

# PENETRATION TEST REPORT

TODOTODO

Application Manager	TODOTODO
Department	TODOTODO
Asset Type	TODOTODO

SHS TE DC CYS CSA

Classification: CONFIDENTIAL

Effective Date: July 10, 2023

Author: TODOTODO	
Approver: TODOTODO	

# Document Information

## Report

This document was issued by SHS TE DC CYS CSA on July 10, 2023.

## Classification

Version "0.5" of this document with the status "RELEASE" has been classified as "CONFIDENTIAL".

## Authors and Reviewers

Name	
<b>Authors</b>	Lukas Nad, Taksh Medhavi, Michal Olencin
<b>Reviewers</b>	Katarina Amrichova
<b>Approvers</b>	Filip Mrocek

Table 1: List of authors and reviewers

---

# Document History

Date	Version	Name	Remarks
2023-06-06	0.1	Lukas Nad	Initial Draft
2023-06-07	0.2	Michal Olencin	Added Findings
2023-06-08	0.3	Taksh Medhavi	Added Findings
2023-06-08	0.4	Katarina Amrichova	Review
2023-06-09	0.5	Lukas Nad	Release

Table 2: Document revision history

# Document Version

Template version: v3.0

# Contents

<b>1</b>	<b>Disclaimer</b>	<b>5</b>
<b>2</b>	<b>Executive Summary</b>	<b>6</b>
<b>3</b>	<b>Project Information</b>	<b>8</b>
3.1	Participants and Contacts . . . . .	8
3.2	Target Information . . . . .	9
3.3	Test Location and Duration . . . . .	10
<b>4</b>	<b>Summary of Findings</b>	<b>11</b>
<b>5</b>	<b>Findings</b>	<b>15</b>
	Finding 1: ePHI is stored on device without encryption . . . . .	15
	Finding 2: Sensitive Information Disclosure via Logging . . . . .	18
	Finding 3: Weak Application Signature . . . . .	21
	Finding 4: Heap Inspection of Sensitive Memory . . . . .	24
	Finding 5: Outdated Components . . . . .	27
	Finding 6: DummyApplication Signed with a Debug Certificate . . . . .	30
	Finding 7: Missing Enforced Updating . . . . .	32
<b>6</b>	<b>Scope and Procedures</b>	<b>34</b>
6.1	Scope . . . . .	34
6.2	Worst Case Scenarios . . . . .	34
6.3	Out Of Scope . . . . .	36
6.4	Environment . . . . .	36
<b>7</b>	<b>Testing Methodology</b>	<b>37</b>
7.1	Tools Used . . . . .	37
7.2	Attack Vectors and Payload Types . . . . .	38
<b>8</b>	<b>Next Steps</b>	<b>39</b>
8.1	Test Cleanup . . . . .	39
8.2	Further Recommendations . . . . .	39
	<b>Bibliography</b>	<b>40</b>
<b>A</b>	<b>Appendix A</b>	<b>42</b>
A.1	Criticality Levels . . . . .	42
A.2	Overall Threat Exposure . . . . .	43
A.3	Testing Approaches . . . . .	43
A.4	Test Protocol . . . . .	45

# 1 Disclaimer

Please note the following aspects of this penetration test:

- Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec lacinia tellus sed dui accumsan placerat sed ut lorem. Cras aliquet.
- Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec lacinia tellus sed dui accumsan placerat sed ut lorem. Cras aliquet.
- Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec lacinia tellus sed dui accumsan placerat sed ut lorem. Cras aliquet.
- Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec lacinia tellus sed dui accumsan placerat sed ut lorem. Cras aliquet.
- Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec lacinia tellus sed dui accumsan placerat sed ut lorem. Cras aliquet.
- Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec lacinia tellus sed dui accumsan placerat sed ut lorem. Cras aliquet.
- Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec lacinia tellus sed dui accumsan placerat sed ut lorem. Cras aliquet.
- Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec lacinia tellus sed dui accumsan placerat sed ut lorem. Cras aliquet.
- Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec lacinia tellus sed dui accumsan placerat sed ut lorem. Cras aliquet.
- Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec lacinia tellus sed dui accumsan placerat sed ut lorem. Cras aliquet.

## 2 Executive Summary

The Penetration Testing team at SHS TE DC CYS CSA in Slovakia conducted a penetration test of **DUMMY PROJECT 1** system in order to assess its overall security posture.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed ultricies pharetra pretium. Cras varius purus eu cursus vehicula. Sed in molestie arcu, id placerat velit. Praesent sagittis purus in neque convallis, a faucibus odio egestas. Nam ultrices, metus et mattis facilisis, felis lectus tempor velit, a interdum nisl libero nec dui. Mauris interdum scelerisque semper. Cras mattis id lacus a ullamcorper. Curabitur fermentum vehicula leo, vel convallis turpis luctus nec. In mollis vitae diam in ornare. Donec molestie augue nisl, malesuada maximus urna gravida quis. Curabitur ac ante turpis. Nulla facilisi. Aenean eleifend ipsum at velit lobortis, in hendrerit arcu dapibus. Proin ut lacus sed tellus maximus euismod. Suspendisse elementum mauris tellus, eget imperdiet leo dictum nec. Fusce tortor mauris, iaculis non tristique ut, condimentum a odio.

Maecenas tincidunt sollicitudin metus id eleifend. Cras justo urna, tempus et mi vestibulum, iaculis pellentesque nunc. Etiam nisi nibh, bibendum sed augue in, molestie lacinia turpis. Ut bibendum pretium mi vel volutpat. Praesent mattis scelerisque neque a vehicula. Cras nec iaculis mi, in rutrum ligula. Suspendisse potenti.

Fusce mollis, erat eget tempus ornare, erat nisl mattis dolor, sed porta mauris quam eget tortor. Etiam bibendum sodales lorem ut fringilla. Phasellus in urna ex. In venenatis turpis a augue egestas efficitur. Interdum et malesuada fames ac ante ipsum primis in faucibus. Cras laoreet odio eu auctor molestie. In congue malesuada sollicitudin.

Integer egestas mollis ex quis semper. Nam vitae diam aliquet, elementum leo molestie, suscipit mauris. Suspendisse ex magna, fermentum eu sem non, convallis faucibus nisl. In id dignissim orci, non sodales ante. Nulla bibendum sem nec turpis porta pulvinar. Curabitur fringilla libero ut ex faucibus, non ultrices nisl volutpat. Quisque imperdiet condimentum diam eu scelerisque. Integer ullamcorper euismod accumsan. Curabitur efficitur, neque non blandit tempus, quam erat viverra risus, eu placerat risus elit eu neque.

## Overall Exposure

The Penetration test identified **0 Critical**, **1 High**, **2 Medium**, **2 Low** and **2 Info** classified findings, which means that the overall threat exposure (see Appendix [A.2](#)) is currently **High**:



Figure 2.1: Overall Threat Exposure

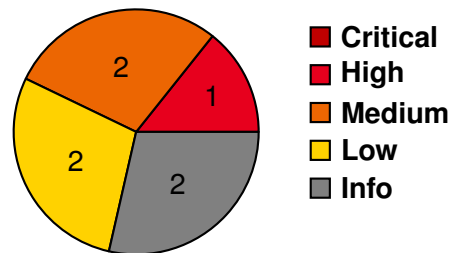


Figure 2.2: Breakdown of finding criticality

It is recommended to remediate all the **Critical**, **High**, **Medium** and **Low** findings to improve the overall security status of our Healthineers assets.

