Supporting Document Mandatory Technical Document: Evaluation Activities for collaborative Protection Profile for Application Software

Foreword

This is a Supporting Document, intended to complement the Common Criteria (CC) version 3 and the associated Common Evaluation Methodology for Information Technology Security Evaluation.

Supporting Documents may be "Guidance Documents", that highlight specific approaches and application of the standard to areas where no mutual recognition of its application is required, and as such, are not of normative nature, or "Mandatory Technical Documents", whose application is mandatory for evaluations whose scope is covered by that of the Supporting Document. The usage of the latter class is not only mandatory, but certificates issued as a result of their application are recognized under the CCRA.

This Supporting Document has been developed by the iTC for Application Software iTC and is designed to be used to support the evaluations of TOEs against the cPP identified in Section 1.1, “Technology Area and Scope of Supporting Document”.

Acknowledgements

This Supporting Document was developed by the iTC for Application Software international Technical Community with representatives from industry, Government agencies, Common Criteria Test Laboratories, and members of academia.

Technical Editor

Revision History

Table 1. Revision history

Version	Date	Description
1.0	2022-04-06	Initial release.
1.1	2023-08-16	Incorporated feedback recevied following initial release.

General Purpose

See Section 1.1, “Technology Area and Scope of Supporting Document”.

Field of Special Use

This Supporting Document applies to the evaluation of TOEs claiming conformance with the collaborative Protection Profile for Application Software.

Table of Contents

1. Introduction

1.1. Technology Area and Scope of Supporting Document

This Supporting Document (SD) is mandatory for evaluations of products that claim conformance to any of the following cPP(s):

- collaborative Protection Profile for Application Software, Version 1.1, 2023-08-16

Although Evaluation Activities (EAs) are defined mainly for the evaluators to follow, in general they will also help developers prepare for evaluation by identifying specific requirements for their Target of Evaluation (TOE). The specific requirements in EAs may in some cases clarify the meaning of Security Functional Requirements (SFRs), and may identify particular requirements for the content of Security Targets (especially the TOE Summary Specification), user guidance documentation, and possibly required supplementary information (e.g. for entropy analysis or cryptographic key management architecture).

1.2. Structure of the Document

Evaluation Activities can be defined for both SFRs and Security Assurance Requirements (SARs). These are defined in separate sections of this SD. The EAs associated with the SFRs are considered to be interpretations of applying the appropriate SAR activity. For instance, activities associated with testing are representative of what is required by ATE_IND.1.

If any Evaluation Activity cannot be successfully completed in an evaluation then the overall verdict for the evaluation is a ‘fail’. In rare cases, there may be acceptable reasons why an Evaluation Activity may be modified or deemed not applicable for a particular TOE, but this must be agreed with the Certification Body for the evaluation.

In general, if all EAs (for both SFRs and SARs) are successfully completed in an evaluation then it would be expected that the overall verdict for the evaluation is a ‘pass’.

In some cases, the Common Evaluation Methodology (CEM) work units have been interpreted to require the evaluator to perform specific EAs. In these instances, EAs will be specified in Section 2, “Evaluation Activities for Mandatory SFRs”, Section 5, “Evaluation Activities for Objective Requirements”, and possibly Section 3, “Evaluation Activities for Optional Requirements” and Section 4, “Evaluation Activities for Selection-Based Requirements”. In cases where there are no CEM interpretations, the CEM activities are to be used to determine if SARs are satisfied and references to the CEM work units are identified as being the sole EAs to be performed.

Finally, there are cases where EAs have rephrased CEM work units to provide clarity on what is required. The EAs are reworded for clarity and interpret the CEM work units such that they will result in more objective and repeatable actions by the evaluator. In these cases, the EA supplements the CEM work unit. These EAs will be specified in Section 6, “Evaluation Activities for SARs”.

2. Evaluation Activities for Mandatory SFRs

The EAs presented in this section capture the actions the evaluator performs to address technology

specific aspects covering specific SARs (e.g., ASE_TSS.1, ADV_FSP.1, AGD_OPE.1, and ATE_IND.1) – this is in addition to the CEM work units that are performed in Section 6, "Evaluation Activities for SARs".

Regarding design descriptions (designated by the subsections labelled TSS, as well as any required supplementary material that may be treated as proprietary), the evaluator must ensure there is specific information that satisfies the EA. For findings regarding the TSS section, the evaluator's verdicts will be associated with the CEM work unit ASE_TSS.1-1. Evaluator verdicts associated with the supplementary evidence will also be associated with ASE_TSS.1-1, since the requirement to provide such evidence is specified in ASE in the cPP.

