security Officers when accounts are modified. Configure the application to notify the system administrator and the ISSO when application accounts are modified [15].</li> <li>✓ V-222419: The application must notify System Administrators and Information System Security Officers of account disabling actions. Configure the application to notify the system administrator and the ISSO when application accounts are disabled [15].</li> <li>✓ V-222420: The application must notify System Administrators and Information System Security Officers of account removal actions. Configure the application to notify the system administrator and the ISSO when application accounts are removed [15].</li> <li>✓ V-222422: The application must notify System Administrators and Information System Security Officers of account enabling actions. Configure the application to notify the system administrator and the ISSO when application accounts are enabled [15].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Providing notifications for account lifecycle events ensures transparency, accountability, and timely responses to unauthorized or unexpected changes, enhancing security and compliance.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Account Creation Notification Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Create a new user account and verify that notifications are sent to the designated administrators and ISSOs.</li> <li>✓ Expected result: Notifications for account creation are sent accurately and include relevant details.</li> </ul> </li> <li><b>2. Account Modification Notification Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Modify an existing account's attributes and verify the notification process.</li> <li>✓ Expected result: Notifications for account modifications are sent with appropriate information.</li> </ul> </li> <li><b>3. Account State Change Notification Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Enable, disable, or remove a user account and verify notification accuracy.</li> <li>✓ Expected result: Notifications for enabling, disabling, or removing accounts are timely and include event details.</li> </ul> </li> <li><b>4. Notification Log Verification:</b> <ul style="list-style-type: none"> <li>✓ Test: Review logs to confirm notifications are consistently generated for all account lifecycle events.</li> <li>✓ Expected result: Logs display all relevant notifications without omissions or errors.</li> </ul> </li> <li><b>5. Notification Delivery Method Validation:</b> <ul style="list-style-type: none"> <li>✓ Test: Verify that notifications are delivered using secure and predefined channels (e.g., email, SMS, or system logs).</li> </ul> </li> </ol>

✓ Expected result: Notifications are delivered via secure channels and reach the intended recipients.
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-SMC-016]
<b>Requirement description:</b> The application must be designed to avoid requiring sensitive permissions by utilizing alternative methods, such as generating unique identifiers internally without accessing device information.
<b>Source:</b>
✓ Design without Sensitive Permissions: Where possible, the application should be designed to avoid the need for sensitive permissions. For example, instead of requesting device information to create unique identifiers, the app must generate a UUID internally [18].
<b>Priority:</b> Not described
<b>Rationale:</b> Minimizing the use of sensitive permissions reduces the risk of exposing user data and enhances compliance with privacy regulations.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li>1. <b>Permission Audit Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Analyze the permissions requested by the application during installation and runtime.</li> <li>✓ Expected result: The app does not request sensitive permissions unnecessarily.</li> </ul> </li> <li>2. <b>Unique Identifier Generation Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Verify that unique identifiers are generated internally without accessing device-specific sensitive information.</li> <li>✓ Expected result: The app generates UUIDs internally without relying on sensitive device attributes.</li> </ul> </li> <li>3. <b>Code Review for Sensitive Permission Usage:</b> <ul style="list-style-type: none"> <li>✓ Test: Perform a static code analysis to identify unnecessary usage of sensitive permissions.</li> <li>✓ Expected result: No sensitive permissions are unnecessarily accessed or declared in the code.</li> </ul> </li> </ol>

<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda
José L Fernández-Alemán
Juan Manuel Carrillo-de-Gea
Joaquín Nicolás
08/01/2026

<b>PUID:</b> [SECM-CAT-SMC-017]
<b>Requirement description:</b> The application must support dynamic permission management, enabling content providers to grant or revoke data access case-by-case and allowing secure, runtime permission additions when necessary.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ Dynamic Permission Management: The content provider must support dynamic permission grants, allowing the system to grant or revoke data access to other apps on a case-by-case basis, ensuring more secure data control [18].</li> <li>✓ Dynamic Permission Addition: The application must have the capability to dynamically add permissions using the addPermission() method, when needed [18].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Dynamic permission management ensures tighter control over data access, enhancing security and reducing unnecessary exposure of sensitive information.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Dynamic Grant/Revoke Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Verify that the application can dynamically grant or revoke permissions for accessing content provider data.</li> <li>✓ Expected result: Permissions are successfully managed on a case-by-case basis, without affecting the functionality.</li> </ul> </li> <li><b>2. addPermission() Method Functionality Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Evaluate the application's capability to dynamically add permissions using the addPermission() method.</li> <li>✓ Expected result: Permissions are added securely and are limited to specific, justified use cases.</li> </ul> </li> <li><b>3. Audit Permission Changes Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Verify that all permission grant and revoke actions are logged for auditing purposes.</li> </ul> </li> </ol>

<ul style="list-style-type: none"> <li>✓ Expected result: Every permission change is recorded with relevant details, such as timestamp and reason.</li> </ul> <p><b>4. Restricted Data Access Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Ensure that dynamically granted permissions are scoped and do not allow unauthorized access to unrelated data.</li> <li>✓ Expected result: Data access is restricted to the specified scope defined during the permission grant.</li> </ul> <p><b>5. Dynamic Permission Security Review:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Conduct a static analysis of the implementation of dynamic permissions to ensure compliance with secure coding practices.</li> <li>✓ Expected result: No vulnerabilities are introduced through the dynamic management of permissions.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-SMC-018]
<b>Requirement description:</b> The mobile health application must minimize the use of dangerous-level permissions and avoid insecure file permissions, ensuring requests are clear and necessary to prevent user confusion or deterred installation.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ Avoiding User Confusion: The application must minimize the use of dangerous-level permissions, avoiding confusing permission requests that may deter users from installing the application [18].</li> <li>✓ Insecure permissions: 1. Avoid storing application files with overly permissive permissions like world-readable and/or world-writable [3].</li> <li>✓ Handling Dangerous Permissions: If a new permission is declared with a "dangerous" protection level, the following considerations must be addressed:           <ul style="list-style-type: none"> <li>○ The permission string must be concise and clearly convey the security decision to users.</li> <li>○ The permission string must be localized into multiple languages to ensure broad understanding.</li> <li>○ The application must handle the possibility that users may choose not to install the app due to confusion or perceived risk associated with the dangerous permission.</li> <li>○ The app must manage requests for the permission even if the defining app has not been installed [18].</li> </ul> </li> </ul>
<b>Priority:</b> Not described

<b>Rationale:</b> Ensuring the clarity and necessity of permission requests reduces user confusion, builds trust, and maintains compliance with security standards critical in handling sensitive health data. Avoiding overly permissive file permissions minimizes unauthorized access risks.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<p><b>1. Permission Request Evaluation Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Verify that all permissions requested by the application are essential for its operation and are communicated clearly to users.</li> <li>✓ Expected Result: The app requests only necessary permissions with concise, localized descriptions of their purpose.</li> </ul>
<p><b>2. File Permission Review Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Inspect application file permissions for any overly permissive settings such as world-readable or world-writable.</li> <li>✓ Expected Result: Application files do not have overly permissive permissions.</li> </ul>
<p><b>3. Dangerous Permission Management Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Assess the app's ability to handle denied dangerous-level permissions gracefully, ensuring app functionality is not critically disrupted.</li> <li>✓ Expected Result: The application operates securely and informs users when dangerous permissions are denied.</li> </ul>
<p><b>4. User Trust Feedback Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Conduct user acceptance testing to measure user trust and clarity of permission requests.</li> <li>✓ Expected Result: Users report understanding and trust in the permissions being requested by the app.</li> </ul>
<p><b>5. Localized Permission Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Verify that dangerous-level permissions are properly localized in multiple languages.</li> <li>✓ Expected Result: Permission descriptions are accurately localized, ensuring broad comprehension across supported languages.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<p><b>Author and date:</b></p> <p>Carlos M. Mejía-Granda      José L Fernández-Alemán      Juan Manuel Carrillo-de-Gea      Joaquín Nicolás      08/01/2026</p>

<b>PUID:</b> [SECM-CAT-SMC-019]
<b>Requirement description:</b> The mobile health application must prevent other applications or non-privileged processes from modifying its software libraries.
<b>Source:</b>
✓ SRG-APP-000133-MAPP-000030: The mobile app must not enable other applications or non-privileged processes to modify software libraries [17].
<b>Priority:</b> Not described
<b>Rationale:</b> Protecting software libraries from unauthorized modification ensures application integrity, prevents exploitation, and safeguards sensitive health-related functionalities and data.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li>1. <b>Library Modification Restriction Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Attempt to modify application software libraries using a non-privileged process or another application.</li> <li>✓ Expected Result: Modification attempts are denied, and no unauthorized changes occur.</li> </ul> </li> <li>2. <b>File Integrity Validation Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Use a checksum or similar method to validate the integrity of software libraries after multiple app operations.</li> <li>✓ Expected Result: Software library integrity remains intact, matching the original checksum.</li> </ul> </li> <li>3. <b>Permissions Audit Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Review the file permissions of software libraries within the application.</li> <li>✓ Expected Result: Libraries are configured with restrictive permissions, preventing modifications by non-privileged processes.</li> </ul> </li> <li>4. <b>Dynamic Analysis Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Perform runtime analysis to monitor access attempts to modify libraries by external processes or applications.</li> <li>✓ Expected Result: No unauthorized processes successfully access or modify the libraries.</li> </ul> </li> <li>5. <b>Security Configuration Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Validate that the app enforces platform-recommended security configurations, such as enforcing SELinux policies or using secured directories.</li> <li>✓ Expected Result: Security configurations are implemented, preventing unauthorized library modifications.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda
José L Fernández-Alemán

Juan Manuel Carrillo-de-Gea  
Joaquín Nicolás  
08/01/2026

**PUID:** [SECM-CAT-SMC-020]

**Requirement description:** The mobile health application must clear all cookies and user-tracking information upon termination to prevent unauthorized access or identity tracking.

**Source:**

- ✓ SRG-APP-000516-MAPP-000066: The mobile app must remove cookies or information used to track a user's identity when it terminates [17].
- ✓ 1.34. In the case that the application includes embedded web browsing capabilities (e.g., WebViews), clear stored cookies on app termination or use in-memory cookie storage [16].

**Priority:** Not described

**Rationale:** Ensuring the removal of cookies and tracking data at app termination minimizes the risk of data leakage, identity theft, and misuse of sensitive health information.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

1. **Cookie Clearing Test:**
  - ✓ Test: Store cookies during a session and terminate the app. Reopen the app and inspect for the presence of previous cookies.
  - ✓ Expected Result: All cookies are cleared, and no previous session data persists.
2. **In-Memory Storage Test:**
  - ✓ Test: Confirm that cookies are stored only in memory during the app's runtime.
  - ✓ Expected Result: Cookies are stored in memory and are automatically deleted when the app terminates.
3. **Session Data Inspection Test:**
  - ✓ Test: Use debugging tools to check for residual tracking or session information post-app termination.
  - ✓ Expected Result: No session or tracking data is retained after the app closes.
4. **WebView Cookie Management Test:**
  - ✓ Test: Validate that WebView instances clear cookies and session data when the app terminates.
  - ✓ Expected Result: WebView clears all stored cookies and session data upon app exit.
5. **Security Policy Validation Test:**
  - ✓ Test: Review the app's cookie and session management policies for compliance with health data protection regulations.
  - ✓ Expected Result: Policies ensure no sensitive information remains accessible after app termination.

**Requested by:** The organization

**Responsible:** Developer

<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026
<b>PUID:</b> [SECM-CAT-SMC-021]
<b>Requirement description:</b> The mobile health application must utilize ports and protocols in alignment with organizational and DoD Ports, Protocols, and Services Management (DoD PPSM) guidelines.
<b>Source:</b> <ul style="list-style-type: none"> <li>✓ SRG-APP-000142-MAPP-000032: The mobile app must utilize ports or protocols in a manner consistent with DoD Ports and Protocols guidance [17].</li> <li>✓ V-222628: New IP addresses, data services, and associated ports used by the application must be submitted to the appropriate approving authority for the organization, which in turn will be submitted through the DoD Ports, Protocols, and Services Management (DoD PPSM) [15].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Proper management of ports and protocols ensures secure communication, prevents unauthorized access, and complies with organizational and DoD security standards.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b> <ol style="list-style-type: none"> <li><b>1. Protocol Configuration Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Review the app's network configuration to ensure only approved ports and protocols are used.</li> <li>✓ Expected Result: The app utilizes only approved ports and protocols consistent with DoD PPSM guidelines.</li> </ul> </li> <li><b>2. Change Approval Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Submit a request for a new port or protocol usage and review the approval process.</li> <li>✓ Expected Result: New ports or protocols are documented and approved by the organization's authority before implementation.</li> </ul> </li> <li><b>3. Network Traffic Analysis Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Monitor network traffic generated by the app to verify that all ports and protocols comply with DoD guidance.</li> <li>✓ Expected Result: All observed traffic adheres to approved configurations.</li> </ul> </li> <li><b>4. Compliance Documentation Review:</b> <ul style="list-style-type: none"> <li>✓ Test: Inspect documentation to confirm submission of all new IP addresses, services, and associated ports to the organization's approving authority.</li> </ul> </li> </ol>

<ul style="list-style-type: none"> <li>✓ Expected Result: Documentation demonstrates compliance with the DoD PPSM process.</li> </ul> <p><b>5. Penetration Testing:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Perform penetration testing to ensure no unauthorized or non-compliant ports or protocols are accessible.</li> <li>✓ Expected Result: No vulnerabilities or unauthorized access via ports or protocols are identified.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-SMC-022]
<b>Requirement description:</b> The mobile health application must synchronize its internal clocks with the authoritative time source provided by the Mobile Operating System (MOS).
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ SRG-APP-000372-MAPP-000100: The mobile app must synchronize internal information system clocks to the MOS-based authoritative time source [17].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Accurate and synchronized system time is critical for secure logging, auditing, and system operation to ensure consistency and traceability.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Time Synchronization Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Verify that the application synchronizes its internal clock with the MOS-provided time source during startup and at predefined intervals.</li> <li>✓ Expected Result: The application's internal clock matches the MOS-provided authoritative time.</li> </ul> </li> <li><b>2. Audit Log Time Consistency Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Review timestamps in the application logs for accuracy and consistency with the authoritative time source.</li> <li>✓ Expected Result: All log entries use the synchronized authoritative time.</li> </ul> </li> <li><b>3. Failure Handling Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Disconnect the app from the time source and observe behavior.</li> </ul> </li> </ol>

<ul style="list-style-type: none"> <li>✓ Expected Result: The app detects the failure and logs the event while maintaining its last synchronized time.</li> </ul> <p><b>4. Configuration Review Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Inspect the app's settings and code to confirm the use of MOS-based time synchronization.</li> <li>✓ Expected Result: Configuration explicitly points to the MOS time source for synchronization.</li> </ul> <p><b>5. Periodic Synchronization Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Simulate a long-running session and ensure periodic time synchronization with the MOS source.</li> <li>✓ Expected Result: The app periodically updates its internal clock to remain synchronized.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<p><b>Author and date:</b>            Carlos M. Mejía-Granda            José L Fernández-Alemán            Juan Manuel Carrillo-de-Gea            Joaquín Nicolás            08/01/2026</p>

<b>PUID:</b> [SECM-CAT-SMC-023]
<b>Requirement description:</b> The mobile health application must activate free security features offered by the toolchain, including byte-code minification, stack protection, Position Independent Executable (PIE) support, and automatic reference counting.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ 7.8: Free security features offered by the toolchain, such as byte-code minification, stack protection, PIE support and automatic reference counting, are activated [19].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Activating these free security features enhances the application's resilience against exploitation by minimizing vulnerabilities at the compilation and runtime levels.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<p><b>1. Toolchain Configuration Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Verify that the app's build configuration enables security features such as byte-code minification, stack protection, PIE support, and automatic reference counting.</li> <li>✓ Expected Result: All security features provided by the toolchain are enabled in the build settings.</li> </ul> <p><b>2. Runtime Protection Test:</b></p>

<ul style="list-style-type: none"> <li>✓ Test: Analyze the application during runtime to confirm that stack protection and PIE mechanisms are active.</li> <li>✓ Expected Result: The app resists stack-based attacks and operates with address randomization.</li> </ul> <p><b>3. Code Minification Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Inspect the compiled bytecode or binary to confirm the application has undergone minification.</li> <li>✓ Expected Result: Non-essential symbols and unnecessary code are removed or obfuscated.</li> </ul> <p><b>4. Memory Management Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Assess memory handling through tools to ensure proper automatic reference counting is implemented.</li> <li>✓ Expected Result: No memory leaks or improper reference handling is detected.</li> </ul> <p><b>5. Static Analysis Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Perform a static analysis on the compiled application to verify adherence to security features offered by the toolchain.</li> <li>✓ Expected Result: The analysis confirms the app's compliance with toolchain security standards.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-SMC-024]
<b>Requirement description:</b> The mobile health application must implement signature-level protection for private data, ensuring that only applications signed with the same developer key can access sensitive features or data.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ Signature Protection for Private Data: For sharing data only between applications signed with the same key, the android:protectionLevel attribute must be set to signature, ensuring controlled access without requiring user confirmation [18].</li> <li>✓ Signature-Level Protection: If a new permission is required, the permission must be implemented with a signature protection level where appropriate. This ensures that only applications signed by the same developer can access the protected features, without requiring user interaction [18].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Signature-level protection enhances security by restricting access to trusted applications within the same development ecosystem, reducing the risk of unauthorized data access or feature exploitation.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0

<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<p><b>1. Permission Attribute Verification Test:</b></p> <ul style="list-style-type: none"> <li>✓ <b>Test:</b> Inspect the app's <code>AndroidManifest.xml</code> file to confirm that the <code>android:protectionLevel</code> attribute for relevant permissions is set to <code>signature</code>.</li> <li>✓ <b>Expected Result:</b> The <code>android:protectionLevel</code> attribute is correctly configured for all applicable permissions.</li> </ul> <p><b>2. Key Match Test:</b></p> <ul style="list-style-type: none"> <li>✓ <b>Test:</b> Validate that the applications sharing protected features are signed with the same key.</li> <li>✓ <b>Expected Result:</b> The signing keys match for all apps accessing protected features.</li> </ul> <p><b>3. Unauthorized Access Test:</b></p> <ul style="list-style-type: none"> <li>✓ <b>Test:</b> Attempt to access the protected feature or data from an app not signed with the same key.</li> <li>✓ <b>Expected Result:</b> The attempt is blocked, and no unauthorized access is granted.</li> </ul> <p><b>4. User Interaction Test:</b></p> <ul style="list-style-type: none"> <li>✓ <b>Test:</b> Confirm that the signature-level permissions do not prompt users for confirmation during runtime.</li> <li>✓ <b>Expected Result:</b> Access to protected features is seamless for authorized apps and does not require user intervention.</li> </ul> <p><b>5. Static Analysis Test:</b></p> <ul style="list-style-type: none"> <li>✓ <b>Test:</b> Perform static code analysis to verify that the signature-level protection is applied consistently across all defined permissions.</li> <li>✓ <b>Expected Result:</b> The analysis confirms the use of <code>signature</code> protection level without misconfigurations.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-SMC-025]
<b>Requirement description:</b> The mobile health application must verify the correct operation of security functions on system startup, restart, or upon command by a privileged user.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ V-222616: The application must perform verification of the correct operation of security functions: upon system startup and/or restart; upon command by a user with privileged</li> </ul>

access; and/or every 30 days. Design the application to verify the correct operation of security functions on command and on application startup and restart [15].
<b>Priority:</b> Not described
<b>Rationale:</b> Periodic and event-driven verification of security functions ensures the reliability and robustness of security mechanisms, reducing risks of undetected failures.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Startup Verification Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Restart the application and monitor for automated security function verification during startup.</li> <li>✓ Expected Result: The application performs security checks and logs the results of the verification.</li> </ul> </li> <li><b>2. Command Trigger Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Execute a command as a privileged user to initiate a security function verification.</li> <li>✓ Expected Result: The security functions are verified, and a confirmation or report is generated.</li> </ul> </li> <li><b>3. 30-Day Interval Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Leave the application running for 30 days and confirm that security function verification is performed automatically.</li> <li>✓ Expected Result: The verification occurs within the required timeframe without user intervention.</li> </ul> </li> <li><b>4. Error Handling Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Simulate a security function failure and observe the application's response.</li> <li>✓ Expected Result: The application identifies the failure and logs or alerts the appropriate parties.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-SMC-026]
<b>Requirement description:</b> The mobile health application must notify the ISSO (Information System Security Officer) and ISSM (Information System Security Manager) of any failed security verification tests.
<b>Source:</b>

✓ V-222617: The application must notify the ISSO and ISSM of failed security verification tests. Configure the application to send notices to the ISSO and ISSM indicating the application failed a verification test [15].
<b>Priority:</b> Not described
<b>Rationale:</b> Prompt notification of failed security tests ensures timely response to potential threats, minimizing risks to system integrity and data security.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Failure Simulation Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Trigger a security verification failure (e.g., by introducing an intentional misconfiguration).</li> <li>✓ Expected Result: The application immediately sends a detailed notification to the designated ISSO and ISSM.</li> </ul> </li> <li><b>2. Notification Accuracy Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Verify the content of the notification for completeness, including details such as the type of failure, time, and recommended actions.</li> <li>✓ Expected Result: Notifications provide sufficient information for ISSO and ISSM to address the issue effectively.</li> </ul> </li> <li><b>3. Delivery Confirmation Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Validate that notifications are delivered successfully to the ISSO and ISSM's communication channels (e.g., email, dashboard, or system alerts).</li> <li>✓ Expected Result: Notifications are received without delays or errors in transmission.</li> </ul> </li> <li><b>4. Event Logging Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Review the application's event logs for the record of the failed verification and notification attempt.</li> <li>✓ Expected Result: Logs include a timestamped entry of the failure and confirmation of notification dispatch.</li> </ul> </li> <li><b>5. Redundancy Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Verify that backup notification mechanisms function if the primary channel fails (e.g., fallback to SMS if email is unavailable).</li> <li>✓ Expected Result: Notifications are delivered via an alternative channel if the primary one is inaccessible.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda
José L Fernández-Alemán
Juan Manuel Carrillo-de-Gea
Joaquín Nicolás
08/01/2026

<b>PUID:</b> [SECM-CAT-SMC-027]
<b>Requirement description:</b> The mobile health application must alert administrators when low resource conditions, such as insufficient memory or CPU overload, are detected.
<b>Source:</b>
✓ V-222668: The system must alert an administrator when low resource conditions are encountered. Implement mechanisms to alert system administrators about a low resource condition [15].
<b>Priority:</b> Not described
<b>Rationale:</b> Timely alerts about low resource conditions enable administrators to take proactive measures to maintain application availability and prevent potential failures.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Simulated Low Resource Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Simulate a low resource condition (e.g., by exceeding memory or CPU thresholds).</li> <li>✓ Expected Result: The application generates and sends an alert to the system administrator with details of the resource condition.</li> </ul> </li> <li><b>2. Alert Content Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Verify the accuracy and comprehensiveness of the alert message.</li> <li>✓ Expected Result: Alerts include specifics about the resource type, severity, and suggested corrective actions.</li> </ul> </li> <li><b>3. Notification Mechanism Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Verify that the alert is delivered through all configured communication channels (e.g., email, SMS, or system dashboard).</li> <li>✓ Expected Result: Notifications are reliably sent and received through all defined channels.</li> </ul> </li> <li><b>4. Threshold Configuration Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Confirm that resource thresholds for triggering alerts can be configured by administrators.</li> <li>✓ Expected Result: Administrators can customize thresholds for memory, CPU, and other resource limits.</li> </ul> </li> <li><b>5. Recovery Logging Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Simulate resource recovery and verify that the system logs the event.</li> <li>✓ Expected Result: Recovery from low resource conditions is accurately logged for auditing purposes.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>

Carlos M. Mejía-Granda  
José L Fernández-Alemán  
Juan Manuel Carrillo-de-Gea  
Joaquín Nicolás  
08/01/2026

**PUID:** [SECM-CAT-SMC-028]

**Requirement description:** Mobile health applications must declare new permissions in the app manifest using the <permission> element, ensuring proper reference and control by dependent applications.

**Source:**

- ✓ V-222668: The system must alert an administrator when low resource conditions are encountered. Implement mechanisms to alert system administrators about a low resource condition [15].

**Priority:** Not described

**Rationale:** Declaring and managing permissions in the manifest provides a secure framework for access control, ensuring permissions are explicit and maintainable across interconnected applications.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

- 1. Manifest Declaration Test:**
  - ✓ Test: Verify that new permissions are declared in the app's manifest using the <permission> element.
  - ✓ Expected Result: All newly created permissions are correctly defined in the manifest with clear descriptions and protection levels.
- 2. Dependent App Reference Test:**
  - ✓ Test: Verify that dependent applications reference the declared permissions using the <uses-permission> element.
  - ✓ Expected Result: Dependent apps include appropriate <uses-permission> elements in their manifests, ensuring correct integration.
- 3. Protection Level Test:**
  - ✓ Test: Check that the declared permission specifies the correct protection level (normal, dangerous, signature).
  - ✓ Expected Result: The protection level aligns with the permission's intended use and security requirements.
- 4. Permission Scope Test:**
  - ✓ Test: Validate that the permission scope is limited to necessary functionalities.
  - ✓ Expected Result: No excessive or overly broad permissions are granted.
- 5. Permission Usage Logging Test:**

<ul style="list-style-type: none"> <li>✓ Test: Enable and monitor logs to track the usage of declared permissions by dependent applications.</li> <li>✓ Expected Result: Usage is logged appropriately, and no unauthorized access is detected.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-SMC-029]
<b>Requirement description:</b> Each service class in mobile health applications must have a <service> declaration in the AndroidManifest.xml file, with explicitly defined accessibility attributes to prevent unintentional exposure to other applications.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ Service Declaration in Manifest: Each service class must have a corresponding &lt;service&gt; declaration in the application's AndroidManifest.xml file. Ensure the android attribute is explicitly set to define whether the service is accessible to other applications. Avoid relying on implicit defaults, especially when intent filters are present [18].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Explicit service declarations and attribute settings prevent unauthorized access, reducing potential security vulnerabilities arising from implicit defaults or misconfigured intent filters.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Manifest Service Declaration Test:</b> <ul style="list-style-type: none"> <li>✓ <b>Test:</b> Verify that all service classes are declared using the &lt;service&gt; element in the AndroidManifest.xml file.</li> <li>✓ <b>Expected Result:</b> All service classes have corresponding &lt;service&gt; declarations.</li> </ul> </li> <li><b>2. Accessibility Attribute Test:</b> <ul style="list-style-type: none"> <li>✓ <b>Test:</b> Check the presence and correctness of the android:exported attribute for each declared service.</li> <li>✓ <b>Expected Result:</b> The android:exported attribute is explicitly set to either true or false as per the service's intended accessibility.</li> </ul> </li> <li><b>3. Intent Filter Evaluation Test:</b></li> </ol>

- ✓ Test: Analyze intent filters in the manifest to confirm services are not unintentionally exposed.
- ✓ Expected Result: Services with intent filters are configured to allow access only to authorized applications.

#### 4. Default Behavior Test:

- ✓ Test: Validate that no service relies on implicit defaults for accessibility attributes.
- ✓ Expected Result: All accessibility settings are explicitly defined in the manifest, avoiding reliance on default behaviors.

#### 5. Dynamic Testing of Service Access:

- ✓ Test: Simulate attempts by unauthorized applications to access the declared services.
- ✓ Expected Result: Unauthorized applications are unable to access services that are not explicitly shared.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda  
José L Fernández-Alemán  
Juan Manuel Carrillo-de-Gea  
Joaquín Nicolás  
08/01/2026

**PUID:** [SECM-CAT-SMC-030]

**Requirement description:** Mobile health applications must restrict access to services by using the android:permission attribute in the <service> declaration. For apps targeting Android 5.0 (API level 21) or higher, JobScheduler must be used instead of continuous background services to optimize resource usage and efficiency.

**Source:**

- ✓ Service Permission Protection for Services: Use the android attribute in the <service> declaration to restrict access. This ensures that only applications with the specified permission, declared via <uses-permission> in their manifest, can start, stop, or bind to the service [18].
- ✓ Use JobScheduler for Background Services (API 21+): For apps targeting Android 5.0 (API level 21) or higher, use JobScheduler instead of running background services continuously. This improves efficiency and reduces resource usage [18].

**Priority:** Not described

**Rationale:** Restricting service access using permissions ensures only authorized apps can interact with sensitive services, enhancing security. Utilizing JobScheduler improves resource management and compliance with modern platform requirements.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<p><b>1. Permission Restriction Test:</b></p> <ul style="list-style-type: none"> <li>✓ <b>Test:</b> Verify the android:permission attribute is specified in all &lt;service&gt; declarations in the manifest.</li> <li>✓ <b>Expected Result:</b> All &lt;service&gt; declarations include the android:permission attribute restricting access to authorized applications.</li> </ul> <p><b>2. Permission Reference Test:</b></p> <ul style="list-style-type: none"> <li>✓ <b>Test:</b> Confirm that permissions specified in &lt;service&gt; declarations are correctly referenced by other applications using &lt;uses-permission&gt; in their manifests.</li> <li>✓ <b>Expected Result:</b> Only applications with the required permissions can start, stop, or bind to the service.</li> </ul> <p><b>3. JobScheduler Implementation Test:</b></p> <ul style="list-style-type: none"> <li>✓ <b>Test:</b> Analyze services targeting Android 5.0+ to ensure background tasks are implemented using JobScheduler instead of continuous background services.</li> <li>✓ <b>Expected Result:</b> JobScheduler is used for background tasks, and no continuous background services are running unnecessarily.</li> </ul> <p><b>4. Dynamic Access Validation Test:</b></p> <ul style="list-style-type: none"> <li>✓ <b>Test:</b> Attempt to access restricted services without the specified permissions.</li> <li>✓ <b>Expected Result:</b> Unauthorized access is blocked, and services respond only to authorized apps.</li> </ul> <p><b>5. Resource Efficiency Test:</b></p> <ul style="list-style-type: none"> <li>✓ <b>Test:</b> Measure system resource usage with JobScheduler-enabled services versus continuous background services.</li> <li>✓ <b>Expected Result:</b> JobScheduler reduces resource consumption compared to continuous background services.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<p><b>Author and date:</b></p> <p>Carlos M. Mejía-Granda      José L Fernández-Alemán      Juan Manuel Carrillo-de-Gea      Joaquín Nicolás      08/01/2026</p>

<b>PUID:</b> [SECM-CAT-SMC-031]
<b>Requirement description:</b> Mobile health applications must explicitly manage export settings and permissions in the manifest to ensure sensitive services are not unintentionally exposed.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ Security Best Practices: Ensure sensitive services are not unintentionally exposed by explicitly managing export settings and permissions in the manifest [18].</li> </ul>

<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Export Setting Verification Test:</b> <ul style="list-style-type: none"> <li>✓ <b>Test:</b> Review the manifest to ensure all &lt;service&gt; declarations explicitly define the android:exported attribute.</li> <li>✓ <b>Expected Result:</b> Sensitive services have android:exported set to false unless external access is explicitly required.</li> </ul> </li> <li><b>2. Permission Implementation Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Verify that permissions are correctly applied to services requiring restricted access.</li> <li>✓ Expected Result: Services with sensitive functionality include proper permissions in the manifest.</li> </ul> </li> <li><b>3. Unauthorized Access Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Attempt to access services configured with restricted export settings from an unauthorized application.</li> <li>✓ Expected Result: Access is denied for unauthorized applications.</li> </ul> </li> <li><b>4. Manifest Security Review Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Perform a static analysis of the AndroidManifest.xml file to identify any services unintentionally exposed.</li> <li>✓ Expected Result: No sensitive services are unintentionally exposed.</li> </ul> </li> <li><b>5. Dynamic Security Audit Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Simulate runtime interactions with services to confirm proper enforcement of permissions and export settings.</li> <li>✓ Expected Result: Sensitive services are accessible only by authorized applications or processes.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda
José L Fernández-Alemán
Juan Manuel Carrillo-de-Gea
Joaquín Nicolás
08/01/2026

<b>PUID:</b> [SECM-CAT-SMC-032]
---------------------------------

<b>Requirement description:</b> Mobile health applications must restrict API key usage by applying IP address, app package name, or certificate fingerprint restrictions and monitor API usage for anomalies.
<b>Source:</b>
✓ Restrict API Key Usage: Use restrictions on API keys to limit their scope, such as restricting access by IP address, app package name, or certificate fingerprints. Implement usage quotas and monitor API usage to detect any suspicious activity. [18].
<b>Priority:</b> Not described
<b>Rationale:</b> Limiting API key usage to defined scopes and monitoring activity enhances security by mitigating unauthorized access and identifying potential misuse.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li>1. <b>Scope Restriction Configuration Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Verify that API keys are configured with scope restrictions (e.g., IP addresses, package names, or certificate fingerprints).</li> <li>✓ Expected Result: API keys are limited to the predefined scopes as per security requirements.</li> </ul> </li> <li>2. <b>Quota Enforcement Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Test API key usage against defined quotas to ensure usage limits are enforced.</li> <li>✓ Expected Result: Quotas are applied, and excess usage is blocked or flagged.</li> </ul> </li> <li>3. <b>Anomaly Detection Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Simulate unusual API usage patterns to confirm detection and alert mechanisms are active.</li> <li>✓ Expected Result: Suspicious activity triggers alerts for administrative review.</li> </ul> </li> <li>4. <b>Key Revocation Simulation Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Attempt to use an API key after it has been revoked.</li> <li>✓ Expected Result: Access is denied immediately upon key revocation.</li> </ul> </li> <li>5. <b>Dynamic Access Control Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Test that API keys restricted by IP or app package fail when accessed from unauthorized IPs or applications.</li> <li>✓ Expected Result: Unauthorized requests are blocked.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda
José L Fernández-Alemán
Juan Manuel Carrillo-de-Gea
Joaquín Nicolás
08/01/2026

<b>PUID:</b> [SECM-CAT-SMC-033]
<b>Requirement description:</b> Mobile health applications must implement regular API key rotation, adhering to ISO 27001 or similar standards, to reduce the risk of key exposure and unauthorized access.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ API Key Rotation and Revocation: Implement regular API key rotation to reduce the risk of long-term key exposure [18].</li> <li>✓ Compliance with ISO 27001 Key Management Framework: Ensure that key rotation and expiration policies are compliant with the ISO 27001 framework or similar security standards to maintain adherence to established information security practices [18].</li> <li>✓ Regular Key Rotation: API keys must be rotated regularly to minimize the risk of unauthorized access. The rotation period should range from 90 days to 6 months, following industry best practices and standards such as ISO 27001 [18].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Regular API key rotation ensures minimized exposure duration for keys, enhancing the security posture by reducing risks associated with stale or compromised credentials.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Key Rotation Policy Verification:</b> <ul style="list-style-type: none"> <li>✓ Test: Review the implemented policy to confirm it specifies key rotation periods (90 days to 6 months).</li> <li>✓ Expected Result: The policy mandates and enforces regular key rotation as per ISO 27001 standards or equivalent.</li> </ul> </li> <li><b>2. Automated Key Rotation Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Verify that API keys are automatically rotated according to the defined schedule without manual intervention.</li> <li>✓ Expected Result: Key rotation occurs seamlessly without affecting application functionality.</li> </ul> </li> <li><b>3. Expired Key Usage Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Attempt to use an expired API key post-rotation.</li> <li>✓ Expected Result: Access is denied for expired keys.</li> </ul> </li> <li><b>4. New Key Activation Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Test the system with a newly rotated API key to ensure proper functionality and access.</li> <li>✓ Expected Result: New keys function correctly while older keys are invalidated.</li> </ul> </li> </ol>

<b>5. Key Revocation Test:</b>
✓ Test: Simulate key compromise and test the revocation mechanism for an active API key.
✓ Expected Result: Revoked keys immediately lose access privileges, and alerts are generated.
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-SMC-034]
<b>Requirement description:</b> Mobile health applications must implement an automated key management system to rotate API keys at regular intervals, reducing human error and ensuring compliance with security policies.
<b>Source:</b>
✓ Automated Key Rotation Process: Implement a key management system that automates the process of rotating API keys at regular intervals, reducing human error and ensuring compliance with security policies [18].
<b>Priority:</b> Not described
<b>Rationale:</b> Automating API key rotation ensures consistency, minimizes human error, and enforces adherence to security best practices, safeguarding sensitive operations and data.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. System Setup Verification:</b> <ul style="list-style-type: none"> <li>✓ Test: Confirm the implementation of an automated key management system for API keys.</li> <li>✓ Expected Result: The system is operational and capable of automating key rotation without manual intervention.</li> </ul> </li> <li><b>2. Key Rotation Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Simulate a scheduled API key rotation event.</li> <li>✓ Expected Result: API keys are rotated automatically, and previous keys are invalidated without disrupting application functionality.</li> </ul> </li> <li><b>3. Compliance Audit:</b></li> </ol>

- ✓ Test: Verify that the automated key rotation intervals align with organizational security policies or standards such as ISO 27001.
- ✓ Expected Result: Key rotation schedules meet predefined policy requirements.