## 3 Project Information

### 3.1 Participants and Contacts

Role	Name	Department	Contact
Application Manager	Anakin Skywalker	SHS DI D&A CEC ITH EH-PLM	<a href="mailto:anakin.skywalker">anakin.skywalker</a> <sup>1</sup>
Technical Contacts	Obi Wan Kenobi	SHS TE DC SVK D&A DIG PTM	<a href="mailto:obi-wan.kenobi">obi-wan.kenobi</a> <sup>1</sup>
	Baby Yoda	SHS DI D&A CEC ITH EH-R&D	<a href="mailto:baby.yoda">baby.yoda</a> <sup>1</sup>
Pentest Coordinator	Alzbeta Vojtusova	SHS TE DC CYS CSA P&PA	<a href="mailto:alzbeta.vojtusova">alzbeta.vojtusova</a> <sup>1</sup>
Pentest Project Lead	Lukas Nad	SHS TE DC CYS CSA P&PA	<a href="mailto:lukas.nad">lukas.nad</a> <sup>1</sup>
Pentest Team	Lukas Nad	SHS TE DC CYS CSA P&PA	<a href="mailto:lukas.nad">lukas.nad</a> <sup>1</sup>
	Michal Olencin	SHS TE DC CYS CSA P&PA	<a href="mailto:michal.olencin">michal.olencin</a> <sup>1</sup>
	Medhavi Taksh	SHS TE DC CYS LAB	<a href="mailto:taksh.bhatt">taksh.bhatt</a> <sup>1</sup>

Table 3.1: Test participants and contacts

<sup>1</sup>Contact e-mail address: {first.last}@siemens-healthineers.com

<sup>2</sup>Contact e-mail address: {first.last}@varian.com



## 3.2 Target Information

In agreement with the **Application Manager**, the Penetration Test has been conducted on the following target:

Property	Target Information
Name	Dummy Project 1
Version	12.1.1.2
Asset Type	Mobile Application
Environment	Testing Environment
Internet Facing	Yes
SNX Connectivity	No
Hosting Location	Special Network
Hosting Provider	N/A
Lifecycle Phase	Pre-Production
ACP Criticality	N/A
ACP ID	N/A
SHARP UUID	N/A
Description	Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed ultricies pharetra pretium. Cras varius purus eu cursus vehicula. Sed in molestie arcu, id placerat velit. Praesent sagittis purus in neque convallis, a faucibus odio egestas. Nam ultrices, metus et mattis facilisis, felis lectus tempor velit, a interdum nisl libero nec dui. Mauris interdum scelerisque semper. Cras mattis id lacus a ullamcorper. Curabitur fermentum vehicula leo, vel convallis turpis luctus nec. In mollis vitae diam in ornare. Donec molestie augue nisl, malesuada maximus urna gravida quis. Curabitur ac ante turpis. Nulla facilisi. Aenean eleifend ipsum at velit lobortis, in hendrerit arcu dapibus. Proin ut lacus sed tellus maximus euismod. Suspendisse elementum mauris tellus, eget imperdiet leo dictum nec. Fusce tortor mauris, iaculis non tristique ut, condimentum a odio.

Table 3.2: Target information

### 3.3 Test Location and Duration

The Penetration test was performed remotely and on site from Siemens Healthineers Bratislava, Kosice and Zilina (Slovakia).

Agreed Timeframe	
Testing timeframe:	10 working days
Pentest Start:	2023-05-29
Pentest End:	2023-06-09
Report due date:	2023-06-12
Comment:	-

Table 3.3: Test Duration

## 4 Summary of Findings

### Finding 1: ePHI is stored on device without encryption

ID: Dummy Project 1-FY23-1

<b>Criticality:</b>	<b>HIGH</b>	<b>Category:</b>	Application Design
<b>Detectability:</b>	Easy	<b>Exploitability:</b>	Easy
<b>CWE / CVE:</b>	<b>CWE-359</b>	<b>CVSS Score:</b>	8.0
<b>CVSS Vector:</b>	CVSS:3.1/AV:A/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H		
<b>Location:</b>	dummyapplication.apk		
<b>Component:</b>	Android local storage		

(Further details on page 15.)

### Finding 2: Sensitive Information Disclosure via Logging

ID: Dummy Project 1-FY23-2

<b>Criticality:</b>	<b>MEDIUM</b>	<b>Category:</b>	Information Disclosure
<b>Detectability:</b>	Easy	<b>Exploitability:</b>	Easy
<b>CWE / CVE:</b>	<b>CWE-532</b>	<b>CVSS Score:</b>	4.0
<b>CVSS Vector:</b>	CVSS:3.1/AV:L/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N		
<b>Location:</b>	dummyapplication.apk, dummyapplication.ipa		
<b>Component:</b>	Logcat		

(Further details on page 18.)

### Finding 3: Weak Application Signature

*ID: Dummy Project 1-FY23-3*

<b>Criticality:</b>	<b>MEDIUM</b>	<b>Category:</b>	Application Design
<b>Detectability:</b>	Easy	<b>Exploitability:</b>	Hard
<b>CWE / CVE:</b>	<a href="#">CWE-328</a>	<b>CVSS Score:</b>	4.7
<b>CVSS Vector:</b>	CVSS:3.1/AV:L/AC:H/PR:L/UI:N/S:U/C:N/I:H/A:N		
<b>Location:</b>	dummyapplication.apk		
<b>Component:</b>	META-INF/CERT.SF		

*(Further details on page [21](#).)*

### Finding 4: Heap Inspection of Sensitive Memory

*ID: Dummy Project 1-FY23-4*

<b>Criticality:</b>	<b>LOW</b>	<b>Category:</b>	Application Design
<b>Detectability:</b>	Easy	<b>Exploitability:</b>	Average
<b>CWE / CVE:</b>	<a href="#">CWE-244</a>	<b>CVSS Score:</b>	3.3
<b>CVSS Vector:</b>	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:L/I:N/A:N		
<b>Location:</b>	dummyapplication.apk, dummyapplication.ipa		
<b>Component:</b>	AppCacheTemplate		

*(Further details on page [24](#).)*

**Finding 5: Outdated Components***ID: Dummy Project 1-FY23-5*

<b>Criticality:</b>	<b>LOW</b>	<b>Category:</b>	Outdated Software
<b>Detectability:</b>	Easy	<b>Exploitability:</b>	Hard
<b>CWE / CVE:</b>	<a href="#">CWE-1104</a>	<b>CVSS Score:</b>	2.5
<b>CVSS Vector:</b>	CVSS:3.1/AV:L/AC:H/PR:L/UI:N/S:U/C:L/I:N/A:N		
<b>Location:</b>	dummyapplication.apk, dummyapplication.ipa		
<b>Component:</b>	Multiple		

*(Further details on page [27](#).)***Finding 6: DummyApplication Signed with a Debug Certificate***ID: Dummy Project 1-FY23-6*

<b>Criticality:</b>	<b>INFO</b>	<b>Category:</b>	Security Configuration
<b>Detectability:</b>	Difficult	<b>Exploitability:</b>	Hard
<b>CWE / CVE:</b>	<a href="#">CWE-296</a>	<b>CVSS Score:</b>	N/A
<b>CVSS Vector:</b>	N/A		
<b>Location:</b>	dummyapplication.apk		
<b>Component:</b>	META-INF/CERT.RSA		

*(Further details on page [30](#).)*

**Finding 7: Missing Enforced Updating***ID: Dummy Project 1-FY23-7*

<b>Criticality:</b>	INFO	<b>Category:</b>	Outdated Software
<b>Detectability:</b>	Easy	<b>Exploitability:</b>	Hard
<b>CWE / CVE:</b>	<a href="#">CWE-691</a>	<b>CVSS Score:</b>	N/A
<b>CVSS Vector:</b>	N/A		
<b>Location:</b>	dummyapplication.apk, dummyapplication.ipa		
<b>Component:</b>	Multiple		

*(Further details on page [32](#).)*

## 5 Findings

### Finding 1: ePHI is stored on device without encryption

ID: Dummy Project 1-FY23-1

<b>Criticality:</b>	<b>HIGH</b>	<b>Category:</b>	Application Design
<b>Detectability:</b>	Easy	<b>Exploitability:</b>	Easy
<b>CWE / CVE:</b>	<b>CWE-359</b>	<b>CVSS Score:</b>	8.0
<b>CVSS Vector:</b>	CVSS:3.1/AV:A/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H		
<b>Location:</b>	dummyapplication.apk		
<b>Component:</b>	Android local storage		

### Details

Application allows user to save ePHI (Electronic Protected Health Information) on device and patient information is stored in plain text in HTML file format. Application stores HTML file in `android > data > com.siemenshealthineers.dummyapplicationapp > files` directory. The file contains sensitive information such as patient vitals, allergies, diagnostic results and medication in it.