For ensuring the guidance documentation provides sufficient information for the administrators/users as it pertains to SFRs, the evaluator's verdicts will be associated with CEM work units ADV_FSP.1-7, AGD_OPE.1-4, and AGD_OPE.1-5.

Finally, the subsection labelled Tests is where the iTC has determined that testing of the product in the context of the associated SFR is necessary. While the evaluator is expected to develop tests, there may be instances where it is more practical for the developer to construct tests, or where the developer may have existing tests. Therefore, it is acceptable for the evaluator to witness developer-generated tests in lieu of executing the tests. In this case, the evaluator must ensure the developer's tests are executing both in the manner declared by the developer and as mandated by the EA. The CEM work units that are associated with the EAs specified in this section are: ATE_IND.1-3, ATE_IND.1-4, ATE_IND.1-5, ATE_IND.1-6, and ATE_IND.1-7.

2.1. Cryptographic Support (FCS)

2.1.1. Random Bit Generation Services (FCS_RBG)

2.1.1.1. FCS_RBG_EXT.1

2.1.1.1.1. TSS

[Conditional] If **use no DRBG functionality** is selected, the evaluator shall inspect the application and its developer documentation and verify that the application needs no random bit generation services.

[Conditional] If **invoke platform-provided DRBG functionality** is selected, the evaluator shall examine the TSS to confirm that it identifies all functions (as described by the SFRs included in the ST) that obtain random numbers from the platform RBG. The evaluator shall determine that for each of these functions, the TSS states which platform interface (API) is used to obtain the random numbers. The evaluator shall confirm that each of these interfaces corresponds to the acceptable interfaces listed for each platform below in Test activities.

[Conditional] If **implement DRBG functionality** is selected, the evaluator shall ensure that FCS_RBG_EXT.2 is included in the ST.

2.1.1.1.2. Operational Guidance

No activities specified.

2.1.1.1.3. Test

[Conditional] If **invoke platform-provided DRBG functionality** is selected the evaluator shall decompile the application binary using an decompiler suitable for the application (TOE). The evaluator shall search the output of the decompiler to determine that, for each API listed in the TSS, that API appears in the output. If the representation of the API does not correspond directly to the strings in the following list, the evaluator shall provide a mapping from the decompiled text to its corresponding API, with a description of why the API text does not directly correspond to the decompiled text and justification that the decompiled text corresponds to the associated API.

It should be noted that there is no expectation that the evaluators attempt to confirm that the APIs are being used “correctly” for the functions identified in the TSS; the activity is to list the used APIs and then do an existence check via decompilation.

The following are the per-platform list of acceptable APIs:

Test 1: [conditional] The evaluator shall verify that rand_s, RtlGenRandom, BCryptGenRandom, or CryptGenRandom API is used for classic desktop applications. The evaluator shall verify the application uses the RNGCryptoServiceProvider class or derives a class from System.Security.Cryptography.RandomNumberGenerator API for Windows Universal Applications. It is only required that the API is called/invoked, there is no requirement that the API be used directly. In future versions of this document, CryptGenRandom may be removed as an option as it is no longer the preferred API per vendor documentation.

Test 2: [conditional] For Linux platform, the evaluator shall verify that the application collects random from /dev/random or /dev/urandom.

Test 3: [conditional] The evaluator shall verify that the application invokes either CCRandomGenerateBytes or CCRandomCopyBytes, or collects random from /dev/random.

If invocation of platform-provided functionality is achieved in another way, the evaluator shall ensure the TSS describes how this is carried out, and how it is equivalent to the methods listed here (e.g. higher-level API invokes identical low-level API).

2.1.2. Storage of Credentials (FCS_STO_EXT)

2.1.2.1. FCS_STO_EXT.1

2.1.2.1.1. TSS

The evaluator shall check the TSS to ensure that it lists all persistent credentials (secret keys, PKI private keys, or passwords) needed to meet the requirements in the ST. For each of these items, the evaluator shall confirm that the TSS lists for what purpose it is used, and how it is stored.

[Conditional] If implement functionality to securely store is selected, the evaluator shall verify that the TSS states how credentials are stored utilizing FCS_COP.1/DataEncryption, FCS_CKM_EXT.1/Hash, FCS_CKM_EXT.1/KeyedHash, or FCS_CKM_EXT.1/PBKDF2.