#### 4. Error Handling Test:

- ✓ Test: Introduce errors during key rotation (e.g., connectivity issues) to test the system's failover mechanisms.
- ✓ Expected Result: The system handles errors gracefully and retries or alerts administrators as needed.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda

José L Fernández-Alemán

Juan Manuel Carrillo-de-Gea

Joaquín Nicolás

08/01/2026

**PUID:** [SECM-CAT-SMC-035]

**Requirement description:** Mobile health applications must define and enforce an API key expiration policy to ensure keys are invalidated and replaced before expiration, reducing the risk of unauthorized access or misuse.

**Source:**

- ✓ Key Expiration Policy: Define an expiration policy for API keys that ensures keys are invalidated and replaced before they reach their expiration date to reduce the risk of misuse [18].

**Priority:** Not described

**Rationale:** Implementing a key expiration policy minimizes the risk of misuse and enhances overall security by ensuring that expired or unused API keys cannot be exploited.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

#### 1. Key Expiration Simulation:

- ✓ Test: Simulate an API key reaching its expiration date.
- ✓ Expected Result: The key is invalidated, and the system prompts for or performs a replacement without disrupting service.

#### 2. Automatic Expiration Check:

- ✓ Test: Verify that API keys are automatically checked against expiration dates.
- ✓ Expected Result: Expired keys are invalidated without requiring manual intervention.

<b>3. Replacement Workflow Verification:</b>
✓ Test: Validate the workflow for replacing expired keys with new ones.
✓ Expected Result: The replacement process is seamless, securely provisioning new keys while deactivating old ones.
<b>4. Alert System Test:</b>
✓ Test: Ensure that the system alerts administrators or developers before API keys approach expiration.
✓ Expected Result: Notifications are sent in advance, allowing time for key renewal or replacement.
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda
José L Fernández-Alemán
Juan Manuel Carrillo-de-Gea
Joaquín Nicolás
08/01/2026

<b>PUID:</b> [SECM-CAT-SMC-036]
<b>Requirement description:</b> Mobile health applications must implement notification and monitoring mechanisms to alert administrators before API key expiration, ensuring sufficient time for key rotation without service interruption.
<b>Source:</b>
✓ Notification and Monitoring for Key Expiry: The system must notify administrators before key expiration, providing sufficient time for the rotation process to complete without service interruption [18].
<b>Priority:</b> Not described
<b>Rationale:</b> Advance notification of key expiry ensures seamless operation by allowing timely key rotation, minimizing risks of downtime or unauthorized access due to expired keys.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
1. <b>Notification Trigger Test:</b>
✓ Test: Verify that the system triggers notifications at a configurable interval before the API key expiration date.
✓ Expected Result: Notifications are sent to designated administrators well in advance of the expiration date.
2. <b>Notification Content Review:</b>

- ✓ Test: Examine the notification messages for accuracy and completeness.
- ✓ Expected Result: Notifications include the key's identifier, expiration date, and instructions for renewal.

### 3. Rotation Timeline Verification:

- ✓ Test: Ensure that the notification timeline provides sufficient lead time for key rotation.
- ✓ Expected Result: The timeline aligns with organizational key rotation policies, allowing uninterrupted service.

### 4. Monitoring System Check:

- ✓ Test: Validate the monitoring system tracks API key status and updates expiration alerts dynamically.
- ✓ Expected Result: The monitoring system reflects real-time status of keys, including updates after renewal or revocation.

### 5. Failure Handling Test:

- ✓ Test: Simulate an API key nearing expiration without renewal.
- ✓ Expected Result: The system escalates notifications or logs critical alerts to prompt immediate administrative action.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda  
José L Fernández-Alemán  
Juan Manuel Carrillo-de-Gea  
Joaquín Nicolás  
08/01/2026

**PUID:** [SECM-CAT-SMC-037]

**Requirement description:** Mobile health applications must support the immediate revocation and replacement of compromised API keys.

**Source:**

- ✓ API Key Rotation and Revocation: In case of potential compromise, enable swift revocation of API keys and update the app to use new keys without disrupting user experience [18].
- ✓ Compliance with ISO 27001 Key Management Framework: Ensure that key rotation and expiration policies are compliant with the ISO 27001 framework or similar security standards to maintain adherence to established information security practices [18].

**Priority:** Not described

**Rationale:** Swift revocation and replacement of API keys in case of compromise mitigate potential risks while aligning with industry standards for secure key management.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Compromised Key Revocation Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Simulate a compromised API key scenario and trigger its revocation through the key management system.</li> <li>✓ Expected Result: The system revokes the key immediately and prevents further use.</li> </ul> </li> <li><b>2. Key Replacement Workflow Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Validate the process of updating the app with new API keys post-revocation.</li> <li>✓ Expected Result: The app continues functioning seamlessly with the new keys without disrupting user sessions.</li> </ul> </li> <li><b>3. Compliance Verification:</b> <ul style="list-style-type: none"> <li>✓ Test: Check that the key management policies align with ISO 27001 or similar frameworks.</li> <li>✓ Expected Result: Policies include rotation, expiration, and revocation mechanisms per industry standards.</li> </ul> </li> <li><b>4. Notification and Logging Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Verify that administrators receive alerts for key revocation events and that all actions are logged.</li> <li>✓ Expected Result: Notifications are sent promptly, and logs provide a detailed audit trail.</li> </ul> </li> <li><b>5. Failover Handling Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Test the app's behavior during the revocation of an API key in active use.</li> <li>✓ Expected Result: The app gracefully switches to the new key without exposing errors to end users.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-SMC-038]
<b>Requirement description:</b> Mobile health applications must implement certificate pinning to restrict trusted certificates to a predefined set, ensuring secure communication with backend servers and mitigating risks from compromised Certificate Authorities.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ General Best Practices: 7. Consider certificate pinning [3].</li> <li>✓ Certificate pinning: For an extra layer of security, you can consider implementing certificate pinning to check which certificates are considered valid [18].</li> <li>✓ 4.8. Introduce certificate pinning. Restrict an application's trusted certificates to a small set of known certificates that are used by the backend servers [16].</li> </ul>

<ul style="list-style-type: none"> <li>✓ iOS Specific Best Practices: 4. Consider using certificate pinning by doing the following: export your certificate, include it in your app bundle, and anchor it to your trust object. Using the NSURL method connection:willSendRequestForAuthenticationChallenge: will now accept your cert [3].</li> <li>✓ MASVS-NETWORK-2: The app performs identity pinning for all remote endpoints under the developer's control: Instead of trusting all the default root CAs of the framework or device, this control will make sure that only very specific CAs are trusted. This practice is typically called certificate pinning or public key pinning [20].</li> <li>✓ Secure network configuration: 1. Disallow cleartext traffic and use certificate pinning when possible [3].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Certificate pinning enhances security by limiting trust to specific certificates, reducing exposure to man-in-the-middle attacks and ensuring communication integrity.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Certificate Pinning Validation Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Verify the app only accepts connections to backend servers using the pinned certificates.</li> <li>✓ Expected Result: The app rejects connections with unpinned or invalid certificates.</li> </ul> </li> <li><b>2. Fallback Handling Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Simulate a scenario where the pinned certificate is revoked or expired.</li> <li>✓ Expected Result: The app fails securely and provides an appropriate error message to users.</li> </ul> </li> <li><b>3. Cleartext Traffic Prevention Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Attempt to establish a connection over HTTP instead of HTTPS.</li> <li>✓ Expected Result: The app rejects the connection and logs the event.</li> </ul> </li> <li><b>4. Dynamic Certificate Update Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Replace the backend server certificate with an updated pinned certificate.</li> <li>✓ Expected Result: The app seamlessly transitions to the new certificate without requiring user intervention.</li> </ul> </li> <li><b>5. Audit of Trust Anchors:</b> <ul style="list-style-type: none"> <li>✓ Test: Review the app's trust store to ensure it only includes pinned certificates or keys for trusted endpoints.</li> <li>✓ Expected Result: Only the expected certificates or public keys are listed in the trust store.</li> </ul> </li> </ol>

**Requested by:** The organization**Responsible:** Developer**Configurable value:** Not described**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda  
 José L Fernández-Alemán  
 Juan Manuel Carrillo-de-Gea  
 Joaquín Nicolás  
 08/01/2026

**PUID: [SECM-CAT-SMC-039]**

**Requirement description:** Mobile health applications must implement a streamlined API key revocation process to address security incidents or vulnerabilities, ensuring continuity of service by quickly issuing new keys.

**Source:**

- ✓ Streamlined Key Revocation: In the event of a security incident or detected vulnerability, the system must allow for immediate revocation of API keys and ensure new keys are issued quickly to maintain service continuity [18].

**Priority:** Not described

**Rationale:** Prompt API key revocation reduces the risk of unauthorized access during security incidents, while efficient reissuance ensures minimal service disruption.

**Number of Children:** 0**Number of parents:** 0**Cycles:** 0**Number Audit:** 0**Child PUIDs:** Not described**Parent PUIDs:** Not described**Exclusion PUIDs:** Not described**Importance:** Not described**Current state:** To be determined**Verification method:** Demonstration/Analysis**Validation criteria:****1. Revocation Process Test:**

- ✓ Test: Trigger an API key revocation event and attempt to use the revoked key for authentication.
- ✓ Expected Result: The revoked key is immediately invalid, and access is denied.

**2. Key Reissuance Validation Test:**

- ✓ Test: Initiate the process for issuing a new API key post-revocation.
- ✓ Expected Result: A new key is generated and distributed securely within an acceptable timeframe.

**3. Service Continuity Test:**

- ✓ Test: Simulate the revocation and replacement of an API key during ongoing service use.
- ✓ Expected Result: The application maintains uninterrupted access for valid users with the updated key.

**4. Audit Logging Test:**

- ✓ Test: Review system logs for API key revocation events.
- ✓ Expected Result: Logs capture the time, reason, and administrator responsible for the revocation.

**5. Notification System Test:**

<ul style="list-style-type: none"> <li>✓ Test: Verify that administrators are notified immediately upon key revocation, detailing the next steps for reissuance.</li> <li>✓ Expected Result: Notifications are promptly sent with clear instructions for handling key updates.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-SMC-040]
<b>Requirement description:</b> Mobile health applications must disable metadata publishing, such as WSDL documents, to prevent unintended exposure of sensitive service metadata.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ 5.3. Disable metadata publishing (e.g., metadata for WSDL documents and for WSDL derived objects), in order to prevent unintentional disclosure of potentially sensitive service metadata [16].</li> <li>✓ 8.1 User endpoint devices: Information stored on, processed by or accessible via user endpoint devices should be protected [6].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Disabling metadata publishing reduces the risk of exposing sensitive information that could assist attackers in understanding application structure and functionality.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Metadata Accessibility Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Attempt to access metadata files, including WSDL documents, through publicly accessible endpoints.</li> <li>✓ Expected Result: Requests for metadata files return an error or are blocked.</li> </ul> </li> <li><b>2. Configuration Review Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Inspect the application server configuration to verify metadata publishing is disabled.</li> <li>✓ Expected Result: Metadata publishing settings are explicitly disabled in the configuration.</li> </ul> </li> </ol>

<p><b>3. Security Assessment Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Conduct a security assessment of application endpoints to ensure metadata files are not accessible.</li> <li>✓ Expected Result: No sensitive metadata is found during the assessment.</li> </ul> <p><b>4. Functional Verification Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Verify that disabling metadata publishing does not impact the functionality of the application.</li> <li>✓ Expected Result: The application operates as expected without metadata publishing enabled.</li> </ul> <p><b>5. Audit Logging Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Review system logs for unauthorized access attempts to metadata endpoints.</li> <li>✓ Expected Result: Any unauthorized attempts are logged and flagged for review.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<p><b>Author and date:</b></p> <p>Carlos M. Mejía-Granda      José L Fernández-Alemán      Juan Manuel Carrillo-de-Gea      Joaquín Nicolás      08/01/2026</p>

<b>PUID:</b> [SECM-CAT-SMC-041]
<b>Requirement description:</b> Mobile health applications must control the export status of services by explicitly setting <code>android:exported</code> attributes to define accessibility.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ Service Export Control: Services are not exported by default unless they include intent filters. If the service needs to be accessed by other applications, explicitly set <code>android</code> <ul style="list-style-type: none"> <li>○ <code>= "true"</code>. Otherwise, ensure <code>android</code></li> <li>○ <code>= "false"</code> to keep the service private [18].</li> </ul> </li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Disabling metadata publishing reduces the risk of exposing sensitive information that could assist attackers in understanding application structure and functionality.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
1. Configuration Test:

<ul style="list-style-type: none"> <li>✓ <b>Test:</b> Inspect the <code>AndroidManifest.xml</code> file to verify the presence of the <code>android:exported</code> attribute for all service declarations.</li> <li>✓ <b>Expected Result:</b> Services without intent filters explicitly set <code>android:exported="false"</code>. Services with intent filters explicitly set <code>android:exported="true"</code> if required.</li> </ul>
<b>2. Access Control Test:</b>
<ul style="list-style-type: none"> <li>✓ Test: Attempt to access private services from unauthorized applications.</li> <li>✓ Expected Result: Unauthorized access attempts are blocked.</li> </ul>
<b>3. Dynamic Intent Handling Test:</b>
<ul style="list-style-type: none"> <li>✓ Test: Verify that services requiring public access handle intents securely and appropriately.</li> <li>✓ Expected Result: Only authorized intents are processed by public services.</li> </ul>
<b>4. Security Assessment Test:</b>
<ul style="list-style-type: none"> <li>✓ Test: Conduct a penetration test to ensure no unintended service exposure.</li> <li>✓ Expected Result: No private services are exposed to external applications.</li> </ul>
<b>5. Functional Verification Test:</b>
<ul style="list-style-type: none"> <li>✓ Test: Confirm that required functionality for exported services operates as intended.</li> <li>✓ Expected Result: Exported services are accessible only as specified, without exposing sensitive functionality.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

### 2.9.3.9 Insecure Data Storage

<b>PUID:</b> [SECM-CAT-IDS-001]
<b>Requirement description:</b> The mobile health application must ensure secure audit logging for all access, updates, or overrides of PHI. This includes logging user interactions with patient records, consent directive overrides, and access to locked or masked data. Logs must remain operational and accessible whenever the system is in use.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ 4.7. Communications and Operations Management</li> </ul> <p>Security Requirement 37 – Logging Transactions in the EHRi: The EHRi must create a secure audit record each time a user:</p> <ul style="list-style-type: none"> <li>a) Accesses, creates or updates PHI of a patient/person via the EHRi;</li> <li>b) Overrides the consent directives of a patient/person via the EHRi;</li> <li>c) Accesses, via the EHRi, data that is locked or masked by instruction of a patient/person; or</li> <li>d) Accesses, creates or updates registration data on an EHRi user</li> </ul>

<p>Security Requirement 41 – Logging Access to PHI in PoS Systems: All PoS systems connected to the EHRI must record in an audit log every instance of a user accessing, updating or archiving PHI [10].</p> <p>✓ 12.4.1 Event logging: The health information system's audit logging facility should be operational at all times while the health information system being audited is available for use [7].</p>
<b>Priority:</b> Not described
<b>Rationale:</b> Ensuring secure and consistent audit logging is critical for accountability and compliance with healthcare data protection regulations. It helps trace unauthorized access or modifications to sensitive health information, thereby improving overall system integrity.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Audit Logging Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Simulate user actions such as accessing, creating, or updating PHI records.</li> <li>✓ Expected result: All actions are logged with accurate timestamps, user identifiers, and descriptions of changes.</li> </ul> </li> <li><b>2. Consent Override Logging Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Perform an override of a patient's consent directive.</li> <li>✓ Expected result: The override is logged with the reason, timestamp, and user responsible.</li> </ul> </li> <li><b>3. Masked Data Access Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Access PHI data locked or masked per patient instructions.</li> <li>✓ Expected result: The access is logged, detailing the user, time, and purpose of access.</li> </ul> </li> <li><b>4. System Uptime Logging Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Verify that the audit logging facility is operational during system uptime.</li> <li>✓ Expected result: Logs are consistently recorded whenever the system is active, without interruptions.</li> </ul> </li> <li><b>5. Log Accessibility Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Attempt to retrieve audit logs during a session.</li> <li>✓ Expected result: Logs are accessible to authorized users with no loss of integrity or data.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda
José L Fernández-Alemán
Juan Manuel Carrillo-de-Gea
Joaquín Nicolás
08/01/2026

<b>PUID:</b> [SECM-CAT-IDS-002]
<b>Requirement description:</b> The mobile health application must ensure audit logs are retained for a period determined by clinical and legal standards, enabling after-the-fact investigation of security incidents. Additionally, the system must provide reporting capabilities to support these investigations.
<b>Source:</b>
<ul style="list-style-type: none"><li>✓ 12.4.1 Event logging: The organization should carefully assess and determine the retention period for these audit logs, with particular reference to clinical professional standards and legal obligations, in order to enable investigations to be carried out when necessary and to provide evidence of misuse where necessary [7].</li><li>✓ V-222494: The application must provide a report generation capability that supports after-the-fact investigations of security incidents. Design or configure the application to provide after-the-fact report generation capability or utilize a centralized utility designed for the purpose of log management and reporting [15].</li></ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Retention of audit logs ensures compliance with legal and professional standards while providing evidence for investigations into potential misuse or security incidents. Reporting capabilities enhance the ability to analyze logs efficiently and take corrective actions.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<b>1. Log Retention Verification:</b>
<ul style="list-style-type: none"><li>✓ Test: Verify that audit logs are retained according to the organization's defined retention policy.</li><li>✓ Expected result: Logs are available for the entire specified retention period without loss or corruption.</li></ul>
<b>2. After-the-Fact Reporting Test:</b>
<ul style="list-style-type: none"><li>✓ Test: Generate a report on a simulated security incident from past audit logs.</li><li>✓ Expected result: The report includes comprehensive details such as timestamps, user actions, and impacted data.</li></ul>
<b>3. Retention Policy Compliance Test:</b>
<ul style="list-style-type: none"><li>✓ Test: Cross-check log retention with clinical and legal retention requirements.</li><li>✓ Expected result: The retention duration aligns with the applicable standards and regulations.</li></ul>
<b>4. Log Accessibility Test:</b>
<ul style="list-style-type: none"><li>✓ Test: Attempt to retrieve specific audit logs from a prior defined period.</li><li>✓ Expected result: Logs are retrievable with accurate and complete data for the specified timeframe.</li></ul>
<b>5. Incident Report Accuracy Test:</b>

<ul style="list-style-type: none"> <li>✓ Test: Simulate a data access breach and generate a corresponding report.</li> <li>✓ Expected result: The report accurately identifies all related events, including unauthorized access attempts.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IDS-003]
<b>Requirement description:</b> The mobile health application must implement user audit controls that log the user's unique identifier, data subject identifier, performed function, and timestamp for all actions involving sensitive data.
<b>Source:</b> <ul style="list-style-type: none"> <li>✓ User audit controls:  Simple audits are in place. While additional security incident and event managers and system log aggregation tools are recommended to maximize security event analysis capabilities, aggregation and analytics tools like these are considered out of scope for this iteration [37].</li> <li>✓ 12.4.1 Event logging  The audit log should uniquely identify the user, uniquely identify the data subject (i.e. the subject of care), identify the function performed by the user (record creation, access, update, etc.), and note the time and date at which the function was performed [7].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Comprehensive audit logs provide essential details for investigating security events, ensuring accountability, and maintaining compliance with regulatory requirements for sensitive data handling in healthcare systems.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b> <ol style="list-style-type: none"> <li>1. <b>User Identification Logging Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Perform actions (e.g., create, access, update records) and verify the audit log entry for the user's unique identifier.</li> </ul> </li> </ol>

<ul style="list-style-type: none"> <li>✓ Expected result: The audit log contains the correct unique identifier for the user performing the action.</li> </ul>
<b>2. Data Subject Identification Logging Test:</b>
<ul style="list-style-type: none"> <li>✓ Test: Access or update sensitive data for a patient and verify the log entry for the data subject's unique identifier.</li> <li>✓ Expected result: The audit log accurately records the unique identifier of the data subject involved in the action.</li> </ul>
<b>3. Action Logging Test:</b>
<ul style="list-style-type: none"> <li>✓ Test: Execute various functions (e.g., record creation, access, update) and verify that the action type is captured in the log.</li> <li>✓ Expected result: The audit log clearly identifies the specific function performed by the user.</li> </ul>
<b>4. Timestamp Accuracy Test:</b>
<ul style="list-style-type: none"> <li>✓ Test: Perform an action and compare the recorded timestamp with the system clock.</li> <li>✓ Expected result: The audit log records an accurate and precise timestamp for the action.</li> </ul>
<b>5. Audit Data Completeness Test:</b>
<ul style="list-style-type: none"> <li>✓ Test: Review a random set of audit logs for user actions involving sensitive data.</li> <li>✓ Expected result: Each log entry includes the user's identifier, data subject identifier, action type, and timestamp with no missing fields.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IDS-004]
<b>Requirement description:</b> The mobile health application must generate audit records that log the start and end times of each user session to ensure accurate tracking of user activity.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ V-222464: The application must generate audit records showing starting and ending time for user access to the system. Configure the application or application server to record the start and end time of user session activity [15].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Tracking session activity enables effective monitoring of user access, supports security audits, and assists in identifying potential misuse or unauthorized access.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis

**Validation criteria:**

- 1. Session Start Time Logging Test:**
  - ✓ Test: Initiate a user session and verify the audit log entry for session start time.
  - ✓ Expected result: The audit log accurately records the timestamp of the session initiation.
- 2. Session End Time Logging Test:**
  - ✓ Test: End the user session and verify the audit log entry for session end time.
  - ✓ Expected result: The audit log accurately records the timestamp of the session termination.
- 3. Session Duration Test:**
  - ✓ Test: Compare the logged start and end times to calculate session duration.
  - ✓ Expected result: The calculated duration matches the actual session length.
- 4. Log Integrity Verification Test:**
  - ✓ Test: Attempt to modify the audit log entries for session start and end times.
  - ✓ Expected result: The system prevents unauthorized modifications, ensuring log integrity.
- 5. Concurrent Session Logging Test:**
  - ✓ Test: Perform multiple simultaneous user sessions and verify each session's start and end times are logged.
  - ✓ Expected result: Audit logs accurately track all concurrent sessions without overlap or data loss.

**Requested by:** The organization**Responsible:** Developer**Configurable value:** Not described**Version history:** v1.0**Author and date:**

Carlos M. Mejía-Granda  
 José L Fernández-Alemán  
 Juan Manuel Carrillo-de-Gea  
 Joaquín Nicolás  
 08/01/2026

**PUID:** [SECM-CAT-IDS-005]

**Requirement description:** The mobile health application must implement centralized audit logging with time-stamped correlation, unique application identifiers, and configurable log content to ensure comprehensive and secure audit record aggregation.

**Source:**

- ✓ V-222439: For applications providing audit record aggregation, the application must compile audit records from organization-defined information system components into a system-wide audit trail that is time-correlated with an organization-defined level of tolerance for the relationship between time stamps of individual records in the audit trail. Configure the application to correlate time stamps when aggregating audit records [15].
- ✓ V-222475: When using centralized logging; the application must include a unique identifier in order to distinguish itself from other application logs. Configure the application logs or the centralized log storage facility so the application name and the hosts hosting the application are uniquely identified in the logs [15].

<ul style="list-style-type: none"> <li>✓ V-222482: The application must be configured to write application logs to a centralized log repository. Configure the application to utilize a centralized log repository and ensure the logs are off-loaded from the application system as quickly as possible [15].</li> <li>✓ V-222480: The application must provide centralized management and configuration of the content to be captured in audit records generated by all application components. Configure the application to utilize a centralized log management system that provides the capability to configure the content of audit records [15].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Centralized audit management enhances the efficiency of tracking security events, ensures precise time correlation for multi-system analysis, and meets regulatory requirements for secure and detailed audit trails in healthcare systems.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Centralized Logging Verification Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Confirm that all audit records generated by the application are transmitted to a centralized log repository.</li> <li>✓ Expected result: Audit logs appear in the centralized repository with complete, time-stamped data.</li> </ul> </li> <li><b>2. Unique Identifier Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Generate logs from multiple instances of the application and verify each log entry includes a unique application identifier and host information.</li> <li>✓ Expected result: Each log entry includes a distinct application and host identifier for accurate log segregation.</li> </ul> </li> <li><b>3. Time Correlation Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Perform time-sensitive actions across distributed components and verify the correlation of timestamps in the centralized log.</li> <li>✓ Expected result: Time-stamps in the logs are synchronized within the organization-defined tolerance.</li> </ul> </li> <li><b>4. Log Content Configuration Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Modify the content settings for audit logs via the centralized management system and verify that changes are reflected in the log entries.</li> <li>✓ Expected result: The centralized log system captures updated log content based on the configured settings.</li> </ul> </li> <li><b>5. Log Offloading Efficiency Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Simulate high-volume log generation and verify that logs are promptly offloaded to the centralized repository.</li> <li>✓ Expected result: Logs are offloaded from the application system to the centralized repository without delay, ensuring system performance is not impacted.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described

<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda
José L Fernández-Alemán
Juan Manuel Carrillo-de-Gea
Joaquín Nicolás
08/01/2026

<b>PUID:</b> [SECM-CAT-IDS-006]
<b>Requirement description:</b> The mobile health application must off-load audit records to a separate system or media at scheduled intervals, ensuring secure and reliable log storage..
<b>Source:</b>
✓ V-222481: The application must off-load audit records onto a different system or media than the system being audited. Configure the application to off-load audit records onto a different system as per approved schedule [15].
<b>Priority:</b> Not described
<b>Rationale:</b> Off-loading audit records to a different system enhances data integrity, prevents log tampering, and ensures availability for forensic analysis in case of system compromise.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
1. <b>Off-Loading Configuration Test:</b>
✓ Test: Verify the configuration settings for off-loading audit logs to a separate system or media.
✓ Expected result: The system is correctly configured to off-load logs to the designated external location.
2. <b>Scheduled Off-Loading Test:</b>
✓ Test: Simulate a scheduled log off-loading event and confirm the logs are successfully transferred to the separate system.
✓ Expected result: Audit records are securely off-loaded to the external system at the scheduled intervals without errors.
3. <b>Data Integrity Test:</b>
✓ Test: Compare the off-loaded logs with the original logs on the application system to ensure data integrity during transfer.
✓ Expected result: The content of the logs matches exactly between the original and the off-loaded records.
4. <b>Tamper Detection Test:</b>
✓ Test: Attempt unauthorized modifications on the off-loaded logs and verify that tamper-proof mechanisms are in place.
✓ Expected result: The off-loaded logs remain protected, with any tampering attempts logged and reported.
5. <b>System Resource Efficiency Test:</b>

<ul style="list-style-type: none"> <li>✓ Test: Monitor system performance during the log off-loading process.</li> <li>✓ Expected result: The off-loading process does not significantly impact the application's performance or availability.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IDS-007]
<b>Requirement description:</b> The mobile health application must provide immediate warnings to system administrators (SA) and Information System Security Officers (ISSO) when the audit record storage reaches 75% of its maximum capacity.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ V-222483: The application must provide an immediate warning to the SA and ISSO (at a minimum) when allocated audit record storage volume reaches 75% of repository maximum audit record storage capacity. Configure the application to send an immediate alarm to the application admin/SA and the ISSO when the allocated log storage capacity exceeds 75% of usage or exceeds the capacity value the SA and ISSO have determined will provide adequate time to plan for capacity expansion [15].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Early alerts for storage capacity issues ensure timely action to prevent potential data loss or disruption in audit logging, which is critical for compliance and forensic investigations.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li>1. <b>Capacity Threshold Configuration Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Verify the application's configuration for defining storage capacity thresholds for audit records.</li> <li>✓ Expected result: The threshold for sending alerts is set at 75% of the total storage capacity or a custom value defined by the SA and ISSO.</li> </ul> </li> <li>2. <b>Warning Notification Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Simulate audit storage usage exceeding the 75% threshold and check for immediate warnings sent to SA and ISSO.</li> </ul> </li> </ol>

<ul style="list-style-type: none"> <li>✓ Expected result: Notifications are delivered promptly to designated recipients, including SA and ISSO, with clear details about storage usage.</li> </ul>
<b>3. Notification Accuracy Test:</b>
<ul style="list-style-type: none"> <li>✓ Test: Review the content of the alert messages.</li> <li>✓ Expected result: Alerts include precise information, such as the current storage usage percentage, remaining capacity, and urgency of action required.</li> </ul>
<b>4. Logging and Alert Tracking Test:</b>
<ul style="list-style-type: none"> <li>✓ Test: Verify that alerts are logged for audit purposes and can be reviewed later for administrative and compliance needs.</li> <li>✓ Expected result: All alerts related to storage thresholds are recorded in a log accessible by authorized personnel.</li> </ul>
<b>5. System Performance Test:</b>
<ul style="list-style-type: none"> <li>✓ Test: Confirm that generating and sending storage alerts does not disrupt application performance or logging functionality.</li> <li>✓ Expected result: The application remains fully operational while generating and delivering alerts.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IDS-008]
<b>Requirement description:</b> The mobile health application must provide real-time alerts to system administrators (SA) and Information System Security Officers (ISSO) for all audit failure events, ensuring immediate notification when the audit system is at risk of failure or has failed.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ V-222484: Applications categorized as having a moderate or high impact must provide an immediate real-time alert to the SA and ISSO (at a minimum) for all audit failure events. Configure the log alerts to send an alarm when the audit system is in danger of failing or has failed. Configure the log alerts to be immediately sent to the application admin/SA and ISSO [15].</li> <li>✓ V-222485: The application must alert the ISSO and SA (at a minimum) in the event of an audit processing failure. Configure the application to send an alarm in the event the audit system has failed or is failing [15].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Real-time alerts for audit system failures are critical to maintaining compliance, ensuring data integrity, and addressing issues promptly to avoid data loss or breaches.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described

<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Audit System Failure Simulation Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Simulate an audit system failure or near-failure scenario.</li> <li>✓ Expected result: Immediate real-time alerts are sent to designated SA and ISSO contacts with details about the failure.</li> </ul> </li> <li><b>2. Alert Notification Content Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Review the content of the alerts sent during an audit failure event.</li> <li>✓ Expected result: Alerts clearly identify the issue, its impact on audit logging, and any recommended actions.</li> </ul> </li> <li><b>3. Alert Delivery Mechanism Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Verify that alerts are delivered via the predefined communication channels (e.g., email, SMS, dashboard notifications).</li> <li>✓ Expected result: Alerts are delivered promptly without delay through all configured channels.</li> </ul> </li> <li><b>4. Logging of Alert Events Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Check whether alert events are logged for future reference.</li> <li>✓ Expected result: Each alert sent for an audit system failure is recorded in a log accessible to authorized personnel.</li> </ul> </li> <li><b>5. System Performance During Alert Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Confirm that the application's performance remains unaffected during the generation and sending of real-time alerts.</li> <li>✓ Expected result: The application functions normally without interruptions to other services while generating and delivering alerts</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IDS-009]
<b>Requirement description:</b> The mobile health application must initiate session auditing immediately upon startup, ensuring all application events are logged from the start of the session.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ V-222468: The application must initiate session auditing upon startup. Configure the application to begin logging application events as soon as the application starts up [15].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Early initiation of session auditing helps maintain a complete audit trail, ensuring no critical events are missed during application startup, which is vital for security monitoring and compliance.

<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Startup Audit Log Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Start the application and inspect the generated logs.</li> <li>✓ Expected result: Logging begins immediately upon application startup, capturing all relevant session events from the start.</li> </ul> </li> <li><b>2. Audit Trail Completeness Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Compare the startup logs with expected session initialization activities.</li> <li>✓ Expected result: All session initialization events are recorded accurately in the audit logs.</li> </ul> </li> <li><b>3. Session Initialization Timing Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Measure the time taken from application startup to the first log entry.</li> <li>✓ Expected result: The first log entry occurs within milliseconds of the application initialization.</li> </ul> </li> <li><b>4. Application Configuration Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Review the application configuration for settings related to audit logging.</li> <li>✓ Expected result: Configuration includes mandatory initiation of session auditing upon application startup.</li> </ul> </li> <li><b>5. Error Handling Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Simulate an issue during startup that could affect logging initiation.</li> <li>✓ Expected result: The application generates an error alert and captures the issue in the audit logs.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda
José L Fernández-Alemán
Juan Manuel Carrillo-de-Gea
Joaquín Nicolás
08/01/2026

<b>PUID:</b> [SECM-CAT-IDS-010]
<b>Requirement description:</b> The mobile health application must generate audit records for any instance of concurrent logons from different devices or workstations, ensuring detailed tracking of such activities.
<b>Source:</b>

✓ V-222672: The application must generate audit records when concurrent logons from different workstations occur. Configure the application to log concurrent logons from different workstations [15].
<b>Priority:</b> Not described
<b>Rationale:</b> Logging concurrent logons from multiple workstations enhances the detection of potentially unauthorized access or credential sharing, improving the application's security and compliance with audit requirements.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li>1. <b>Concurrent Logon Detection Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Log into the application simultaneously from two different devices or workstations using the same credentials.</li> <li>✓ Expected result: Audit records are generated, capturing details of both logon events, including timestamps, device identifiers, and IP addresses.</li> </ul> </li> <li>2. <b>Audit Log Consistency Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Review the audit logs after concurrent logons.</li> <li>✓ Expected result: Logs clearly indicate that the same user accessed the system from different devices or workstations concurrently.</li> </ul> </li> <li>3. <b>Logon Alert Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Simulate concurrent logons from different devices and monitor for alerts.</li> <li>✓ Expected result: The application generates alerts (if configured) to notify the system administrator of concurrent logons.</li> </ul> </li> <li>4. <b>Configuration Review Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Inspect application configuration for parameters governing audit record generation for concurrent logons.</li> <li>✓ Expected result: Configuration explicitly includes concurrent logon tracking and ensures log generation for such events.</li> </ul> </li> <li>5. <b>Stress Test for Multiple Logons:</b> <ul style="list-style-type: none"> <li>✓ Test: Simulate multiple concurrent logons across a large number of devices to assess performance and logging accuracy.</li> <li>✓ Expected result: The application successfully generates accurate audit records without performance degradation or missed entries.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IDS-011]
<b>Requirement description:</b> The mobile health application must log all shutdown events in the event logs to ensure comprehensive audit trail coverage.
<b>Source:</b> <ul style="list-style-type: none"><li>✓ V-222469: The application must log application shutdown events. Configure the application or application server to record application shutdown events in the event logs [15].</li></ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Logging shutdown events provides critical insights for troubleshooting and security auditing, enabling the detection of unexpected or unauthorized shutdowns that may indicate potential security incidents or system issues.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b> <ol style="list-style-type: none"><li><b>1. Shutdown Event Logging Test:</b><ul style="list-style-type: none"><li>✓ Test: Initiate a controlled shutdown of the application.</li><li>✓ Expected result: A log entry is created with details of the shutdown event, including the timestamp, triggering user or process, and system status at shutdown.</li></ul></li><li><b>2. Unexpected Shutdown Test:</b><ul style="list-style-type: none"><li>✓ Test: Force the application to close unexpectedly (e.g., simulate a crash or abrupt termination).</li><li>✓ Expected result: A log entry is created documenting the abrupt shutdown, including a note on whether the shutdown was abnormal.</li></ul></li><li><b>3. Log Content Verification Test:</b><ul style="list-style-type: none"><li>✓ Test: Review the application event logs for shutdown entries.</li><li>✓ Expected result: Shutdown logs include essential information such as the timestamp, initiating user/process, and reason (normal or abnormal termination).</li></ul></li><li><b>4. Configuration Review Test:</b><ul style="list-style-type: none"><li>✓ Test: Inspect the application server configuration for settings that ensure shutdown events are logged.</li><li>✓ Expected result: Configuration explicitly mandates the logging of all application shutdown events.</li></ul></li><li><b>5. Log Integrity Test:</b><ul style="list-style-type: none"><li>✓ Test: Attempt to modify or suppress shutdown event logs.</li><li>✓ Expected result: The application prevents tampering or ensures logs maintain integrity, with any modification attempts being logged separately</li></ul></li></ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>

Carlos M. Mejía-Granda  
José L Fernández-Alemán  
Juan Manuel Carrillo-de-Gea  
Joaquín Nicolás  
08/01/2026

**PUID:** [SECM-CAT-IDS-012]

**Requirement description:** The mobile health application must generate and log audit records for all successful and unsuccessful logon attempts, and raise security events for detected log-on breaches.