### Impact

Third party application installed in mobile devices can access ePHI stored in application data directory and since data at rest is stored without encryption, attacker can read contents of file which can lead to loss of confidentiality and violation of healthcare compliance. Application with external storage read/write permission can affect file integrity, as well as, application availability. In another scenario, attacker with local access of device can use file manager to access patient data.

## Repeatability

User can download patient summary details in device in plaintext HTML file as shown in Figure 5.1a. Figure 5.1b shows that HTML file contains ePHI such as patient vitals, allergies, diagnostic results and medication in it.

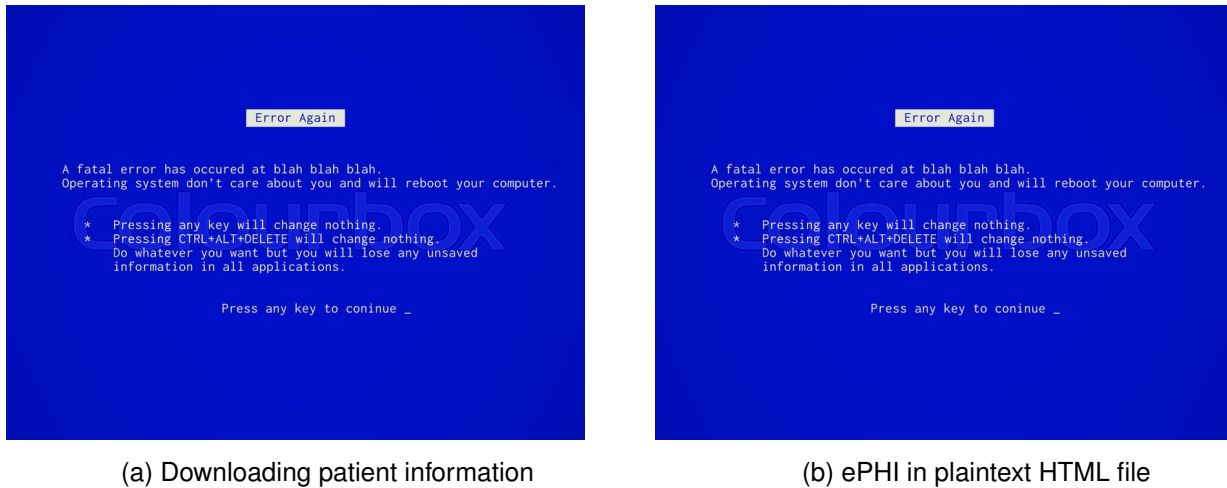


Figure 5.1: ePHI without data at rest encryption



User can also open patient summary HTML file using file manager and access ePHI as shown in Figure 5.2.



Figure 5.2: EPHI access via file manager

## Countermeasures

Application allows ePHI to be downloaded in plain HTML without encryption at rest. Application should handle patient data as per healthcare compliances applicable and should provide PDF report with password.

## References

This finding references the following information sources:

- [CVSS 8.0](#)
- MITRE. CWE-359: Exposure of Private Personal Information to an Unauthorized Actor. <http://cwe.mitre.org/data/definitions/359.html>, 2021. [Online; accessed 1-December-2021]

**Finding 2: Sensitive Information Disclosure via Logging***ID: Dummy Project 1-FY23-2*

<b>Criticality:</b>	<b>MEDIUM</b>	<b>Category:</b>	Information Disclosure
<b>Detectability:</b>	Easy	<b>Exploitability:</b>	Easy
<b>CWE / CVE:</b>	<b>CWE-532</b>	<b>CVSS Score:</b>	4.0
<b>CVSS Vector:</b>	CVSS:3.1/AV:L/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N		
<b>Location:</b>	dummyapplication.apk, dummyapplication.ipa		
<b>Component:</b>	Logcat		

**Details**

Sensitive information like JWT and exception stacktrace is being written to the logs, making it accessible to unauthorized parties.

**Impact**

The presence of sensitive information in the console creates a security risk as attackers could potentially access and misuse the information, leading to data breaches and other negative consequences.

**Repeatability**

During the static analysis it was observed, that the value of the cache (including JWT) is logged at level `info` in the method `saveString` (see Figure 5.3). Additionally, observe that the exception stack trace is logged at level `error` in the method `saveString` (see Figure 5.4). In Figure 5.5 is snippet that sets the log level. Since the level is set to `info`, both JWT and exception stacktrace are logged.

```
@mustCallSuper
@visibleForOverriding
Future<void> saveString(CacheKeys key, String value) async {
    _appLogger.info("Storing pair: $key - $value as string");
}
```

Figure 5.3: Snippet of the `saveString` method

Figure 5.4: Snippet of the `runZonedGuarded` method

```
// logger initialization
Logger.root.level = kReleaseMode ? Level.INFO : Level.ALL;
```

Figure 5.5: Snippet of the code for setting log level

The aforementioned code snippets can be observed during runtime using the Logcat tool. The following screenshot (Figure 5.6) shows Logcat output during the authentication.



Figure 5.6: Logcat output with set filter

## Countermeasures

It is recommended not to enable logging in the production build or to remove sensitive information from logging, such as JWT and exception stack trace.

## References

This finding references the following information sources:

- [CVSS 4.0](#)
- MITRE. CWE-532: Insertion of Sensitive Information into Log File. <https://cwe.mitre.org/data/definitions/532.html>, 2021. [Online; accessed 1-December-2021]

### Finding 3: Weak Application Signature

*ID: Dummy Project 1-FY23-3*

<b>Criticality:</b>	<b>MEDIUM</b>	<b>Category:</b>	Application Design
<b>Detectability:</b>	Easy	<b>Exploitability:</b>	Hard
<b>CWE / CVE:</b>	<a href="#">CWE-328</a>	<b>CVSS Score:</b>	4.7
<b>CVSS Vector:</b>	CVSS:3.1/AV:L/AC:H/PR:L/UI:N/S:U/C:N/I:H/A:N		
<b>Location:</b>	dummyapplication.apk		
<b>Component:</b>	META-INF/CERT.SF		

#### Details

Application uses SHA1 for application signing. Also, application supports Android v1 signing. For Android 5.0 to 8.1, it can lead to Janus vulnerability. Currently application is supporting Android 5.0 and higher versions.

#### Impact

SHA1 has flaws with regards to collisions and it is considered weak hashing algorithm. In Janus vulnerability scenario, the attacker can modify the code in applications without affecting their signatures.

## Repeatability

Application is using deprecated SHA1 algorithm and supports v1 also as shown in Figure 5.7.