2.1.2.1.2. Operational Guidance

No activities specified.

2.1.2.1.3. Test

[Conditional] **If invoke the functionality provided by the platform to securely store** is selected, the evaluator shall perform the following actions which vary per platform.

Test 1: [conditional] For Windows platform the evaluator shall verify that all certificates are stored in the Windows Certificate Store. The evaluator shall verify that other credentials, like passwords, are stored in the Windows Credential Manager or stored using the Data Protection API (DPAPI). For Windows Universal Applications, the evaluator shall verify that the application is using the ProtectData class and storing credentials in IsolatedStorage.

Test 2: [conditional] For Linux platform the evaluator shall verify that all keys are stored using Linux keyrings.

Test 3: [conditional] For macOS platform the evaluator shall verify that all credentials are stored within Keychain.

2.2. User Data Protection (FDP)

2.2.1. Network Communications (FDP_NET_EXT)

2.2.1.1. FDP_NET_EXT.1

2.2.1.1.1. TSS

The evaluator shall check the TSS and verify that for each connection, inbound and outbound, the protocols and ports used have been listed.

2.2.1.1.2. Operational Guidance

No activities specified.

2.2.1.1.3. Test

The evaluator shall perform the following test:

Test 1: [conditional] If **outbound connections** is selected then the evaluator shall run the application. While the application is running, the evaluator shall sniff network traffic ignoring all non-application associated traffic and verify that any network communications witnessed are documented in the TSS or are user-initiated.

Test 2: [conditional] If **inbound connections** is selected then the evaluator shall run the application. After the application initializes, the evaluator shall verify that any ports opened by the application have been captured in the TSS. This includes connection-based protocols (e.g. TCP) as well as connectionless protocols (e.g. UDP).

Test 3: [conditional] If **no network communication** is selected then the evaluator shall initialize and run the application. The evaluator shall verify that no ports are opened by the application and no other connectivity is observed.

2.3. Security Management (FMT)

2.3.1. Default Configuration (FMT_CFG)

2.3.1.1. FMT_CFG_EXT.1.1

2.3.1.1.1. TSS

The evaluator shall check the TSS to determine if the application requires any type of credentials and if the application installs with default credentials. If the TSF doesn't support credentials, the TSS will shall document this.

2.3.1.1.2. Operational Guidance

The evaluator shall check the Guidance documentation to check if any default credentials are provided and description of how they are changed at installation or before the application is operational.

2.3.1.1.3. Test

If the application uses any default credentials the evaluator shall run the following tests:

Test 1: [conditional] If default credentials are required to be changed **during installation** the evaluator shall install the application and verify that the application requires that the default credentials are changed.

Test 2: [conditional] If default credentials are required to be changed **before application is operational** the evaluator shall install and run the application without generating or loading new credentials and verify that only the minimal application functionality required to set new credentials is available.

Test 3: The evaluator shall run the application after establishing new credentials and verify that the original default credentials no longer provide access to the application.

2.3.1.2. FMT_CFG_EXT.1.2

2.3.1.2.1. TSS

No activities specified

2.3.1.2.2. Operational Guidance

No activities specified

2.3.1.2.3. Test

The evaluator shall install and run the application. The evaluator shall inspect the filesystem of the platform (to the extent possible) for any files created by the application and ensure that their permissions are adequate to protect them. The method of doing so varies per platform.

Test 1: [conditional] If the application is being tested on Windows, the evaluator shall run the

SysInternals tools, Process Monitor and Access Check (or tools of equivalent capability, like icacls.exe) for Classic Desktop applications to verify that files written to disk during an application's installation have the correct file permissions, such that a standard user cannot modify the application or its data files. For Windows Universal Applications the evaluator shall consider the requirement met because of the AppContainer sandbox.

Test 2: [conditional] If the application is being tested on Linux, the evaluator shall run the command find -L . -perm /002 inside the application's data directories to ensure that all files are not world-writable. The command should not print any files.

Test 3: [conditional] If the application is being tested on macOS, the evaluator shall run the command find . -perm +002 inside the application's data directories to ensure that all files are not world-writable. The command should not print any files.

2.3.2. Specification of Management Functions (FMT_SMF)

2.3.2.1. FMT_SMF.1

2.3.2.1.1. TSS

No activities specified

2.3.2.1.2. Operational Guidance

The evaluator shall verify that every management function specified in the SFR is described in the operational guidance. If multiple management interfaces are supported, the guidance documentation must describe which interfaces may be used to perform the management functions.