**Source:**

- ✓ V-222462: The application must generate audit records when successful/unsuccessful logon attempts occur. Configure the application or application server to write a log entry when successful and unsuccessful logon events occur [15].
- ✓ 9.4.2 Secure log-on procedures: f) log unsuccessful and successful attempts; [7].
- ✓ 9.4.2 Secure log-on procedures: g) raise a security event if a potential attempted or successful breach of log-on controls is detected; [7]
- ✓ 164.308 Administrative safeguards.
  - (a) A covered entity or business associate must, in accordance with § 164.306:
    - (5)
      - (ii) Implementation specifications:
        - (B) Security reminders (Addressable)
          - i. Log-in monitoring (Addressable)

Procedures for monitoring log-in attempts and reporting discrepancies [9].

**Priority:** Not described

**Rationale:** Logging and monitoring logon events provide critical insights into potential unauthorized access attempts and user activity, helping to maintain security and comply with regulatory requirements.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

1. **Successful Logon Test:**
  - ✓ Test: Perform a valid logon with correct credentials.
  - ✓ Expected result: A log entry is created documenting the successful logon, including the timestamp, user ID, and source IP/device.
2. **Unsuccessful Logon Test:**

<ul style="list-style-type: none"> <li>✓ Test: Attempt logons with incorrect credentials (e.g., invalid username/password combinations).</li> <li>✓ Expected result: Log entries are created for each failed attempt, capturing the timestamp, user ID, and source IP/device.</li> </ul>
<b>3. Security Event Test:</b>
<ul style="list-style-type: none"> <li>✓ Test: Simulate a brute-force attack or repeated logon failures.</li> <li>✓ Expected result: A security event is triggered, and administrators are notified of potential log-on breaches.</li> </ul>
<b>4. Log Content Verification Test:</b>
<ul style="list-style-type: none"> <li>✓ Test: Review log entries for both successful and unsuccessful logon events.</li> <li>✓ Expected result: Logs include timestamps, user details, result (success/failure), and the originating device/IP.</li> </ul>
<b>5. Compliance Review Test:</b>
<ul style="list-style-type: none"> <li>✓ Test: Compare logging and notification mechanisms against HIPAA § 164.308 and organizational policies.</li> <li>✓ Expected result: The application complies with log-in monitoring and reporting requirements, ensuring audit trail completeness and security.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IDS-013]
<b>Requirement description:</b> The mobile health application must generate and log audit records for privileged activities and system-level access events.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ V-222463: The application must generate audit records for privileged activities or other system-level access. Configure the application to write a log entry when privileged activities or other system-level events occur [15].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Logging privileged activities and system-level access helps detect and prevent unauthorized or malicious actions, ensuring compliance with security policies and regulations.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>

- 1. Privileged Action Log Test:**
  - ✓ Test: Perform a privileged action, such as modifying system configurations or accessing restricted data.
  - ✓ Expected result: A log entry is created capturing the action type, user ID, timestamp, and originating device.
- 2. Unauthorized Privileged Access Attempt Test:**
  - ✓ Test: Attempt a privileged action with insufficient permissions.
  - ✓ Expected result: A log entry is created documenting the failed attempt, including details of the user, action, and timestamp.
- 3. Log Content Verification Test:**
  - ✓ Test: Review audit logs for privileged actions.
  - ✓ Expected result: Logs include the nature of the action, associated user ID, success/failure status, and detailed contextual metadata.
- 4. Alert Generation Test:**
  - ✓ Test: Trigger multiple unauthorized privileged access attempts within a short time frame.
  - ✓ Expected result: The system raises an alert and notifies the administrator of potential misuse.
- 5. Compliance Verification Test:**
  - ✓ Test: Compare logging mechanisms with organizational policies and regulatory standards (e.g., HIPAA).
  - ✓ Expected result: Logging meets requirements for tracking privileged access and system-level events comprehensively.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda  
 José L Fernández-Alemán  
 Juan Manuel Carrillo-de-Gea  
 Joaquín Nicolás  
 08/01/2026

**PUID:** [SECM-CAT-IDS-014]

**Requirement description:** The mobile health application must generate audit records for both successful and unsuccessful attempts to grant user privileges.

**Source:**

- ✓ V-222450: The application must generate audit records when successful/unsuccessful attempts to grant privileges occur. Configure the application to audit successful and unsuccessful attempts to grant privileges [15].

**Priority:** Not described

**Rationale:** Auditing privilege grant attempts ensures accountability and helps detect unauthorized or malicious attempts to escalate user access, enhancing security compliance.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Successful Privilege Grant Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Assign additional privileges to a user account.</li> <li>✓ Expected result: A log entry is generated documenting the success of the operation, including the user performing the action, the target user, timestamp, and the privileges granted.</li> </ul> </li> <li><b>2. Failed Privilege Grant Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Attempt to assign privileges without the necessary authorization.</li> <li>✓ Expected result: A log entry is created capturing the failed attempt, including the user performing the action, the target user, timestamp, and reason for failure.</li> </ul> </li> <li><b>3. Log Content Verification Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Inspect the generated audit logs for privilege grant attempts.</li> <li>✓ Expected result: Logs contain detailed information, such as the initiating user, target user, privileges involved, success/failure status, and associated timestamps.</li> </ul> </li> <li><b>4. Privilege Escalation Monitoring Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Trigger multiple privilege grant attempts in a short period.</li> <li>✓ Expected result: The system identifies suspicious activity, creates appropriate log entries, and sends an alert to the administrator.</li> </ul> </li> <li><b>5. Compliance Verification Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Compare the application's privilege auditing mechanisms with regulatory and organizational requirements (e.g., HIPAA or ISO 27001).</li> <li>✓ Expected result: The system's auditing approach complies with mandated standards for logging privilege-related activities</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IDS-015]
<b>Requirement description:</b> The mobile health application must generate audit records for both successful and unsuccessful attempts to access security or application objects.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ V-222451: The application must generate audit records when successful/unsuccessful attempts to access security objects occur. Configure the application to create an audit record for both successful and unsuccessful attempts to access security objects [15].</li> </ul>

✓ V-222465: The application must generate audit records when successful/unsuccessful accesses to objects occur. Configure the application to log successful and unsuccessful access to application objects [15].
<b>Priority:</b> Not described
<b>Rationale:</b> Monitoring access to security and application objects ensures unauthorized attempts are identified, and security compliance is maintained, protecting sensitive health information.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Successful Object Access Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Access a security-sensitive object with valid permissions.</li> <li>✓ Expected result: A log entry is created recording the user, object accessed, timestamp, and access type (e.g., read, write).</li> </ul> </li> <li><b>2. Unsuccessful Object Access Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Attempt to access a security-sensitive object without sufficient permissions.</li> <li>✓ Expected result: A log entry is created capturing the failed access attempt, including the user, target object, timestamp, and reason for denial.</li> </ul> </li> <li><b>3. Comprehensive Audit Verification Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Review audit logs after a series of successful and unsuccessful object access attempts.</li> <li>✓ Expected result: Logs should reflect detailed information for each access attempt, including user ID, object ID, access type, outcome (success/failure), and timestamps.</li> </ul> </li> <li><b>4. Alert Trigger Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Attempt multiple unauthorized accesses to a security-sensitive object in a short timeframe.</li> <li>✓ Expected result: The system generates audit logs for each attempt and triggers an alert for potential malicious activity.</li> </ul> </li> <li><b>5. Compliance Review Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Evaluate the application's logging mechanism against relevant security standards (e.g., HIPAA, ISO 27001).</li> <li>✓ Expected result: The logging functionality meets or exceeds compliance requirements for security and object access auditing.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda
José L Fernández-Alemán
Juan Manuel Carrillo-de-Gea
Joaquín Nicolás
08/01/2026

<b>PUID:</b> [SECM-CAT-IDS-016]
<b>Requirement description:</b> The mobile health application must generate audit records for both successful and unsuccessful attempts to modify security objects.
<b>Source:</b> <ul style="list-style-type: none"><li>✓ V-222455: The application must generate audit records when successful/unsuccessful attempts to modify security objects occur. Configure the application to create an audit record for both successful and unsuccessful attempts to modify security objects [15].</li></ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Capturing modification attempts on security objects ensures any unauthorized or unintended changes are detected and helps maintain the integrity and confidentiality of sensitive health data.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b> <ol style="list-style-type: none"><li><b>1. Successful Modification Test:</b><ul style="list-style-type: none"><li>✓ Test: Modify a security-sensitive object with appropriate permissions.</li><li>✓ Expected result: A log entry is created capturing the user ID, object modified, timestamp, and details of the changes made.</li></ul></li><li><b>2. Unsuccessful Modification Test:</b><ul style="list-style-type: none"><li>✓ Test: Attempt to modify a security-sensitive object without the necessary permissions.</li><li>✓ Expected result: A log entry is generated recording the failed attempt, including the user ID, object targeted, timestamp, and reason for denial.</li></ul></li><li><b>3. Audit Log Verification Test:</b><ul style="list-style-type: none"><li>✓ Test: Perform a series of successful and unsuccessful modification attempts on security objects and review the corresponding audit logs.</li><li>✓ Expected result: Logs accurately reflect the events, including user identity, object details, modification type, success/failure status, and timestamps.</li></ul></li><li><b>4. Unauthorized Changes Alert Test:</b><ul style="list-style-type: none"><li>✓ Test: Simulate multiple unauthorized modification attempts on security-sensitive objects within a defined period.</li><li>✓ Expected result: The system generates logs for each attempt and triggers an alert for possible malicious activity.</li></ul></li><li><b>5. Compliance Audit Test:</b><ul style="list-style-type: none"><li>✓ Test: Assess the logging mechanism against applicable standards or regulations (e.g., HIPAA, GDPR).</li><li>✓ Expected result: Logging features comply with regulatory requirements, supporting detailed auditing of security object modifications.</li></ul></li></ol>
<b>Requested by:</b> The organization

<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IDS-017]
<b>Requirement description:</b> The mobile health application must generate audit records for successful and unsuccessful attempts to delete database security objects.
<b>Source:</b>
✓ V-222460: The application must generate audit records when successful/unsuccessful attempts to delete application database security objects occur. Configure the application to create an audit record for both successful and unsuccessful attempts to delete database security objects [15].
<b>Priority:</b> Not described
<b>Rationale:</b> Logging deletion attempts on database security objects ensures accountability, detects potential misuse, and preserves the integrity of sensitive healthcare data.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<b>1. Successful Deletion Test:</b>
✓ Test: Delete a security-sensitive database object with appropriate permissions. ✓ Expected result: A log entry is created capturing the user ID, object deleted, timestamp, and details of the deletion action.
<b>2. Unsuccessful Deletion Test:</b>
✓ Test: Attempt to delete a security-sensitive database object without the necessary permissions. ✓ Expected result: A log entry is generated recording the failed attempt, including the user ID, object targeted, timestamp, and reason for denial.
<b>3. Audit Log Accuracy Test:</b>
✓ Test: Perform both successful and unsuccessful deletion attempts on database security objects and review the audit logs. ✓ Expected result: Logs accurately reflect all deletion attempts, with detailed information such as user identity, object details, success/failure status, and timestamps.
<b>4. Unauthorized Deletion Alert Test:</b>

<ul style="list-style-type: none"> <li>✓ Test: Simulate multiple unauthorized deletion attempts on security-sensitive database objects within a short timeframe.</li> <li>✓ Expected result: The system generates logs for each attempt and triggers an alert for potential malicious activity.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<p><b>Author and date:</b>            Carlos M. Mejía-Granda            José L Fernández-Alemán            Juan Manuel Carrillo-de-Gea            Joaquín Nicolás            08/01/2026</p>

<b>PUID:</b> [SECM-CAT-IDS-018]
<b>Requirement description:</b> The mobile health application must generate audit records for successful and unsuccessful attempts to access security levels.
<p><b>Source:</b></p> <ul style="list-style-type: none"> <li>✓ V-222452: The application must generate audit records when successful/unsuccessful attempts to access security levels occur. Configure the application to create an audit record for both successful and unsuccessful attempts to access security levels. [15].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Capturing attempts to access security levels ensures accountability, helps identify unauthorized access attempts, and protects sensitive healthcare data from misuse or compromise.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<p><b>Validation criteria:</b></p> <ol style="list-style-type: none"> <li><b>1. Successful Access Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Access a specified security level with appropriate credentials and permissions.</li> <li>✓ Expected result: A log entry is created with details including the user ID, accessed security level, timestamp, and the action result.</li> </ul> </li> <li><b>2. Unsuccessful Access Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Attempt to access a security level without the necessary permissions or with incorrect credentials.</li> <li>✓ Expected result: A log entry is generated capturing the failed access attempt, including the user ID, security level targeted, timestamp, and reason for denial.</li> </ul> </li> <li><b>3. Audit Log Completeness Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Perform multiple access attempts (both successful and unsuccessful) across various security levels and review the generated audit logs.</li> </ul> </li> </ol>

<ul style="list-style-type: none"> <li>✓ Expected result: All attempts are accurately recorded with necessary details such as user identity, security level, success/failure status, and timestamps.</li> </ul>
<b>4. Alert on Repeated Unsuccessful Access:</b>
<ul style="list-style-type: none"> <li>✓ Test: Simulate repeated unauthorized access attempts to a high-security level.</li> <li>✓ Expected result: The system generates logs for each attempt and triggers an alert to notify administrators of potential malicious activity.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IDS-019]
<b>Requirement description:</b> The mobile health application must generate audit records for all successful and unsuccessful attempts to modify security levels.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ V-222456: The application must generate audit records when successful/unsuccessful attempts to modify security levels occur. Configure the application to create an audit record for both successful and unsuccessful attempts to modify security levels [15].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Recording modifications to security levels ensures traceability, accountability, and early detection of potential misuse, which is critical for safeguarding sensitive health data in compliance with regulatory standards.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Successful Modification Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Modify a security level using appropriate credentials and permissions.</li> <li>✓ Expected result: A log entry is created, detailing the user ID, security level modified, timestamp, changes made, and the result of the action.</li> </ul> </li> <li><b>2. Unsuccessful Modification Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Attempt to modify a security level without sufficient permissions or with incorrect credentials.</li> <li>✓ Expected result: A log entry is generated, capturing the user ID, security level targeted, timestamp, and reason for the failed attempt.</li> </ul> </li> <li><b>3. Comprehensive Audit Verification Test:</b></li> </ol>

<ul style="list-style-type: none"> <li>✓ Test: Perform a series of modifications (both successful and unsuccessful) to various security levels, then review the generated audit logs.</li> <li>✓ Expected result: All attempts are logged accurately with complete details, including the action result and timestamps.</li> </ul> <p><b>4. Alert on Unauthorized Modifications:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Simulate repeated unauthorized modification attempts to a high-security level.</li> <li>✓ Expected result: The system logs each attempt and triggers an alert to notify administrators of potential security threats.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<p><b>Author and date:</b></p> <p>Carlos M. Mejía-Granda      José L Fernández-Alemán      Juan Manuel Carrillo-de-Gea      Joaquín Nicolás      08/01/2026</p>

<b>PUID:</b> [SECM-CAT-IDS-020]
<b>Requirement description:</b> The mobile health application must generate audit records for all successful and unsuccessful attempts to access protected categories of information, such as classification levels.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ V-222453: The application must generate audit records when successful/unsuccessful attempts to access categories of information (e.g., classification levels) occur. Configure the application to create an audit record for both successful and unsuccessful attempts to access protected categories of information [15].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Logging access to protected categories of information ensures compliance with healthcare data privacy regulations, supports forensic investigations, and enhances accountability for safeguarding sensitive health data.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Successful Access Test:</b></li> </ol> <ul style="list-style-type: none"> <li>✓ Test: Access a protected category of information with valid credentials and permissions.</li> </ul>

<ul style="list-style-type: none"> <li>✓ Expected result: An audit log is generated, capturing the user ID, category accessed, timestamp, and outcome of the access.</li> </ul>
<b>2. Unsuccessful Access Test:</b>
<ul style="list-style-type: none"> <li>✓ Test: Attempt to access a protected category of information without sufficient permissions or using invalid credentials.</li> <li>✓ Expected result: An audit log entry is created, documenting the user ID, category targeted, timestamp, and reason for the failed attempt.</li> </ul>
<b>3. Detailed Log Verification Test:</b>
<ul style="list-style-type: none"> <li>✓ Test: Perform a series of successful and unsuccessful attempts to access various categories of protected information.</li> <li>✓ Expected result: Audit logs contain detailed entries for each attempt, including user ID, timestamp, action result, and category accessed.</li> </ul>
<b>4. Alert for Repeated Unauthorized Access Attempts:</b>
<ul style="list-style-type: none"> <li>✓ Test: Simulate multiple unauthorized access attempts to a high-security category.</li> <li>✓ Expected result: The system generates logs for each attempt and triggers an alert to notify administrators of potential security threats.</li> </ul>
<b>5. Compliance Validation Test:</b>
<ul style="list-style-type: none"> <li>✓ Test: Assess the logging mechanism for access attempts to protected categories against relevant healthcare compliance standards (e.g., HIPAA, GDPR).</li> <li>✓ Expected result: Audit records meet the required standards for documentation and monitoring of access to sensitive information.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IDS-021]
<b>Requirement description:</b> The mobile health application must generate audit records for all successful and unsuccessful attempts to modify protected categories of information, such as classification levels.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ V-222457: The application must generate audit records when successful/unsuccessful attempts to modify categories of information (e.g., classification levels) occur. Configure the application to create an audit record for both successful and unsuccessful attempts to modify protected categories of information [15].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Logging modification attempts to protected information categories ensures data integrity, supports regulatory compliance, and enables effective monitoring of unauthorized changes to sensitive health data.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0

<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Successful Modification Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Modify a protected category of information with valid credentials and permissions.</li> <li>✓ Expected result: An audit log entry is generated, capturing the user ID, category modified, timestamp, and details of the modification.</li> </ul> </li> <li><b>2. Unsuccessful Modification Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Attempt to modify a protected category of information without sufficient permissions or using invalid credentials.</li> <li>✓ Expected result: An audit log entry is created, documenting the user ID, category targeted, timestamp, and reason for the failed modification attempt.</li> </ul> </li> <li><b>3. Detailed Log Verification Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Perform a series of successful and unsuccessful attempts to modify various categories of protected information.</li> <li>✓ Expected result: Audit logs contain detailed entries for each attempt, including user ID, timestamp, modification result, and category modified.</li> </ul> </li> <li><b>4. Alert for Unauthorized Modification Attempts:</b> <ul style="list-style-type: none"> <li>✓ Test: Simulate multiple unauthorized modification attempts to high-security categories.</li> <li>✓ Expected result: The system generates logs for each attempt and triggers an alert to notify administrators of potential security threats.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IDS-022]
<b>Requirement description:</b> The mobile health application must generate audit records for all successful and unsuccessful attempts to delete protected categories of information, such as classification levels.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ V-222461: The application must generate audit records when successful/unsuccessful attempts to delete categories of information (e.g., classification levels) occur. Configure the application to create an audit record for both successful and unsuccessful attempts to delete protected categories of information [15].</li> </ul>
<b>Priority:</b> Not described

<b>Rationale:</b> Logging deletion attempts ensures the integrity and availability of critical health data, supports regulatory compliance, and facilitates investigations into unauthorized actions.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Successful Deletion Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Delete a protected category of information with valid credentials and sufficient permissions.</li> <li>✓ Expected result: An audit log entry is generated, recording the user ID, category deleted, timestamp, and the successful deletion action.</li> </ul> </li> <li><b>2. Unsuccessful Deletion Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Attempt to delete a protected category of information without proper permissions or using invalid credentials.</li> <li>✓ Expected result: An audit log entry is generated, capturing the user ID, category targeted, timestamp, and reason for the failed deletion attempt.</li> </ul> </li> <li><b>3. Detailed Log Verification Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Perform multiple deletion attempts (both successful and unsuccessful) on various categories of protected information.</li> <li>✓ Expected result: Audit logs include comprehensive details for each attempt, including user ID, timestamp, result of the attempt, and category involved.</li> </ul> </li> <li><b>4. Alert for Unauthorized Deletion Attempts:</b> <ul style="list-style-type: none"> <li>✓ Test: Simulate multiple unauthorized deletion attempts on high-security categories of information.</li> <li>✓ Expected result: The system generates detailed logs for each attempt and triggers alerts to notify administrators of potential security threats.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IDS-023]
<b>Requirement description:</b> The mobile health application must generate audit records for all successful and unsuccessful attempts to delete security levels.
<b>Source:</b>

<ul style="list-style-type: none"> <li>✓ V-222459: The application must generate audit records when successful/unsuccessful attempts to delete security levels occur. Configure the application to create an audit record for both successful and unsuccessful attempts to delete security levels [15].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Maintaining detailed logs of deletion activities ensures accountability, aids in forensic investigations, and upholds the integrity of security configurations within the mobile health application.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Successful Deletion Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Perform a valid deletion of a security level with appropriate permissions.</li> <li>✓ Expected result: An audit log entry is generated, capturing the user ID, security level deleted, timestamp, and confirmation of the successful action.</li> </ul> </li> <li><b>2. Unsuccessful Deletion Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Attempt to delete a security level without the required permissions or with invalid credentials.</li> <li>✓ Expected result: An audit log entry is generated, recording the user ID, security level targeted, timestamp, and details of the failed attempt.</li> </ul> </li> <li><b>3. Comprehensive Log Verification Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Perform multiple deletion attempts (successful and unsuccessful) for various security levels.</li> <li>✓ Expected result: Audit logs comprehensively record each action, including user ID, timestamp, outcome of the attempt, and the affected security level.</li> </ul> </li> <li><b>4. Security Alert Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Simulate repeated unauthorized deletion attempts on high-priority security levels.</li> <li>✓ Expected result: The system generates audit log entries for each attempt and triggers alerts to notify administrators.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IDS-024]
---------------------------------

<b>Requirement description:</b> The mobile health application must generate audit records for both successful and unsuccessful attempts to modify privileges.
<b>Source:</b>
✓ V-222454: The application must generate audit records when successful/unsuccessful attempts to modify privileges occur. Configure the application to audit successful and unsuccessful attempts to modify privileges [15].
<b>Priority:</b> Not described
<b>Rationale:</b> Auditing modifications to user privileges is essential for maintaining security and ensuring that unauthorized privilege escalation or changes are detected and appropriately addressed.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li>1. <b>Successful Privilege Modification Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Perform a valid modification of user privileges (e.g., increasing user access level).</li> <li>✓ Expected result: The audit log records the event with details such as user ID, privilege modification details, timestamp, and confirmation of the successful change.</li> </ul> </li> <li>2. <b>Unsuccessful Privilege Modification Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Attempt to modify user privileges without the required permissions.</li> <li>✓ Expected result: The audit log captures the event, noting the user ID, the privilege change attempt, timestamp, and the reason for failure (e.g., insufficient privileges).</li> </ul> </li> <li>3. <b>Comprehensive Log Verification Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Perform multiple privilege modification attempts (both successful and unsuccessful) for different users.</li> <li>✓ Expected result: The audit logs accurately record each modification attempt, including user ID, timestamp, action taken, and outcome.</li> </ul> </li> <li>4. <b>Alert Trigger Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Simulate unauthorized privilege modification attempts for a high-level user or admin.</li> <li>✓ Expected result: The audit system generates logs for each attempt and sends alerts to security administrators indicating a possible privilege escalation attempt.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda
José L Fernández-Alemán
Juan Manuel Carrillo-de-Gea
Joaquín Nicolás
08/01/2026

<b>PUID:</b> [SECM-CAT-IDS-025]
<b>Requirement description:</b> The mobile health application must log the destination IP addresses for all connections made by the application.
<b>Source:</b>
✓ V-222470: The application must log destination IP addresses. Configure the application to record the destination IP address of the remote system [15].
<b>Priority:</b> Not described
<b>Rationale:</b> Logging destination IP addresses is crucial for monitoring network activity, identifying malicious behavior, and supporting after-the-fact forensic investigations in compliance with healthcare security requirements.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Connection Log Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Establish a connection from the application to a remote server.</li> <li>✓ Expected result: The audit log records the destination IP address, timestamp, and the purpose of the connection.</li> </ul> </li> <li><b>2. Multiple Connections Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Initiate connections to various remote systems during typical application operations.</li> <li>✓ Expected result: The audit logs contain entries for all destination IP addresses, along with relevant metadata such as connection type and timestamp.</li> </ul> </li> <li><b>3. Unauthorized Connection Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Attempt to initiate an unauthorized or unexpected connection.</li> <li>✓ Expected result: The audit log captures the destination IP address, flags the connection as unauthorized, and generates an alert for security personnel.</li> </ul> </li> <li><b>4. Retention and Accessibility Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Review logs after a defined retention period.</li> <li>✓ Expected result: Logs containing destination IP addresses remain intact, accessible, and comply with the organization's log retention policies.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda
José L Fernández-Alemán
Juan Manuel Carrillo-de-Gea
Joaquín Nicolás
08/01/2026

<b>PUID:</b> [SECM-CAT-IDS-026]
<b>Requirement description:</b> The mobile health application must log user actions involving data access and changes, specifying the data elements accessed or modified and the corresponding user actions.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ V-222471: The application must log user actions involving access to data. Identify the specific data elements requiring protection and audit access to the data [15].</li> <li>✓ V-222472: The application must log user actions involving changes to data. Configure the application to log all changes to application data [15].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Logging user access to and changes in sensitive healthcare data ensures compliance with regulatory standards, supports forensic investigations, and helps detect unauthorized activities.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Data Access Logging Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Access sensitive data within the application as an authorized user.</li> <li>✓ Expected result: The audit log records the user ID, timestamp, type of data accessed, and the purpose of access.</li> </ul> </li> <li><b>2. Data Modification Logging Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Modify a data entry (e.g., updating patient information) in the application.</li> <li>✓ Expected result: The audit log captures the user ID, timestamp, original data value, modified data value, and action performed.</li> </ul> </li> <li><b>3. Unauthorized Access Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Attempt to access sensitive data without proper permissions.</li> <li>✓ Expected result: The application denies access, logs the user ID, attempted action, timestamp, and generates a security alert.</li> </ul> </li> <li><b>4. Data Deletion Logging Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Delete a data entry as an authorized user.</li> <li>✓ Expected result: The log records the user ID, timestamp, and details of the data removed, including metadata where applicable.</li> </ul> </li> <li><b>5. Retention and Review Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Access audit logs after a defined retention period.</li> <li>✓ Expected result: Logs remain intact, accessible, and provide detailed records of data access and modification actions for review.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda

José L Fernández-Alemán  
 Juan Manuel Carrillo-de-Gea  
 Joaquín Nicolás  
 08/01/2026

<b>PUID:</b> [SECM-CAT-IDS-027]
<b>Requirement description:</b> The mobile health application must log all successful and unsuccessful attempts to delete user privileges.
<b>Source:</b>
✓ V-222458: The application must generate audit records when successful/unsuccessful attempts to delete privileges occur. Configure the application to audit successful and unsuccessful attempts to delete privileges [15].
<b>Priority:</b> Not described
<b>Rationale:</b> Logging attempts to delete user privileges ensures transparency, accountability, and detection of unauthorized or malicious actions, protecting sensitive health data and maintaining user role integrity.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Successful Privilege Deletion Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Delete a user's privilege as an administrator.</li> <li>✓ Expected result: The audit log records the administrator's ID, timestamp, user ID whose privilege was deleted, and the privilege removed.</li> </ul> </li> <li><b>2. Unsuccessful Privilege Deletion Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Attempt to delete a privilege without proper authorization.</li> <li>✓ Expected result: The application denies the action, logs the unauthorized user ID, timestamp, attempted privilege deletion, and generates a security alert.</li> </ul> </li> <li><b>3. Privilege Change Review Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Retrieve logs of privilege deletion actions over a specified period.</li> <li>✓ Expected result: Logs provide a complete and accurate record of all privilege deletion attempts, including details of success or failure.</li> </ul> </li> <li><b>4. Concurrent Session Audit Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Attempt to delete privileges during an active user session.</li> <li>✓ Expected result: The application logs the attempt, specifies the session state, and includes any associated warnings or system responses.</li> </ul> </li> <li><b>5. Retention Policy Verification Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Verify that privilege deletion logs are retained for the defined audit retention period.</li> <li>✓ Expected result: Logs remain accessible and intact for review throughout the retention period.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization

<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IDS-028]
<b>Requirement description:</b> The mobile health application must generate detailed and time-synchronized audit records containing event type, date and time, user identity, and associated outcomes for security and accountability.
<b>Source:</b>
<ul style="list-style-type: none"><li>✓ V-222446: The application must record a time stamp indicating when the event occurred. Configure the application to record the time the event occurred when recording the event [15].</li><li>✓ 4.7 Communications and Operations Management: Consideration ID-15 – Consider the Definition of Minimum Requirements and Specifications for Audit and Logging Capabilities to be Used by Member Organizations: Member organizations should consider use of common logging and auditing functions to facilitate trust relationships and to meet accountability obligations [10].</li><li>✓ V-222476: The application must produce audit records that contain information to establish the outcome of the events. Configure the application to include the outcome of application functions or events [15].</li><li>✓ V-222477: The application must generate audit records containing information that establishes the identity of any individual or process associated with the event. Configure the application to log the identity of the user and/or the process associated with the event [15].</li><li>✓ V-222478: The application must generate audit records containing the full-text recording of privileged commands or the individual identities of group account users. Configure the application to log the full text recording of privileged commands or the individual identities of group users [15].</li><li>✓ 4.7 Communications and Operations Management: Security Requirement 42 – Minimum Content of Audit Logs: The EHRI audit log and the audit logs of PoS systems connecting to the EHRI must contain:<ol style="list-style-type: none"><li>a) The user ID of the accessing user;</li><li>b) The role the user is exercising;</li><li>c) The organization of the accessing user (at least in those cases where an individual accesses information on behalf of more than one organization);</li><li>d) The patient ID of the data subject (patient/person);</li></ol></li></ul>

- e) The function performed by the accessing user;
  - f) A timestamp;
  - g) In the case of access override to blocked or masked records or portions of records, a reason for the override, as chosen by the user making the access; and
  - h) In the case of changes to consent directives made by a substitute decision-maker, the identity of the decision-maker.
- ✓ V-222448: The application must provide audit record generation capability for connecting system IP addresses. Configure the application or application server to log all connecting IP address information [15].
  - ✓ V-222497: The applications must use internal system clocks to generate time stamps for audit records. Configure the application to use the hosting systems internal clock for audit record generation [15].
  - ✓ V-222498: The application must record time stamps for audit records that can be mapped to Coordinated Universal Time (UTC) or Greenwich Mean Time (GMT). Configure the application to use the underlying system clock that maps to relevant UTC or GMT time zone [15].
  - ✓ V-222499: The application must record time stamps for audit records that meet a granularity of one second for a minimum degree of precision. Configure the application to leverage the underlying operating system as the time source when recording time stamps or design the application to ensure granularity of 1 second as the minimum degree of precision [15].
  - ✓ V-222473: The application must produce audit records containing information to establish when (date and time) the events occurred. Configure the application or application server to include the date and the time of the event in the audit logs [15].
  - ✓ V-222449: The application must record the username or user ID of the user associated with the event. Configure the application to record the user ID of the user responsible for the log event entry [15].
  - ✓ AU-8 TIME STAMPS

Control:

- a. Use internal system clocks to generate time stamps for audit records; and
- b. Record time stamps for audit records that meet [Assignment: organization-defined granularity of time measurement] and that use Coordinated Universal Time, have a fixed local time offset from Coordinated Universal Time, or that include the local time offset as part of the time stamp [11].

- ✓ AU-3 CONTENT OF AUDIT RECORDS

Control: Ensure that audit records contain information that establishes the following:

- a. What type of event occurred;
- b. When the event occurred;
- c. Where the event occurred;
- d. Source of the event;
- e. Outcome of the event; and
- f. Identity of any individuals, subjects, or objects/entities associated with the event [11].

- ✓ V-222474: The application must produce audit records containing enough information to establish which component, feature or function of the application triggered the audit event. Configure the application to log which component, feature or functionality of the application triggered the event [15].

- ✓ 12.4.1 Event logging

Control: Event logs recording user activities, exceptions, faults and information security events should be produced, kept and regularly reviewed.

Event logs should include, when relevant:

- user IDs;
- system activities;
- dates, times and details of key events, e.g. log-on and log-off;
- device identity or location if possible and system identifier;
- records of successful and rejected system access attempts;
- records of successful and rejected data and other resource access attempts;
- changes to system configuration;
- use of privileges;
- use of system utilities and applications;
- files accessed and the kind of access;
- network addresses and protocols;
- alarms raised by the access control system;
- activation and de-activation of protection systems, such as anti-virus systems and intrusion detection systems;
- records of transactions executed by users in applications [7].

**Priority:** Not described

**Rationale:** Comprehensive and precise audit logging supports forensic analysis, accountability, compliance, and incident response in mobile health applications, ensuring integrity and security of operations.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

1. **Timestamp Verification Test:**

- ✓ Test: Log an event and check the timestamp recorded in the log.
- ✓ Expected result: The timestamp is accurate to the second and aligns with the system's UTC or GMT configuration.

2. **User ID Logging Test:**

- ✓ Test: Perform an action requiring authentication and check the audit log.

<ul style="list-style-type: none"> <li>✓ Expected result: The log entry contains the correct user ID or username associated with the action.</li> </ul> <p><b>3. Event Outcome Logging Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Execute both successful and failed operations (e.g., data access or privilege modification).</li> <li>✓ Expected result: Logs clearly indicate the outcome (success or failure) and details of the event.</li> </ul> <p><b>4. IP Address Logging Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Access the application from different IP addresses.</li> <li>✓ Expected result: Logs include the originating IP address for each session or action.</li> </ul> <p><b>5. Granularity and Content Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Generate audit records for various operations and review the log details.</li> <li>✓ Expected result: Logs contain event type, date, time, user ID, system component triggering the event, and event outcomes with one-second precision.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IDS-029]
<b>Requirement description:</b> The mobile health application must automatically log all account lifecycle actions, including creation, modification, enabling, disabling, and removal, capturing account name, event type, and timestamp.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ Control Enhancements:           <ul style="list-style-type: none"> <li>(4) ACCOUNT MANAGEMENT   AUTOMATED AUDIT ACTIONS</li> </ul> <p>Automatically audit account creation, modification, enabling, disabling, and removal actions [11].</p> </li> </ul> <ul style="list-style-type: none"> <li>✓ V-222467: The application must generate audit records for all account creations, modifications, disabling, and termination events. Configure the application to log user account creation, modification, disabling, and termination events [15].</li> <li>✓ V-222413: The application must automatically audit account creation. Configure the application to write a log entry when a new user account is created. At a minimum, ensure account name, date and time of the event are recorded [15].</li> <li>✓ V-222414: The application must automatically audit account modification. Configure the application to write a log entry when a user account is modified. At a minimum, ensure account name, date and time of the event are recorded [15].</li> </ul>

<ul style="list-style-type: none"> <li>✓ V-222415: The application must automatically audit account disabling actions. Configure the application to write a log entry when a user account is disabled. At a minimum, ensure account name, date and time of the event are recorded [15].</li> <li>✓ V-222416: The application must automatically audit account removal actions. Configure the application to write a log entry when a user account is removed. At a minimum, ensure account name, date and time of the event are recorded [15].</li> <li>✓ V-222421: The application must automatically audit account enabling actions. Configure the application to write a log entry when a user account is enabled. At a minimum, ensure account name, date and time of the event are recorded [15].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Automated logging of account lifecycle events ensures traceability, supports forensic investigations, and meets regulatory compliance requirements for securing sensitive healthcare data.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Account Creation Audit Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Create a new user account and verify the corresponding log entry.</li> <li>✓ Expected result: The log records the account name, event type, and exact timestamp of the creation action.</li> </ul> </li> <li><b>2. Account Modification Audit Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Modify an existing user account (e.g., change roles or attributes) and check the logs.</li> <li>✓ Expected result: The log contains details of the modification, including the account name, event type, and timestamp.</li> </ul> </li> <li><b>3. Account Disabling Audit Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Disable a user account and review the log entry.</li> <li>✓ Expected result: The log records the disabling event with the account name, event type, and timestamp.</li> </ul> </li> <li><b>4. Account Enabling Audit Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Re-enable a previously disabled account and inspect the log.</li> <li>✓ Expected result: The log captures the enabling action, account name, event type, and timestamp.</li> </ul> </li> <li><b>5. Account Removal Audit Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Delete a user account and verify the corresponding log entry.</li> <li>✓ Expected result: The log includes the removal event with the account name, event type, and timestamp.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0

**Author and date:**

Carlos M. Mejía-Granda  
 José L Fernández-Alemán  
 Juan Manuel Carrillo-de-Gea  
 Joaquín Nicolás  
 08/01/2026

**PUID: [SECM-CAT-IDS-030]**

**Requirement description:** The mobile health application must log all configuration changes, capturing details of the user account making the change, event type, and timestamp.