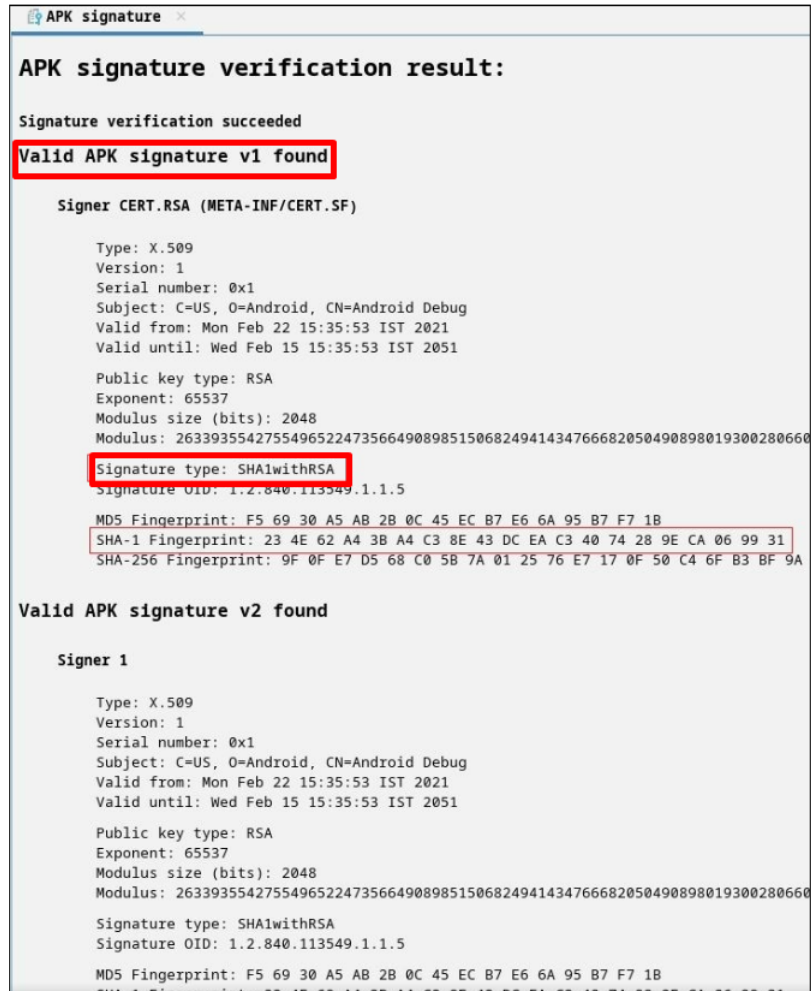


Figure 5.7: Application signed with v1

Application also supports Android minimum SDK version 21 which allows application to be installed on devices with Android 5.0 and higher.

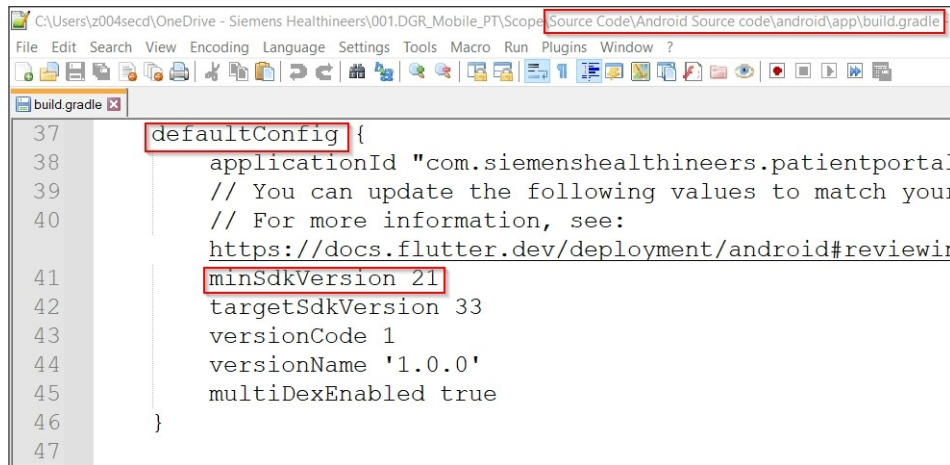


Figure 5.8: Application supports minSdkVersion 21

## Countermeasures

Sign the application with v2 signature with SHA256 hash. For Android device below version 8.0, Janus vulnerability is applicable for all versions of signature, so application should support Android device above 8.0 to avoid it. It can be achieved by changing default configuration value of "minSdkVersion" parameter in build.gradle file.

## References

This finding references the following information sources:

- CVSS 4.7
- Janus Vulnerability: CVE-2017-13156
- MITRE. CWE-328: Use of Weak Hash. <https://cwe.mitre.org/data/definitions/328.html>, 2021. [Online; accessed 1-December-2021]

## Finding 4: Heap Inspection of Sensitive Memory

*ID: Dummy Project 1-FY23-4*

<b>Criticality:</b>	<b>LOW</b>	<b>Category:</b>	Application Design
<b>Detectability:</b>	Easy	<b>Exploitability:</b>	Average
<b>CWE / CVE:</b>	<a href="#">CWE-244</a>	<b>CVSS Score:</b>	3.3
<b>CVSS Vector:</b>	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:L/I:N/A:N		
<b>Location:</b>	dummyapplication.apk, dummyapplication.ipa		
<b>Component:</b>	AppCacheTemplate		

### Details

The unprotected in-memory storage of plaintext sensitive data exposes its contents to potential disclosure. An absence of secure deletion mechanisms further extend the attack surface past the necessary longevity of the contents.

### Impact

Sensitive data residing in memory could be exposed to an attacker during a "heap inspection" attack. For instance, plaintext account credentials could be exposed during the exploitation of a memory disclosure vulnerability or the execution of a memory dump.



## Repeatability

By using static analysis, one can observe the variable `pJwt` is stored as `string` type in the `AppCacheTemplate` (see Figure 5.9).

```
abstract class AppCacheTemplate {
    @protected
    String pUrlBase = "";
    @protected
    bool pIsCustomer = false;
    @protected
    bool pFirstLaunch = true;
    @protected
    bool pIsBiometry = false;
    @protected
    bool pIsOnBoarding = false;
    @protected
    String pAppConfig = "";
    @protected
    String pUUID = "";
    @protected
    String pJwt = "";
```

Figure 5.9: Snippet of the `AppCacheTemplate` class

By following the following steps, contents of the `pJwt` variable can be read from memory:

1. Click **View > Tool Windows > Profiler** in the Android Studio.
2. Select the device and app process.
3. In memory profiler capture the memory dump (see Figure 5.10).
4. Extract a JWT from memory (see Figure 5.11).