2.3.2.1.3. Test

The evaluator shall perform the following test:

Test 1: For each selection in FMT_SMF.1, the evaluator shall configure the TOE for that option. The evaluator shall then verify that the TOE does or does not transmit sensitive information/PII, as appropriate for the configuration specified. Each function should be tested on each management interface on which the functionality is supported.

Test 2: [conditional] If **other management functions** are specified, the evaluator shall test the application's ability to provide each management function by configuring the application and testing each function specified. The evaluator is expected to test these functions in all the ways in which the ST and guidance documentation state the configuration can be managed. Each function should be tested on each management interface on which the functionality is supported.

2.3.3. Management Roles (FMT_SMR)

2.3.3.1. FMT_SMR.2 (Security Management Roles)

2.3.3.1.1. TSS

The evaluator shall examine the TSS to determine that it details the TOE supported roles and any

restrictions of the roles involving administration of the TOE.

2.3.3.1.2. Guidance Documentation

The evaluator shall review the guidance documentation to ensure that it contains instructions for administering the TOE.

2.3.3.1.3. Test

While performing the testing activities for the evaluation, the evaluator shall ensure that each supported method of administering the TOE that conforms to the requirements of this cPP be tested.

2.4. Protection of the TSF (FPT)

2.4.1. Anti-Exploitation Capabilities (FPT_AEX_EXT)

2.4.1.1. FPT_AEX_EXT.1.1

2.4.1.1.1. TSS

The evaluator shall ensure that the TSS describes the compiler flags used to enable ASLR when the application is compiled. If no compiler flags are required to be set (the default behaviour satisfies the SFR), this shall be noted in the TSS.

2.4.1.1.2. Operational Guidance

No activities specified

2.4.1.1.3. Test

The evaluator shall perform either a static or dynamic analysis to determine that no memory mappings are placed at an explicit and consistent address. The method of doing so varies per platform.

Test 1: [conditional] If the application is being tested on Windows, the evaluator shall run the same application twice on the same system and run a tool that will list all memory mapped addresses for the application. The evaluator shall then verify the two different instances share no mapping locations. The Microsoft sysinternals tool, VMMap, could be used to view memory addresses of a running application. The evaluator shall use a tool such as Microsoft's BinScope Binary Analyzer to confirm that the application has ASLR enabled.

Test 2: [conditional] If the application is being tested on Linux, the evaluator shall run the same application twice and then compare their memory maps using pmap -x PID to ensure the two different instances share no mapping locations.

Test 3: [conditional] If the application is being tested on macOS, the evaluator shall run the same application twice and then compare their memory maps using vmmap PID to ensure the two different instances share no mapping locations

2.4.1.2. FPT_AEX_EXT.1.2

2.4.1.2.1. TSS

No activities specified

2.4.1.2.2. Operational Guidance

No activities specified

2.4.1.2.3. Test

The evaluator shall verify that no memory mapping requests are made with write and execute permissions. The method of doing so varies per platform.

Test 1: [conditional] If the application is being tested on Windows, the evaluator shall use a tool such as Microsoft's BinScope Binary Analyzer to confirm that the application passes the NXCheck. The evaluator may also ensure that the /NXCOMPAT flag was used during compilation to verify that DEP protections are enabled for the application.

Test 2: [conditional] If the application is being tested on Linux, the evaluator shall perform static analysis on the application to verify that both

- mmap is never be invoked with both the PROT_WRITE and PROT_EXEC permissions, and
- mprotect is never invoked with the PROT_EXEC permission.

Test 3: [conditional] If the application is being tested on macOS, the evaluator shall perform static analysis on the application to verify that mprotect is never invoked with the PROT_EXEC permission.

2.4.1.3. FPT_AEX_EXT.1.3

2.4.1.3.1. TSS

No activities specified

2.4.1.3.2. Operational Guidance

No activities specified

2.4.1.3.3. Test

The evaluator shall configure the platform in the ascribed manner and carry out one of the prescribed tests:

Test 1 [conditional]: If the application is being tested on Windows which supports Windows Defender Exploit Guard (Windows 10 version 1709 or later), then the evaluator shall ensure that the application can run successfully with Windows Defender Exploit Guard Exploit Protection configured with the following minimum mitigations enabled; Control Flow Guard (CFG), Randomize memory allocations (Bottom-Up ASLR), Export address filtering (EAF), Import address filtering (IAF), and Data Execution Prevention (DEP).