**Source:**

- ✓ V-222512: The application must audit who makes configuration changes to the application. Configure the application to create log entries that can be used to identify the user accounts that make application configuration changes [15].

**Priority:** Not described

**Rationale:** Logging configuration changes enhances traceability, supports accountability, and helps in investigating unauthorized or erroneous modifications critical to healthcare applications.

**Number of Children:** 0**Number of parents:** 0**Cycles:** 0**Number Audit:** 0**Child PUIDs:** Not described**Parent PUIDs:** Not described**Exclusion PUIDs:** Not described**Importance:** Not described**Current state:** To be determined**Verification method:** Demonstration/Analysis**Validation criteria:****1. Configuration Change Audit Test:**

- ✓ Test: Perform a configuration change in the application (e.g., modify system settings) and check the logs.
- ✓ Expected result: The log entry includes the user account responsible, the type of configuration change, and the exact timestamp.

**2. Unauthorized Change Detection Test:**

- ✓ Test: Attempt an unauthorized configuration change with a user account lacking required permissions.
- ✓ Expected result: The log captures the denied attempt, including the user account, event type, and timestamp.

**3. Multiple User Audit Test:**

- ✓ Test: Have different users make configuration changes and verify log entries for each.
- ✓ Expected result: The log contains distinct entries for each user, accurately reflecting their actions and timestamps.

**4. Configuration Reversion Test:**

- ✓ Test: Revert a configuration change and inspect the logs for both the original and reversion events.

<ul style="list-style-type: none"> <li>✓ Expected result: The log includes entries for both the initial change and the reversion, with user account details and timestamps.</li> </ul> <p><b>5. Audit Integrity Verification Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Attempt to tamper with the configuration audit logs.</li> <li>✓ Expected result: Logs are protected against tampering, ensuring their integrity and evidentiary value.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IDS-031]
<b>Requirement description:</b> The mobile health application must securely log all accesses, modifications, or archiving of personal health information (PHI), ensuring accountability and non-repudiation for all user interactions.
<b>Source:</b> <ul style="list-style-type: none"> <li>✓ 12.4.1 Event logging: health information systems processing personal health information should create a secure audit record each time a user accesses, creates, updates or archives personal health information via the system [7].</li> <li>✓ 4.7. Communications and Operations Management: Security Requirement 46 – Reporting Every Access to a Patient/Person's HER: The EHRi must be capable of identifying all users who have accessed or modified a given patient/person's record(s) over a given period of time [10].</li> <li>✓ AU-10 NON-REPUDIATION Control: Provide irrefutable evidence that an individual (or process acting on behalf of an individual) has performed [Assignment: organization-defined actions to be covered by non-repudiation] [11].</li> <li>✓ 7) Non-Repudiation: It guarantees the reliability of the data communicated in epidemic situation applications, where the originator of the data is not able to deny the validity of the signature transmitted [31].</li> <li>✓ V-222438: The application must protect against an individual (or process acting on behalf of an individual) falsely denying having performed organization-defined actions to be covered by non-repudiation. Configure the application to provide users with a non-repudiation function in the form of digital signatures when it is required by the organization or by the application design and architecture [15].</li> </ul>

<ul style="list-style-type: none"> <li>✓ 4.7. Communications and Operations Management: Security Requirement 47 – Reporting Every Access by a User: The EHR must be capable of identifying all patients/persons whose records have been accessed or modified by a given user over a given period of time [10].</li> <li>✓ V-222512: The application must audit who makes configuration changes to the application. Configure the application to create log entries that can be used to identify the user accounts that make application configuration changes [15].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Secure logging of PHI interactions strengthens accountability, enables auditability, and supports compliance with healthcare regulations by identifying users involved in PHI actions.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<p><b>1. Audit Log Creation Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Access, modify, or archive PHI through the application.</li> <li>✓ Expected result: The log records the user ID, action performed (e.g., access, modify, archive), and timestamp.</li> </ul>
<p><b>2. Non-Repudiation Mechanism Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Verify that all user actions involving PHI are digitally signed or secured to prevent denial of performed actions.</li> <li>✓ Expected result: Digital signatures or equivalent mechanisms ensure non-repudiation for recorded actions.</li> </ul>
<p><b>3. Access Reporting Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Query for all users who accessed a specific patient's records within a defined period.</li> <li>✓ Expected result: The report lists all user IDs and timestamps accurately, matching log data.</li> </ul>
<p><b>4. Modification Reporting Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Query for all patients whose records were modified by a specific user within a defined period.</li> <li>✓ Expected result: The report identifies all patient records and corresponding modification timestamps.</li> </ul>
<p><b>5. Configuration Change Logging Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Make configuration changes to the application and inspect the logs.</li> <li>✓ Expected result: Logs contain entries with the user ID, configuration change details, and timestamps for each action.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda
José L Fernández-Alemán
Juan Manuel Carrillo-de-Gea

Joaquín Nicolás 08/01/2026
-------------------------------

**PUID:** [SECM-CAT-IDS-032]

**Requirement description:** The mobile health application must enforce strict access controls for configuration settings, ensuring only authorized users can modify or view application configurations.

**Source:**

- ✓ V-222511: The application must enforce access restrictions associated with changes to application configuration. Configure the application to limit access to configuration settings to only authorized users [15].

**Priority:** Not described

**Rationale:** Restricting access to application configurations reduces the risk of unauthorized changes that may compromise system security or functionality, safeguarding sensitive healthcare operations.

**Number of Children:** 0**Number of parents:** 0**Cycles:** 0**Number Audit:** 0**Child PUIDs:** Not described**Parent PUIDs:** Not described**Exclusion PUIDs:** Not described**Importance:** Not described**Current state:** To be determined**Verification method:** Demonstration/Analysis**Validation criteria:****1. Access Control Test:**

- ✓ Test: Attempt to modify application configurations as an unauthorized user.
- ✓ Expected result: Unauthorized access is denied, and an alert is logged.

**2. Authorized User Test:**

- ✓ Test: Modify application configurations as an authorized user.
- ✓ Expected result: The changes are applied successfully, and the activity is logged with the user ID and timestamp.

**3. Audit Log Test:**

- ✓ Test: Review the logs for configuration access and modifications.
- ✓ Expected result: Logs contain entries for all configuration changes, including the user ID, action performed, and timestamp.

**4. Configuration Access Restriction Test:**

- ✓ Test: Verify that configuration settings are not accessible through unauthorized APIs or interfaces.
- ✓ Expected result: Configuration settings are accessible only through secure, authenticated channels.

**5. Role-Based Access Test:**

- ✓ Test: Assign roles to users and verify that only users with the appropriate roles can access configuration settings.
- ✓ Expected result: Access permissions align with role definitions, and unauthorized roles are denied access.

**Requested by:** The organization**Responsible:** Developer

<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda
José L Fernández-Alemán
Juan Manuel Carrillo-de-Gea
Joaquín Nicolás
08/01/2026

<b>PUID:</b> [SECM-CAT-IDS-033]
<b>Requirement description:</b> The mobile health application must utilize transaction recovery logs for all database operations, ensuring the ability to restore data integrity in the event of failure or rollback.
<b>Source:</b>
✓ V-222479: The application must implement transaction recovery logs when transaction based. Configure the application database to utilize transactional logging
Transaction recovery through logs allows you to record transactional operations performed in the database, including both modifications and insertions and deletions of data [15].
<b>Priority:</b> Not described
<b>Rationale:</b> Transaction recovery logging ensures that all changes to the database are tracked, enabling the restoration of the system to a consistent state after errors, interruptions, or failures, which is critical for maintaining data integrity in health-related applications.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
1. <b>Transaction Logging Test:</b>
✓ Test: Perform a series of database operations, including data insertions, deletions, and modifications.
✓ Expected result: All operations are logged in the transaction recovery logs with sufficient detail to enable restoration.
2. <b>Rollback Functionality Test:</b>
✓ Test: Simulate a transaction failure and attempt a rollback.
✓ Expected result: The database reverts to its previous consistent state as recorded in the logs.
3. <b>Log Inspection Test:</b>
✓ Test: Review the transaction recovery logs after performing database operations.
✓ Expected result: Logs include detailed entries for each operation, including timestamps, affected data, and user IDs.
4. <b>Data Recovery Test:</b>

- ✓ Test: Simulate a system crash during a transaction and attempt recovery using transaction logs.
- ✓ Expected result: Data integrity is restored, and the database reflects the last committed state.

### 5. Log Protection Test:

- ✓ Test: Verify that the transaction recovery logs are secured against unauthorized access and modification.
- ✓ Expected result: Only authorized users can access or modify the logs, and all access attempts are logged.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda  
José L Fernández-Alemán  
Juan Manuel Carrillo-de-Gea  
Joaquín Nicolás  
08/01/2026

**PUID:** [SECM-CAT-IDS-034]

**Requirement description:** The mobile health application must retain historical records of personal health information updates, including the original data, changes made, and associated audit details such as user identity and timestamps.

**Source:**

- ✓ 12.4.1 Event logging: When personal health information is updated, a record of the former content of the data and the associated audit record (i.e. who entered the data on what date) should be retained [7].
- ✓ 8.15 Logging: Logs that record activities, exceptions, faults and other relevant events should be produced, stored, protected and analyzed. (It could be used for logging and auditing) [6].

**Priority:** Not described

**Rationale:** Retaining comprehensive logs of data updates is essential for accountability, non-repudiation, and compliance with health information privacy and security regulations. These logs support auditability and forensic investigations while maintaining data integrity.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

#### 1. Historical Data Logging Test:

- ✓ Test: Update a patient's personal health information and check the system logs.

<ul style="list-style-type: none"> <li>✓ Expected result: Logs capture the original data, updated content, user identity, and timestamp of the change.</li> </ul>
<b>2. Audit Record Review Test:</b>
<ul style="list-style-type: none"> <li>✓ Test: Access audit records related to a specific update.</li> <li>✓ Expected result: Logs include complete details of who made the update, what changes were made, and when they occurred.</li> </ul>
<b>3. Data Integrity Test:</b>
<ul style="list-style-type: none"> <li>✓ Test: Verify that historical data and associated logs are stored securely without modification.</li> <li>✓ Expected result: Data remains immutable, and access is limited to authorized users only.</li> </ul>
<b>4. Retention Policy Test:</b>
<ul style="list-style-type: none"> <li>✓ Test: Validate compliance with data retention policies by reviewing logs stored for the required duration.</li> <li>✓ Expected result: Logs are available for the defined retention period and properly archived after expiration.</li> </ul>
<b>5. Access Control Test:</b>
<ul style="list-style-type: none"> <li>✓ Test: Attempt to view or modify logs with different user roles.</li> <li>✓ Expected result: Only authorized roles can access or modify audit logs, and all access attempts are recorded</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IDS-035]
<b>Requirement description:</b> The mobile health application must implement an emergency access procedure to allow authorized access to electronic protected health information (ePHI) during emergencies.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ § 164.312 Technical safeguards. <ul style="list-style-type: none"> <li>(a) <ul style="list-style-type: none"> <li>(2) Implementation specifications: <ul style="list-style-type: none"> <li>(i) 1. Emergency access procedure (Required): Establish (and implement as needed) procedures for obtaining necessary electronic protected health information during an emergency [9].</li> </ul> </li> </ul> </li> </ul> </li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Emergency access ensures the availability of critical health information to authorized personnel in life-threatening or urgent scenarios, complying with § 164.312 technical safeguards for ePHI access.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0

<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Emergency Access Simulation Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Simulate an emergency scenario requiring access to ePHI.</li> <li>✓ Expected result: Authorized personnel can access the necessary ePHI securely and promptly.</li> </ul> </li> <li><b>2. Access Restriction Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Attempt unauthorized access to ePHI using the emergency access feature.</li> <li>✓ Expected result: Access is denied, and the attempt is logged with details of the user and action.</li> </ul> </li> <li><b>3. Audit Logging Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Review audit logs after an emergency access event.</li> <li>✓ Expected result: Logs capture the identity of the user, the data accessed, the time of access, and the justification provided.</li> </ul> </li> <li><b>4. Procedure Verification Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Review the documented emergency access procedure.</li> <li>✓ Expected result: The procedure is well-documented, aligns with regulatory requirements, and is accessible to authorized personnel.</li> </ul> </li> <li><b>5. Periodic Procedure Review Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Verify the periodic review and testing of the emergency access procedure.</li> <li>✓ Expected result: The procedure is reviewed and updated as needed to remain effective and compliant.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IDS-036]
<b>Requirement description:</b> The mobile health application must maintain a transmission log for messages containing personal health information (PHI), recording metadata such as time, date, origin, and destination, while excluding message content.
<b>Source:</b>

<ul style="list-style-type: none"> <li>✓ 12.4.1 Event logging: Messaging systems used to transmit messages containing personal health information should keep a log of message transmissions (such a log should contain the time, date, origin and destination of the message, but not its content) [7].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Logging metadata for message transmissions ensures accountability and traceability without compromising the confidentiality of sensitive health information, aligning with health data logging standards.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b> <ol style="list-style-type: none"> <li><b>1. Message Transmission Logging Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Send a message containing PHI through the application.</li> <li>✓ Expected result: The transmission log records the time, date, origin, and destination of the message without logging its content.</li> </ul> </li> <li><b>2. Content Exclusion Validation Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Review the transmission log.</li> <li>✓ Expected result: Only metadata (time, date, origin, and destination) is logged; no message content is present.</li> </ul> </li> <li><b>3. Log Integrity Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Attempt unauthorized modification of transmission logs.</li> <li>✓ Expected result: Modifications are prevented, or any changes are logged and flagged.</li> </ul> </li> <li><b>4. Log Review Access Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Attempt to access the transmission log using an unauthorized account.</li> <li>✓ Expected result: Access is denied, and the attempt is recorded in an audit log.</li> </ul> </li> <li><b>5. Compliance Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Verify that the logged data adheres to relevant privacy and security regulations.</li> <li>✓ Expected result: The transmission log format complies with legal and regulatory standards for handling PHI.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IDS-037]
---------------------------------

<b>Requirement description:</b> The mobile health application must implement organization-defined data mining detection techniques to monitor and safeguard data storage objects from unauthorized data mining attempts.
<b>Source:</b>
✓ V-222424: The application must utilize organization-defined data mining detection techniques for organization-defined data storage objects to adequately detect data mining attempts. Utilize and implement data mining protections when requirements specify it [15].
<b>Priority:</b> Not described
<b>Rationale:</b> Detecting and preventing data mining ensures that sensitive patient information stored within the application remains secure and is not misused or accessed inappropriately.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Data Mining Detection Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Simulate a data mining attempt by generating abnormal access patterns to data storage objects.</li> <li>✓ Expected result: The application detects and logs the attempt, triggering an appropriate alert.</li> </ul> </li> <li><b>2. Behavior Analysis Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Perform regular access operations and compare against data mining detection thresholds.</li> <li>✓ Expected result: Normal operations are not flagged as data mining attempts.</li> </ul> </li> <li><b>3. Alert Notification Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Trigger a simulated data mining attempt.</li> <li>✓ Expected result: An alert is sent to the system administrator or security team.</li> </ul> </li> <li><b>4. Logging Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Review logs for a flagged data mining attempt.</li> <li>✓ Expected result: Detailed records of the suspicious activity, including user ID, timestamps, and accessed data objects, are present.</li> </ul> </li> <li><b>5. Compliance Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Validate detection mechanisms against organization-defined policies and industry standards for data mining protections.</li> <li>✓ Expected result: The implemented techniques align with organizational and regulatory requirements.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda
José L Fernández-Alemán
Juan Manuel Carrillo-de-Gea
Joaquín Nicolás

08/01/2026

<b>PUID:</b> [SECM-CAT-IDS-038]
<b>Requirement description:</b> The mobile health application must include facilities for analyzing logs and audit trails to identify users who have accessed or modified patient records, as well as patients whose records were accessed by specific users, within a defined time frame.
<b>Source:</b>
<ul style="list-style-type: none"><li>✓ 4.7. Communications and Operations Management: Security Requirement 48 – Analyzing EHRI Audit Logs for Patients/Persons at Elevated Risk: The EHRI must provide functions for analyzing logs and audit trails to allow the identification of all users who have accessed or modified such record(s) over a given period of time [10].</li><li>✓ 12.4.1 Event logging: 7. Health information systems containing personal health information should be provided with facilities for analyzing logs and audit trails that:<ul style="list-style-type: none"><li>a) allow the identification of all system users who have accessed or modified a given subject of care's record(s) over a given period of time;</li><li>b) allow the identification of all subjects of care whose records have been accessed or modified by a given system user over a given period of time. [7].</li></ul></li></ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Enabling comprehensive log analysis ensures accountability and enhances security by identifying potential misuse or unauthorized access to sensitive health records.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"><li><b>1. Access Analysis Test:</b><ul style="list-style-type: none"><li>✓ Test: Query the log analysis function for all users who accessed or modified a specific patient's record within a specified timeframe.</li><li>✓ Expected result: A detailed report lists all relevant users with timestamps and actions performed.</li></ul></li><li><b>2. Reverse Lookup Test:</b><ul style="list-style-type: none"><li>✓ Test: Query the log analysis function for all patients whose records were accessed by a specific user during a specified period.</li><li>✓ Expected result: A report lists all patients accessed by the user, along with timestamps and details of access or modifications.</li></ul></li><li><b>3. Audit Trail Integrity Test:</b><ul style="list-style-type: none"><li>✓ Test: Verify that log entries remain unaltered and accurately represent system activities.</li><li>✓ Expected result: All log entries are intact, tamper-proof, and provide a reliable history.</li></ul></li><li><b>4. Log Search Performance Test:</b><ul style="list-style-type: none"><li>✓ Test: Perform queries on logs with large datasets to retrieve specific user or patient activities.</li></ul></li></ol>

✓ Expected result: Results are returned within an acceptable timeframe as defined by performance requirements.
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IDS-039]
<b>Requirement description:</b> The mobile health application must include an audit reduction capability for on-demand reporting, real-time review, and forensic investigations, supporting both centralized and local audit log analysis.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ V-222489: The application must provide an audit reduction capability that supports on-demand reporting requirements. Configure the application to generate soft copy, hard copy and/or screen-based reports based on the selected filtered event data [15].</li> <li>✓ V-222490: The application must provide an audit reduction capability that supports on-demand audit review and analysis. Configure the application to log to a centralized auditing capability that provides on-demand reports based on the filtered audit event data or design or configure the application to meet the requirement [15].</li> <li>✓ V-222491: The application must provide an audit reduction capability that supports after-the-fact investigations of security incidents. Configure the application to provide an audit reduction capability that supports forensic investigations [15].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Audit reduction capabilities streamline security incident investigations, enabling efficient reporting, analysis, and compliance with regulatory requirements.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li>1. <b>On-Demand Reporting Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Generate a report based on filtered audit event data (e.g., by date, user, or action).</li> <li>✓ Expected result: A soft copy or screen-based report is produced accurately reflecting the filtered data.</li> </ul> </li> <li>2. <b>Centralized Logging Test:</b></li> </ol>

<ul style="list-style-type: none"> <li>✓ Test: Verify integration with a centralized auditing system to enable on-demand audit reviews.</li> <li>✓ Expected result: The system successfully logs to a centralized system and generates filtered event reports as requested.</li> </ul>
<b>3. Forensic Investigation Capability Test:</b>
<ul style="list-style-type: none"> <li>✓ Test: Perform an investigation simulation using the audit logs to trace a simulated security incident.</li> <li>✓ Expected result: Logs provide all necessary details for an accurate reconstruction of the incident.</li> </ul>
<b>4. Report Format Test:</b>
<ul style="list-style-type: none"> <li>✓ Test: Validate that reports can be generated in different formats, such as soft copy, hard copy, or displayed on the user interface.</li> <li>✓ Expected result: Reports are generated in the required formats without errors.</li> </ul>
<b>5. Performance Test:</b>
<ul style="list-style-type: none"> <li>✓ Test: Measure the time taken to generate audit reduction reports under heavy log data loads.</li> <li>✓ Expected result: Reports are generated within acceptable performance thresholds.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IDS-040]
<b>Requirement description:</b> Ensure the mobile health application provides an audit reduction and reporting capability that preserves the original content and time ordering of audit records.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ V-222495: The application must provide an audit reduction capability that does not alter original content or time ordering of audit records. Configure the application to not alter original log content or time ordering of audit records [15].</li> <li>✓ V-222496: The application must provide a report generation capability that does not alter original content or time ordering of audit records. Configure and design the application to not modify source logs when filtering events [15].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Maintaining unaltered audit logs ensures data integrity, compliance with regulations, and reliability for forensic investigations.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described

<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Audit Integrity Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Filter audit logs to generate a report without modifying the original logs.</li> <li>✓ Expected result: Original logs remain intact with no changes to content or time ordering.</li> </ul> </li> <li><b>2. Timestamp Verification Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Verify timestamps in reports match those in the source logs.</li> <li>✓ Expected result: All timestamps in the generated report are consistent with the original logs.</li> </ul> </li> <li><b>3. Log Content Consistency Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Validate that the log content in the audit records remains unchanged after the audit reduction process.</li> <li>✓ Expected result: Filtered logs retain their original content without alterations.</li> </ul> </li> <li><b>4. Forensic Reliability Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Use filtered logs for a simulated forensic analysis.</li> <li>✓ Expected result: Logs provide consistent and reliable information for investigation without discrepancies.</li> </ul> </li> <li><b>5. Data Export Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Export filtered logs in a secure format without modifying the source logs.</li> <li>✓ Expected result: Exported data is consistent with the original logs, maintaining integrity.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IDS-041]
<b>Requirement description:</b> Ensure the mobile health application provides filtering capabilities for audit records, enabling event searches based on organization-defined criteria.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ V-222488: The application must provide the capability to filter audit records for events of interest based upon organization-defined criteria. Configure the application filters to search event logs based on defined criteria [15].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Filtering audit logs based on specific criteria supports efficient incident investigation, compliance, and operational monitoring.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described

<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<p><b>1. Filtering Capability Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Apply filters to the audit logs based on organization-defined criteria (e.g., user ID, event type, timestamp).</li> <li>✓ Expected result: Only records matching the defined criteria are displayed in the filtered results.</li> </ul> <p><b>2. Search Efficiency Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Perform multiple searches with varied criteria to assess performance and response time.</li> <li>✓ Expected result: The filtering process retrieves relevant records within an acceptable time frame.</li> </ul> <p><b>3. Filter Accuracy Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Validate that the filtering mechanism excludes irrelevant events and includes all relevant ones.</li> <li>✓ Expected result: Filtered results are accurate and complete for the defined criteria.</li> </ul> <p><b>4. Customization Capability Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Modify the organization-defined filtering criteria and reapply the filters.</li> <li>✓ Expected result: The system successfully adapts to the new criteria and displays appropriate results.</li> </ul> <p><b>5. Audit Retention Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Verify that applying filters does not alter or remove original audit records.</li> <li>✓ Expected result: Original logs remain intact and unaltered after filtering operations.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IDS-042]
<b>Requirement description:</b> Ensure the mobile health application employs cryptographic mechanisms to protect the integrity of audit logs, preventing tampering or unauthorized modifications.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ V-222507: The application must use cryptographic mechanisms to protect the integrity of audit information. Configure the application to create an integrity check consisting of a cryptographic hash or one-way digest that can be used to establish the integrity when storing log files [15].</li> </ul>

**Priority:** Not described

<b>Rationale:</b> Cryptographic protection ensures the integrity of audit information, safeguarding it against tampering and preserving its value for forensic analysis and compliance audits.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Integrity Check Implementation Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Generate audit logs and compute cryptographic hashes (e.g., SHA-256) for each entry.</li> <li>✓ Expected result: Each log entry is accompanied by a valid cryptographic hash or one-way digest.</li> </ul> </li> <li><b>2. Tampering Detection Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Modify an audit log entry and verify its hash against the stored digest.</li> <li>✓ Expected result: The system detects the modification and flags the entry as tampered.</li> </ul> </li> <li><b>3. Secure Storage Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Store the audit logs and their integrity hashes in a secure, non-rewriteable location.</li> <li>✓ Expected result: Logs and hashes are securely stored and accessible only to authorized personnel or processes.</li> </ul> </li> <li><b>4. Audit Verification Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Run periodic integrity checks on stored logs using their cryptographic hashes.</li> <li>✓ Expected result: All logs pass the integrity check unless tampered.</li> </ul> </li> <li><b>5. Compliance Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Verify the cryptographic mechanisms meet industry standards (e.g., FIPS 140-2 compliance).</li> <li>✓ Expected result: The system adheres to recognized cryptographic standards.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IDS-043]
<b>Requirement description:</b> The mobile health application must safeguard audit tools and audit logs from unauthorized deletion by implementing access controls and role-based permissions.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ V-222505: The application must protect audit tools from unauthorized deletion. Configure the application to protect audit tools from unauthorized deletions. Limit users to roles that</li> </ul>

are assigned the rights to edit or delete audit tools and establish file permissions that control access to the audit tools and audit tool capabilities and configuration settings [15].
✓ V-222502: The application must protect audit information from unauthorized deletion. Configure the application to protect audit data from unauthorized deletion. Limit users to roles that are assigned the rights to delete audit data and establish permissions that control access to the audit logs and audit configuration settings [15].
✓ V-222500: The application must protect audit information from any type of unauthorized read access. Configure the application to protect audit data from unauthorized access. Limit users to roles that are assigned the rights to view, edit or copy audit data, and establish permissions that control access to the audit logs and audit configuration settings [15].
<b>Priority:</b> Not described
<b>Rationale:</b> Protecting audit tools and logs ensures the integrity and availability of audit data for compliance, security monitoring, and forensic investigations.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li>1. <b>Access Control Test for Audit Tools:</b> <ul style="list-style-type: none"> <li>✓ Test: Attempt to delete or modify audit tools using a non-privileged account.</li> <li>✓ Expected result: The application denies access and logs the unauthorized attempt.</li> </ul> </li> <li>2. <b>Role-Based Permission Verification:</b> <ul style="list-style-type: none"> <li>✓ Test: Verify only authorized roles (e.g., system administrator) can manage audit tools and logs.</li> <li>✓ Expected result: Only designated roles can edit, delete, or configure audit tools and logs.</li> </ul> </li> <li>3. <b>File Permission Configuration Check:</b> <ul style="list-style-type: none"> <li>✓ Test: Review the file system permissions on audit tools and logs.</li> <li>✓ Expected result: Permissions are configured to restrict access to authorized users and roles.</li> </ul> </li> <li>4. <b>Unauthorized Deletion Attempt Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Attempt to delete an audit log entry with an unauthorized user account.</li> <li>✓ Expected result: Deletion is prevented, and the attempt is logged.</li> </ul> </li> <li>5. <b>Audit Log Retention Verification:</b> <ul style="list-style-type: none"> <li>✓ Test: Ensure the application retains logs as per the defined retention policy, even in cases of unauthorized deletion attempts.</li> <li>✓ Expected result: Logs remain intact and accessible per retention policies.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán

Juan Manuel Carrillo-de-Gea  
 Joaquín Nicolás  
 08/01/2026

**PUID:** [SECM-CAT-IDS-044]

**Requirement description:** The mobile health application must safeguard audit tools and audit logs from unauthorized deletion by implementing access controls and role-based permissions.

**Source:**

- ✓ V-222501: The application must protect audit information from unauthorized modification. Configure the application to protect audit data from unauthorized modification and changes. Limit users to roles that are assigned the rights to edit audit data and establish permissions that control access to the audit logs and audit configuration settings [15].
- ✓ V-222504: The application must protect audit tools from unauthorized modification. Configure the application to protect audit tools from unauthorized modifications. Limit users to roles that are assigned the rights to edit or update audit tools and establish file permissions that control access to the audit tools and audit tool capabilities and configuration settings [15].
- ✓ V-222503: The application must protect audit tools from unauthorized access. Configure the application to protect audit data from unauthorized access. Limit users to roles that are assigned the rights to view, edit or copy audit data, and establish file permissions that control access to the audit tools and audit tool capabilities and configuration settings [15].

**Priority:** Not described

**Rationale:** Protecting audit tools and logs ensures the integrity and availability of audit data for compliance, security monitoring, and forensic investigations.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

1. **Access Control Test for Viewing Audit Logs:**
  - ✓ Test: Attempt to access audit logs with a non-privileged account.
  - ✓ Expected result: The application denies access and logs the unauthorized attempt.
2. **Role-Based Access Verification:**
  - ✓ Test: Validate that only authorized roles (e.g., system administrators, auditors) can view or manage audit logs.
  - ✓ Expected result: Only users with designated roles can access audit logs.
3. **File Permission Configuration Check:**
  - ✓ Test: Inspect file system permissions for audit logs.
  - ✓ Expected result: Permissions are configured to prevent unauthorized viewing, ensuring only authorized accounts have access.

<p><b>4. Audit Log Monitoring Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Attempt to read audit logs using unauthorized methods (e.g., direct file access).</li> <li>✓ Expected result: Unauthorized read attempts are denied and logged by the system.</li> </ul> <p><b>5. Secure Configuration Validation:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Verify audit log encryption is applied during storage and transmission.</li> <li>✓ Expected result: Logs are encrypted, preventing unauthorized read access.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<p><b>Author and date:</b></p> <p>Carlos M. Mejía-Granda      José L Fernández-Alemán      Juan Manuel Carrillo-de-Gea      Joaquín Nicolás      08/01/2026</p>

<b>PUID:</b> [SECM-CAT-IDS-045]
<b>Requirement description:</b> The mobile health application must secure access to audit logs and tools by implementing cryptographic hashing, tamper-proof storage, and access safeguards to prevent misuse or compromise.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ 4.7. Communications and Operations Management: Security Requirement 49 – Securing Access to EHRI Audit Logs: The EHRI must secure access to audit records and must safeguard access to system audit tools and audit trails to prevent misuse or compromise [10]</li> <li>✓ V-222508: Application audit tools must be cryptographically hashed. Cryptographically hash the audit tool files used by the application. Store and protect the generated hash values for future reference [15].</li> <li>✓ 12.4.2. Protection of log Information: Audit records shall be secure and tamper-proof. Access to system audit tools and audit trails shall be safeguarded to prevent misuse or compromise [7].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Protecting audit records and tools from unauthorized access or tampering ensures data integrity, supports compliance with healthcare regulations, and preserves audit reliability.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li>1. <b>Access Restriction Validation:</b> <ul style="list-style-type: none"> <li>✓ Test: Attempt unauthorized access to audit logs and tools.</li> </ul> </li> </ol>

<ul style="list-style-type: none"> <li>✓ Expected result: Access is denied, and the attempt is logged as an event.</li> </ul>
<b>2. Cryptographic Hash Verification:</b>
<ul style="list-style-type: none"> <li>✓ Test: Generate and verify cryptographic hashes for audit tools.</li> <li>✓ Expected result: Any tampering with the tools is detected by hash mismatches.</li> </ul>
<b>3. Audit Log Tamper Protection:</b>
<ul style="list-style-type: none"> <li>✓ Test: Simulate tampering attempts on stored audit logs.</li> <li>✓ Expected result: Logs remain tamper-proof, and unauthorized modification attempts are flagged.</li> </ul>
<b>4. Access Control Review:</b>
<ul style="list-style-type: none"> <li>✓ Test: Check that access permissions for audit tools and logs are restricted to authorized roles only.</li> <li>✓ Expected result: Only designated administrators or auditors can access or modify these records.</li> </ul>
<b>5. Encryption of Audit Records:</b>
<ul style="list-style-type: none"> <li>✓ Test: Verify that audit logs are encrypted at rest and during transmission.</li> <li>✓ Expected result: Logs are encrypted, and unauthorized access attempts cannot reveal sensitive data.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IDS-046]
<b>Requirement description:</b> The mobile health application must implement automated backup processes to securely copy audit records to a separate system or storage component at least every seven days, in compliance with organizational policies and risk-level guidelines.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ V-222506: The application must back up audit records at least every seven days onto a different system or system component than the system or component being audited. Configure application backup settings to backup application audit logs every 7 days [15].</li> <li>✓ V-222638: Data backup must be performed at required intervals in accordance with DoD policy. Develop and implement backup procedures based on risk level of the system and in accordance with DoD policy [15].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Regularly backing up audit records ensures data availability, supports forensic analysis, and prevents data loss in the event of system failures or security incidents.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described

<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Automated Backup Process Validation:</b> <ul style="list-style-type: none"> <li>✓ Test: Verify that audit logs are automatically backed up every seven days.</li> <li>✓ Expected result: Backup operations occur as scheduled without manual intervention.</li> </ul> </li> <li><b>2. Backup Location Separation:</b> <ul style="list-style-type: none"> <li>✓ Test: Confirm that audit backups are stored on a separate system or storage medium from the primary application system.</li> <li>✓ Expected result: Backup locations are physically or logically separate from the audited system.</li> </ul> </li> <li><b>3. Backup Integrity Verification:</b> <ul style="list-style-type: none"> <li>✓ Test: Check that backed-up audit logs maintain data integrity during and after transfer.</li> <li>✓ Expected result: Backups are complete, accurate, and free from corruption.</li> </ul> </li> <li><b>4. Compliance with Policy:</b> <ul style="list-style-type: none"> <li>✓ Test: Ensure backup processes align with organizational policies and DoD guidelines.</li> <li>✓ Expected result: Backup intervals and procedures meet specified compliance requirements.</li> </ul> </li> <li><b>5. Restoration Testing:</b> <ul style="list-style-type: none"> <li>✓ Test: Perform a restoration test to recover audit logs from backups.</li> <li>✓ Expected result: Restored logs match original logs in content and structure.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IDS-047]
<b>Requirement description:</b> The mobile health application must implement a process to validate the integrity of audit tools by periodically verifying their cryptographic hash values to detect and prevent tampering.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ V-222509: The integrity of the audit tools must be validated by checking the files for changes in the cryptographic hash value. Establish a process to periodically check the audit tool cryptographic hashes to ensure the audit tools have not been tampered with [15].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Validating the integrity of audit tools ensures their reliability, prevents unauthorized modifications, and maintains the security of audit processes.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0

<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<p><b>1. Hash Generation and Storage:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Verify that cryptographic hash values are generated and securely stored for all audit tools.</li> <li>✓ Expected result: All audit tools have corresponding hash values stored securely and access to them is restricted.</li> </ul> <p><b>2. Periodic Hash Validation Process:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Confirm that a scheduled process is in place to validate the integrity of audit tools by comparing stored hash values with recalculated ones.</li> <li>✓ Expected result: The hash validation process executes periodically without failure or manual intervention.</li> </ul> <p><b>3. Tampering Detection and Alerting:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Simulate a modification to an audit tool and observe the system's response.</li> <li>✓ Expected result: The system detects the change, logs the event, and generates an alert for administrators.</li> </ul> <p><b>4. Cryptographic Standards Compliance:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Validate that the hash function used complies with recognized cryptographic standards (e.g., SHA-256).</li> <li>✓ Expected result: Hash values are generated using industry-standard cryptographic algorithms.</li> </ul> <p><b>5. Audit Log for Validation Events:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Verify that all hash validation activities, including outcomes, are logged.</li> <li>✓ Expected result: Validation events are logged with timestamps, tool identifiers, and validation outcomes.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IDS-048]
<b>Requirement description:</b> The mobile health application must ensure that during the uninstallation process, all confidential user data, including health information and app-specific credentials, is securely deleted from the device, storage media, and execution environment.
<b>Source:</b>

<ul style="list-style-type: none"> <li>✓ 1.11. Ensure that during application removal (uninstall operation), any confidential user data and the corresponding app-specific credentials are deleted from the execution environment, the device, and any other storage medium [16].</li> <li>✓ Delete all stored health information before discarding or reusing the mobile device [32].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Secure removal of user data during app uninstallation mitigates risks of data leakage, unauthorized access, and potential privacy violations.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Comprehensive Data Removal:</b> <ul style="list-style-type: none"> <li>✓ Test: Verify that the app deletes all user data and credentials stored locally and in associated app directories upon uninstallation.</li> <li>✓ Expected result: No residual data, including health information or credentials, remains on the device or in app-specific storage.</li> </ul> </li> <li><b>2. Remote Data Deletion:</b> <ul style="list-style-type: none"> <li>✓ Test: If applicable, ensure the app removes user data from connected cloud services or external storage upon user confirmation.</li> <li>✓ Expected result: User data stored remotely is deleted as per the app's data removal policy.</li> </ul> </li> <li><b>3. Secure Deletion Methods:</b> <ul style="list-style-type: none"> <li>✓ Test: Confirm that the app employs secure deletion mechanisms (e.g., overwriting or encryption) for sensitive data during uninstallation.</li> <li>✓ Expected result: Data deleted using secure methods cannot be recovered.</li> </ul> </li> <li><b>4. User Notification:</b> <ul style="list-style-type: none"> <li>✓ Test: Verify that the app informs the user about the data deletion process during uninstallation.</li> <li>✓ Expected result: Users are clearly notified that all data, including health records, will be removed.</li> </ul> </li> <li><b>5. Data Cleanup Verification:</b> <ul style="list-style-type: none"> <li>✓ Test: Perform forensic analysis after uninstallation to confirm the absence of residual sensitive data.</li> <li>✓ Expected result: Forensic tools find no trace of deleted user data.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda
José L Fernández-Alemán
Juan Manuel Carrillo-de-Gea
Joaquín Nicolás

08/01/2026

**PUID:** [SECM-CAT-IDS-049]

**Requirement description:** The mobile health application must store configuration and control files in a directory separate from user data to enhance security and prevent unauthorized access to application settings.