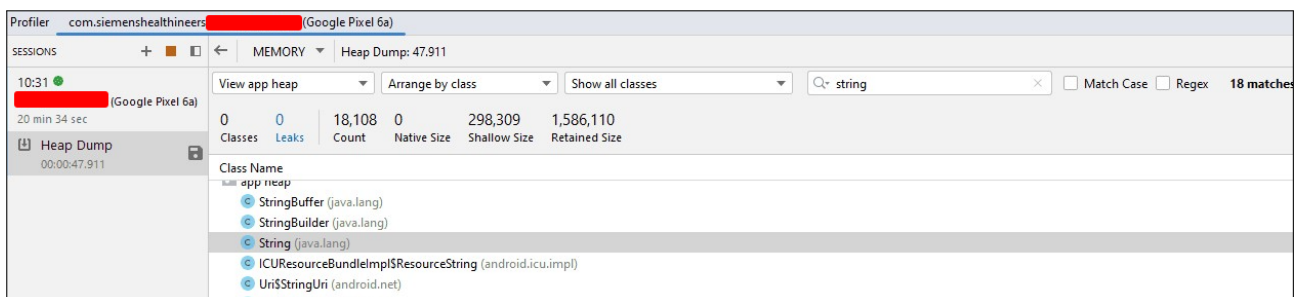
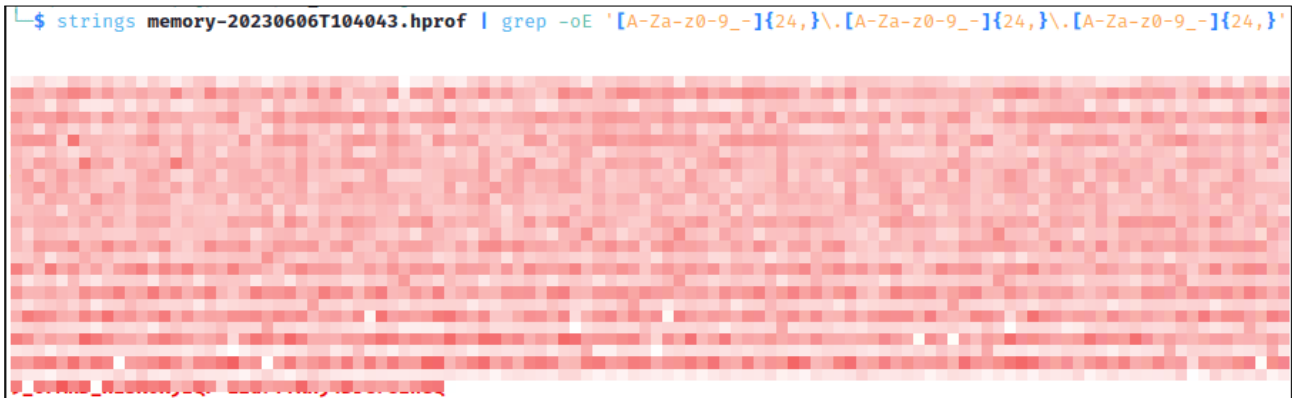


Figure 5.10: Capturing the memory dump



```
$ strings memory-20230606T104043.hprof | grep -oE '[A-Za-z0-9_-]{24,}\.[A-Za-z0-9_-]{24,}\.[A-Za-z0-9_-]{24,}'
```

Figure 5.11: Extracting a JWT from memory

## Countermeasures

As countermeasure, don't save JWT to variable in the `AppCacheTemplate` class. Instead, directly access JWT value from `FlutterSecureStorage` instance.

Review source code for storing sensitive data in memory. Substitute ordinary `String` objects with `byte[]`, which can be cleared from memory when no longer needed. However, it is important to ensure that the `byte[]` is securely encrypted and cleared from memory when no longer needed to prevent sensitive information from being disclosed.

## References

This finding references the following information sources:

- [CVSS 3.3](#)
- [Memory Profiler](#)
- MITRE. CWE-244: Improper Clearing of Heap Memory Before Release ('Heap Inspection'). <https://cwe.mitre.org/data/definitions/244.html>, 2021. [Online; accessed 1-December-2021]

## Finding 5: Outdated Components

ID: Dummy Project 1-FY23-5

<b>Criticality:</b>	<b>LOW</b>	<b>Category:</b>	Outdated Software
<b>Detectability:</b>	Easy	<b>Exploitability:</b>	Hard
<b>CWE / CVE:</b>	<b>CWE-1104</b>	<b>CVSS Score:</b>	2.5
<b>CVSS Vector:</b>	CVSS:3.1/AV:L/AC:H/PR:L/UI:N/S:U/C:L/I:N/A:N		
<b>Location:</b>	dummyapplication.apk, dummyapplication.ipa		
<b>Component:</b>	Multiple		

### Details

The application uses outdated third party components. Also the application uses API level 21 (Android 5.0) as minimum API level required for the application to run.

### Impact

Outdated third party components can compromise the security of an application, leading to data theft, business disruption, legal issues, and harm to reputation. Upgrading these packages and monitoring the system's security is necessary to avoid these consequences. The best practice is to use the latest versions of components.

By setting the minimum API level to 21, the application can be installed on an older version of Android that has multiple unfixed vulnerabilities. This increases the risk of known vulnerabilities in older Android versions being exploited by attackers, potentially compromising the security of the application and user data.

### Repeatability

Observe the set value `minSdkVersion` in the file `build.gradle` to check Android minimum API level (see Figure 5.12).

```
defaultConfig {
    applicationId "com.siemenshealthineers.██████████.developer"
    // You can update the following values to match your application needs.
    // For more information, see: https://docs.flutter.dev/deployment/android#reviewing-the-gradle-build-configuration.
    minSdkVersion 21
    targetSdkVersion 33
    versionCode 1
    versionName '1.0.0'
    multiDexEnabled true
}
```

Figure 5.12: Snippet of the `build.gradle` file

Observe the output of the command `flutter pub outdated` to check outdated third party components (see Figure 5.13).

```
PS: 06/05/2023 15:14:35>flutter pub outdated
Showing outdated packages.
[*] indicates versions that are not the latest available.
```

Package Name	Current	Upgradable	Resolvable	Latest
<b>direct dependencies:</b>				
collection	*1.17.0	-	*1.17.1	1.17.2
connectivity_plus	*3.0.4	-	4.0.1	4.0.1
device_info_plus	*8.2.0	-	9.0.2	9.0.2
dio	*5.1.1	-	5.1.2	5.1.2
file_picker	*5.2.10	-	*5.3.1	5.3.2
flutter_bloc	*8.1.2	-	8.1.3	8.1.3
flutter_local_notifications	*13.0.0	-	14.1.1	14.1.1
flutter_svg	*2.0.5	-	2.0.6	2.0.6
freezed	*2.3.3	-	2.3.5	2.3.5
go_router	*6.5.9	-	7.1.1	7.1.1
http	*0.13.6	-	*0.13.6	1.0.0
i18n_extension	*6.0.0	-	9.0.2	9.0.2
image_picker	*0.8.7+4	-	0.8.7+5	0.8.7+5
injectable	*2.1.1	-	2.1.2	2.1.2
local_auth_android	*1.0.28	-	1.0.30	1.0.30
local_auth_ios	*1.1.1	-	1.1.3	1.1.3
logging	*1.1.1	-	1.2.0	1.2.0
package_info_plus	*3.1.2	-	4.0.2	4.0.2
webview_flutter	*4.2.0	-	4.2.1	4.2.1
webview_flutter_android	*3.7.0	-	3.7.1	3.7.1
webview_flutter_wkwebview	*3.4.3	-	3.4.4	3.4.4
<b>dev_dependencies:</b>				
build_runner	*2.3.3	-	2.4.4	2.4.4
injectable_generator	*2.1.5	-	2.1.6	2.1.6
json_serializable	*6.6.2	-	6.7.0	6.7.0
mockito	*5.4.0	-	5.4.1	5.4.1

```
No resolution was found. Try running `flutter pub upgrade --dry-run` to explore why.

25 dependencies are constrained to versions that are older than a resolvable version.
To update these dependencies, edit pubspec.yaml, or run `flutter pub upgrade --major-versions`.
```

Figure 5.13: Output of the command `flutter pub outdated`

## Countermeasures

The following countermeasures are recommended:

- Review all used libraries for vulnerable versions and upgrade the outdated ones.
- Consider signing up to [Security Vulnerability Monitoring](#).
- Perform regular updates of used components.
- Increase Android minimum API level. Setting API level 29 (Android 10) is recommended.