If the application is being tested on Windows which only supports the Enhanced Mitigation Experience Toolkit (EMET) (can be installed on Windows 10 version 1703 and earlier), then the evaluator shall ensure that the application can run successfully with EMET configured with the following minimum mitigations enabled; Memory Protection Check, Randomize memory allocations (Bottom-Up ASLR), Export address filtering (EAF), and Data Execution Prevention (DEP).

Test 2 [conditional]: If the application is being tested on Linux, the evaluator shall ensure that the application can successfully run on a system with either SELinux (or equivalent platform vendor recommended security features) enabled and in enforce mode.

Test 3 [conditional]: If the application is being tested on macOS, the evaluator shall ensure that the application can successfully run without disabling any OS security functions.

2.4.1.4. FPT_AEX_EXT.1.4

2.4.1.4.1. TSS

No activities specified

2.4.1.4.2. Operational Guidance

No activities specified

2.4.1.4.3. Test

Test 1: The evaluator shall run the application and determine where it writes its files. For files where the user does not choose the destination, the evaluator shall check whether the destination directory contains executable files.

Test 2: The evaluator shall run the application, mimicking normal usage, and note where all files are written. The evaluator shall ensure that there are no executable files stored in the same directories to which the application wrote and no data files in the application's install directory.

2.4.1.5. FPT_AEX_EXT.1.5

2.4.1.5.1. TSS

The evaluator shall ensure that the TSS section of the ST describes the compiler flag used to enable stack-based buffer overflow protection in the application. The following flags should be used depending on the compiler:

Table 2. Mapping between Compilers and Compiler Flags

Compiler	**Flag**
Visual Studio	GS
GCC or Xcode	-fstack-protector-all **(preferred)** -fstack-protector-strong
clang	-fsanitize=address

Windows Applications that run as Managed Code in the .NET Framework do not require these stack protections. Applications developed in Object Pascal using the Delphi IDE compiled with RangeChecking enabled comply with this element.

2.4.1.5.2. Operational Guidance

No activities specified

2.4.1.5.3. Test

Test 1: [conditional] If the application is evaluated on Windows platform, evaluator shall run a tool, like BinScope, that can verify the correct usage of /GS

The evaluator will inspect every native executable included in the TOE to ensure that stack-based buffer overflow protection is present.

Test 2: [conditional] If the application is PE type, the evaluator will disassemble each and ensure the following sequence appears:

```
mov rcx, QWORD PTR [rsp+(...)]
xor rcx, (...)
call (...)
```

Test 3: [conditional] If the application contains ELF executables, the evaluator will ensure that each contains references to the symbol **__stack_chk_fail**.

The test is considered passing as long as at least one instance of the above mentioned sequence/symbol is found in each compiled binary that makes up the TOE.

Tools such as Canary Detector (https://github.com/commoncriteria/canary-detector) may help automate these activities.

2.4.2. Trusted Update (FPT_TUD_EXT.1)

2.4.2.1. FPT_TUD_EXT.1.1

2.4.2.1.1. TSS

No activities specified

2.4.2.1.2. Operational Guidance

The evaluator must verify that the operational user guidance (AGD_OPE.1) identifies the method to query the current version of the application.

2.4.2.1.3. Test

The evaluator shall query the application for the current version of the software and verify that the current version matches that of the documented and installed version.

2.4.2.2. FPT_TUD_EXT.1.2

2.4.2.2.1. TSS

The evaluator shall verify that the TSS identifies:

- How the application installation package and updates to it are signed by an authorized source.
- The definition of an authorized source.

2.4.2.2.2. Operational Guidance

The evaluator shall verify that the TOE operational guidance documentation covers installation and update procedures for the application.

2.4.2.2.3. Test

Test 1: [conditional] If the TOE verifies the digital signature of the installation package and its updates, the evaluator shall obtain or produce illegitimate updates as defined below, and attempt to install them. The evaluator shall verify that the updates are rejected for all illegitimate updates. The evaluator shall perform this test using all of the following forms of illegitimate updates:

1. A modified version (e.g. using a hex editor) of a legitimately signed update
2. An update that has not been signed
3. An update signed with an invalid signature (e.g. by using a different key as expected for creating the signature or by manual modification of a legitimate signature)

2.5. Trusted Channels (FTP)

2.5.1. Data in Transit (FTP_DIT)

2.5.1.1. FTP_DIT_EXT.1

2.5.1.1.1. TSS

The evaluator shall verify that the TSS describes whether the application transmits any data over a network. If data is transmitted, the TSS shall identify whether all data or only sensitive data is transmitted. It must also identify types of sensitive data transmitted.