**Source:**

- ✓ V-222626: The designer must ensure the application does not store configuration and control files in the same directory as user data. Separate the application user data into a different directory than the application code and user file permissions to restrict user access to application configuration settings [15].

**Priority:** Not described

**Rationale:** Segregating application configuration files from user data reduces the risk of accidental or malicious modification of configuration files and enhances overall application security.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

**1. Directory Segregation:**

- ✓ Test: Verify that configuration and control files are stored in a distinct directory from user data.
- ✓ Expected result: Configuration files are located in a separate directory with no overlap with user data.

**2. Access Restrictions:**

- ✓ Test: Confirm that user-level permissions restrict access to the directory containing configuration and control files.
- ✓ Expected result: Only authorized processes and roles can access configuration files.

**3. Secure Path Validation:**

- ✓ Test: Inspect the file paths used by the application to ensure user data and configuration files are not co-located.
- ✓ Expected result: Application paths maintain strict separation between user data and configuration files.

**4. Data Integrity:**

- ✓ Test: Verify that unauthorized attempts to access or modify configuration files are logged and denied.
- ✓ Expected result: Audit logs capture unauthorized access attempts, and files remain unaltered.

**5. Configuration Review:**

- ✓ Test: Perform periodic reviews of the directory structure and permissions.
- ✓ Expected result: Directory structure and permissions comply with the application's security design.

<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda
José L Fernández-Alemán
Juan Manuel Carrillo-de-Gea
Joaquín Nicolás
08/01/2026

<b>PUID:</b> [SECM-CAT-IDS-050]
<b>Requirement description:</b> The mobile health application must ensure the confidentiality and integrity of stored information by implementing approved cryptographic mechanisms to protect sensitive data elements at rest, in compliance with DoD policy and data owner requirements.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ V-222587: The application must protect the confidentiality and integrity of stored information when required by DoD policy or the information owner. Identify data elements that require protection. Document the data types and specify protection requirements and methods used [15].</li> <li>✓ V-222588: The application must implement approved cryptographic mechanisms to prevent unauthorized modification of organization-defined information at rest on organization-defined information system components. Identify data elements that require protection. Document the data types and specify encryption requirements. Encrypt data according to DoD policy or data owner requirements [15].</li> <li>✓ V-222589: The application must use appropriate cryptography in order to protect stored DoD information when required by the information owner or DoD policy [15].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Protecting data at rest through encryption and integrity mechanisms mitigates the risk of unauthorized access or modification, ensuring compliance with security standards and safeguarding sensitive health information.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li>1. <b>Data Classification and Protection:</b> <ul style="list-style-type: none"> <li>✓ Test: Verify that all sensitive data elements requiring protection are identified and documented.</li> <li>✓ Expected result: A complete list of protected data elements with corresponding protection methods is available and aligns with DoD policy.</li> </ul> </li> </ol>

- |  |
|--|
| <p><b>2. Cryptographic Implementation:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Ensure that all sensitive data at rest is encrypted using approved cryptographic algorithms (e.g., AES-256).</li> <li>✓ Expected result: Data encryption mechanisms meet or exceed DoD cryptographic standards.</li> </ul> <p><b>3. Integrity Assurance:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Confirm that mechanisms, such as cryptographic hashes or digital signatures, are used to maintain data integrity.</li> <li>✓ Expected result: Data integrity mechanisms are in place and functional for all protected data.</li> </ul> <p><b>4. Access Control:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Validate that access to protected data is restricted to authorized users and processes.</li> <li>✓ Expected result: Access control policies effectively enforce restrictions, with unauthorized attempts logged.</li> </ul> <p><b>5. Policy Compliance:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Compare application implementation against DoD policy and data owner requirements for cryptographic protection.</li> <li>✓ Expected result: The application complies fully with documented policies and standards for data at rest.</li> </ul> |
|--|

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda  
 José L Fernández-Alemán  
 Juan Manuel Carrillo-de-Gea  
 Joaquín Nicolás  
 08/01/2026

**PUID:** [SECM-CAT-IDS-051]

**Requirement description:** The mobile health application must prevent the use of insecure cached data in HTTP connections and embedded web browsing components, such as WebViews, by ensuring cache storage is secure and not tampered with.

**Source:**

- ✓ 12.14. Do not use insecure cached data in HTTP connections and in embedded web browsing capabilities (e.g., WebViews). Caches are usually located on the device file system. Many platforms allow applications to place this cached data to insecure locations (e.g., sdcard on Android) in which they can be easily tampered [16].

**Priority:** Not described

**Rationale:** Storing HTTP or WebView cache data in insecure locations can lead to data tampering and unauthorized access, compromising sensitive health information and application integrity.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<p><b>1. Secure Cache Storage:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Verify that cached data is stored in a secure location, such as application-specific directories with restricted permissions.</li> <li>✓ Expected result: All cached data is stored securely and cannot be accessed or modified by unauthorized users or applications.</li> </ul> <p><b>2. Tamper Protection:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Ensure cached data is protected from tampering through measures such as cryptographic validation or access restrictions.</li> <li>✓ Expected result: Any attempt to modify cached data results in detection or prevention mechanisms being triggered.</li> </ul> <p><b>3. Cache Control Policies:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Validate that HTTP headers and WebView configurations enforce cache control policies (e.g., no-store, no-cache, must-revalidate).</li> <li>✓ Expected result: Cache control policies align with application security requirements, preventing sensitive data from being cached unnecessarily.</li> </ul> <p><b>4. Audit of Cache Usage:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Confirm the application logs or monitors attempts to access cached data in an unauthorized manner.</li> <li>✓ Expected result: Logs provide visibility into access attempts and demonstrate compliance with secure cache handling practices.</li> </ul> <p><b>5. Insecure Location Avoidance:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Check that cache data is not stored in insecure locations such as shared external storage (e.g., SD card on Android).</li> <li>✓ Expected result: Cache data is restricted to secure, application-controlled storage areas.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IDS-052]
<b>Requirement description:</b> The mobile health application must support remote wipe and kill switch capabilities to securely remove sensitive data in case of theft or loss of the device.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ 1.14. Applications on managed devices should leverage remote wipe and kill switch APIs to remove sensitive information from the device in the event of theft or loss [16].</li> <li>✓ Install and activate remote wiping or remote disabling: As discussed, the deletion system for personalized mobile health data can be deleted at the app level and the individual level [32].</li> </ul>

✓ IPC-006 Remote removal of my personal data on a lost mobile device [26].
<b>Priority:</b> Not described
<b>Rationale:</b> Remote wipe and kill switch mechanisms ensure sensitive health information is protected by enabling quick and secure data removal when a device is compromised, thereby mitigating risks of unauthorized access.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Remote Wipe Functionality:</b> <ul style="list-style-type: none"> <li>✓ Test: Simulate a remote wipe request to confirm that all sensitive data, including user credentials and health information, is securely deleted.</li> <li>✓ Expected result: Data is completely removed from the device without residual traces.</li> </ul> </li> <li><b>2. Kill Switch Activation:</b> <ul style="list-style-type: none"> <li>✓ Test: Simulate a kill switch activation to verify that the application becomes inaccessible and data is rendered unreadable.</li> <li>✓ Expected result: Application access is immediately revoked</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IDS-053]
<b>Requirement description:</b> Incorporate an application-specific "data kill switch" feature to securely delete sensitive data associated with the mobile health application, ensuring strong user authentication to prevent misuse.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ 1.15. Application developers may want to incorporate an application-specific "data kill switch" into their products, to allow the per-app deletion of their application's sensitive data when needed (strong authentication is required to protect misuse of such a feature) [16].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Protects personal health information (PHI) and other sensitive data by allowing secure, app-specific data deletion, enhancing data security in emergency scenarios or user-initiated actions.
<b>Number of Children:</b> 0

<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Kill Switch Activation:</b> <ul style="list-style-type: none"> <li>✓ Test: Trigger the data kill switch feature using the application's interface.</li> <li>✓ Expected result: All sensitive data specific to the application is securely deleted without affecting other system data.</li> </ul> </li> <li><b>2. Authentication Verification:</b> <ul style="list-style-type: none"> <li>✓ Test: Initiate a kill switch request without providing user authentication.</li> <li>✓ Expected result: The application denies the action and requests strong user authentication before proceeding.</li> </ul> </li> <li><b>3. Event Logging:</b> <ul style="list-style-type: none"> <li>✓ Test: Verify if a log entry is created upon activation of the data kill switch.</li> <li>✓ Expected result: A secure log is generated with details of the action, including timestamp and authenticated user identity.</li> </ul> </li> <li><b>4. Misuse Prevention:</b> <ul style="list-style-type: none"> <li>✓ Test: Simulate multiple unauthorized attempts to activate the kill switch.</li> <li>✓ Expected result: The application detects misuse and temporarily locks access to the feature after a defined number of failed attempts.</li> </ul> </li> <li><b>5. Secure Recovery Mechanism:</b> <ul style="list-style-type: none"> <li>✓ Test: Evaluate the application's ability to recover functionality post-data deletion.</li> <li>✓ Expected result: The app remains operational for non-sensitive features or prompts the user for a secure setup if reinstallation is required.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IDS-054]
<b>Requirement description:</b> Ensure the mobile health application does not store sensitive information in hidden fields or embed authentication data in any code, configuration files, or other files.
<b>Source:</b> <ul style="list-style-type: none"> <li>✓ V-222601: The application must not store sensitive information in hidden fields. Design and configure the application to not store sensitive information in hidden fields [15].</li> </ul>

✓ V-222642: The application must not contain embedded authentication data. Remove embedded authentication data stored in code, configuration files, scripts, HTML file, or any ASCII files [15].
<b>Priority:</b> Not described
<b>Rationale:</b> Storing sensitive data in hidden fields or embedding authentication credentials increases the risk of unauthorized access or data breaches, compromising patient confidentiality and application security.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Hidden Field Audit:</b> <ul style="list-style-type: none"> <li>✓ Test: Inspect the application's forms and elements for hidden fields containing sensitive data.</li> <li>✓ Expected result: No sensitive data is stored in hidden fields across all application features.</li> </ul> </li> <li><b>2. Authentication Data Review:</b> <ul style="list-style-type: none"> <li>✓ Test: Analyze code, configuration files, and scripts for embedded authentication data.</li> <li>✓ Expected result: No authentication credentials are stored in any files or scripts.</li> </ul> </li> <li><b>3. Data Transmission Validation:</b> <ul style="list-style-type: none"> <li>✓ Test: Verify sensitive data is securely transmitted through appropriate encrypted methods, avoiding exposure in hidden fields.</li> <li>✓ Expected result: All sensitive data transmission adheres to secure encryption standards.</li> </ul> </li> <li><b>4. Static Code Analysis:</b> <ul style="list-style-type: none"> <li>✓ Test: Run static code analysis tools to identify embedded sensitive data or credentials in the source code.</li> <li>✓ Expected result: The analysis returns no findings of embedded sensitive information.</li> </ul> </li> <li><b>5. Post-Deployment Monitoring:</b> <ul style="list-style-type: none"> <li>✓ Test: Use runtime monitoring tools to ensure sensitive data is not stored or exposed during application operation.</li> <li>✓ Expected result: The monitoring confirms compliance with data handling standards.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IDS-055]
<b>Requirement description:</b> Ensure that sensitive data stored by the mobile health application is encrypted and protected based on its classification and retained only for the duration necessary for its functionality.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ MASVS-STORAGE-1: The app securely stores sensitive data: Any sensitive data that is intentionally stored by the app is properly protected independently of the target location [20].</li> <li>✓ 1.1. Classify data storage according to sensitivity and apply controls accordingly (e.g. passwords, personal data, location, error logs, etc.). Process, store and use data according to its classification [5].</li> <li>✓ 1.5. Do not store/cache sensitive data (including keys) unless they are encrypted and if possible stored in a platform supported tamper-proof area [5].</li> <li>✓ 1.7. Do not store historical location data or other sensitive information on the device beyond the period required by the application. Assume that shared storage is untrusted - information may easily leak in unexpected ways through any shared storage. In particular: <ul style="list-style-type: none"> <li>○ Be aware of caches and temporary storage as a possible leakage channel, when shared with other apps.</li> <li>○ Be aware of shared storage such as address book, media gallery, audio files, as a possible leakage channel. For example, storing images with location metadata in the media-gallery allows that information to be shared in unintended ways.</li> <li>○ Do not store temporary cached data in a world readable directory [16]</li> </ul> </li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Ensure that sensitive data stored by the mobile health application is encrypted and protected based on its classification and retained only for the duration necessary for its functionality.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Data Classification and Protection:</b> <ul style="list-style-type: none"> <li>✓ Test: Review the application's data storage mechanism to confirm that all data is classified based on sensitivity and protected according to its classification.</li> <li>✓ Expected result: All sensitive data is encrypted and stored in secure, platform-supported areas.</li> </ul> </li> <li><b>2. Encryption Enforcement:</b> <ul style="list-style-type: none"> <li>✓ Test: Verify that encryption is applied to all stored sensitive data, including cached data and application-specific keys.</li> <li>✓ Expected result: All sensitive data is encrypted using industry-standard algorithms (e.g., AES-256).</li> </ul> </li> <li><b>3. Retention Policy Compliance:</b></li> </ol>

- ✓ Test: Assess the application's data retention mechanisms to ensure sensitive data is only stored for the necessary duration.
- ✓ Expected result: Sensitive data, such as location history, is deleted once it is no longer needed for application functionality.

#### 4. Storage Location Audit:

- ✓ Test: Inspect storage locations (e.g., media galleries, caches) for leakage of sensitive data or metadata.
- ✓ Expected result: Sensitive data is not stored in untrusted or world-readable locations.

#### 5. Shared Storage Usage:

- ✓ Test: Analyze the use of shared storage to ensure no sensitive information is unintentionally exposed.
- ✓ Expected result: Sensitive data is securely stored in tamper-proof areas or encrypted before being stored in shared locations.

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda  
José L Fernández-Alemán  
Juan Manuel Carrillo-de-Gea  
Joaquín Nicolás  
08/01/2026

**PUID:** [SECM-CAT-IDS-056]

**Requirement description:** Ensure the mobile health application securely deletes all cached data upon termination to prevent sensitive information exposure.

**Source:**

- ✓ 1.31. Delete application caches on app termination [16].

**Priority:** Not described

**Rationale:** Retaining cached data after app termination increases the risk of sensitive information being accessed by unauthorized users or other applications.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

#### 1. Cache Deletion on Termination:

- ✓ Test: Terminate the application and verify whether all cached data is securely deleted.
- ✓ Expected result: No cached data should remain accessible post-termination.

#### 2. Temporary Storage Clearance:

- ✓ Test: Inspect temporary storage directories (e.g., app cache folders) after app termination.

<ul style="list-style-type: none"> <li>✓ Expected result: Temporary storage should be cleared entirely upon app exit.</li> </ul> <p><b>3. Logging and Debug Data Clearance:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Confirm that no sensitive logging or debug data persists in the cache post-termination.</li> <li>✓ Expected result: All temporary logs and debug data are securely removed.</li> </ul> <p><b>4. Encryption During Runtime:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Verify that cached sensitive data is encrypted during app execution to reduce risk if not immediately deleted.</li> <li>✓ Expected result: Sensitive cache data is encrypted during runtime.</li> </ul> <p><b>5. Compliance with Platform Guidelines:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Ensure the app complies with platform-specific cache management practices (e.g., Android, iOS).</li> <li>✓ Expected result: Cache management adheres to best practices and platform security standards.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IDS-057]
<b>Requirement description:</b> Ensure the mobile health application removes all temporary files upon termination to prevent unauthorized access to sensitive data.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ SRG-APP-000516-MAPP-000065: The mobile app must remove temporary files when it terminates [17].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Retaining temporary files after application termination increases the risk of exposing sensitive information to unauthorized users or malicious applications.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Temporary File Deletion on Termination:</b></li> </ol> <ul style="list-style-type: none"> <li>✓ Test: Terminate the application and inspect the designated temporary file directories.</li> <li>✓ Expected result: No temporary files created by the application remain accessible after termination.</li> </ul>

<p><b>2. Secure Deletion Method:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Verify that temporary files are securely deleted, not merely unlinked, to prevent data recovery.</li> <li>✓ Expected result: Deleted temporary files cannot be recovered using standard data recovery tools.</li> </ul> <p><b>3. Runtime File Management:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Ensure that the application removes unnecessary temporary files during runtime as soon as they are no longer needed.</li> <li>✓ Expected result: Temporary files are deleted immediately after use to minimize risk.</li> </ul> <p><b>4. Logging and Debug Data Clearance:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Verify that no sensitive debug or log data remains in temporary file locations post-termination.</li> <li>✓ Expected result: All debug and log data is securely removed.</li> </ul> <p><b>5. Platform Compliance:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Confirm that the app adheres to platform-specific guidelines for temporary file storage and removal (e.g., Android cache directories, iOS sandboxing rules).</li> <li>✓ Expected result: Temporary file handling complies with platform security best practices.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<p><b>Author and date:</b>            Carlos M. Mejía-Granda            José L Fernández-Alemán            Juan Manuel Carrillo-de-Gea            Joaquín Nicolás            08/01/2026</p>

<b>PUID:</b> [SECM-CAT-IDS-058]
<b>Requirement description:</b> Ensure the mobile health application restricts the upload of files to only those of secure and approved types, implementing validation and processing controls to mitigate risks associated with dangerous file types.
<p><b>Source:</b></p> <ul style="list-style-type: none"> <li>✓ CWE-434: Unrestricted Upload of File with Dangerous Type            The product allows the attacker to upload or transfer files of dangerous types that can be automatically processed within the product's environment.            The phrase could be interpreted as the lack of restrictions on the size or number of uploaded files, which is a resource consumption issue. This weakness is caused by missing a security tactic during the architecture and design phase [35].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Allowing unrestricted file uploads can lead to exploitation by attackers who may upload malicious files, leading to unauthorized access, data corruption, or compromise of system integrity.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described

<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<p><b>1. File Type Validation:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Attempt to upload files of various types, including non-approved and potentially harmful formats.</li> <li>✓ Expected result: Only files of explicitly approved and safe formats are accepted.</li> </ul> <p><b>2. File Content Inspection:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Verify that uploaded files are scanned for potentially malicious content, even if they have an allowed extension.</li> <li>✓ Expected result: Files with dangerous content (e.g., embedded scripts, executables) are rejected.</li> </ul> <p><b>3. Size and Volume Restriction:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Attempt to upload files exceeding allowed size limits or upload a large volume of files.</li> <li>✓ Expected result: Uploads exceeding predefined size or volume limits are blocked.</li> </ul> <p><b>4. Secure Storage:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Inspect the location where uploaded files are stored for access control and encryption.</li> <li>✓ Expected result: Uploaded files are stored securely, with access limited to authorized processes or users.</li> </ul> <p><b>5. Error Handling and Logging:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Simulate file upload failures or attempts to upload restricted files.</li> <li>✓ Expected result: Errors are properly logged without exposing sensitive system information, and the user is notified appropriately.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IDS-059]
<b>Requirement description:</b> Ensure that database files containing sensitive data, such as iOS WebView caches, are securely and completely removed from the file system to prevent residual data exposure.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ 1.32. Database files that contain sensitive data (e.g., iOS WebView caches) must be manually removed from the file system. Deleting records using the database API will not necessarily lead to complete data removal from database structure [16].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Using the database API to delete records does not guarantee that sensitive information is fully removed from the database structure, leaving potential security vulnerabilities.
<b>Number of Children:</b> 0

<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Secure Deletion Implementation:</b> <ul style="list-style-type: none"> <li>✓ Test: Evaluate the mechanism used to delete sensitive database files.</li> <li>✓ Expected result: Files are fully removed from the file system using secure deletion methods.</li> </ul> </li> <li><b>2. Residual Data Check:</b> <ul style="list-style-type: none"> <li>✓ Test: Scan the file system for remnants of deleted database records after deletion.</li> <li>✓ Expected result: No residual sensitive data should be recoverable.</li> </ul> </li> <li><b>3. Automation of Manual Removal:</b> <ul style="list-style-type: none"> <li>✓ Test: Verify the implementation of automated processes for secure database file removal during application workflows.</li> <li>✓ Expected result: Sensitive database files are removed without manual intervention.</li> </ul> </li> <li><b>4. Configuration Validation:</b> <ul style="list-style-type: none"> <li>✓ Test: Inspect application configuration for safeguards ensuring all database files with sensitive content are identified and securely removed.</li> <li>✓ Expected result: Proper configurations are in place to identify and target sensitive files for secure deletion.</li> </ul> </li> <li><b>5. End-of-Life Data Handling:</b> <ul style="list-style-type: none"> <li>✓ Test: Evaluate the application's ability to securely erase sensitive data during uninstallation or decommissioning.</li> <li>✓ Expected result: All sensitive database files are removed securely during application uninstallation or decommissioning.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IDS-060]
<b>Requirement description:</b> Ensure the mobile application does not write data to persistent memory that is accessible to other applications to prevent unauthorized data access or leakage.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ SRG-APP-000243-MAPP-000049: The mobile app must not write data to persistent memory accessible to other applications [17].</li> </ul>

<b>Priority:</b> Not described
<b>Rationale:</b> Storing sensitive or private data in persistent memory locations accessible to other applications increases the risk of data breaches and violates security best practices.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Memory Access Restriction:</b> <ul style="list-style-type: none"> <li>✓ Test: Attempt to access data stored by the app using another unauthorized application.</li> <li>✓ Expected result: Unauthorized applications are unable to access any data written by the app.</li> </ul> </li> <li><b>2. Storage Location Audit:</b> <ul style="list-style-type: none"> <li>✓ Test: Inspect all storage operations to verify data is only saved in private app directories.</li> <li>✓ Expected result: All data is written to private, app-specific directories with restricted access.</li> </ul> </li> <li><b>3. Code Review for Persistent Storage:</b> <ul style="list-style-type: none"> <li>✓ Test: Conduct a review of the codebase for instances of persistent storage operations.</li> <li>✓ Expected result: No instances of data being stored in publicly accessible memory locations.</li> </ul> </li> <li><b>4. Secure Configuration Validation:</b> <ul style="list-style-type: none"> <li>✓ Test: Validate app configurations to ensure compliance with secure data storage policies.</li> <li>✓ Expected result: Configurations align with policies restricting data storage to private locations.</li> </ul> </li> <li><b>5. Dynamic Testing:</b> <ul style="list-style-type: none"> <li>✓ Test: Perform runtime testing of storage behaviors during various app operations.</li> <li>✓ Expected result: Data remains inaccessible to external apps under all conditions.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IDS-061]
<b>Requirement description:</b> The mobile health application must ensure correct handling of electronic health records (EHR). It must uniquely identify records and provide functionality to merge multiple records for the same patient/person, ensuring seamless consolidation of unintentional duplicates.

<b>Source:</b>
✓ 4.9.2. Correct Processing of Information Security Requirement 75 – Uniquely Identifying: b) Be capable of merging two or more EHR records if it is determined that multiple records for the same patient/person have been unintentionally created [10].
<b>Priority:</b> Not described
<b>Rationale:</b> Accurate record management ensures that healthcare providers access complete and consolidated patient information, supporting effective care delivery and compliance with healthcare standards.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li>1. <b>EHR Record Uniqueness:</b> <ul style="list-style-type: none"> <li>✓ Test: Validate the application's ability to uniquely identify each EHR record using defined unique identifiers.</li> <li>✓ Expected result: Each record is assigned a unique identifier, preventing duplication.</li> </ul> </li> <li>2. <b>Record Merge Capability:</b> <ul style="list-style-type: none"> <li>✓ Test: Test the application's ability to merge multiple records for the same patient/person after confirmation of duplication.</li> <li>✓ Expected result: Duplicate records are accurately merged while retaining data integrity.</li> </ul> </li> <li>3. <b>Audit Trail Verification:</b> <ul style="list-style-type: none"> <li>✓ Test: Inspect generated audit logs for each merge operation.</li> <li>✓ Expected result: Logs include details of the merged records, timestamps, and the user performing the merge.</li> </ul> </li> <li>4. <b>Regulatory Compliance Check:</b> <ul style="list-style-type: none"> <li>✓ Test: Validate the merging process against established health data standards and regulations.</li> <li>✓ Expected result: Merging complies with all relevant standards, maintaining patient data security and privacy.</li> </ul> </li> <li>5. <b>Dynamic Testing of Merge Operations:</b> <ul style="list-style-type: none"> <li>✓ Test: Perform runtime tests to evaluate the merging behavior under different scenarios.</li> <li>✓ Expected result: The system accurately merges records and maintains operational integrity.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

**PUID:** [SECM-CAT-IDS-062]

**Requirement description:** The mobile health application must ensure protection Against Tampered Executables.

**Source:**

- ✓ Protection Against Tampered Versions: The system must ensure that interactions from modified or tampered app binaries are identified and handled securely to protect app functionality and data [18].
- ✓ Cryptographic Verification of Executables: If the system retrieves executable files from external storage, those files must be signed and cryptographically verified before they are dynamically loaded to ensure their authenticity and integrity [18].

**Priority:** Not described

**Rationale:** The mobile health application must ensure protection against tampered app binaries and dynamically loaded executables. Interactions from modified binaries must be identified and handled securely to maintain the integrity of app functionality and data. Executable files retrieved from external storage must be signed and cryptographically verified prior to dynamic loading.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

**1. Tampered Binary Detection:**

- ✓ Test: Execute the app with a modified or tampered binary.
- ✓ Expected result: The app detects the tampered binary and terminates or restricts operations securely.

**2. Cryptographic Signature Validation:**

- ✓ Test: Attempt to load executable files from external storage with invalid or missing cryptographic signatures.
- ✓ Expected result: The system rejects unsigned or invalid executables and prevents their execution.

**3. Audit Logging:**

- ✓ Test: Verify audit logs for tampered or invalid executable file loading attempts.
- ✓ Expected result: Logs capture details of the tampered or invalid attempts, including timestamps and file metadata.

**4. Code Review for Verification Logic:**

- ✓ Test: Review the codebase for proper implementation of cryptographic signature verification mechanisms.
- ✓ Expected result: All dynamically loaded executables undergo signature checks using secure cryptographic methods.

**5. Dynamic Testing Under Adversarial Conditions:**

- ✓ Test: Simulate various tampering scenarios and evaluate the app's response.

✓ Expected result: The app consistently identifies tampered interactions and enforces protective measures without data leakage.
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IDS-063]
<b>Requirement description:</b> Sensitive data must never be stored in publicly accessible locations, as external storage is globally readable, writable, and modifiable by other applications.
<b>Source:</b> <ul style="list-style-type: none"> <li>✓ Non-Sensitive Data Storage: Files created on external storage, such as SD cards, are globally readable and writable. Because external storage can be removed by the user and also modified by any application, only store non-sensitive information using external storage [18].</li> <li>✓ MASVS-STORAGE-2: The app prevents leakage of sensitive data: Prevent the unintentional storage or exposure of sensitive data in publicly accessible locations. This control addresses potential leaks that can be avoided through proactive measures by the developer [20].</li> <li>✓ 1.1. Classify data storage according to sensitivity and apply controls accordingly (e.g. passwords, personal data, location, error logs, etc.). Process, store and use data according to its classification [16].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> By limiting sensitive data storage to private locations, the application protects against unauthorized access, tampering, and exposure, ensuring the security and privacy of user information.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b> <ol style="list-style-type: none"> <li><b>1. Data Classification and Control:</b> <ul style="list-style-type: none"> <li>✓ Test: Review the classification and storage logic implemented in the application.</li> <li>✓ Expected result: Sensitive data is classified appropriately and stored only in secure, private locations.</li> </ul> </li> <li><b>2. External Storage Content Audit:</b> <ul style="list-style-type: none"> <li>✓ Test: Inspect files written to external storage during app operations.</li> </ul> </li> </ol>

<ul style="list-style-type: none"> <li>✓ Expected result: No sensitive data, such as passwords, personal identifiers, or logs containing confidential information, is stored externally.</li> </ul> <p><b>3. Code Review for Storage Operations:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Analyze the codebase for storage methods involving external storage.</li> <li>✓ Expected result: Only non-sensitive information is written to external storage, and sensitive information is securely stored internally.</li> </ul> <p><b>4. Dynamic Testing for Storage Behavior:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Simulate app scenarios that involve file storage and examine resulting file locations and access permissions.</li> <li>✓ Expected result: All sensitive data remains inaccessible to external entities and is securely stored.</li> </ul> <p><b>5. Tampering and Data Integrity Testing:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Modify external storage data and observe the app's handling of altered data.</li> <li>✓ Expected result: The app does not rely on or process tampered external storage data, ensuring no compromise in functionality or security.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IDS-064]
<b>Requirement description:</b> The mobile health application must implement monitoring mechanisms to oversee all data interactions with external storage. Only non-sensitive and validated data may be written to or read from external storage to maintain application security and integrity.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ External Storage Monitoring: The system must monitor any data being written to or read from external storage and ensure that only non-sensitive, validated data is accessed to maintain application integrity [18].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> By actively monitoring external storage operations, the application ensures the protection of its data integrity, prevents unauthorized tampering, and reduces the risk of security breaches due to improper data handling.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis

<b>Validation criteria:</b>
<b>1. External Storage Write Monitoring:</b>
✓ Test: Track all data being written to external storage during app operations.
✓ Expected result: Only non-sensitive and validated data is stored on external storage.
<b>2. External Storage Read Validation:</b>
✓ Test: Inspect the application's process for reading data from external storage.
✓ Expected result: Data read from external storage is validated for integrity and complies with the application's security requirements.
<b>3. Code Inspection for External Storage Handling:</b>
✓ Test: Conduct a code review to identify storage methods involving external storage interactions.
✓ Expected result: All external storage operations are restricted to non-sensitive data and include validation checks.
<b>4. Dynamic Testing for Unauthorized Access:</b>
✓ Test: Simulate unauthorized modifications to external storage and observe app behavior.
✓ Expected result: The application detects tampering, ignores invalid data, and prevents functionality compromise.
<b>5. Runtime Monitoring for External Access:</b>
✓ Test: Use dynamic tools to monitor runtime behavior for external storage reads and writes.
✓ Expected result: All storage operations are logged, monitored, and comply with defined security standards.
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IDS-065]
<b>Requirement description:</b> The mobile health application must implement hardware, software, and procedural mechanisms to log and monitor all activities within systems that process or access electronic protected health information (ePHI).
<b>Source:</b>
✓ § 164.312 Technical safeguards. (b) Standard: Audit controls. Implement hardware, software, and/or procedural mechanisms that record and examine activity in information systems that contain or use electronic protected health information [9].
✓ Logging and Monitoring of API Activity: Implement API usage logging and monitoring mechanisms to detect suspicious activity, unauthorized access, or abnormal usage patterns, and trigger alerts when necessary [18].
<b>Priority:</b> Not described

<b>Rationale:</b> Implementing robust logging and monitoring mechanisms ensures the application meets regulatory compliance, such as HIPAA, and mitigates risks associated with unauthorized access or misuse of ePHI, safeguarding patient confidentiality.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. API Activity Logging:</b> <ul style="list-style-type: none"> <li>✓ Test: Execute API calls involving ePHI under various scenarios.</li> <li>✓ Expected result: All API activity is logged, including request details, timestamps, and user identities.</li> </ul> </li> <li><b>2. Monitoring Abnormal API Usage:</b> <ul style="list-style-type: none"> <li>✓ Test: Simulate abnormal API usage patterns, such as rapid consecutive calls or access from unknown IP addresses.</li> <li>✓ Expected result: The system identifies abnormal patterns and triggers an alert.</li> </ul> </li> <li><b>3. Access Control Logging:</b> <ul style="list-style-type: none"> <li>✓ Test: Attempt authorized and unauthorized access to ePHI systems.</li> <li>✓ Expected result: Both successful and failed access attempts are logged with sufficient detail.</li> </ul> </li> <li><b>4. Alert Generation for Suspicious Activity:</b> <ul style="list-style-type: none"> <li>✓ Test: Simulate suspicious activities such as data access attempts from untrusted sources.</li> <li>✓ Expected result: Alerts are generated and sent to the designated administrators.</li> </ul> </li> <li><b>5. Audit Log Integrity Verification:</b> <ul style="list-style-type: none"> <li>✓ Test: Inspect audit logs for consistency, completeness, and integrity.</li> <li>✓ Expected result: Logs remain tamper-proof and maintain data integrity under all circumstances.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IDS-066]
<b>Requirement description:</b> The mobile health application must utilize internal storage for storing sensitive data wherever feasible. This approach eliminates the dependency on external storage and reduces the need for storage-related permissions, enhancing the security of sensitive information
<b>Source:</b>

✓ Internal Storage for Data: The app must use internal storage instead of external storage, where feasible, to avoid the need for storage-related permissions [18].
<b>Priority:</b> Not described
<b>Rationale:</b> By using internal storage, the application ensures enhanced protection for sensitive data, minimizing the risk of unauthorized access through external storage mechanisms and complying with secure development practices for mobile health applications.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Internal Storage Configuration:</b> <ul style="list-style-type: none"> <li>✓ Test: Attempt to access data stored by the app in internal storage using unauthorized external applications.</li> <li>✓ Expected result: Unauthorized access to internal storage data is denied.</li> </ul> </li> <li><b>2. Permission Optimization:</b> <ul style="list-style-type: none"> <li>✓ Test: Inspect the app's manifest for storage-related permissions.</li> <li>✓ Expected result: The app does not request unnecessary storage-related permissions.</li> </ul> </li> <li><b>3. Secure Data Handling:</b> <ul style="list-style-type: none"> <li>✓ Test: Verify that sensitive data operations, such as saving or retrieving, are confined to internal storage locations.</li> <li>✓ Expected result: Sensitive data is exclusively stored in app-specific directories within internal storage.</li> </ul> </li> <li><b>4. Dynamic Testing for Storage Behavior:</b> <ul style="list-style-type: none"> <li>✓ Test: Monitor data storage operations during runtime using debugging tools.</li> <li>✓ Expected result: All sensitive data remains restricted to internal storage throughout app operations.</li> </ul> </li> <li><b>5. Code Review for Storage Paths:</b> <ul style="list-style-type: none"> <li>✓ Test: Review the codebase to ensure no sensitive data is written to external storage.</li> <li>✓ Expected result: All storage operations involving sensitive data use internal storage paths.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda
José L Fernández-Alemán
Juan Manuel Carrillo-de-Gea
Joaquín Nicolás
08/01/2026

<b>PUID:</b> [SECM-CAT-IDS-067]
---------------------------------

**Requirement description:** The mobile health application must utilize content providers for securely sharing data with other app processes. Content providers must implement fine-grained, controlled read and write permissions, ensuring the secure exchange of data while preventing unauthorized access or misuse.