## References

This finding references the following information sources:

- [Security Vulnerability Monitoring](#)
- [CVSS 2.5](#)
- MITRE. CWE-1104: Use of Unmaintained Third Party Components. <https://cwe.mitre.org/data/definitions/1104.html>, 2021. [Online; accessed 1-December-2021]

**Finding 6: DummyApplication Signed with a Debug Certificate***ID: Dummy Project 1-FY23-6*

<b>Criticality:</b>	INFO	<b>Category:</b>	Security Configuration
<b>Detectability:</b>	Difficult	<b>Exploitability:</b>	Hard
<b>CWE / CVE:</b>	<a href="#">CWE-296</a>	<b>CVSS Score:</b>	N/A
<b>CVSS Vector:</b>	N/A		
<b>Location:</b>	dummyapplication.apk		
<b>Component:</b>	META-INF/CERT.RSA		

**Details**

The `dummyapplication.apk` is signed with a debug certificate.

**Impact**

Debug certificates do not meet security standards of the release certificates.

## Repeatability

The `dummyapplication.apk` is signed with a debug certificate (CERT.RSA), which can be found in the META-INF folder. The certificate properties are shown in Figure 5.14.

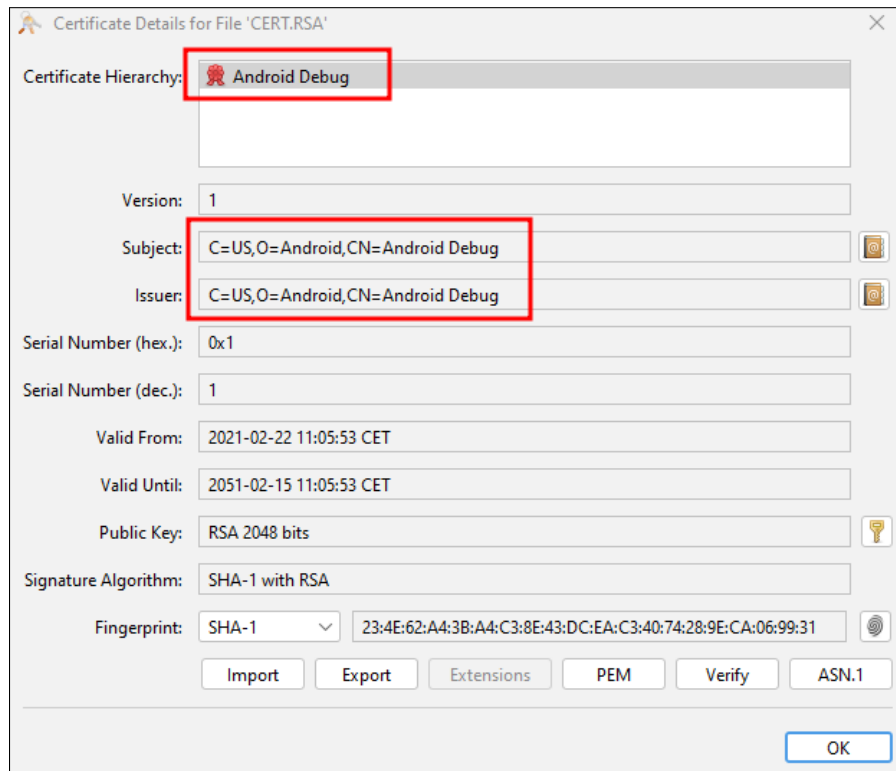


Figure 5.14: Android debug certificate properties

## Countermeasures

Make sure that release version of the application is signed with the organization certificate of appropriate RSA (2048-bit) and SHA-2 key sizes.

## References

This finding references the following information sources:

- Siemens Healthineers Guidance for Secure Software Architecture, Design and Development: 8.4 Code-Signing
- MITRE. CWE-296: Improper Following of a Certificate's Chain of Trust. <https://cwe.mitre.org/data/definitions/296.html>, 2021. [Online; accessed 1-December-2021]

## Finding 7: Missing Enforced Updating

*ID: Dummy Project 1-FY23-7*

<b>Criticality:</b>	INFO	<b>Category:</b>	Outdated Software
<b>Detectability:</b>	Easy	<b>Exploitability:</b>	Hard
<b>CWE / CVE:</b>	<a href="#">CWE-691</a>	<b>CVSS Score:</b>	N/A
<b>CVSS Vector:</b>	N/A		
<b>Location:</b>	dummyapplication.apk, dummyapplication.ipa		
<b>Component:</b>	Multiple		

### Details

The both applications, Android and iOS, are missing enforced updating, which assures running up-to-date and fully patched application.

### Impact

The missing enforced updating poses the risk of exploitation of known vulnerabilities, increases the attack surface of outdated app, and may result in compliance violations.

### Repeatability

Try to change version of the application to an older version. Rebuild the application and run it. User is able to interact with the older version of the application.

### Countermeasures

When the app is opened, check whether any new updates have been released for the application. If the app is outdated, do not allow the user to interact with the application until it is updated.

To check for new updates, the [AppUpdateManager](#) for Android applications can be used.

For iOS applications, the `http://itunes.apple.com/lookup?id=<BundleId>` API call can be used.



## References

This finding references the following information sources:

- [AppUpdateManager Documentation](#)
- [iTunes Search API](#)
- MITRE. CWE-691: Insufficient Control Flow Management. <https://cwe.mitre.org/data/definitions/691.html>, 2021. [Online; accessed 1-December-2021]

# 6 Scope and Procedures

## 6.1 Scope

In alignment with the Application Manager, the following components of **Dummy Project 1** have been tested during this assessment:

Components	Details
APK file	Android application
IPA file	iOS application
Source code	Static analysis

Table 6.1: Scope

## 6.2 Worst Case Scenarios

Below is a list of worst-case scenarios defined together with the **Application Manager**.

#	Worst-case scenario
WS1	Information leakage of personal /patient data/customer data
WS2	Modification or corruption of data
WS3	Unauthorized read/write access to application/database
WS4	Asset becomes partly or completely unavailable

Table 6.2: Worst-case scenarios

The following table contains all worst-case scenarios and findings and if they could be realized within the Penetration test:

Finding #	Description	WS1	WS2	WS3	WS4
1	ePHI is stored on device without encryption	•	•		
2	Sensitive Information Disclosure via Logging	•	•	•	
3	Weak Application Signature			•	•
4	Heap Inspection of Sensitive Memory	•	•	•	
5	Outdated Components				
6	DummyApplication Signed with a Debug Certificate				
7	Missing Enforced Updating				

Table 6.3: Findings case scenarios

## 6.3 Out Of Scope

In alignment with the Application Manager, the following components of **Dummy Project 1** were not tested during this assessment:

Components	Details
3rd party plugins	Plugins not developed by Siemens
Underlying operating systems	Android and iOS
REST APIs	Was already tested

Table 6.4: Out Of Scope

## 6.4 Environment

CYS CSA obtained following access methods to DummyApplication:

- Android APK file,
- iOS IPA file,
- application source code,
- test user credentials.

## 7 Testing Methodology

### 7.1 Tools Used

During the course of the penetration test, the following tools were utilized:

Tool	Version	Test Type	Work Type
adb	1.0.41	Android debugging	Bridge to Andorid device
Android Studio	2022.2.1	Android Development, Emulator	Official integrated development environment for Google's Android operating system, with emulator capabilities.
apktool	v2.7.0-dirty	Reverse Engineering	APK decompiler
Burp Suite Professional	2023.4.5	Automatic scan, Manual verification	Intercepting web traffic, request modification, etc.
Checkmarx	9.5.4.1000	Automated static analysis	Source code verification
Genymotion	3.4.0	Virtualization	Android emulator.
Jadx	1.4.7	Reverse Engineering	APK decompiler
MobSF	3.6.0	Static and Dynamic Analysis	Automated, all-in-one mobile application pen-testing, malware analysis and security assessment framework.
strings	2.40	Reverse Engineering	String finder in binary files

Table 7.1: Tools employed

## **7.2 Attack Vectors and Payload Types**

Tests on Dummy Project 1 included, but were not limited to:

- static analysis,
- file system analysis,
- debugging,
- workflow analysis,
- client-side testing,
- testing for weak cryptography,
- testing error handling.