2.5.1.1.2. Operational Guidance

No activities specified

2.5.1.1.3. Test

Test 1: [conditional] The evaluator shall exercise the application (attempting to transmit data; for example by connecting to remote systems or websites) while capturing packets from the application. Based on the selection in the ST, the evaluator shall verify from the packet capture that the traffic is encrypted with HTTPS, TLS, DTLS or SSH.

Test 2: [conditional] The evaluator shall exercise the application (attempting to transmit data; for

example by connecting to remote systems or websites) while capturing packets from the application. The evaluator shall review the packet capture and verify that no sensitive data is transmitted in the clear.

Test 3: [conditional] If credentials are transmitted the evaluator shall set the credential to a known value. The evaluator shall capture packets from the application while causing credentials to be transmitted as described in the TSS. The evaluator shall perform a string search of the captured network packets and verify that the plaintext credential previously set by the evaluator is not found.

3. Evaluation Activities for Optional Requirements

3.1. Cryptographic Support (FCS)

3.1.1. Cryptographic Key Management (FCS_CKM)

3.1.1.1. FCS_CKM_EXT.1/Symmetric Cryptographic Key Generation

3.1.1.1.1. TSS

The evaluator shall review the TSS to determine that it describes how the functionality described by FCS_RBG_EXT.1 is invoked. The TSS shall identify the key generation algorithms, key sizes, and usage.

If the application is relying on random bit generation from the host platform, the evaluator shall verify the TSS includes the name/manufacturer of the external RBG and describes the function call and parameters used when calling the external DRBG function. If different external RBGs are used for different platforms, the evaluator shall verify the TSS identifies each RBG for each platform. Also, the evaluator shall verify the TSS includes a short description of the vendor's assumption for the amount of entropy seeding the external DRBG. The evaluator uses the description of the RBG functionality in FCS_RBG_EXT or documentation available for the operational environment to determine that the key size being requested is identical to the key size and mode to be used for the encryption/decryption of the user data.

3.1.1.1.2. Operational Guidance

No activities specified.

3.1.1.1.3. Test

No activities specified.

3.2. Protection of the TSF (FPT)

3.2.1. Use of Supported Services and APIs (FPT_API_EXT.2)

3.2.1.1. FPT_API_EXT.2.1

3.2.1.1.1. TSS

If **use platform-provided libraries for parsing** is selected, the evaluator shall verify that the TSS lists the IANA MIME media types (as described by http://www.iana.org/assignments/media-types) for all formats the application processes and that it maps those formats to parsing services provided by the platform.

If **does not perform parsing** is selected there is no activity specified.

3.2.1.1.2. Operational Guidance

No activities specified.

3.2.1.1.3. Test

No activities specified.

4. Evaluation Activities for Selection-Based Requirements

4.1. Cryptograhic Support (FCS)

4.1.1. Random Bit Generation (FCS_RBG_EXT.2)

Documentation shall be produced—and the evaluator shall perform the activities—in accordance with Appendix D of [SWAppcPP].

4.1.1.1. TSS

The evaluator shall examine the TSS to determine that it specifies the DRBG type, identifies the entropy source(s) seeding the DRBG, and state the assumed or calculated min-entropy supplied either separately by each source or the min-entropy contained in the combined seed value.

4.1.1.2. Operational Guidance

The evaluator shall confirm that the guidance documentation contains appropriate instructions for configuring the RNG functionality.

4.1.1.3. Test

The evaluator shall perform 15 trials for the RNG implementation. If the RNG is configurable, the evaluator shall perform 15 trials for each configuration.

If the RNG has prediction resistance enabled, each trial consists of (1) instantiate DRBG, (2) generate

the first block of random bits (3) generate a second block of random bits (4) uninstantiate. The evaluator verifies that the second block of random bits is the expected value. The evaluator shall generate eight input values for each trial. The first is a count (0 – 14). The next three are entropy input, nonce, and personalization string for the instantiate operation. The next two are additional input and entropy input for the first call to generate. The final two are additional input and entropy input for the second call to generate. These values are randomly generated. “generate one block of random bits” means to generate random bits with number of returned bits equal to the Output Block Length (as defined in NIST SP800-90A).