**Source:**

- ✓ Use Content Provider for Data Sharing: The system must utilize a content provider for sharing data with other app processes. This approach must enable controlled read and write permissions for other apps, ensuring secure data sharing [18].
- ✓ 1.17. Restrict the data that is shared with other applications (e.g., by implementing an Android Content Provider). This can be accomplished using fine-grained permissions (ensure permissions are protected using signature protection level on Android) [16].
- ✓ Write Permission Misuse Prevention: Avoid assuming that the write permission is inherently secure. Developers must be aware that SQL statements with creative WHERE clauses may be used by attackers to infer or confirm sensitive data. Write permissions must be treated with caution, especially if the content provider's data structure is predictable [18].

**Priority:** Not described

**Rationale:** Implementing secure content providers for data sharing reduces the risk of unauthorized data access and ensures compliance with best practices for secure development in mobile health applications.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

**1. Controlled Data Sharing Implementation:**

- ✓ Test: Attempt to access the shared data through the content provider using unauthorized applications.
- ✓ Expected result: Unauthorized applications are denied access to read or write the shared data.

**2. Permission Configuration Validation:**

- ✓ Test: Inspect the app's manifest and content provider configurations for proper permission settings.
- ✓ Expected result: Read and write permissions are protected using the signature protection level.

**3. SQL Injection Mitigation:**

- ✓ Test: Validate SQL queries used by the content provider for susceptibility to SQL injection or data inference attacks.
- ✓ Expected result: Queries are parameterized, and no creative WHERE clauses allow data inference.

**4. Data Scope Restriction:**

- ✓ Test: Verify that only the necessary data is exposed through the content provider.
- ✓ Expected result: Content provider shares only the minimum required data for app functionality.

<b>5. Dynamic Behavior Testing:</b>
✓ Test: Monitor runtime behavior of the content provider during data sharing operations.
✓ Expected result: Data sharing operations respect defined permissions, and sensitive data is securely shared.
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IDS-068]
<b>Requirement description:</b> The mobile health application must enforce secure practices for accessing content providers. All data handling operations, including reading and writing data, must mitigate potential security threats such as unauthorized access and data manipulation.
<b>Source:</b>
✓ Secure Access to Content Provider: Any application that accesses a content provider must follow secure practices, ensuring all data handling is performed in a way that mitigates potential security threats, including unauthorized access and data manipulation [18].
<b>Priority:</b> Not described
<b>Rationale:</b> Enforcing secure access to content providers ensures sensitive data within mobile health applications is protected against unauthorized access and manipulation, safeguarding user privacy and data integrity.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Access Control Testing:</b> <ul style="list-style-type: none"> <li>✓ Test: Attempt to access the content provider using unauthorized applications or processes.</li> <li>✓ Expected result: Unauthorized access is denied, and only applications with appropriate permissions can access the content provider.</li> </ul> </li> <li><b>2. Data Manipulation Prevention:</b> <ul style="list-style-type: none"> <li>✓ Test: Attempt to modify data within the content provider without proper authorization.</li> <li>✓ Expected result: Unauthorized modifications are blocked, and the integrity of the data is maintained.</li> </ul> </li> </ol>

<p><b>3. Configuration Verification:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Review the manifest file to ensure permissions and access settings for the content provider are correctly configured.</li> <li>✓ Expected result: The content provider is configured with secure read and write permissions, leveraging signature-level protection where appropriate.</li> </ul> <p><b>4. SQL Injection Mitigation:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Analyze SQL queries used by the content provider for vulnerabilities to injection attacks.</li> <li>✓ Expected result: Queries are parameterized, and no mechanisms allow unauthorized inference or modification of data.</li> </ul> <p><b>5. Runtime Behavior Monitoring:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Monitor real-time interactions with the content provider during app operation.</li> <li>✓ Expected result: All access and data handling comply with predefined secure practices, and sensitive data remains protected.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<p><b>Author and date:</b>            Carlos M. Mejía-Granda            José L Fernández-Alemán            Juan Manuel Carrillo-de-Gea            Joaquín Nicolás            08/01/2026</p>

<b>PUID:</b> [SECM-CAT-IDS-069]
<b>Requirement description:</b> For mobile health applications, content providers must use the <code>android:exported</code> attribute to control external access. If a content provider is not intended to be accessed by other applications, the <code>android:exported</code> attribute in the manifest must be set to <code>false</code> . For cases where external access is required, the attribute must be set to <code>true</code> , ensuring proper data sharing while maintaining security.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ Export Control: If a content provider is not intended to be accessed by other applications, the <code>android:exported</code> attribute in the manifest must be set to <code>false</code>. If external access is needed, the attribute must be set to <code>true</code> to allow proper access to the data [18].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Explicitly managing the <code>android:exported</code> attribute ensures that only intended content providers are accessible externally, reducing the risk of unauthorized data access or manipulation in mobile health applications.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined

<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Manifest Attribute Validation:</b> <ul style="list-style-type: none"> <li>✓ Test: Inspect the android:exported attribute for all content provider declarations in the manifest file.</li> <li>✓ Expected result: The android:exported attribute is explicitly set to false for private content providers and to true only for those requiring external access.</li> </ul> </li> <li><b>2. Access Behavior Testing:</b> <ul style="list-style-type: none"> <li>✓ Test: Attempt to access a non-exported content provider from an external application.</li> <li>✓ Expected result: Access to the non-exported content provider is denied.</li> </ul> </li> <li><b>3. Security Review of Exported Providers:</b> <ul style="list-style-type: none"> <li>✓ Test: Evaluate the security measures in place for content providers where android:exported is set to true.</li> <li>✓ Expected result: Adequate security measures, such as permissions and validation checks, are implemented for exported content providers.</li> </ul> </li> <li><b>4. Dynamic Behavior Testing:</b> <ul style="list-style-type: none"> <li>✓ Test: Test runtime behavior to ensure content providers with android:exported=false are inaccessible externally.</li> <li>✓ Expected result: Non-exported content providers remain inaccessible during app operations.</li> </ul> </li> <li><b>5. Compliance with Data Sharing Policies:</b> <ul style="list-style-type: none"> <li>✓ Test: Ensure exported content providers comply with data sharing policies and permissions required for external applications.</li> <li>✓ Expected result: Exported providers securely share data only with authorized applications and processes.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IDS-070]
<b>Requirement description:</b> Permissions must align with the principle of least privilege, ensuring access is limited to the minimum necessary for the task. Additional permissions may be extended only when explicitly required.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ Export Control: Permission Limitation: When creating an exported content provider, specify distinct permissions for read and write operations, or use a single permission if both operations are required. Ensure permissions are limited to what is necessary for the task, with the option to extend permissions later if required [18].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Defining specific permissions for exported content providers ensures controlled data access and supports compliance with security policies, reducing the risk of unauthorized operations in mobile health applications.

<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Permission Definition Audit:</b> <ul style="list-style-type: none"> <li>✓ Test: Review the manifest file to ensure exported content providers declare permissions for read and write operations.</li> <li>✓ Expected result: Separate or combined permissions are defined for read and write access as appropriate, and permissions align with the intended use.</li> </ul> </li> <li><b>2. Access Control Testing:</b> <ul style="list-style-type: none"> <li>✓ Test: Attempt unauthorized read and write operations on the exported content provider.</li> <li>✓ Expected result: Unauthorized access is denied, and only authorized entities can perform the operations.</li> </ul> </li> <li><b>3. Least Privilege Assessment:</b> <ul style="list-style-type: none"> <li>✓ Test: Inspect the scope of permissions assigned to the content provider.</li> <li>✓ Expected result: Permissions are restricted to the necessary operations, minimizing exposure of sensitive data.</li> </ul> </li> <li><b>4. Permission Extension Validation:</b> <ul style="list-style-type: none"> <li>✓ Test: Verify the process for adding new permissions to the content provider.</li> <li>✓ Expected result: Additional permissions are granted only through a controlled and documented process.</li> </ul> </li> <li><b>5. Dynamic Behavior Testing:</b> <ul style="list-style-type: none"> <li>✓ Test: Perform runtime testing of read and write operations using assigned permissions.</li> <li>✓ Expected result: The provider enforces permissions effectively, and operations are executed securely.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IDS-071]
<b>Requirement description:</b> Mobile health applications must restrict data access to authorized applications as defined by predefined permissions. Shared data files must not be globally accessible. Access should be granted only to applications or entities with the appropriate private and corresponding public keys, ensuring secure and controlled data sharing.
<b>Source:</b>

<ul style="list-style-type: none"> <li>✓ Limit Data Access to Specific Applications: The system must ensure that data access is restricted to specific applications as per predefined permissions, avoiding global access to shared data files [18].</li> <li>✓ File or data sharing is not enabled at the app layer, although the blockchain resides on a decentralized network. The access to data is provided only to those with the private and corresponding public key [32].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Restricting data access to authorized applications enhances security and prevents unauthorized entities from accessing sensitive health data. Key-based access control aligns with the principles of secure data sharing in decentralized networks, ensuring data confidentiality and integrity.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Access Control Validation:</b> <ul style="list-style-type: none"> <li>✓ Test: Attempt unauthorized access to shared data files.</li> <li>✓ Expected result: Unauthorized applications or entities cannot access the data.</li> </ul> </li> <li><b>2. Predefined Permissions Audit:</b> <ul style="list-style-type: none"> <li>✓ Test: Inspect the permission settings for data access in the application configuration.</li> <li>✓ Expected result: Permissions are explicitly defined for specific applications, with no global or broad access settings.</li> </ul> </li> <li><b>3. Key-Based Access Testing:</b> <ul style="list-style-type: none"> <li>✓ Test: Attempt to access shared data using incorrect or unauthorized private-public key pairs.</li> <li>✓ Expected result: Access is denied for unauthorized key pairs, and only valid keys are accepted.</li> </ul> </li> <li><b>4. Data Sharing Restriction Review:</b> <ul style="list-style-type: none"> <li>✓ Test: Validate that no unnecessary data sharing mechanisms are enabled at the application layer.</li> <li>✓ Expected result: Data sharing is limited to the blockchain or equivalent decentralized network infrastructure with defined access protocols.</li> </ul> </li> <li><b>5. Runtime Behavior Analysis:</b> <ul style="list-style-type: none"> <li>✓ Test: Monitor data access operations during app runtime to ensure compliance with predefined permissions and key-based access controls.</li> <li>✓ Expected result: All data access operations comply with defined permissions and utilize secure keys.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda
José L Fernández-Alemán

Juan Manuel Carrillo-de-Gea  
 Joaquín Nicolás  
 08/01/2026

**PUID:** [SECM-CAT-IDS-072]

**Requirement description:** Mobile health applications must not retain sensitive data in memory for longer than necessary. Sensitive data must be explicitly cleared from memory after its use to reduce the risk of unauthorized access or memory dumps.

**Source:**

- ✓ 2.10: Verify that the app does not hold sensitive data in memory longer than necessary, and memory is cleared explicitly after use [19].

**Priority:** Not described

**Rationale:** Explicitly clearing sensitive data from memory minimizes exposure risks, including unauthorized access and memory-based attacks. This practice ensures compliance with secure coding standards and enhances overall application security.

**Number of Children:** 0**Number of parents:** 0**Cycles:** 0**Number Audit:** 0**Child PUIDs:** Not described**Parent PUIDs:** Not described**Exclusion PUIDs:** Not described**Importance:** Not described**Current state:** To be determined**Verification method:** Demonstration/Analysis**Validation criteria:****1. Memory Retention Audit:**

- ✓ Test: Analyze memory allocation and retention for sensitive data during app operations.
- ✓ Expected result: Sensitive data is retained only during active usage and cleared immediately afterward.

**2. Explicit Memory Clearance Validation:**

- ✓ Test: Review the codebase for explicit memory clearance methods (e.g., overwriting variables).
- ✓ Expected result: All sensitive data is explicitly cleared from memory after use.

**3. Runtime Memory Testing:**

- ✓ Test: Perform runtime analysis using debugging tools to detect residual sensitive data in memory.
- ✓ Expected result: No residual sensitive data remains in memory after its intended use.

**4. Dynamic Behavior Testing:**

- ✓ Test: Execute dynamic tests during sensitive data operations to verify memory clearing functionality.
- ✓ Expected result: Memory allocated for sensitive data is freed promptly after use.

**5. Security Policy Compliance Review:**

- ✓ Test: Assess compliance with policies requiring minimal retention of sensitive data in memory.
- ✓ Expected result: The application meets security policies for sensitive data handling and memory management.

**Requested by:** The organization**Responsible:** Developer

<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda
José L. Fernández-Alemán
Juan Manuel Carrillo-de-Gea
Joaquín Nicolás
08/01/2026

<b>PUID:</b> [SECM-CAT-IDS-073]
<b>Requirement description:</b> Mobile health applications must utilize secure system credential storage facilities to safeguard sensitive data, including user credentials and cryptographic keys. The application should leverage industry-standard key management practices, storing data using tools like Credential Manager or equivalent secure mechanisms, and avoiding direct storage of sensitive data such as passwords on the device.
<b>Source:</b>
<ul style="list-style-type: none"><li>✓ 2.1: Verify that system credential storage facilities are used appropriately to store sensitive data, such as user credentials or cryptographic keys [19].</li><li>✓ Secure Credential Storage: Use Credential Manager for passwordless authentication via passkeys or federated sign-ins (e.g., Sign in with Google). Do not store user IDs or passwords on the device. Instead, use a short-lived, service-specific authorization token after initial authentication [18].</li><li>✓ MASVS-CRYPTO-2: The app performs key management according to industry best practices: Even the strongest cryptography would be compromised by poor key management. This control covers the management of cryptographic keys throughout their lifecycle, including key generation, storage and protection [20].</li><li>✓ 1.1. Classify data storage according to sensitivity and apply controls accordingly (e.g. passwords, personal data, location, error logs, etc.). Process, store and use data according to its classification [16].</li></ul>

<b>Priority:</b> Not described
<b>Rationale:</b> Ensuring the secure storage and management of sensitive data and credentials minimizes the risk of unauthorized access, reduces exposure to attacks, and aligns with best practices in cryptographic key management and authentication. This is critical for maintaining user trust and regulatory compliance in mobile health applications.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
1. <b>System Credential Storage Usage:</b>

<ul style="list-style-type: none"> <li>✓ Test: Verify that sensitive data, including user credentials, is stored using secure system storage facilities (e.g., Credential Manager).</li> <li>✓ Expected result: Sensitive data is securely stored using platform-recommended mechanisms.</li> </ul>
<b>2. Passwordless Authentication Implementation:</b>
<ul style="list-style-type: none"> <li>✓ Test: Assess the use of passkeys or federated sign-ins for user authentication.</li> <li>✓ Expected result: Application supports secure, passwordless authentication methods.</li> </ul>
<b>3. Key Management Practices Audit:</b>
<ul style="list-style-type: none"> <li>✓ Test: Review cryptographic key management practices to ensure adherence to industry standards, including secure generation, storage, and usage.</li> <li>✓ Expected result: Keys are managed securely throughout their lifecycle.</li> </ul>
<b>4. Short-lived Token Verification:</b>
<ul style="list-style-type: none"> <li>✓ Test: Inspect the implementation of service-specific short-lived authorization tokens.</li> <li>✓ Expected result: Tokens are used in place of passwords for continued user sessions.</li> </ul>
<b>5. Data Classification Review:</b>
<ul style="list-style-type: none"> <li>✓ Test: Analyze storage operations to confirm data classification is applied, and controls are implemented according to sensitivity.</li> <li>✓ Expected result: Sensitive data is classified and protected according to its sensitivity level.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IDS-074]
<b>Requirement description:</b> Mobile health applications must implement encryption and decryption mechanisms for all electronic protected health information (ePHI) stored on client devices or transmitted through the application.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ § 164.312 Technical safeguards. <ul style="list-style-type: none"> <li>(a) <ul style="list-style-type: none"> <li>(2) Implementation specifications: <ul style="list-style-type: none"> <li>(iii) Encryption and decryption (Addressable): <ul style="list-style-type: none"> <li>Implement a mechanism to encrypt and decrypt electronic protected health information [9].</li> </ul> </li> </ul> </li> </ul> </li> </ul> </li> <li>✓ Secure Avoid Weak Patterns: 4. If client-side data storage is necessary, encrypt the data using an encryption key securely derived from the user's login credentials. However, there are additional risks that the data will be decrypted via binary attacks. [3].</li> <li>✓ Install and enable encryption: All data are encrypted at multiple levels [32].</li> <li>✓ Sensitive Information should be encrypted [28].</li> <li>✓ 4.3.3. Device integrity:</li> </ul>

Encryption of data at rest
All systems that serve, manage, and protect systems that serve patient information use disk encryption. All archived patient information and server system files are stored off-site/remotely via encrypted communication with a backup service [37].
<b>Priority:</b> Not described
<b>Rationale:</b> Encrypting ePHI ensures data confidentiality, aligns with technical safeguard requirements for health information, and protects against data breaches. Proper encryption minimizes risks associated with unauthorized access, ensuring compliance with privacy regulations and fostering user trust in mobile health applications.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Data Encryption Validation:</b> <ul style="list-style-type: none"> <li>✓ Test: Verify that all ePHI is encrypted using strong, industry-standard encryption algorithms (e.g., AES-256).</li> <li>✓ Expected result: All sensitive data is encrypted and cannot be accessed in plaintext.</li> </ul> </li> <li><b>2. Encryption Key Management:</b> <ul style="list-style-type: none"> <li>✓ Test: Inspect the key management process to confirm that encryption keys are securely derived and stored.</li> <li>✓ Expected result: Keys are protected using a secure key management system or derived securely from user credentials.</li> </ul> </li> <li><b>3. Secure Backup Encryption:</b> <ul style="list-style-type: none"> <li>✓ Test: Validate that all archived patient information and backups are encrypted during storage and transmission.</li> <li>✓ Expected result: Archived and transmitted data is encrypted with strong protocols (e.g., TLS 1.3 for communication).</li> </ul> </li> <li><b>4. Device Integrity Testing:</b> <ul style="list-style-type: none"> <li>✓ Test: Validate disk encryption mechanisms on devices storing patient data.</li> <li>✓ Expected result: Device integrity is ensured through enabled and verified disk encryption.</li> </ul> </li> <li><b>5. Binary Attack Prevention:</b> <ul style="list-style-type: none"> <li>✓ Test: Perform security testing to detect vulnerabilities in client-side storage encryption, focusing on binary attack resistance.</li> <li>✓ Expected result: Encryption mechanisms withstand attempts to decrypt data via binary analysis or reverse engineering.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda
José L Fernández-Alemán
Juan Manuel Carrillo-de-Gea
Joaquín Nicolás

08/01/2026

<b>PUID:</b> [SECM-CAT-IDS-075]
<b>Requirement description:</b> Mobile health applications must ensure that the "Remember Me" functionality does not store user passwords on the device. Instead, secure mechanisms such as short-lived tokens or encrypted identifiers should be used to maintain session persistence without compromising the user's credentials.
<b>Source:</b>
✓ Secure Avoid Weak Patterns: 5. The "Remember Me" functionality should never store a user's password on the device [3].
<b>Priority:</b> Not described
<b>Rationale:</b> Storing user passwords on devices, even temporarily, introduces a significant security risk, including exposure to unauthorized access or malicious exploitation. By utilizing alternative mechanisms, applications ensure enhanced security while providing a seamless user experience for returning users.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<b>1. Password Storage Prohibition:</b>
✓ Test: Inspect the codebase for instances where user passwords are stored locally, including plaintext or encrypted formats.
✓ Expected result: No instances of password storage for "Remember Me" functionality are found.
<b>2. Token Implementation Validation:</b>
✓ Test: Verify that secure tokens are used to maintain user sessions instead of passwords.
✓ Expected result: Short-lived tokens or other secure alternatives are implemented for session persistence.
<b>3. Session Expiry Testing:</b>
✓ Test: Evaluate session behavior after a prolonged period of inactivity or token expiration.
✓ Expected result: Expired sessions require user reauthentication.
<b>4. Dynamic Testing for Credential Leaks:</b>
✓ Test: Perform runtime analysis to identify potential credential leaks during session persistence.
✓ Expected result: No credentials are exposed during the operation of the "Remember Me" feature.
<b>5. Encryption and Token Security:</b>
✓ Test: Validate those tokens or identifiers are encrypted and securely stored within device-specific secure storage mechanisms (e.g., Keychain, Keystore).
✓ Expected result: Tokens are securely stored with encryption and inaccessible to unauthorized applications.

<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IDS-076]
<b>Requirement description:</b> API keys and encryption keys used within mobile health applications must not be embedded directly in the source code to prevent exposure through reverse engineering or decompilation. Instead, they must be securely stored using operating system-provided mechanisms, such as the Android Keystore or iOS Keychain.
<b>Source:</b> <ul style="list-style-type: none"><li>✓ Secure API Key Storage: API keys used within the app must not be embedded directly in the source code to prevent exposure through reverse engineering or decompilation. Use secure storage mechanisms such as Android Keystore to protect API keys within the app [18].</li><li>✓ Secure Storage of Encryption Keys: 1. Ensure encryption keys are securely stored on the mobile device. Avoid storing keys in plain text or easily accessible locations. Consider using secure storage mechanisms provided by the operating system or utilizing hardware-based secure storage options [3].</li><li>✓ Strong Key Storage: API keys must be securely stored using Android Keystore to ensure that sensitive data is protected from unauthorized access [18].</li><li>✓ Application Verification: If credentials are used exclusively by a single application, securely store them in a KeyStore instead of embedding them within the app or device [18].</li><li>✓ Minimize API Key Exposure: API keys should be fetched dynamically from a secure server or environment during runtime where applicable [18].</li><li>✓ 1.3. When storing sensitive data on the device, use a file encryption API provided by the OS or other trusted source. Some platforms (e.g., iOS and Android) provide file encryption API's which use a secret key protected by the device unlock code and deletable on remote wipe. If this is available, it should be used as it increases the security of the encryption without creating extra burden on the end-user. It also makes stored data safer in the case of loss or theft. However, it should be borne in mind that even when protected by the device unlock key, if data is stored on the device, its security is dependent on the security of the device unlock code if remote deletion of the key is for any reason not possible [16].</li></ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Directly embedding keys in application code exposes sensitive credentials to potential attackers through reverse engineering or device compromise. By using secure storage mechanisms, the risk of unauthorized access and misuse of API keys or encryption keys is minimized, aligning with best practices for secure app development in healthcare.
<b>Number of Children:</b> 0

<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Static Code Review:</b> <ul style="list-style-type: none"> <li>✓ Test: Analyze the source code to ensure no API keys or encryption keys are hardcoded.</li> <li>✓ Expected result: No keys are found embedded in the codebase.</li> </ul> </li> <li><b>2. Secure Key Storage Validation:</b> <ul style="list-style-type: none"> <li>✓ Test: Verify that keys are stored in the Android Keystore, iOS Keychain, or other equivalent secure storage mechanisms.</li> <li>✓ Expected result: All keys are securely stored using OS-provided secure storage.</li> </ul> </li> <li><b>3. Runtime Key Fetching:</b> <ul style="list-style-type: none"> <li>✓ Test: Confirm that API keys are dynamically fetched from a secure server or environment during runtime.</li> <li>✓ Expected result: API keys are not locally stored but fetched securely as needed.</li> </ul> </li> <li><b>4. Encryption of Stored Keys:</b> <ul style="list-style-type: none"> <li>✓ Test: Ensure that keys are encrypted and can only be accessed by the application under secure conditions.</li> <li>✓ Expected result: Keys are encrypted and securely tied to the app's execution environment.</li> </ul> </li> <li><b>5. Dynamic Security Testing:</b> <ul style="list-style-type: none"> <li>✓ Test: Perform penetration testing to identify potential leaks or exposure of stored keys during app operations.</li> <li>✓ Expected result: No unauthorized access or exposure of keys is detected.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IDS-077]
<b>Requirement description:</b> Cryptographic keys and user credentials must be securely managed throughout their lifecycle, including generation, storage, transmission, and rotation. Keys must be stored using secure OS-provided mechanisms such as KeyStore or Keychain.
<b>Source:</b> <ul style="list-style-type: none"> <li>✓ Long-Term Key Storage: Store keys securely using KeyStore, which provides long-term storage and retrieval for cryptographic keys [18].</li> </ul>

<ul style="list-style-type: none"> <li>✓ Properly Handle User Credentials: User credentials should always be stored, transmitted, and authenticated securely:           <ul style="list-style-type: none"> <li>○ Encrypt credentials during transmission.</li> <li>○ Do not store user credentials on the device. Instead, consider using secure, revocable access tokens.</li> <li>○ Implement strong user authentication protocols.</li> <li>○ Regularly update and rotate any used API keys or tokens [3].</li> </ul> </li> <li>✓ MASVS-CRYPTO-2: The app performs key management according to industry best practices: Even the strongest cryptography would be compromised by poor key management. This control covers the management of cryptographic keys throughout their lifecycle, including key generation, storage and protection [20].</li> <li>✓ 1.3. When storing sensitive data on the device, use a file encryption API provided by the OS or other trusted source. Some platforms (e.g., iOS and Android) provide file encryption API's which use a secret key protected by the device unlock code and deletable on remote wipe. If this is available, it should be used as it increases the security of the encryption without creating extra burden on the end-user. It also makes stored data safer in the case of loss or theft. However, it should be borne in mind that even when protected by the device unlock key, if data is stored on the device, its security is dependent on the security of the device unlock code if remote deletion of the key is for any reason not possible [16].</li> </ul>
--

**Priority:** Not described

**Rationale:** Improper key and credential management exposes applications to potential security breaches, including unauthorized access, data leaks, and identity theft. Secure storage, encryption, and lifecycle management mitigate these risks and align with healthcare data protection regulations and industry best practices.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

1. **Key Storage Audit:**
  - ✓ Test: Verify that all cryptographic keys are securely stored in the KeyStore, Keychain, or equivalent secure storage mechanism.
  - ✓ Expected result: No cryptographic keys are stored in unprotected locations.
2. **Credential Transmission Security:**
  - ✓ Test: Inspect network traffic to ensure that credentials are encrypted during transmission.
  - ✓ Expected result: All credentials are transmitted securely using industry-standard encryption (e.g., TLS 1.2 or higher).
3. **Token Implementation Review:**
  - ✓ Test: Ensure that user authentication utilizes secure, revocable access tokens instead of directly storing user credentials.

<ul style="list-style-type: none"> <li>✓ Expected result: Tokens are used exclusively for authentication, and user credentials are not stored on the device.</li> </ul>
<b>4. Key Rotation Mechanism:</b>
<ul style="list-style-type: none"> <li>✓ Test: Review the implementation for automated or manual key and token rotation mechanisms.</li> <li>✓ Expected result: Keys and tokens are regularly rotated to reduce the risk of long-term exposure.</li> </ul>
<b>5. Dynamic Testing:</b>
<ul style="list-style-type: none"> <li>✓ Test: Conduct penetration testing to identify potential vulnerabilities in key and credential handling during app operations.</li> <li>✓ Expected result: No vulnerabilities related to unauthorized access or exposure of keys and credentials are found.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IDS-078]
<b>Requirement description:</b> Applications must utilize platform-provided key-store mechanisms (e.g., Android Keystore, iOS Keychain) at the highest supported security level. Keys and sensitive data must be protected using device passcodes, configured to remain inaccessible when the device is locked.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ 3.3. Leverage the provided key-store mechanisms at the highest supported security level and only when a device passcode has been set. If possible, request key-store items to be protected after the device is locked and to remain only in the current device (e.g., exclude these items from backups and cloud synchronization) [16].</li> <li>✓ Implement Secure Storage Mechanisms: 1. Store sensitive data in secure storage locations that are inaccessible to unauthorized users. Use platform-specific secure storage mechanisms provided by the mobile operating system, such as Keychain (iOS) or Keystore (Android) [3].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Leveraging native key-store mechanisms ensures sensitive data is stored securely and is protected from unauthorized access. Limiting backup and cloud synchronization further mitigates risks of data compromise, aligning with best practices for safeguarding healthcare information.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described

<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<b>1. Key-Store Integration:</b> <ul style="list-style-type: none"><li>✓ Test: Inspect the app's implementation of secure storage to ensure that sensitive data is stored exclusively in the Keychain or Keystore.</li><li>✓ Expected result: Sensitive data is securely stored using the highest supported security level of the platform-provided key-store.</li></ul>
<b>2. Device Lock Protection:</b> <ul style="list-style-type: none"><li>✓ Test: Verify that key-store items are inaccessible when the device is locked.</li><li>✓ Expected result: Keys and sensitive data cannot be accessed while the device is locked.</li></ul>
<b>3. Backup and Synchronization Restriction:</b> <ul style="list-style-type: none"><li>✓ Test: Inspect configurations to ensure key-store items are excluded from backups and cloud synchronization.</li><li>✓ Expected result: No sensitive key-store items are included in backups or synchronized to the cloud.</li></ul>
<b>4. Dynamic Testing:</b> <ul style="list-style-type: none"><li>✓ Test: Attempt to access sensitive data through unauthorized means, such as device backups or cloud data.</li><li>✓ Expected result: Sensitive data remains protected and inaccessible to unauthorized users.</li></ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IDS-079]
<b>Requirement description:</b> Sensitive data must be classified appropriately, stored using file encryption APIs provided by the operating system, and protected by device-level security mechanisms like unlock codes.
<b>Source:</b> <ul style="list-style-type: none"><li>✓ Strong Key Storage: Encrypt stored API keys using robust encryption libraries, such as Tink Java, to safeguard them from potential threats [18].</li><li>✓ 1.1. Classify data storage according to sensitivity and apply controls accordingly (e.g. passwords, personal data, location, error logs, etc.). Process, store and use data according to its classification. Validate the security of API calls applied to sensitive data [16].</li><li>✓ 1.3. When storing sensitive data on the device, use a file encryption API provided by the OS or other trusted source. Some platforms (e.g., iOS and Android) provide file encryption API's which use a secret key protected by the device unlock code and delete-able on remote wipe. If this is available, it should be used as it increases the security of the encryption without creating extra burden on the end-user. It also makes stored data safer in the case of loss or theft. However, it should be borne in mind that even when protected by the device</li></ul>

<p>unlock key, if data is stored on the device, its security is dependent on the security of the device unlock code if remote deletion of the key is for any reason not possible [16].</p> <ul style="list-style-type: none"> <li>✓ data encryption and secure key management are especially important [16]</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Storing API keys without encryption exposes applications to potential reverse engineering and security breaches. Employing robust encryption and secure key management practices ensures compliance with data protection requirements, enhances application resilience, and mitigates risks associated with device loss or theft.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b> <ol style="list-style-type: none"> <li><b>1. API Key Encryption:</b> <ul style="list-style-type: none"> <li>✓ Test: Inspect the app's encryption implementation to verify that API keys are encrypted using Tink Java or a similarly robust encryption library.</li> <li>✓ Expected result: API keys are securely encrypted and cannot be accessed in plaintext.</li> </ul> </li> <li><b>2. File Encryption API Usage:</b> <ul style="list-style-type: none"> <li>✓ Test: Review the implementation to ensure sensitive data is stored using file encryption APIs provided by the operating system.</li> <li>✓ Expected result: Sensitive data is encrypted and tied to the device unlock code for additional security.</li> </ul> </li> <li><b>3. Remote Wipe Validation:</b> <ul style="list-style-type: none"> <li>✓ Test: Simulate remote wipe actions to verify the deletion of API keys and sensitive data.</li> <li>✓ Expected result: All encrypted data and keys are securely erased during a remote wipe.</li> </ul> </li> <li><b>4. Dynamic Testing for Unauthorized Access:</b> <ul style="list-style-type: none"> <li>✓ Test: Attempt to access API keys through reverse engineering or tampered binaries.</li> <li>✓ Expected result: API keys remain inaccessible and are protected by robust encryption mechanisms.</li> </ul> </li> <li><b>5. Key Classification Verification:</b> <ul style="list-style-type: none"> <li>✓ Test: Evaluate data storage classification policies and their application to API keys and sensitive data.</li> <li>✓ Expected result: API keys are classified as sensitive data and adhere to stringent security protocols.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IDS-080]
<b>Requirement description:</b> Applications must leverage hardware-level encryption to secure files, utilizing the highest supported security level provided by the device. Application files must be configured to remain protected after the device is locked to ensure data confidentiality.
<b>Source:</b>
✓ 1.25. Leverage the hardware-level encryption support for files at the highest supported security level. If possible request application's files to be protected after the device is locked [16].
<b>Priority:</b> Not described
<b>Rationale:</b> Hardware-level encryption provides robust security for sensitive data stored on mobile devices, minimizing risks of unauthorized access. Protecting files after the device is locked ensures that data remains secure even in the event of theft or loss.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li>1. <b>Encryption Mechanism Check:</b> <ul style="list-style-type: none"> <li>✓ Test: Verify the use of hardware-level encryption in securing application files.</li> <li>✓ Expected result: Files are encrypted using the device's hardware security features.</li> </ul> </li> <li>2. <b>File Lock Protection:</b> <ul style="list-style-type: none"> <li>✓ Test: Confirm that application files are inaccessible when the device is locked.</li> <li>✓ Expected result: Encrypted files cannot be accessed while the device is in a locked state.</li> </ul> </li> <li>3. <b>Dynamic Testing During Lock/Unlock:</b> <ul style="list-style-type: none"> <li>✓ Test: Perform runtime testing to access application files during lock and unlock states.</li> <li>✓ Expected result: Files remain encrypted and secure while the device is locked.</li> </ul> </li> <li>4. <b>Code Review for Encryption APIs:</b> <ul style="list-style-type: none"> <li>✓ Test: Conduct a code review to verify the implementation of hardware-level encryption APIs.</li> <li>✓ Expected result: APIs utilized comply with the highest security standards supported by the device.</li> </ul> </li> <li>5. <b>Compatibility Testing:</b> <ul style="list-style-type: none"> <li>✓ Test: Validate encryption functionality across supported device models and operating system versions.</li> <li>✓ Expected result: Hardware-level encryption operates consistently across all supported configurations.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda
José L Fernández-Alemán

Juan Manuel Carrillo-de-Gea  
Joaquín Nicolás  
08/01/2026

**PUID:** [SECM-CAT-IDS-081]

**Requirement description:** Applications must prioritize the use of platform-provided framework functionalities, such as Android Content Providers, for data sharing. Avoid using file system permissions or custom access schemes for managing shared data.

**Source:**

- ✓ 1.27. Prefer using framework functionality (e.g., Android Content Provider) for data sharing instead of using file system permissions or a custom access scheme on platforms that support this (e.g., Android) [16].

**Priority:** Not described

**Rationale:** Framework functionalities provide standardized, secure mechanisms for data sharing, reducing risks associated with improperly configured file system permissions or custom access implementations. Leveraging these features ensures better alignment with platform security practices and mitigates the potential for unauthorized access.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

**Parent PUIDs:** Not described

**Exclusion PUIDs:** Not described

**Importance:** Not described

**Current state:** To be determined

**Verification method:** Demonstration/Analysis

**Validation criteria:**

**1. Framework Integration Check:**

- ✓ Test: Verify that the application uses platform-provided frameworks, such as Content Providers, for data sharing.
- ✓ Expected result: All shared data is accessed and managed via the platform's framework functionalities.

**2. File System Permissions Review:**

- ✓ Test: Conduct a review of file system permissions to ensure no sensitive data is shared using file-level permissions.
- ✓ Expected result: Sensitive data is not shared via file system permissions.

**3. Dynamic Testing for Unauthorized Access:**

- ✓ Test: Attempt unauthorized access to shared data using file system manipulation or custom schemes.
- ✓ Expected result: Unauthorized access is blocked, and the framework enforces proper access control.

**4. Code Review for Data Sharing Implementation:**

- ✓ Test: Examine the implementation of data-sharing features in the codebase.
- ✓ Expected result: Data sharing is implemented using Android Content Providers or equivalent framework functionalities.