## 8 Next Steps

### 8.1 Test Cleanup

Over the course of a security assessment it may be necessary to create testing accounts with the sole purpose of testing the various components of an *Asset*. Additionally, firewall rules may be modified to enable tester access to the various components of the *Asset*. These exceptions and testing accounts are no longer necessary after the end of an assessment and as such should be removed and/or revoked after testing has been completed. Note the following example scenarios:

- Code may be inserted into the application or server
- Escalation and/or modification of the user accounts
- Creation of additional user accounts within the application
- Modification of database content or other internal application information

In order to ensure that this penetration test will not negatively impact future developments, deployments or operations of the testing environment should be inspected and purged of all accounts and objects that have been tampered with or controlled by CYS CSA. Additionally, any exploits declared within this report should be inspected and addressed to ensure all payloads have been removed from the *Asset*.

### 8.2 Further Recommendations

This report contains a set of findings. Each finding describes a security issue found in the *Asset* along with a recommendation about possible countermeasures.

However, while fixing the current issues is important keep in mind that it is just a reactive patch and does not necessarily address the root cause. Root cause analysis answers why this security issue was introduced into the product or service in the first place and why it was not detected by standard testing during the development phase. Therefore, root cause analysis may reveal weaknesses in the development process. Unless remediated, these weaknesses could result in the same or similar security issues in future versions of the target of evaluation.

#### 8.2.1 Static Application Security Testing

Our security experts are dedicated to support you beyond the snapshot of security status as provided by this report. Many security issues are already introduced in the development phase and are prime targets for attackers, such as **Cross-Site Scripting (XSS)** vulnerabilities, **SQL Injection**, **Cross-Site Request Forgery (CSRF)**, **Buffer Overflow**, **Security Misconfigurations** and **Cryptographic Failures**.

**SAST** Service provides automated static source code analysis that enables you to find vulnerabilities in source code:

- Identification of thousands of known code vulnerabilities (SQL Injection, Cross-Site Scripting, Code Injection, Buffer Overflow, Unvalidated Input, Log Forgery, etc.)
- Ensures coverage of security standards (OWASP Top 10, SANS 25, CWE and more)
- Provides overview of GD41 compliance
- SDLC integration into CI/CD pipelines & plugins for IDEs
- To achieve maximum benefit from security testing, SAST should be utilized during the development phase, when the cost of fixing a security weakness is lower than in later stages of the product lifecycle (saves at least 50% remediation costs)
- With the support of SAST, developers are empowered to deliver secure code

E-mail: [SASTservice@siemens-healthineers.com](mailto:SASTservice@siemens-healthineers.com)



# Bibliography

- [1] Siemens Healthineers. SHS Infosec Policies. [https://intranet.for.healthineers.siemens.com/wll/0047/en/ISEQ/SHS%20Information%20Security%20Policies/SHS%20Information%20Security%20Controls/SHS\\_ISEC\\_C\\_01.pdf](https://intranet.for.healthineers.siemens.com/wll/0047/en/ISEQ/SHS%20Information%20Security%20Policies/SHS%20Information%20Security%20Controls/SHS_ISEC_C_01.pdf), 2020. [Intranet; accessed 1-June-2021].
- [2] MITRE. CWE-1104: Use of Unmaintained Third Party Components. <https://cwe.mitre.org/data/definitions/1104.html>, 2021. [Online; accessed 1-December-2021].
- [3] MITRE. CWE-244: Improper Clearing of Heap Memory Before Release ('Heap Inspection'). <https://cwe.mitre.org/data/definitions/244.html>, 2021. [Online; accessed 1-December-2021].
- [4] MITRE. CWE-296: Improper Following of a Certificate's Chain of Trust. <https://cwe.mitre.org/data/definitions/296.html>, 2021. [Online; accessed 1-December-2021].
- [5] MITRE. CWE-328: Use of Weak Hash. <https://cwe.mitre.org/data/definitions/328.html>, 2021. [Online; accessed 1-December-2021].
- [6] MITRE. CWE-359: Exposure of Private Personal Information to an Unauthorized Actor. <https://cwe.mitre.org/data/definitions/359.html>, 2021. [Online; accessed 1-December-2021].
- [7] MITRE. CWE-532: Insertion of Sensitive Information into Log File. <https://cwe.mitre.org/data/definitions/532.html>, 2021. [Online; accessed 1-December-2021].
- [8] MITRE. CWE-691: Insufficient Control Flow Management. <https://cwe.mitre.org/data/definitions/691.html>, 2021. [Online; accessed 1-December-2021].

# A Appendix A

## A.1 Criticality Levels

Each finding in the Penetration Test Report is associated with a certain criticality level. The criticality level is an estimation of the potential impact of a finding and the likelihood of its exploitation. Furthermore, laws on data privacy, Siemens Information Security Regulations [1] and information security best practices are incorporated into the rating process.

The following criticality levels are used in this Penetration Test Report:

Severity	CVSS Score	Description
<b>Critical</b>	9.0 - 10.0	Critical findings can be readily compromised with publicly available malware or exploits.
<b>High</b>	7.0 - 8.9	Vulnerabilities that can lead to worst case scenarios in a very direct way without too complex preconditions, e.g. a missing security patch that allows taking over an operating system, an SQL Injection that allows direct access to the database or a privilege escalation to an administrative account.
<b>Medium</b>	4.0 - 6.9	Vulnerabilities that might trigger worst case scenarios in an indirect way or that might only work under certain circumstances, e.g. a cross-side request- forgery-attack (CSRF) where an attacker needs to send an email to a certain person, this person needs to click on a link in this email and this person already needs to be logged into a certain application to trigger a command execution in an application (phishing attacks).
<b>Low</b>	0.1 - 3.9	Vulnerabilities that are neither directly, nor indirectly exploitable but increase the likelihood or impact of another vulnerability. Additionally, vulnerabilities are classified low if they require highly advanced hacking skills or very complex preconditions and, hence, the likelihood of exploitation is extremely low, or the impact is minimal. Examples are error message revealing software version numbers or internal path information.
<b>Information</b>	N/A	Additional observations or notes from a security perspective: observations of problematic behavior for which no clear evidence could be found, or which do not pose a direct security risk, but should still be reviewed by the asset owner.

Table A.1: Criticality Levels

## A.2 Overall Threat Exposure

The “**Overall Threat Exposure**” determines the current security state of the *Asset*. The overall criticality is based on two key factors:

- Criticality of the findings
- Exploitation of defined worst-case scenarios

The following triggers determine the criticality of the “Overall Threat Exposure”:

Threat Exposure	Trigger
<b>Critical</b>	At least one <b>Critical</b> finding.
<b>High</b>	At least one <b>High</b> finding <i>OR/AND</i> At least one worst-case scenario triggered by a <b>High</b> or <b>Medium</b> finding.
<b>Medium</b>	No High findings. No worst-case scenario triggered. At least one Medium finding.
<b>Low</b>	No High or Medium findings. No worst-case scenarios triggered. Only Low findings.

Table A.2: Overall Threat Exposure

In the case of a worst-case scenario being triggered by a **Low** finding and no other findings higher than **Low** being found, the overall threat exposure might be rated higher than **Low**. This will be determined on a case-by-case basis.

## A.3 Testing Approaches

**Manual Testing** Manual testing is done by professional security experts. It requires collection of security data about the *Asset* manually in order to identify vulnerabilities and exploit them.

**Automatic Scans** Automatic scans are done by in-house and third-party tools and are much faster than manual testing. The testing software performs automatic actions, which would otherwise be very time consuming for a security expert. The scan then generates a report with information about every finding that was found during the scan.