If the RNG does not have prediction resistance, each trial consists of (1) instantiate DRBG, (2) generate the first block of random bits (3) reseed, (4) generate a second block of random bits (5) uninstantiate. The evaluator verifies that the second block of random bits is the expected value. The evaluator shall generate eight input values for each trial. The first is a count (0 – 14). The next three are entropy input, nonce, and personalization string for the instantiate operation. The fifth value is additional input to the first call to generate. The sixth and seventh are additional input and entropy input to the call to reseed. The final value is additional input to the second generate call.

The following paragraphs contain more information on some of the input values to be generated/selected by the evaluator.

Entropy input: the length of the entropy input value must equal the seed length.

Nonce: If a nonce is supported (CTR_DRBG with no Derivation Function does not use a nonce), the nonce bit length is one-half the seed length.

Personalization string: The length of the personalization string must be ⇐ seed length. If the implementation only supports one personalization string length, then the same length can be used for both values. If more than one string length is support, the evaluator shall use personalization strings of two different lengths. If the implementation does not use a personalization string, no value needs to be supplied.

Additional input: the additional input bit lengths have the same defaults and restrictions as the personalization string lengths.

4.1.2. Cryptographic Key Generation Services (FCS_CKM_EXT)

4.1.2.1. FCS_CKM_EXT.1/Asymetric

4.1.2.1.1. TSS

The evaluator shall inspect the application and its developer documentation to determine if the application needs asymmetric key generation services. If not, the evaluator shall verify the **generate no asymmetric cryptographic keys** selection is present in the ST. Otherwise, the evaluation activities shall be performed as stated below.

If asymetric keys are generated, The evaluator shall ensure that the TSS identifies the key sizes supported by the TOE. If the ST specifies more than one scheme, the evaluator shall examine the TSS to verify that it identifies the usage for each scheme.

4.1.2.1.2. Operational Guidance

The evaluator shall verify that the AGD guidance instructs the administrator how to configure the TOE to use the selected key generation scheme(s) and key size(s) for all cryptographic protocols defined in the Security Target.

4.1.2.1.3. Test

Note: The following tests require the developer to provide access to a test platform that provides the evaluator with tools that are typically not found on factory products. Generation of long-term cryptographic keys (i.e. keys that are not ephemeral keys/session keys) might be performed automatically (e.g. during initial start-up). Testing of key generation must cover not only administrator invoked key generation but also automated key generation (if supported).

Key Generation for FIPS PUB 186-4 RSA Schemes The evaluator shall verify the implementation of RSA Key Generation by the TOE using the Key Generation test. This test verifies the ability of the TSF to correctly produce values for the key components including the public verification exponent e, the private prime factors p and q, the public modulus n and the calculation of the private signature exponent *d*.

Key Pair generation specifies 5 ways (or methods) to generate the primes *p* and *q*. These include:

Random Primes: * Provable primes * Probable primes

Primes with Conditions: * Primes p1, p2, q1,q2, p and q shall all be provable primes * Primes p1, p2, q1, and q2 shall be provable primes and p and q shall be probable primes * Primes p1, p2, q1,q2, p and q shall all be probable primes

To test the key generation method for the Random Provable primes method and for all the Primes with Conditions methods, the evaluator must seed the TSF key generation routine with sufficient data to deterministically generate the RSA key pair. This includes the random seed(s), the public exponent of the RSA key, and the desired key length. For each key length supported, the evaluator shall have the TSF generate 25 key pairs. The evaluator shall verify the correctness of the TSF's implementation by comparing values generated by the TSF with those generated from a known good implementation.

Key Generation for Elliptic Curve Cryptography (ECC)

FIPS 186-4 ECC Key Generation Test

For each supported NIST curve, i.e., P-256, P-384 and P-521, the evaluator shall require the implementation under test (IUT) to generate 10 private/public key pairs. The private key shall be generated using an approved random bit generator (RBG). To determine correctness, the evaluator shall submit the generated key pairs to the public key verification (PKV) function of a known good implementation.

FIPS 186-4 Public Key Verification (PKV) Test For each supported NIST curve, i.e., P-256, P-384 and P-521, the evaluator shall generate 10 private/public key pairs using the key generation function of a known good implementation and modify five of the public key values so that they are incorrect, leaving five values unchanged (i.e., correct). The evaluator shall obtain in response a set of 10 PASS/FAIL values.