**5. Compatibility Across Platform Versions:**

<ul style="list-style-type: none"> <li>✓ Test: Validate that the framework functionality works consistently across supported versions of the operating system.</li> <li>✓ Expected result: Secure data sharing operates correctly across all target platform versions.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IDS-082]
<b>Requirement description:</b> Applications must prevent interpreted code from directly accessing user data and encrypted storage. Implement strict isolation and controlled access mechanisms to ensure that interpreted code cannot bypass security controls or interact with sensitive data storage areas.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ 10.4. Deny interpreted code direct access to user data and encrypted storage [16].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Interpreted code poses a higher security risk due to its runtime nature, which may expose vulnerabilities that can be exploited. Restricting its access to sensitive data mitigates potential risks such as unauthorized data exposure or corruption.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li>1. <b>Access Control Validation:</b> <ul style="list-style-type: none"> <li>✓ Test: Attempt to access user data and encrypted storage using interpreted code.</li> <li>✓ Expected result: Access is denied, and appropriate security controls are enforced.</li> </ul> </li> <li>2. <b>Isolation Mechanism Testing:</b> <ul style="list-style-type: none"> <li>✓ Test: Evaluate the implementation of isolation mechanisms for interpreted code.</li> <li>✓ Expected result: Interpreted code operates in a sandboxed environment without direct access to sensitive storage.</li> </ul> </li> <li>3. <b>Dynamic Code Analysis:</b> <ul style="list-style-type: none"> <li>✓ Test: Conduct runtime testing to simulate potential exploitation attempts by interpreted code.</li> </ul> </li> </ol>

<ul style="list-style-type: none"> <li>✓ Expected result: Exploitation attempts fail, and sensitive data remains protected.</li> </ul>
<b>4. Code Review for Storage Access:</b>
<ul style="list-style-type: none"> <li>✓ Test: Analyze the codebase to ensure no direct storage access is granted to interpreted code.</li> <li>✓ Expected result: Code complies with secure access principles, and interpreted code interactions are strictly controlled.</li> </ul>
<b>5. Platform-Specific Testing:</b>
<ul style="list-style-type: none"> <li>✓ Test: Verify that platform-specific security features, such as Android Storage Access Framework, are used to manage data access.</li> <li>✓ Expected result: Interpreted code leverages platform-approved methods for secure data interactions.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IDS-083]
<b>Requirement description:</b> Applications must avoid using deprecated file modes such as MODE_WORLD_WRITEABLE and MODE_WORLD_READABLE for IPC files to prevent uncontrolled access to sensitive data. Additionally, sensitive IPC must not rely on localhost network ports, which can be accessed by other apps on the device.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ Avoid Deprecated File Modes: The system must not use deprecated file modes such as MODE_WORLD_WRITEABLE and MODE_WORLD_READABLE for inter-process communication (IPC) files, as they do not offer control over data access or format [18].</li> <li>✓ Avoid Localhost for Sensitive IPC: The application must not use localhost network ports for sensitive Inter-Process Communication (IPC), as these ports are accessible to other apps on the device. Instead, secure IPC mechanisms such as Android Service with authentication must be used [18].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Using deprecated file modes or localhost for IPC can lead to unauthorized access and data leakage, compromising user security and privacy. Employing secure IPC mechanisms ensures robust access control and mitigates potential vulnerabilities.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described

<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. File Mode Validation:</b> <ul style="list-style-type: none"> <li>✓ Test: Attempt to access IPC files created with deprecated file modes (MODE_WORLD_WRITEABLE or MODE_WORLD_READABLE).</li> <li>✓ Expected result: No IPC files are accessible with these modes, ensuring secure configurations.</li> </ul> </li> <li><b>2. Secure IPC Mechanism Validation:</b> <ul style="list-style-type: none"> <li>✓ Test: Verify the use of Android Services with authentication for IPC operations.</li> <li>✓ Expected result: IPC operations utilize secure and authenticated channels.</li> </ul> </li> <li><b>3. Localhost Port Validation:</b> <ul style="list-style-type: none"> <li>✓ Test: Attempt to establish IPC communication via localhost network ports.</li> <li>✓ Expected result: Localhost ports are not used for sensitive IPC.</li> </ul> </li> <li><b>4. Code Review for IPC Security:</b> <ul style="list-style-type: none"> <li>✓ Test: Inspect the codebase for instances of deprecated file modes or localhost IPC mechanisms.</li> <li>✓ Expected result: Code adheres to secure IPC practices, avoiding deprecated or insecure configurations.</li> </ul> </li> <li><b>5. Dynamic Security Testing:</b> <ul style="list-style-type: none"> <li>✓ Test: Conduct runtime analysis to identify unauthorized IPC access attempts.</li> <li>✓ Expected result: All unauthorized attempts are blocked, and secure IPC mechanisms function correctly.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-IDS-084]
<b>Requirement description:</b> The application must implement and maintain robust data recovery practices to restore critical enterprise assets, including sensitive health data, to a pre-incident and trusted state. This includes establishing secure and reliable mechanisms for data backup, encryption, and restoration processes.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ CIS Critical Security Control 11: Data Recovery: Establish and maintain data recovery practices sufficient to restore in-scope enterprise assets to a pre-incident and trusted state [18].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Data recovery is essential to maintain operational continuity and ensure patient safety in the event of system failure, data corruption, or security incidents. Compliance with established standards for secure data recovery practices minimizes downtime and supports regulatory requirements.
<b>Number of Children:</b> 0

<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<b>1. Backup Mechanism Verification:</b> ✓ Test: Validate the implementation of automated backup systems for sensitive health data. ✓ Expected result: Backups are performed regularly and stored securely using encryption.
<b>2. Recovery Procedure Testing:</b> ✓ Test: Simulate a data recovery scenario to ensure data can be restored to its original state without loss or tampering. ✓ Expected result: Data recovery processes restore data accurately and securely.
<b>3. Incident Response Alignment:</b> ✓ Test: Verify that data recovery practices align with the organization's incident response plan. ✓ Expected result: Recovery practices integrate seamlessly with incident response procedures, ensuring minimal disruption.
<b>4. Encryption Audit:</b> ✓ Test: Assess the encryption methods used for data backup storage. ✓ Expected result: Data backups utilize strong encryption algorithms to prevent unauthorized access.
<b>5. Integrity Validation Post-Recovery:</b> ✓ Test: Compare restored data with pre-incident state to verify integrity and completeness. ✓ Expected result: Data integrity is preserved, and no discrepancies are found.
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

#### 2.9.3.10 Insufficient Cryptography

<b>PUID:</b> [SECM-CAT-ICR-001]
<b>Requirement description:</b> The mobile application must implement NIST-approved or NSA-approved key management technologies and processes for the production, control, and distribution of asymmetric cryptographic keys. The system must utilize PKI Class 3 or Class 4 certificates and, where applicable, hardware tokens to safeguard private keys. Prepositioned keying materials or approved PKI certificates must be employed to meet recognized cryptographic security standards.
<b>Source:</b>

<ul style="list-style-type: none"> <li>✓ SRG-APP-000516-MAPP-000041: Mobile apps involved in the production, control, and distribution of asymmetric cryptographic keys must use approved PKI Class 3 or class 4 certificates and hardware tokens that protect the user's private key [17].</li> <li>✓ SRG-APP-000516-MAPP-000040: Mobile apps involved in the production, control, and distribution of asymmetric cryptographic keys must use approved PKI Class 3 certificates or prepositioned keying material [17].</li> <li>✓ SRG-APP-000516-MAPP-000039: Mobile apps involved in the production, control, and distribution of asymmetric cryptographic keys must use NIST approved or NSA approved key management technology and processes [17].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Asymmetric key management is critical to ensuring the confidentiality, integrity, and authenticity of sensitive data in mobile health applications. By employing secure key management practices, the application mitigates risks of unauthorized key access, data breaches, and compromised communications.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Certificate Validation:</b> <ul style="list-style-type: none"> <li>✓ Test: Validate that the app exclusively uses PKI Class 3 or Class 4 certificates for cryptographic key management.</li> <li>✓ Expected Result: Certificates are verified to meet the required standards and are actively employed in the app's cryptographic processes.</li> </ul> </li> <li><b>2. Hardware Token Integration:</b> <ul style="list-style-type: none"> <li>✓ Test: Ensure hardware tokens are implemented to protect private keys.</li> <li>✓ Expected Result: Private keys are securely stored in hardware tokens, preventing unauthorized access.</li> </ul> </li> <li><b>3. Key Management Compliance:</b> <ul style="list-style-type: none"> <li>✓ Test: Evaluate the application's key management procedures against NIST and NSA-approved standards.</li> <li>✓ Expected Result: Key management processes adhere fully to established cryptographic standards.</li> </ul> </li> <li><b>4. Prepositioned Keying Material Usage:</b> <ul style="list-style-type: none"> <li>✓ Test: Inspect the app's configurations to confirm the secure use of prepositioned keying material.</li> <li>✓ Expected Result: Keying material is securely managed, ensuring no exposure to vulnerabilities.</li> </ul> </li> <li><b>5. Dynamic Testing for Key Protection:</b> <ul style="list-style-type: none"> <li>✓ Test: Conduct runtime tests to simulate unauthorized attempts to access private keys.</li> <li>✓ Expected Result: Unauthorized access is denied, and cryptographic integrity is maintained under all tested scenarios.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer

<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda
José L Fernández-Alemán
Juan Manuel Carrillo-de-Gea
Joaquín Nicolás
08/01/2026

<b>PUID:</b> [SECM-CAT-ICR-002]
<b>Requirement description:</b> The mobile application must utilize NIST-approved or NSA-approved key management technologies and processes for the production, control, and distribution of symmetric cryptographic keys. These technologies must ensure robust encryption standards and secure key lifecycle management, including generation, storage, rotation, and destruction..
<b>Source:</b>
✓ SRG-APP-000516-MAPP-000038: Mobile apps involved in the production, control, and distribution of symmetric cryptographic keys must use NIST approved or NSA approved key management technology and processes. [17].
<b>Priority:</b> Not described
<b>Rationale:</b> Secure symmetric key management is essential to protecting sensitive health data in mobile applications. By adhering to recognized cryptographic standards, the application can safeguard against unauthorized access, mitigate the risk of data breaches, and maintain the confidentiality and integrity of transmitted and stored information.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
1. <b>Key Management Technology Validation:</b>
✓ Test: Verify the application's use of NIST-approved or NSA-approved key management solutions.
✓ Expected Result: Only approved technologies are implemented for symmetric key production and lifecycle management.
2. <b>Secure Key Generation:</b>
✓ Test: Evaluate the methods used for generating symmetric keys.
✓ Expected Result: Keys are generated using secure algorithms compliant with NIST standards.
3. <b>Key Storage Audit:</b>
✓ Test: Inspect storage locations and mechanisms for storing symmetric keys.
✓ Expected Result: Keys are stored in secure environments (e.g., hardware security modules) inaccessible to unauthorized processes or users.
4. <b>Key Rotation Procedure:</b>

<ul style="list-style-type: none"> <li>✓ Test: Simulate a key rotation event to assess compliance with secure rotation practices.</li> <li>✓ Expected Result: Symmetric keys are rotated according to policy without exposing sensitive data.</li> </ul>
<b>5. Key Destruction Validation:</b>
<ul style="list-style-type: none"> <li>✓ Test: Test the process of securely destroying keys after their lifecycle ends.</li> <li>✓ Expected Result: Symmetric keys are irretrievably deleted, ensuring no remnants are left accessible.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-ICR-003]
<b>Requirement description:</b> The application must store passwords only in cryptographic representations, using secure hashing algorithms and salting mechanisms to protect against unauthorized access. Passwords must never be stored in plain text or reversible encrypted formats.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ Implement V-222542: The application must only store cryptographic representations of passwords [15].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Storing cryptographic representations of passwords ensures the security and confidentiality of user credentials. This approach mitigates risks of password exposure in the event of data breaches, supporting compliance with best practices for cryptographic security and protecting sensitive health information managed by mobile health applications.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Password Hashing Verification:</b> <ul style="list-style-type: none"> <li>✓ Test: Validate that passwords are hashed using secure algorithms such as bcrypt, Argon2, or PBKDF2.</li> <li>✓ Expected Result: Passwords are hashed using strong, industry-standard algorithms with appropriate parameters for computational hardness.</li> </ul> </li> <li><b>2. Salting Implementation Check:</b> <ul style="list-style-type: none"> <li>✓ Test: Confirm that a unique, randomly generated salt is added to each password before hashing.</li> </ul> </li> </ol>

<ul style="list-style-type: none"> <li>✓ Expected Result: All passwords are salted uniquely, preventing attacks such as rainbow table lookups.</li> </ul> <p><b>3. Plain Text Password Storage Audit:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Conduct a code review and inspect storage mechanisms to ensure no passwords are stored in plain text or reversible encrypted formats.</li> <li>✓ Expected Result: No instances of plain text or insecure password storage are found.</li> </ul> <p><b>4. Security Against Hash Collisions:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Verify the application uses secure hashing algorithms that mitigate collision risks.</li> <li>✓ Expected Result: Algorithms implemented ensure low collision probability, enhancing password security.</li> </ul> <p><b>5. Data Breach Response Simulation:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Simulate a data breach scenario to confirm that cryptographically represented passwords remain secure.</li> <li>✓ Expected Result: Extracted hashed passwords are computationally infeasible to reverse or compromise.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-ICR-004]
<b>Requirement description:</b> The application must implement salting for all password hashing processes, using strong and unique random salts for each password. Salts must be securely generated and stored alongside cryptographic hash values, ensuring robust protection against precomputed or rainbow table attacks.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ Implement Salting: 1. Always use a strong random salt when hashing passwords. Salting adds an extra layer of security by making it harder for attackers to use precomputed tables or rainbow tables to crack passwords [3].</li> <li>✓ V-222542: Use strong cryptographic hash functions when creating password hash values. Utilize random salt values when creating the password hash. Ensure strong access control permissions on data files containing authentication data [15].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Salting strengthens password security by adding uniqueness to each hash, even when users choose identical passwords. This practice significantly increases the computational effort required for an attacker to crack password hashes, ensuring compliance with secure password management standards and safeguarding sensitive health application data.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described

<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<p><b>1. Salt Generation Verification:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Inspect the application's password hashing logic to confirm that random salts are generated for each password.</li> <li>✓ Expected Result: Every password hash includes a unique, randomly generated salt.</li> </ul> <p><b>2. Secure Hashing Algorithm Validation:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Ensure the use of secure cryptographic hash functions such as bcrypt, Argon2, or PBKDF2.</li> <li>✓ Expected Result: Passwords are hashed using secure, industry-recommended algorithms.</li> </ul> <p><b>3. Salt Storage Review:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Examine how salts are stored and associated with their respective password hashes.</li> <li>✓ Expected Result: Salts are securely stored and readily available for authentication processes without compromising security.</li> </ul> <p><b>4. Access Control Audit:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Verify that access controls are applied to files or databases containing salted password hashes.</li> <li>✓ Expected Result: Authentication data is accessible only to authorized processes and users.</li> </ul> <p><b>5. Password Collision Testing:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Test scenarios where multiple users create identical passwords.</li> <li>✓ Expected Result: Hashes for identical passwords are unique due to distinct salts, preventing collisions.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-ICR-005]
<b>Requirement description:</b> The application must employ Key Derivation Functions (KDFs), such as PBKDF2, bcrypt, or scrypt, for password hashing. These functions must include mechanisms like iteration counts and adaptive work factors to enhance security against brute-force attacks and ensure secure derivation of cryptographic keys from user passwords.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ Use Key Derivation Functions (KDFs): 1. For password hashing, use Key Derivation Functions like PBKDF2, bcrypt, or scrypt. These functions are specifically designed for</li> </ul>

securely deriving cryptographic keys from passwords and provide additional security features like iteration counts to slow down brute-force attacks [3].
<b>Priority:</b> Not described
<b>Rationale:</b> Key Derivation Functions are purpose-built to securely transform user passwords into cryptographic keys. By integrating iteration counts and adaptive mechanisms, KDFs increase computational cost for attackers, reducing the feasibility of brute-force attacks and ensuring compliance with secure password management standards.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<p><b>1. Algorithm Verification:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Inspect the application code to confirm the use of PBKDF2, bcrypt, or scrypt for password hashing.</li> <li>✓ Expected Result: Password hashing processes are implemented using one of the specified KDFs.</li> </ul> <p><b>2. Iteration Count Testing:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Verify that the KDF includes a high iteration count to slow down hashing operations for attackers.</li> <li>✓ Expected Result: Iteration counts meet industry-recommended thresholds (e.g., 100,000+ iterations for PBKDF2).</li> </ul> <p><b>3. Adaptive Mechanism Review:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Evaluate whether the KDF supports adaptive mechanisms to allow adjustment of work factors over time.</li> <li>✓ Expected Result: The system can increase computational work as processing power advances.</li> </ul> <p><b>4. Secure Key Output Validation:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Ensure the output of the KDF is cryptographically strong and suitable for secure storage or encryption.</li> <li>✓ Expected Result: Derived keys are robust and meet security requirements.</li> </ul> <p><b>5. Brute-Force Resistance Testing:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Simulate a brute-force attack against hashed passwords to assess resistance.</li> <li>✓ Expected Result: Password hashing resists brute-force attacks within acceptable security thresholds.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda
José L Fernández-Alemán
Juan Manuel Carrillo-de-Gea
Joaquín Nicolás
08/01/2026

<b>PUID:</b> [SECM-CAT-ICR-006]
<b>Requirement description:</b> The application must implement NSA-approved cryptographic mechanisms to protect classified information and comply with applicable federal laws, directives, and industry standards. It must utilize FIPS-validated cryptographic modules for encryption, authentication, and signing application components.
<b>Source:</b>
<ul style="list-style-type: none"><li>✓ V-254803: The application must implement NSA-approved cryptography to protect classified information in accordance with applicable federal laws, Executive Orders, directives, policies, regulations, and standards [15].</li><li>✓ V-222555: The application must use mechanisms meeting the requirements of applicable federal laws, Executive Orders, directives, policies, regulations, standards, and guidance for authentication to a cryptographic module. Use FIPS-approved cryptographic modules [15].</li><li>✓ V-222570: The application must utilize FIPS-validated cryptographic modules when signing application components. Utilize FIPS-validated algorithms when signing application components [15].</li><li>✓ MASVS-CRYPTO-1: The app employs current strong cryptography and uses it according to industry best practices: Cryptography plays an especially important role in securing the user's data - even more so in a mobile environment, where attackers having physical access to the user's device is a likely scenario. This control covers general cryptography best practices, which are typically defined in external standards [20].</li><li>✓ 8.24 Use of cryptography: Rules for the effective use of cryptography, including cryptographic key management, should be defined and implemented [6].</li><li>✓ SRG-APP-000416-MAPP-000100: The mobile app must implement NSA-approved cryptography to protect classified information in accordance with applicable federal laws, Executive Orders, directives, policies, regulations, and standards [17].</li><li>✓ Follow Industry Standards and Best Practices: 1. Stay updated with industry standards and best practices related to cryptography. Organizations like NIST (National Institute of Standards and Technology) and IETF (Internet Engineering Task Force) provide guidelines and recommendations for secure cryptographic practices [3].</li><li>✓ 3.2: Verify that the app uses proven implementations of cryptographic primitives [19].</li><li>✓ 3.3: Verify that the app uses cryptographic primitives that are appropriate for the particular use-case, configured with parameters that adhere to industry best practices [19].</li><li>✓ Use Strong Encryption: 1. Implement robust encryption algorithms and practices to protect sensitive data both at rest and in transit. Utilize industry-standard encryption algorithms and ensure that encryption keys are securely stored and managed [3].</li></ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Strong cryptographic implementations safeguard sensitive and classified data, ensuring compliance with federal regulations and minimizing risks associated with unauthorized access or data breaches. Using FIPS-validated modules and industry-recommended practices strengthens the security posture of mobile health applications, especially in scenarios involving classified or sensitive patient data.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0

<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<p><b>1. Compliance Validation:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Verify that all cryptographic modules used in the application are FIPS-validated and comply with NSA-approved cryptographic standards.</li> <li>✓ Expected Result: Only FIPS-validated modules and NSA-approved algorithms are employed.</li> </ul> <p><b>2. Cryptographic Algorithm Audit:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Inspect the codebase for the use of proven cryptographic primitives appropriate for specific use cases, such as AES for encryption and RSA for key exchange.</li> <li>✓ Expected Result: All cryptographic algorithms conform to current NIST and NSA recommendations.</li> </ul> <p><b>3. Encryption Key Management:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Evaluate encryption key storage and management practices, ensuring secure key generation, storage, and lifecycle management.</li> <li>✓ Expected Result: Encryption keys are securely generated, stored in hardware-backed solutions like KeyStore, and managed according to best practices.</li> </ul> <p><b>4. Digital Signature Validation:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Ensure that digital signatures for application components use FIPS-validated cryptographic algorithms.</li> <li>✓ Expected Result: Signing mechanisms utilize robust, industry-approved algorithms such as RSA or ECDSA.</li> </ul> <p><b>5. Secure Data Protection:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Perform dynamic testing to validate that sensitive data is encrypted during storage and transit.</li> <li>✓ Expected Result: Sensitive data remains encrypted using strong encryption mechanisms and is inaccessible without decryption.</li> </ul> <p><b>6. Industry Standards Review:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Confirm that the application adheres to NIST and IETF standards for cryptography and key management.</li> <li>✓ Expected Result: All cryptographic implementations align with the latest industry standards and guidelines.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-ICR-007]
<b>Requirement description:</b> The application must use FIPS-validated cryptographic modules for generating cryptographic hashes. All hash generation must be performed using FIPS-validated algorithms, ensuring compliance with federal standards for cryptographic security.
<b>Source:</b> <ul style="list-style-type: none"><li>✓ V-222571: The application must utilize FIPS-validated cryptographic modules when generating cryptographic hashes. Configure the application to use a FIPS-validated hashing algorithm when creating a cryptographic hash [15].</li></ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Using FIPS-validated hashing algorithms ensures that the cryptographic hash functions meet rigorous security standards, protecting sensitive data from tampering and ensuring the integrity of the application. This is critical for applications handling sensitive health information to comply with regulatory requirements and safeguard data integrity.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b> <ol style="list-style-type: none"><li><b>1. Hash Algorithm Validation:</b><ul style="list-style-type: none"><li>✓ Test: Verify that the application uses only FIPS-validated cryptographic modules for hashing operations.</li><li>✓ Expected Result: All hashing operations are performed using FIPS-compliant algorithms such as SHA-256 or SHA-3.</li></ul></li><li><b>2. Hash Integrity Test:</b><ul style="list-style-type: none"><li>✓ Test: Inspect the generated hash values for consistency and integrity during operations such as file verification or data storage.</li><li>✓ Expected Result: Hash values are reproducible and unique, ensuring no data collisions.</li></ul></li><li><b>3. Implementation Audit:</b><ul style="list-style-type: none"><li>✓ Test: Review the codebase to confirm the integration of FIPS-approved cryptographic libraries for hashing.</li><li>✓ Expected Result: Only FIPS-validated modules, such as OpenSSL configured for FIPS, are utilized.</li></ul></li><li><b>4. Performance Benchmark:</b><ul style="list-style-type: none"><li>✓ Test: Measure the performance of hashing operations to ensure they meet operational requirements while maintaining security.</li><li>✓ Expected Result: Hashing operations are efficient and secure without introducing significant delays.</li></ul></li><li><b>5. Compliance Review:</b><ul style="list-style-type: none"><li>✓ Test: Assess the application's adherence to FIPS guidelines and federal cryptographic standards.</li><li>✓ Expected Result: Hashing mechanisms are fully compliant with FIPS requirements and other relevant standards.</li></ul></li></ol>
<b>Requested by:</b> The organization

<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-ICR-008]
<b>Requirement description:</b> The application must implement NIST FIPS-validated cryptographic mechanisms for generating digital signatures, cryptographic hashes, and ensuring confidentiality of sensitive data in compliance with federal regulations.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ V-222572: The application must utilize FIPS-validated cryptographic modules when protecting unclassified information that requires cryptographic protection. Configure the application to use a FIPS-validated cryptographic module [15].</li> <li>✓ SRG-APP-000514-MAPP-000100: If the underlying MOS does not provide NIST FIPS-validated crypto modules, the mobile app must implement NIST FIPS-validated cryptography for the following: to provision digital signatures; to generate cryptographic hashes; and to protect unclassified information requiring confidentiality and cryptographic protection in accordance with applicable federal laws, Executive Orders, directives, policies, regulations, and standards [17].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Utilizing FIPS-validated cryptographic modules ensures adherence to federal security standards, enhancing the confidentiality, integrity, and authenticity of unclassified information. This is especially critical in mobile health applications to safeguard sensitive data against unauthorized access and tampering.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li>1. <b>Cryptographic Module Validation:</b> <ul style="list-style-type: none"> <li>✓ Test: Verify the application integrates FIPS-validated cryptographic modules for all cryptographic operations.</li> <li>✓ Expected Result: All cryptographic operations, including encryption, hashing, and signing, utilize FIPS-validated modules.</li> </ul> </li> <li>2. <b>Encryption Verification:</b></li> </ol>

<ul style="list-style-type: none"> <li>✓ Test: Validate that data requiring confidentiality is encrypted using FIPS-approved algorithms, such as AES-256.</li> <li>✓ Expected Result: Data is securely encrypted and compliant with FIPS standards.</li> </ul>
<b>3. Hash and Signature Integrity Testing:</b>
<ul style="list-style-type: none"> <li>✓ Test: Confirm that cryptographic hashes and digital signatures are generated using FIPS-compliant algorithms, such as SHA-256 and RSA.</li> <li>✓ Expected Result: Generated hashes and signatures meet FIPS validation requirements and ensure data integrity.</li> </ul>
<b>4. Code Inspection:</b>
<ul style="list-style-type: none"> <li>✓ Test: Conduct a review of the codebase to ensure integration of FIPS-approved cryptographic libraries (e.g., OpenSSL in FIPS mode).</li> <li>✓ Expected Result: Only FIPS-validated cryptographic functions are present in the codebase.</li> </ul>
<b>5. Compliance Audit:</b>
<ul style="list-style-type: none"> <li>✓ Test: Assess the application's cryptographic implementations against federal laws, directives, and FIPS standards.</li> <li>✓ Expected Result: The application complies with all relevant cryptographic protection requirements for unclassified information.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-ICR-009]
<b>Requirement description:</b> The application must establish and manage cryptographic keys according to defined key management requirements, ensuring secure generation, distribution, storage, access, and destruction.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ SC-12 CRYPTOGRAPHIC KEY ESTABLISHMENT AND MANAGEMENT</li> </ul> <p>Control: Establish and manage cryptographic keys when cryptography is employed within the system in accordance with the following key management requirements: [Assignment: organization-defined requirements for key generation, distribution, storage, access, and destruction].</p> <p>Discussion: Cryptographic key management and establishment can be performed using manual procedures or automated mechanisms with supporting manual procedures. Organizations define key management requirements in accordance with applicable laws, executive orders, directives, regulations, policies, standards, and guidelines and specify appropriate options, parameters, and levels. Organizations manage trust stores to ensure that only approved trust anchors are part of such trust stores. This includes certificates with visibility external to organizational systems and certificates related to the internal operations of systems. [NIST CMVP] and [NIST CAVP] provide additional information on validated cryptographic modules and algorithms that can be used in cryptographic key management and establishment [11].</p>

<b>Priority:</b> Not described
<b>Rationale:</b> Effective cryptographic key management ensures that sensitive data is protected from unauthorized access, tampering, and misuse. By adhering to robust key management practices, the application can maintain the confidentiality, integrity, and authenticity of cryptographic operations critical to mobile health application security.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<b>1. Key Generation Audit:</b> <ul style="list-style-type: none"><li>✓ Test: Verify that cryptographic keys are generated using approved algorithms and adhere to defined organizational requirements.</li><li>✓ Expected Result: All keys are securely generated and comply with NIST and organizational standards.</li></ul>
<b>2. Key Distribution Validation:</b> <ul style="list-style-type: none"><li>✓ Test: Assess mechanisms used to distribute cryptographic keys within the system.</li><li>✓ Expected Result: Key distribution employs secure methods, preventing unauthorized access during transmission.</li></ul>
<b>3. Key Storage Examination:</b> <ul style="list-style-type: none"><li>✓ Test: Validate that keys are stored in a secure, tamper-resistant location, such as a hardware security module (HSM) or a secure enclave.</li><li>✓ Expected Result: Keys are securely stored and inaccessible to unauthorized entities.</li></ul>
<b>4. Access Control Testing:</b> <ul style="list-style-type: none"><li>✓ Test: Verify access restrictions to cryptographic keys, ensuring only authorized processes or users can retrieve them.</li><li>✓ Expected Result: Access to cryptographic keys is strictly controlled and logged.</li></ul>
<b>5. Key Destruction Validation:</b> <ul style="list-style-type: none"><li>✓ Test: Confirm that expired or unused keys are securely destroyed using organization-defined procedures.</li><li>✓ Expected Result: Cryptographic keys are irrecoverable after destruction, ensuring no residual risk.</li></ul>
<b>6. Compliance Audit:</b> <ul style="list-style-type: none"><li>✓ Test: Assess the key management implementation against applicable regulations, including NIST guidelines.</li><li>✓ Expected Result: Key management practices are fully compliant with federal and organizational standards.</li></ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-ICR-010]
<b>Requirement description:</b> The application must encrypt all messages when the SessionIndex is associated with privacy data to safeguard sensitive information during transmission and mitigate risks of unauthorized access or interception.
<b>Source:</b>
✓ V-222406: The application must ensure messages are encrypted when the SessionIndex is tied to privacy data. Encrypt messages when the SessionIndex is tied to privacy data [15].
<b>Priority:</b> Not described
<b>Rationale:</b> Encrypting messages tied to privacy data ensures that sensitive information remains secure and complies with privacy regulations. This practice protects against data breaches and unauthorized disclosures, fostering trust in mobile health applications.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<b>1. Encryption Verification:</b>
✓ Test: Verify that messages containing privacy data tied to the SessionIndex are encrypted using strong encryption algorithms.
✓ Expected Result: All messages associated with privacy data are encrypted before transmission.
<b>2. SessionIndex Validation:</b>
✓ Test: Inspect the implementation to confirm that the SessionIndex is accurately identified when tied to privacy data.
✓ Expected Result: The system correctly identifies and flags SessionIndex instances that require encryption.
<b>3. Transmission Monitoring:</b>
✓ Test: Simulate data transmission and inspect network traffic for unencrypted messages linked to the SessionIndex.
✓ Expected Result: No unencrypted messages containing privacy data are observed in the network traffic.
<b>4. Decryption Testing:</b>
✓ Test: Attempt to decrypt the encrypted messages using the authorized decryption process.
✓ Expected Result: Decryption succeeds only with the correct cryptographic keys and authorized processes.
<b>5. Compliance Audit:</b>
✓ Test: Assess the encryption implementation against regulatory standards for protecting privacy data during transmission.
✓ Expected Result: The encryption approach complies with applicable privacy regulations and standards.
<b>Requested by:</b> The organization

<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-ICR-011]
<b>Requirement description:</b> The application must implement robust validation and authentication mechanisms to verify the integrity and authenticity of all parties involved in the encryption process.
<b>Source:</b>
✓ Validate and Authenticate: 1. Implement strong validation and authentication mechanisms to verify the integrity and authenticity of parties involved in the encryption process. Perform proper validation of certificates, digital signatures, or other mechanisms used for authentication [3].
<b>Priority:</b> Not described
<b>Rationale:</b> Ensuring the authenticity and integrity of parties in the encryption process is critical to prevent man-in-the-middle attacks, unauthorized access, or data tampering. Proper validation and authentication mechanisms enhance trust and compliance with security standards in mobile health applications.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Certificate Validation:</b> <ul style="list-style-type: none"> <li>✓ Test: Verify the application's ability to validate certificates against trusted certificate authorities.</li> <li>✓ Expected Result: Certificates are validated and rejected if not issued by a trusted authority or if they are expired or revoked.</li> </ul> </li> <li><b>2. Digital Signature Verification:</b> <ul style="list-style-type: none"> <li>✓ Test: Test the application's ability to verify the integrity of messages using digital signatures.</li> <li>✓ Expected Result: Only messages with valid digital signatures are processed.</li> </ul> </li> </ol>

<p><b>3. Authentication Challenge Testing:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Simulate an unauthorized party attempting to authenticate in the encryption process.</li> <li>✓ Expected Result: Unauthorized attempts are rejected, and the system logs the event.</li> </ul> <p><b>4. Mechanism Robustness Assessment:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Inspect all validation and authentication mechanisms for adherence to industry best practices and standards, such as FIPS or NIST guidelines.</li> <li>✓ Expected Result: Mechanisms comply with recognized security standards and perform reliably under all conditions.</li> </ul> <p><b>5. Dynamic Testing for Validation Flow:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Perform runtime testing to ensure validation and authentication processes execute securely during live operations.</li> <li>✓ Expected Result: The system securely validates and authenticates all encryption-related operations without performance degradation or failure.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<p><b>Author and date:</b></p> <p>Carlos M. Mejía-Granda  José L Fernández-Alemán  Juan Manuel Carrillo-de-Gea  Joaquín Nicolás  08/01/2026</p>

<b>PUID:</b> [SECM-CAT-ICR-012]
<b>Requirement description:</b> The application must verify during the first run that no other application has previously requested the same custom permissions, particularly on platforms older than Android 5.0. This ensures that the permissions are securely associated with the intended application and are not subject to unauthorized access or conflict.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ 12.5. In the case that the application requests custom permissions, and older platforms are supported (e.g., earlier than Android 5.0), always verify on the first run of the app that no other application has previously requested the same permissions [16].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Older platforms may lack the robust permission-handling mechanisms available in newer systems, leading to potential security vulnerabilities. By validating custom permissions during the initial run, the application mitigates risks associated with permission hijacking or unauthorized use.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined

<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<b>1. Initial Permission Check:</b> ✓ Test: Execute the application on a platform older than Android 5.0 and inspect its ability to validate the ownership of custom permissions during the first run. ✓ Expected Result: The application identifies and resolves any conflicts if another app has requested the same permissions.
<b>2. Conflict Resolution Assessment:</b> ✓ Test: Simulate a scenario where another app has claimed the same custom permission before the first run. ✓ Expected Result: The application detects the conflict and either reclaims or denies usage of the permission securely.
<b>3. Cross-Platform Testing:</b> ✓ Test: Validate the custom permission-checking mechanism across supported Android platform versions, particularly focusing on versions prior to 5.0. ✓ Expected Result: Permission validation operates reliably and securely across all supported versions.
<b>4. Permission Association Integrity Check:</b> ✓ Test: Review the application's permission association process for adherence to security best practices. ✓ Expected Result: Custom permissions are securely linked to the application and cannot be exploited by other apps.
<b>5. Dynamic Behavior Testing:</b> ✓ Test: Test runtime behavior when custom permissions are accessed during the app lifecycle. ✓ Expected Result: Permissions are handled securely, and unauthorized access attempts are logged or blocked
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-ICR-013]
<b>Requirement description:</b> The application must adhere to the domain name registration infrastructure when declaring custom permissions to prevent collisions or conflicts with permissions declared by other applications.
<b>Source:</b> ✓ 12.6. Always follow the domain name registration infrastructure to declare a custom permission, in order to avoid any collisions with other apps [16].
<b>Priority:</b> Not described
<b>Rationale:</b> Using the domain name registration infrastructure ensures that custom permissions are uniquely identifiable and associated with the correct application, mitigating risks of unauthorized access, permission hijacking, and compatibility issues across the application ecosystem.