**Manual Verification** Automatic scans produce a lot of results with many false positives/false negatives. Therefore, a manual check by a professional security expert is required to verify the results and eliminate all false positives/negatives.

## A.4 Test Protocol

**Target:** https://TODOTODO

OWASP Control	OWASP Testing Method	Result	Comment
OTG-INFO-001	Conduct Search Engine Discovery and Re-connaissance for Information Leakage	TODOTODO	
OTG-INFO-002	Fingerprint Web Server	TODOTODO	
OTG-INFO-003	Review Webserver Metafiles for Information Leakage	TODOTODO	
OTG-INFO-004	Enumerate Applications on Webserver	TODOTODO	
OTG-INFO-005	Review Webpage Comments and Metadata for Information Leakage	TODOTODO	
OTG-INFO-006	Identify application entry points	TODOTODO	
OTG-INFO-007	Map execution paths through application	TODOTODO	
OTG-INFO-008	Fingerprint Web Application Framework	TODOTODO	
OTG-INFO-009	Fingerprint Web Application	TODOTODO	
OTG-INFO-010	Map Application Architecture	TODOTODO	
OTG-CONFIG-001	Test Network/Infrastructure Configuration	TODOTODO	
OTG-CONFIG-002	Test Application Platform Configuration	TODOTODO	
OTG-CONFIG-003	Test File Extensions Handling for Sensitive Information	TODOTODO	
OTG-CONFIG-004	Review Old Backup and Unreferenced Files for Sensitive Information	TODOTODO	
OTG-CONFIG-005	Enumerate Infrastructure and Application Admin Interfaces	TODOTODO	
OTG-CONFIG-006	Test HTTP Methods	TODOTODO	

OTG-CONFIG-007	Test HTTP Strict Transport Security	TODOTODO	
OTG-CONFIG-008	Test RIA cross domain policy	TODOTODO	
OTG-IDENT-001	Test Role Definitions	TODOTODO	
OTG-IDENT-002	Test User Registration Process	TODOTODO	
OTG-IDENT-003	Test Account Provisioning Process	TODOTODO	
OTG-IDENT-004	Testing for Account Enumeration and Guessable User Account	TODOTODO	
OTG-IDENT-005	Testing for Weak or unenforced username policy	TODOTODO	
OTG-AUTHN-001	Testing for Credentials Transported over an Encrypted Channel	TODOTODO	
OTG-AUTHN-002	Testing for default credentials	TODOTODO	
OTG-AUTHN-003	Testing for Weak lock out mechanism	TODOTODO	
OTG-AUTHN-004	Testing for bypassing authentication schema	TODOTODO	
OTG-AUTHN-005	Test remember password functionality	TODOTODO	
OTG-AUTHN-006	Testing for Browser cache weakness	TODOTODO	
OTG-AUTHN-007	Testing for Weak password policy	TODOTODO	
OTG-AUTHN-008	Testing for Weak security question/answer	TODOTODO	
OTG-AUTHN-009	Testing for weak password change or reset functionalities	TODOTODO	
OTG-AUTHN-010	Testing for Weaker authentication in alternative channel	TODOTODO	
OTG-AUTHZ-001	Testing Directory traversal/file include	TODOTODO	
OTG-AUTHZ-002	Testing for bypassing authorization schema	TODOTODO	
OTG-AUTHZ-003	Testing for Privilege Escalation	TODOTODO	

OTG-AUTHZ-004	Testing for Insecure Direct Object References	TODOTODO	
OTG-SESS-001	Testing for Bypassing Session Management Schema	TODOTODO	
OTG-SESS-002	Testing for Cookies attributes	TODOTODO	
OTG-SESS-003	Testing for Session Fixation	TODOTODO	
OTG-SESS-004	Testing for Exposed Session Variables	TODOTODO	
OTG-SESS-005	Testing for Cross Site Request Forgery (CSRF)	TODOTODO	
OTG-SESS-006	Testing for logout functionality	TODOTODO	
OTG-SESS-007	Test Session Timeout	TODOTODO	
OTG-SESS-008	Testing for Session puzzling	TODOTODO	
OTG-INPVAL-001	Testing for Reflected Cross Site Scripting	TODOTODO	
OTG-INPVAL-002	Testing for Stored Cross Site Scripting	TODOTODO	
OTG-INPVAL-003	Testing for HTTP Verb Tampering	TODOTODO	
OTG-INPVAL-004	Testing for HTTP Parameter pollution	TODOTODO	
OTG-INPVAL-005	Testing for SQL Injection	TODOTODO	
OTG-INPVAL-006	Testing for LDAP Injection	TODOTODO	
OTG-INPVAL-007	Testing for ORM Injection	TODOTODO	
OTG-INPVAL-008	Testing for XML Injection	TODOTODO	
OTG-INPVAL-009	Testing for SSI Injection	TODOTODO	
OTG-INPVAL-010	Testing for XPath Injection	TODOTODO	
OTG-INPVAL-011	IMAP/SMTP Injection	TODOTODO	
OTG-INPVAL-012	Testing for Code Injection	TODOTODO	

OTG-INPVAL-013	Testing for Command Injection	TODOTODO	
OTG-INPVAL-014	Testing for Buffer overflow	TODOTODO	
OTG-INPVAL-015	Testing for incubated vulnerabilities	TODOTODO	
OTG-INPVAL-016	Testing for HTTP Splitting/Smuggling	TODOTODO	
OTG-ERR-001	Analysis of Error Codes	TODOTODO	
OTG-ERR-002	Analysis of Stack Traces	TODOTODO	
OTG-CRYPST-001	Testing for Weak SSL/TLS Ciphers Insufficient Transport Layer Protection	TODOTODO	
OTG-CRYPST-002	Testing for Padding Oracle	TODOTODO	
OTG-CRYPST-003	Testing for Sensitive information sent via un-encrypted channels	TODOTODO	
OTG-BUSLOGIC-001	Test Business Logic Data Validation	TODOTODO	
OTG-BUSLOGIC-002	Test Ability to Forge Requests	TODOTODO	
OTG-BUSLOGIC-003	Test Integrity Checks	TODOTODO	
OTG-BUSLOGIC-004	Test for Process Timing	TODOTODO	
OTG-BUSLOGIC-005	Test Number of Times a Function Can be Used Limits	TODOTODO	
OTG-BUSLOGIC-006	Testing for the Circumvention of Work Flows	TODOTODO	
OTG-BUSLOGIC-007	Test Defenses Against Application Misuse	TODOTODO	
OTG-BUSLOGIC-008	Test Upload of Unexpected File Types	TODOTODO	
OTG-BUSLOGIC-009	Test Upload of Malicious Files	TODOTODO	
OTG-CLIENT-001	Testing for DOM based Cross Site Scripting	TODOTODO	
OTG-CLIENT-002	Testing for JavaScript Execution	TODOTODO	
OTG-CLIENT-003	Testing for HTML Injection	TODOTODO	



OTG-CLIENT-004	Testing for Client-Side URL Redirect	TODOTODO	
OTG-CLIENT-005	Testing for CSS Injection	TODOTODO	
OTG-CLIENT-006	Testing for Client-Side Resource Manipulation	TODOTODO	
OTG-CLIENT-007	Test Cross Origin Resource Sharing	TODOTODO	
OTG-CLIENT-008	Testing for Cross Site Flashing	TODOTODO	
OTG-CLIENT-009	Testing for Clickjacking	TODOTODO	
OTG-CLIENT-010	Testing WebSockets	TODOTODO	
OTG-CLIENT-011	Test Web Messaging	TODOTODO	
OTG-CLIENT-012	Test Local Storage	TODOTODO	

Table A.3: OWASP Testing Guide v4