<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<p><b>1. Domain-Based Naming Validation:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Review the naming convention of declared custom permissions in the application.</li> <li>✓ Expected Result: All custom permissions are prefixed with the app's domain name, following the domain name registration infrastructure.</li> </ul> <p><b>2. Collision Detection Simulation:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Simulate the registration of permissions with identical names by multiple apps.</li> <li>✓ Expected Result: The application's custom permissions remain unique due to adherence to the domain-based naming convention.</li> </ul> <p><b>3. Cross-App Compatibility Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Test the application in an environment with multiple apps using custom permissions.</li> <li>✓ Expected Result: No collisions or conflicts occur between the custom permissions of the application and those of other apps.</li> </ul> <p><b>4. Code Review for Custom Permissions:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Perform a static code analysis to ensure the correct usage of domain-prefixed custom permissions throughout the application.</li> <li>✓ Expected Result: Custom permissions are consistently declared and used without deviations from the domain-based naming standard.</li> </ul> <p><b>5. Dynamic Environment Testing:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Validate custom permission behavior during app interactions in a multi-application ecosystem.</li> <li>✓ Expected Result: The application's permissions function securely and without interference from other apps.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-ICR-014]
<b>Requirement description:</b> The application must restrict access to its components, such as Activities, Services, and Content Providers, ensuring that only authorized applications can start or interact with them. This must be implemented using strict permissions and access controls.
<b>Source:</b>

<p>✓ 12.7. Restrict what apps can cause an application component (e.g., Android Activity) to start or are able to interact with it (e.g., Android Service and Content Provider). This can be accomplished using strict permissions [16].</p>
<b>Priority:</b> Not described
<b>Rationale:</b> Restricting component access prevents unauthorized or malicious applications from interacting with sensitive parts of the application, reducing the risk of privilege escalation, data leakage, or other security vulnerabilities.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<p><b>1. Component Access Control Test:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Attempt to access application components (e.g., Activities, Services, Content Providers) using unauthorized apps.</li> <li>✓ Expected Result: Unauthorized apps are unable to access or interact with any protected components.</li> </ul> <p><b>2. Permission Definition Verification:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Review the application manifest file to verify the use of strict permissions for component access.</li> <li>✓ Expected Result: All components have permissions defined to restrict unauthorized access.</li> </ul> <p><b>3. Dynamic Interaction Testing:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Use an external app to send requests to the application's components during runtime.</li> <li>✓ Expected Result: Requests from unauthorized apps are blocked or denied.</li> </ul> <p><b>4. Code Review for Access Restrictions:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Conduct a static code analysis to ensure strict permissions are implemented for all accessible components.</li> <li>✓ Expected Result: All publicly accessible components have well-defined, enforced permissions.</li> </ul> <p><b>5. Security Policy Validation:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Inspect the application's security policies to confirm alignment with secure component interaction standards.</li> <li>✓ Expected Result: The policies explicitly address restricted access to components and enforce proper permissions.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda
José L Fernández-Alemán
Juan Manuel Carrillo-de-Gea
Joaquín Nicolás

08/01/2026

<b>PUID:</b> [SECM-CAT-ICR-015]
<b>Requirement description:</b> The application must disable device identifiers after 35 days of inactivity unless cryptographic certificates, such as PKI credentials, are used for authentication.
<b>Source:</b>
<ul style="list-style-type: none"><li>✓ V-222535: The application must disable device identifiers after 35 days of inactivity unless a cryptographic certificate is used for authentication. Configure the application to disable device accounts after 35 days of inactivity or to utilize DoD PKI certificates that provide an expiration date [15].</li><li>✓ SRG-APP-000391-MAPP-000100: The mobile app must accept Public Key Infrastructure (PKI) credentials [17].</li></ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Limiting the duration of inactive device identifiers mitigates risks of unauthorized access due to outdated or unused accounts. Utilizing PKI credentials ensures a secure and verifiable authentication process, aligned with industry standards for cryptographic security.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<b>1. Inactivity Enforcement Test:</b> <ul style="list-style-type: none"><li>✓ Test: Simulate 35 days of account inactivity and attempt to authenticate using the device identifier.</li><li>✓ Expected Result: The application denies access for inactive device identifiers.</li></ul>
<b>2. PKI Credential Usage Verification:</b> <ul style="list-style-type: none"><li>✓ Test: Authenticate using PKI credentials with an expiration date after a 35-day inactivity period.</li><li>✓ Expected Result: Authentication succeeds only with valid PKI credentials.</li></ul>
<b>3. Configuration Review:</b> <ul style="list-style-type: none"><li>✓ Test: Inspect application settings to verify implementation of inactivity-based identifier disabling and PKI certificate acceptance.</li><li>✓ Expected Result: Settings reflect compliance with the 35-day inactivity rule and enable PKI credential validation.</li></ul>
<b>4. Certificate Expiration Validation:</b> <ul style="list-style-type: none"><li>✓ Test: Attempt authentication using an expired PKI certificate.</li><li>✓ Expected Result: The application rejects expired certificates.</li></ul>
<b>5. Policy Documentation Audit:</b> <ul style="list-style-type: none"><li>✓ Test: Review security policies to confirm the inclusion of a 35-day inactivity rule and PKI credential handling.</li><li>✓ Expected Result: Policies explicitly define and enforce both requirements.</li></ul>
<b>Requested by:</b> The organization

<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-ICR-016]
<b>Requirement description:</b> The application must use standard cryptographic algorithms provided by the platform, such as AES, RSA, or ECC, through validated cryptographic libraries or frameworks. Avoid the implementation of custom cryptographic algorithms or the use of deprecated cryptographic protocols and ensure compliance with industry-recommended practices for encryption and decryption processes.
<b>Source:</b> <ul style="list-style-type: none"><li>✓ Use Standard Cryptography Algorithms: Use existing cryptographic algorithms provided by Android, such as AES and RSA, through the Cipher class. Avoid implementing custom cryptographic algorithms [18].</li><li>✓ Use Strong Encryption Algorithms: 1. Implement widely accepted and secure encryption algorithms, such as AES (Advanced Encryption Standard), RSA (Rivest-Shamir-Adleman), or Elliptic Curve Cryptography (ECC). Stay updated with current cryptographic standards and avoid deprecated or weak algorithms [3].</li><li>✓ 3.4: Verify that the app does not use cryptographic protocols or algorithms that are widely considered depreciated for security purposes [19].</li><li>✓ 4.5. Use strong and standardized encryption algorithms (e.g., AES) and appropriate key lengths (check recommendations for the algorithm you use e.g. for the TLS configuration). Remove support for weak ciphers [16]</li><li>✓ 4.11. Always use platform supported or vetted frameworks for establishing secure communication channels. Avoid using custom solutions [16].</li><li>✓ Implement Encryption Correctly: 1. Carefully implement encryption and decryption processes in the mobile application, adhering to established cryptographic libraries and frameworks. Avoid custom encryption implementations, as they are more prone to errors and vulnerabilities [3].</li></ul>
<b>Priority:</b> Not described

<b>Rationale:</b> Adopting standardized cryptographic algorithms reduces vulnerabilities associated with custom implementations and ensures robustness against evolving threats. Utilizing platform-supported cryptography improves security, compatibility, and maintainability, while avoiding deprecated algorithms minimizes exposure to known attacks.
---

<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described

<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<p><b>1. Algorithm Verification:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Inspect the codebase to verify the use of platform-supported algorithms (e.g., AES, RSA).</li> <li>✓ Expected Result: Only validated algorithms from the platform-provided cryptographic libraries are used.</li> </ul> <p><b>2. Custom Algorithm Prohibition:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Conduct a code review to ensure no custom cryptographic implementations exist.</li> <li>✓ Expected Result: No instances of custom cryptographic algorithms are found.</li> </ul> <p><b>3. Protocol Audit:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Evaluate the application's cryptographic protocols to ensure deprecated or insecure protocols are not utilized.</li> <li>✓ Expected Result: Only current and secure cryptographic protocols are implemented.</li> </ul> <p><b>4. Encryption Configuration Review:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Validate encryption key lengths and configurations against recommended standards (e.g., AES with 256-bit keys).</li> <li>✓ Expected Result: Encryption keys and configurations comply with industry best practices.</li> </ul> <p><b>5. Dynamic Testing of Encryption Processes:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Simulate encryption and decryption operations during runtime to verify correct implementation and secure behavior.</li> <li>✓ Expected Result: Encryption and decryption processes execute without leaks or errors, ensuring data integrity and confidentiality.</li> </ul> <p><b>6. Compliance Validation:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Review security policies and implementation documentation to ensure alignment with cryptographic standards such as NIST or ISO.</li> <li>✓ Expected Result: Policies and implementation meet established cryptographic security requirements.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-ICR-017]
<b>Requirement description:</b> The application must implement Advanced Encryption Standard (AES) for encrypting sensitive data, such as Protected Health Information (PHI). AES encryption must utilize 256-bit keys where supported. If 256-bit AES is unavailable, fallback to 128-bit AES encryption, ensuring compliance with secure cryptographic practices.
<b>Source:</b>

<ul style="list-style-type: none"> <li>✓ AES Encryption: For commercial encryption, use 256-bit AES encryption where available. If 256-bit AES is unavailable, use 128-bit AES as a fallback [18].</li> <li>✓ Use Advanced Encryption Standard (AES) to encrypt PHI. The cryptographic key used must have at least 128 bits. This method offers better encryption times than other techniques [23].</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> AES is a widely recognized and efficient encryption standard for securing sensitive data. Utilizing 256-bit keys provides robust security, and falling back to 128-bit keys ensures encryption availability without compromising data protection. Adopting this standard safeguards PHI against unauthorized access and aligns with industry best practices.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Key Length Verification:</b> <ul style="list-style-type: none"> <li>✓ Test: Review the application's encryption configurations to confirm the use of 256-bit AES encryption.</li> <li>✓ Expected Result: The application uses 256-bit AES encryption where available and falls back to 128-bit AES if necessary.</li> </ul> </li> <li><b>2. PHI Encryption Testing:</b> <ul style="list-style-type: none"> <li>✓ Test: Encrypt sample PHI data and verify that AES encryption is applied.</li> <li>✓ Expected Result: PHI is encrypted with the specified AES key length and securely stored.</li> </ul> </li> <li><b>3. Performance Audit:</b> <ul style="list-style-type: none"> <li>✓ Test: Evaluate encryption and decryption times for both 256-bit and 128-bit AES.</li> <li>✓ Expected Result: Encryption performance aligns with acceptable benchmarks without compromising security.</li> </ul> </li> <li><b>4. Configuration Inspection:</b> <ul style="list-style-type: none"> <li>✓ Test: Inspect configuration files and code for encryption settings.</li> <li>✓ Expected Result: Only AES encryption is implemented with specified key lengths.</li> </ul> </li> <li><b>5. Dynamic Security Testing:</b> <ul style="list-style-type: none"> <li>✓ Test: Conduct penetration testing to ensure encrypted data is resistant to attacks such as brute force.</li> <li>✓ Expected Result: Encrypted data remains secure under all tested attack scenarios.</li> </ul> </li> <li><b>6. Compliance Review:</b> <ul style="list-style-type: none"> <li>✓ Test: Validate encryption methods against applicable standards (e.g., NIST).</li> <li>✓ Expected Result: AES encryption methods and key lengths comply with established standards and regulations.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán

Juan Manuel Carrillo-de-Gea  
Joaquín Nicolás  
08/01/2026

<b>PUID:</b> [SECM-CAT-ICR-018]
<b>Requirement description:</b> The application must implement Advanced Encryption Standard (AES) for all sensitive data encryption, prioritizing 256-bit keys for robust security. If 256-bit AES is unavailable, the system must fallback to 128-bit AES encryption to maintain secure data protection.
<b>Source:</b>
✓ AES Encryption: For commercial encryption, use 256-bit AES encryption where available. If 256-bit AES is unavailable, use 128-bit AES as a fallback [18].
<b>Priority:</b> Not described
<b>Rationale:</b> AES encryption is a globally recognized standard for securing sensitive data, providing strong resistance against unauthorized access. The use of 256-bit keys ensures a higher level of security, while a fallback to 128-bit keys maintains encryption availability in environments with limited support, without compromising the baseline security.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
1. <b>Key Length Enforcement:</b>
✓ Test: Inspect the encryption configuration in the application to verify the default use of 256-bit AES encryption.
✓ Expected Result: The application defaults to 256-bit AES encryption and switches to 128-bit AES only when 256-bit is unsupported.
2. <b>Encryption Verification:</b>
✓ Test: Encrypt a dataset and confirm that the AES algorithm with the specified key length is applied.
✓ Expected Result: Data is encrypted using AES with the appropriate key length (256-bit or 128-bit fallback).
3. <b>Data Decryption Validation:</b>
✓ Test: Decrypt the encrypted dataset to ensure data integrity and usability.
✓ Expected Result: The decryption process accurately restores the data, confirming the encryption was performed correctly.
4. <b>Configuration Review:</b>
✓ Test: Conduct a review of encryption libraries and settings used in the application.
✓ Expected Result: Only AES encryption with supported key lengths is configured in the system.
5. <b>Dynamic Security Testing:</b>
✓ Test: Perform penetration testing to ensure encrypted data withstands cryptographic attacks.

✓ Expected Result: Encrypted data is resilient to attacks such as brute force and known vulnerabilities.
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-ICR-019]
<b>Requirement description:</b> The application must use the appropriate block cipher mode for encryption based on the context of use. For general encryption, Cipher Block Chaining (CBC) or Galois/Counter Mode (GCM) must be implemented. For streaming data, Counter (CTR) mode is recommended, ensuring no reuse of initialization vectors (IVs) or counters to maintain security.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ Block Mode Selection: Use the appropriate block mode for encryption:           <ul style="list-style-type: none"> <li>○ CBC or GCM for general encryption.</li> <li>○ CTR for streaming encryption, while avoiding IV/counter reuse. [18].</li> </ul> </li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> The selection of an appropriate block cipher mode is critical to ensure the security and integrity of encrypted data. CBC and GCM provide robust options for general encryption, with GCM offering additional authenticated encryption capabilities. CTR is suitable for streaming data, but improper IV or counter reuse can compromise encryption. Adhering to these standards ensures strong cryptographic practices aligned with industry guidelines.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li><b>1. Block Mode Configuration:</b> <ul style="list-style-type: none"> <li>✓ Test: Review the encryption implementation to confirm the selected block mode aligns with the data use case (e.g., CBC or GCM for general encryption, CTR for streaming).</li> <li>✓ Expected Result: The block mode is correctly configured for the intended encryption scenario.</li> </ul> </li> <li><b>2. Initialization Vector (IV) Management:</b> <ul style="list-style-type: none"> <li>✓ Test: Inspect IV generation processes to ensure uniqueness and unpredictability for each encryption operation.</li> <li>✓ Expected Result: Each encryption operation generates a unique and random IV.</li> </ul> </li> </ol>

- |   |
|---|
| <p><b>3. Counter Management in CTR Mode:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Validate that counters in CTR mode do not repeat or overlap across encryption operations.</li> <li>✓ Expected Result: Counters are managed to prevent reuse, maintaining data confidentiality.</li> </ul> <p><b>4. Authentication Testing (GCM Mode):</b></p> <ul style="list-style-type: none"> <li>✓ Test: Encrypt and decrypt data using GCM mode to verify authentication tags and data integrity.</li> <li>✓ Expected Result: Decryption succeeds only when the authentication tag is valid, confirming data integrity.</li> </ul> <p><b>5. Encryption Process Audit:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Audit the implementation of encryption libraries and configurations for adherence to best practices.</li> <li>✓ Expected Result: The encryption process uses approved libraries and configurations that comply with security standards.</li> </ul> <p><b>6. Dynamic Encryption Validation:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Perform runtime encryption and decryption operations to validate block mode functionality under operational conditions.</li> <li>✓ Expected Result: Data is encrypted and decrypted successfully, with no leakage or integrity compromise.</li> </ul> |
|---|

**Requested by:** The organization

**Responsible:** Developer

**Configurable value:** Not described

**Version history:** v1.0

**Author and date:**

Carlos M. Mejía-Granda  
 José L Fernández-Alemán  
 Juan Manuel Carrillo-de-Gea  
 Joaquín Nicolás  
 08/01/2026

**PUID:** [SECM-CAT-ICR-020]

**Requirement description:** The application must not rely solely on symmetric cryptography with hardcoded keys for encryption. Instead, implement secure key management practices, such as dynamically generating or securely retrieving keys at runtime, to enhance the cryptographic security of the application.

**Source:**

- ✓ 3.1: Verify that the app does not rely on symmetric cryptography with hardcoded keys as a sole method of encryption [19].

**Priority:** Not described

**Rationale:** Hardcoded symmetric keys can easily be extracted through reverse engineering, exposing encrypted data to unauthorized access. Adopting dynamic key generation and secure key retrieval mechanisms mitigates the risk of key compromise and strengthens the overall security posture of the application.

**Number of Children:** 0

**Number of parents:** 0

**Cycles:** 0

**Number Audit:** 0

**Child PUIDs:** Not described

<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<p><b>1. Key Hardcoding Review:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Perform a static analysis of the application code to detect hardcoded cryptographic keys.</li> <li>✓ Expected Result: No hardcoded cryptographic keys are found in the codebase.</li> </ul> <p><b>2. Key Management Verification:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Validate the implementation of key management practices, ensuring keys are dynamically generated or retrieved securely at runtime.</li> <li>✓ Expected Result: Keys are not embedded in the application but managed securely using runtime mechanisms.</li> </ul> <p><b>3. Dynamic Key Testing:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Simulate the application's encryption processes to verify that keys are dynamically assigned and securely exchanged.</li> <li>✓ Expected Result: Encryption and decryption operations use dynamically generated or securely retrieved keys.</li> </ul> <p><b>4. Reverse Engineering Resistance:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Use reverse engineering techniques on the application binary to confirm that symmetric keys cannot be extracted.</li> <li>✓ Expected Result: Symmetric keys are not recoverable from the application binary.</li> </ul> <p><b>5. Encryption Security Audit:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Audit the cryptographic implementation for adherence to best practices, such as using secure algorithms and proper key rotation policies.</li> <li>✓ Expected Result: Encryption mechanisms comply with security standards, and key management practices are robust and secure.</li> </ul> <p><b>6. End-to-End Testing:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Encrypt and decrypt data in various operational scenarios to ensure secure and reliable cryptographic behavior.</li> <li>✓ Expected Result: Encrypted data remains secure, and the process functions without exposing or compromising the keys.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-ICR-021]
<b>Requirement description:</b> The application must ensure that cryptographic keys are not reused for multiple purposes, such as encrypting data and signing messages. Separate cryptographic keys must be utilized for distinct cryptographic operations to maintain security and prevent key misuse.
<b>Source:</b>

✓ 3.5: Verify that the app doesn't re-use the same cryptographic key for multiple purposes [19].
<b>Priority:</b> Not described
<b>Rationale:</b> Reusing cryptographic keys for multiple purposes can lead to vulnerabilities, such as weakening the encryption scheme or enabling cryptographic attacks. Implementing distinct keys for each purpose ensures that the security of one function does not compromise another, maintaining the integrity and confidentiality of sensitive data.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described
<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<ol style="list-style-type: none"> <li>1. <b>Key Purpose Isolation Audit:</b> <ul style="list-style-type: none"> <li>✓ Test: Review the cryptographic implementation to verify that keys are assigned unique purposes, such as encryption, decryption, or signing.</li> <li>✓ Expected Result: Each cryptographic key is exclusively used for a single purpose.</li> </ul> </li> <li>2. <b>Dynamic Key Generation Verification:</b> <ul style="list-style-type: none"> <li>✓ Test: Validate that keys are dynamically generated for each operation and not reused across different cryptographic functions.</li> <li>✓ Expected Result: Cryptographic keys are unique for their specific operation and purpose.</li> </ul> </li> <li>3. <b>Cryptographic Configuration Review:</b> <ul style="list-style-type: none"> <li>✓ Test: Inspect the application's cryptographic configuration to ensure separate key management mechanisms are in place for each type of cryptographic operation.</li> <li>✓ Expected Result: The application uses distinct keys for encryption, signing, and other cryptographic operations.</li> </ul> </li> <li>4. <b>Algorithm-Specific Key Use Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Evaluate the application's cryptographic workflows to confirm that algorithms requiring unique keys for specific purposes (e.g., AES for encryption, RSA for signing) adhere to best practices.</li> <li>✓ Expected Result: Cryptographic workflows do not mix or reuse keys between different algorithms or purposes.</li> </ul> </li> <li>5. <b>Security Penetration Test:</b> <ul style="list-style-type: none"> <li>✓ Test: Conduct a penetration test targeting cryptographic key reuse vulnerabilities.</li> <li>✓ Expected Result: The application resists attacks aimed at exploiting key reuse, maintaining data integrity and confidentiality.</li> </ul> </li> </ol>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b>
Carlos M. Mejía-Granda
José L Fernández-Alemán
Juan Manuel Carrillo-de-Gea
Joaquín Nicolás
08/01/2026

**PUID:** [SECM-CAT-ICR-022]

**Requirement description:** The application must utilize a secure and cryptographically strong random number generator, such as `SecureRandom`, to generate random values for cryptographic purposes, including key initialization, nonce generation, and secure tokens. This ensures the randomness required to maintain encryption strength and protect sensitive data.

**Source:**

- ✓ 3.6: Verify that all random values are generated using a sufficiently secure random number generator [19].
- ✓ Random Number Generation: Use `SecureRandom` to initialize cryptographic keys. Cryptographic keys generated without a secure random number generator may weaken the encryption and allow offline attacks [18]
- ✓ Encryption Best Practices: When handling sensitive data, follow established practices for encryption and secure communications. Always use secure key storage and transmission methods to protect data integrity and confidentiality [18]

**Priority:** Not described

**Rationale:** Using non-secure random number generators can result in predictable values, undermining cryptographic protocols and exposing sensitive data to attacks such as brute force or replay. A secure random number generator provides high entropy and ensures randomness critical for secure encryption and token generation.

**Number of Children:** 0**Number of parents:** 0**Cycles:** 0**Number Audit:** 0**Child PUIDs:** Not described**Parent PUIDs:** Not described**Exclusion PUIDs:** Not described**Importance:** Not described**Current state:** To be determined**Verification method:** Demonstration/Analysis**Validation criteria:****1. Random Number Source Validation:**

- ✓ Test: Inspect the application's random number generation methods to ensure the use of a secure generator like `SecureRandom`.
- ✓ Expected Result: The application consistently uses secure random number generators for all cryptographic operations.

**2. Entropy Verification:**

- ✓ Test: Evaluate the entropy of generated random values using statistical tests to ensure high randomness.
- ✓ Expected Result: Random values exhibit high entropy and do not follow predictable patterns.

**3. Key Generation Audit:**

<ul style="list-style-type: none"> <li>✓ Test: Verify that all cryptographic keys are initialized using a secure random number generator.</li> <li>✓ Expected Result: Keys are securely generated with no predictable or reused values.</li> </ul>
<b>4. Token Security Test:</b>
<ul style="list-style-type: none"> <li>✓ Test: Analyze generated tokens or nonces for randomness and resistance to replay or guessing attacks.</li> <li>✓ Expected Result: Tokens are unpredictable and unique for each session or operation.</li> </ul>
<b>5. Penetration Testing for Randomness Weakness:</b>
<ul style="list-style-type: none"> <li>✓ Test: Conduct security testing to identify potential weaknesses in the application's random value generation process.</li> <li>✓ Expected Result: The application demonstrates resilience to attacks targeting randomness vulnerabilities.</li> </ul>
<b>6. Logging and Monitoring of Randomness Failures:</b>
<ul style="list-style-type: none"> <li>✓ Test: Verify that the application logs and alerts on any detected failures in random number generation.</li> <li>✓ Expected Result: Failures or abnormalities in randomness generation are logged and trigger security alerts.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

<b>PUID:</b> [SECM-CAT-ICR-023]
<b>Requirement description:</b> The application must ensure that Initialization Vectors (IVs) or counters used in CTR (Counter) mode encryption are cryptographically random and never reused. This protects the confidentiality and integrity of encrypted data by preventing vulnerabilities related to predictability or repeated usage of cryptographic parameters.
<b>Source:</b>
<ul style="list-style-type: none"> <li>✓ Avoid Counter Reuse: In CTR mode, ensure IV/counters are cryptographically random to prevent vulnerabilities [18]</li> </ul>
<b>Priority:</b> Not described
<b>Rationale:</b> Using non-secure random number generators can result in predictable values, undermining cryptographic protocols and exposing sensitive data to attacks such as brute force or replay. A secure random number generator provides high entropy and ensures randomness critical for secure encryption and token generation.
<b>Number of Children:</b> 0
<b>Number of parents:</b> 0
<b>Cycles:</b> 0
<b>Number Audit:</b> 0
<b>Child PUIDs:</b> Not described
<b>Parent PUIDs:</b> Not described
<b>Exclusion PUIDs:</b> Not described
<b>Importance:</b> Not described

<b>Current state:</b> To be determined
<b>Verification method:</b> Demonstration/Analysis
<b>Validation criteria:</b>
<p><b>1. Randomness Audit:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Verify that IVs or counters are generated using secure random number generators, such as SecureRandom.</li> <li>✓ Expected Result: Generated IVs and counters exhibit cryptographic randomness and high entropy.</li> </ul> <p><b>2. Replay Attack Simulation:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Simulate a replay attack to ensure the application does not accept reused IVs or counters.</li> <li>✓ Expected Result: The application rejects reused IVs or counters, ensuring data integrity.</li> </ul> <p><b>3. Code Review for Counter Reuse:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Conduct a thorough review of the codebase to ensure no instances of counter reuse exist.</li> <li>✓ Expected Result: All counters and IVs are unique and securely generated for each operation.</li> </ul> <p><b>4. Dynamic Encryption Testing:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Perform runtime testing of the encryption module to observe IV/counter generation during operations.</li> <li>✓ Expected Result: Encryption consistently generates unique, random IVs or counters without reuse.</li> </ul> <p><b>5. Integrity Validation Post-Encryption:</b></p> <ul style="list-style-type: none"> <li>✓ Test: Verify the integrity and security of encrypted data when processed using CTR mode with generated IVs/counters.</li> <li>✓ Expected Result: Data remains secure and resistant to cryptographic analysis or attacks.</li> </ul>
<b>Requested by:</b> The organization
<b>Responsible:</b> Developer
<b>Configurable value:</b> Not described
<b>Version history:</b> v1.0
<b>Author and date:</b> Carlos M. Mejía-Granda José L Fernández-Alemán Juan Manuel Carrillo-de-Gea Joaquín Nicolás 08/01/2026

## 2.9.4 Maintainability

*[[specify attributes of software that relate to the ease of maintenance of the software itself. These may include requirements for certain modularity, interfaces or complexity limitation. Requirements should not be placed here just because they are thought to be good design practices.]]*

Not described.

## 2.9.5 Portability

*[[specify attributes of software that relate to the ease of porting the software to other host machines and/or operating systems, including:*

- 1) percentage of elements with host-dependent code;
- 2) percentage of code that is host dependent;

- 3) use of a proven portable language;
- 4) use of a particular compiler or language subset; and
- 5) use of a particular operating system.]]

Not described.

### 3. Verification

*[[Provide the verification approaches and methods planned to qualify the software. The information items for verification are recommended to be given in a parallel manner with the information items in 9.6.10 to 9.6.18]]*

Not described.

### 4. Supporting Information

*[[Additional supporting information to be considered includes:*

- a) sample input/output formats, descriptions of cost analysis studies or results of user surveys;
- b) supporting or background information that can help the readers of the SRS;
- c) a description of the problems to be solved by the software; and
- d) special packaging instructions for the code and the media to meet security, export, initial loading or other requirements.

*The SRS should explicitly state whether or not these information items are to be considered part of the requirements.]]*

Not described.

### 5. References

- [1] “IEEE SA - IEEE/ISO/IEC 29148-2018.” Accessed: May 11, 2024. [Online]. Available: <https://standards.ieee.org/ieee/29148/6937/>
- [2] “IEEE SA - IEEE 830-1998.” Accessed: Jul. 05, 2024. [Online]. Available: <https://standards.ieee.org/ieee/830/1222/>
- [3] OWASP Foundation, “OWASP Mobile Top 10.” Accessed: May 11, 2024. [Online]. Available: <https://owasp.org/www-project-mobile-top-10/>
- [4] “IEEE Guide for Developing System Requirements Specifications,” *IEEE Std 1233, 1998 Edition*, pp. 1–36, 1998, doi: 10.1109/IEEESTD.1998.88826.
- [5] “ISO/IEC 27001 Standard – Information Security Management Systems.” Accessed: May 11, 2024. [Online]. Available: <https://www.iso.org/standard/27001>
- [6] “ISO/IEC 27002:2022 - Information security, cybersecurity and privacy protection — Information security controls.” Accessed: May 11, 2024. [Online]. Available: <https://www.iso.org/standard/75652.html>
- [7] “ISO 27799:2016 - Health informatics — Information security management in health using ISO/IEC 27002.” Accessed: Jul. 04, 2024. [Online]. Available: <https://www.iso.org/standard/62777.html>

- [8] “Health Insurance Portability and Accountability Act of 1996 | ASPE.” Accessed: Nov. 10, 2022. [Online]. Available: <https://aspe.hhs.gov/reports/health-insurance-portability-accountability-act-1996>
- [9] “eCFR :: 45 CFR Part 164 Subpart C -- Security Standards for the Protection of Electronic Protected Health Information.” Accessed: Sep. 27, 2023. [Online]. Available: <https://www.ecfr.gov/current/title-45/subtitle-A/subchapter-C/part-164/subpart-C>
- [10] “Privacy and Security Requirements and Considerations for Digital Health Solutions | Canada Health Infoway.” Accessed: Nov. 11, 2023. [Online]. Available: <https://www.infoway-inforoute.ca/en/component/edocman/resources/technical-documents/architecture/2154-privacy-and-security-requirements-and-considerations-for-digital-health-solutions>
- [11] J. T. Force, “Security and Privacy Controls for Information Systems and Organizations,” Jul. 2020, doi: 10.6028/NIST.SP.800-53R5.
- [12] G. O’Brien *et al.*, “Securing Electronic Health Records on Mobile Devices,” *NIST SPECIAL PUBLICATION*, pp. 1800–1801, Jul. 2018, doi: 10.6028/NIST.SP.1800-1.
- [13] “Top 25 Software Errors | SANS Institute.” Accessed: Aug. 16, 2024. [Online]. Available: <https://www.sans.org/top25-software-errors/>
- [14] Center for Internet Security, “CIS Critical Security Controls Version 8.1.” Accessed: Sep. 26, 2024. [Online]. Available: <https://www.cisecurity.org/controls/v8-1>
- [15] “Application Security and Development Security Technical Implementation Guide.” [Online]. Available: [https://www.stigviewer.com/stig/application\\_security\\_and\\_development/](https://www.stigviewer.com/stig/application_security_and_development/)
- [16] “Smartphone Secure Development Guidelines — ENISA.” Accessed: Sep. 25, 2024. [Online]. Available: <https://www.enisa.europa.eu/publications/smartphone-secure-development-guidelines-2016>
- [17] “Mobile Application Security Requirements Guide,” Jul. 22, 2014. [Online]. Available: [https://www.stigviewer.com/stig/mobile\\_application\\_security\\_requirements\\_guide/](https://www.stigviewer.com/stig/mobile_application_security_requirements_guide/)
- [18] Android Open Source Project, “Improve your app’s security ┌ App quality ┌ Android Developers,” Apr. 18, 2024. [Online]. Available: <https://developer.android.com/privacy-and-security/security-best-practices#java>
- [19] owasp.org, “OWASP MAS Checklist.” Accessed: Jun. 11, 2024. [Online]. Available: [https://wiki.owasp.org/images/1/1b/Mobile\\_App\\_Security\\_Checklist\\_0.9.3.xlsx](https://wiki.owasp.org/images/1/1b/Mobile_App_Security_Checklist_0.9.3.xlsx)
- [20] owasp.org, “OWASP MASVS - OWASP Mobile Application Security.” Accessed: Sep. 26, 2024. [Online]. Available: <https://mas.owasp.org/MASVS/>
- [21] P. Llorens-Vernet and J. Miró, “Standards for Mobile Health–Related Apps: Systematic Review and Development of a Guide,” *JMIR Mhealth Uhealth*, vol. 8, no. 3, p. e13057, 2020, doi: 10.2196/13057.
- [22] K. Escarfuller, C. Moore, S. Tucker, and J. Wei, “An object-oriented mobile health system with usability features,” *Int. J. Electron. Healthc.*, vol. 7, no. 1, pp. 53–67, 2012, doi: 10.1504/IJEH.2012.048669.
- [23] B. Martínez-Pérez, I. de la Torre-Díez, and M. López-Coronado, “Privacy and Security in Mobile Health Apps: A Review and Recommendations,” *J. Med. Syst.*, vol. 39, no. 1, 2015, doi: 10.1007/s10916-014-0181-3.
- [24] D. D. Luxton, R. A. Kayl, and M. C. Mishkind, “MHealth data security: The need for HIPAA-compliant standardization,” *Telemedicine and e-Health*, vol. 18, no. 4, pp. 284–288, 2012, doi: 10.1089/tmj.2011.0180.
- [25] E. P. Morera, I. de la Torre Díez, B. Garcia-Zapirain, M. López-Coronado, and J. Arambarri, “Security Recommendations for mHealth Apps: Elaboration of a Developer’s Guide,” *J. Med. Syst.*, vol. 40, no. 6, 2016, doi: 10.1007/s10916-016-0513-6.
- [26] L. Zhou, J. Bao, V. Watzlaf, and B. Parmanto, “Barriers to and Facilitators of the Use of Mobile Health Apps From a Security Perspective: Mixed-Methods Study,” *JMIR Mhealth Uhealth*, vol. 7, no. 4, p. e11223, 2019, doi: 10.2196/11223.
- [27] M. Wazid, S. Zeadally, A. K. Das, and V. Odelu, “Analysis of Security Protocols for Mobile Healthcare,” *J. Med. Syst.*, vol. 40, no. 11, 2016, doi: 10.1007/s10916-016-0596-0.

- [28] A. Papageorgiou, M. Strigkos, E. Politou, E. Alepis, A. Solanas, and C. Patsakis, “Security and Privacy Analysis of Mobile Health Applications: The Alarming State of Practice,” *IEEE Access*, vol. 6, pp. 9390–9403, 2018, doi: 10.1109/ACCESS.2018.2799522.
- [29] P. Llorens-Vernet and J. Miró, “The mobile app development and assessment guide (MAG): Delphi-Based validity study,” *JMIR Mhealth Uhealth*, vol. 8, no. 7, 2020, doi: 10.2196/17760.
- [30] D. Sethia, D. Gupta, and H. Saran, “Smart health record management with secure NFC-enabled mobile devices,” *Smart Health*, vol. 13, p. 100063, 2019, doi: <https://doi.org/10.1016/j.smhl.2018.11.001>.
- [31] M. A. Ferrag, L. Shu, and K.-K. R. Choo, “Fighting COVID-19 and Future Pandemics with the Internet of Things: Security and Privacy Perspectives,” *IEEE/CAA Journal of Automatica Sinica*, vol. 8, no. 9, pp. 1477–1499, 2021, doi: 10.1109/JAS.2021.1004087.
- [32] A. Sengupta and H. Subramanian, “User Control of Personal mHealth Data Using a Mobile Blockchain App: Design Science Perspective,” *JMIR Mhealth Uhealth*, vol. 10, no. 1, 2022, doi: 10.2196/32104.
- [33] R. Rezaee, M. Khashayar, S. Saeedinezhad, M. Nasiri, and S. Zare, “Critical Criteria and Countermeasures for Mobile Health Developers to Ensure Mobile Health Privacy and Security: Mixed Methods Study,” *JMIR Mhealth Uhealth*, vol. 11, 2023, doi: 10.2196/39055.
- [34] C. Esposito, R. Horne, L. Robaldo, B. Buelens, and E. Goesaert, “Assessing the Solid Protocol in Relation to Security and Privacy Obligations,” *Information (Switzerland)*, vol. 14, no. 7, 2023, doi: 10.3390/info14070411.
- [35] cwe.mitre.org, “CWE - 2022 CWE Top 25 Most Dangerous Software Weaknesses.” Accessed: Mar. 22, 2024. [Online]. Available: [https://cwe.mitre.org/top25/archive/2022/2022\\_cwe\\_top25.html](https://cwe.mitre.org/top25/archive/2022/2022_cwe_top25.html)
- [36] mitre.org, “CWE - 2025 CWE Top 25 Most Dangerous Software Weaknesses,” 2026. [Online]. Available: [https://cwe.mitre.org/top25/archive/2025/2025\\_cwe\\_top25.html](https://cwe.mitre.org/top25/archive/2025/2025_cwe_top25.html)
- [37] G. O’Brien *et al.*, “Securing Electronic Health Records on Mobile Devices,” *NIST SPECIAL PUBLICATION*, pp. 1800–1801, Jul. 2018, doi: 10.6028/NIST.SP.1800-1.
- [38] U.S. Centers for Disease Control and Prevention, “Health Insurance Portability and Accountability Act of 1996 (HIPAA) | Public Health Law | CDC.” Accessed: Jul. 08, 2024. [Online]. Available: <https://www.cdc.gov/phlp/php/resources/health-insurance-portability-and-accountability-act-of-1996-hipaa.html>