Key Generation for Finite-Field Cryptography (FFC)

The evaluator shall verify the implementation of the Parameters Generation and the Key Generation for FFC by the TOE using the Parameter Generation and Key Generation test. This test verifies the ability of the TSF to correctly produce values for the field prime p, the cryptographic prime q (dividing p-1), the cryptographic group generator g, and the calculation of the private key x and public key y.

The Parameter generation specifies 2 ways (or methods) to generate the cryptographic prime q and the field prime p:

- Primes q and p shall both be provable primes
- Primes q and field prime p shall both be probable primes

and two ways to generate the cryptographic group generator g:

- Generator g constructed through a verifiable process
- Generator g constructed through an unverifiable process.

The Key generation specifies 2 ways to generate the private key x: * len(q) bit output of RBG where 1 ⇐x ⇐ q-1 * len(q) + 64 bit output of RBG, followed by a mod q-1 operation and a +1 operation, where 1⇐ x⇐q-1.

The security strength of the RBG must be at least that of the security offered by the FFC parameter set.

To test the cryptographic and field prime generation method for the provable primes method and/or the group generator g for a verifiable process, the evaluator must seed the TSF parameter generation routine with sufficient data to deterministically generate the parameter set.

For each key length supported, the evaluator shall have the TSF generate 25 parameter sets and key pairs. The evaluator shall verify the correctness of the TSF's implementation by comparing values generated by the TSF with those generated from a known good implementation. Verification must also confirm * g != 0,1 * q divides p-1 * g^q mod p = 1 * g^x mod p = y

for each FFC parameter set and key pair.

FFC Schemes using "safe-prime" groups

Testing for FFC Schemes using safe-prime groups is done as part of testing in FCS_CKM.2.1.

4.1.2.2. FCS_CKM_EXT.1/PBKDF2

4.1.2.2.1. TSS

Support for PBKDF: The evaluator shall examine the password hierarchy TSS to ensure that the formation of all password based derived keys is described and that the key sizes match that described by the ST author. The evaluator shall check that the TSS describes the method by which the password/passphrase is first encoded and then fed to the SHA algorithm. The settings for the algorithm (padding, blocking, etc.) shall be described, and the evaluator shall verify that these are

supported by the selections in this component as well as the selections concerning the hash function itself. The evaluator shall verify that the TSS contains a description of how the output of the hash function is used to form the submask that will be input into the function. For the NIST SP 800-132-based conditioning of the password/passphrase, the required evaluation activities will be performed when doing the evaluation activities for the appropriate requirements (FCS_COP.1.1/KeyedHash). No explicit testing of the formation of the submask from the input password is required. FCS_CKM_EXT.1/PBKDF: The ST author shall provide a description in the TSS regarding the salt generation. The evaluator shall confirm that the salt is generated using an RBG described in FCS_RBG_EXT.2.

4.1.2.2.2. Operational Guidance

No activities specified.

4.1.2.2.3. Test

No activities specified.

4.1.2.3. FCS_CKM.2.1

4.1.2.3.1. TSS

The evaluator shall ensure that the supported key establishment schemes correspond to the key generation schemes identified in FCS_CKM_EXT.1/Asymetric. If the ST specifies more than one scheme, the evaluator shall examine the TSS to verify that it identifies the usage for each scheme (including whether the TOE acts as a sender, a recipient, or both).

The intent of this activity is to be able to identify the scheme being used by each service. This would mean, for example, one way to document scheme usage could be:

Scheme	SFR	Service
RSA	FCS_TLSS_EXT.1	Administration
ECDH	FCS_SSHC_EXT.1	Audit Server
Diffie-Hellman (Group 14)	FCS_SSHC_EXT.1	Backup Server

The information provided in the example above does not necessarily have to be included as a table but can be presented in other ways as long as the necessary data is available.

4.1.2.3.2. Guidance Documentation

The evaluator shall verify that the AGD guidance instructs the administrator how to configure the TOE to use the selected key establishment scheme(s).

4.1.2.3.3. Test

Key Establishment Schemes The evaluator shall verify the implementation of the key establishment schemes of the supported by the TOE using the applicable tests below.

SP800-56A Key Establishment Schemes

The evaluator shall verify a TOE's implementation of SP800-56A key agreement schemes using the following Function and Validity tests. These validation tests for each key agreement scheme verify that a TOE has implemented the components of the key agreement scheme according to the specifications in the Recommendation. These components include the calculation of the DLC primitives (the shared secret value Z) and the calculation of the derived keying material (DKM) via the Key Derivation Function (KDF). If key confirmation is supported, the evaluator shall also verify that the components of key confirmation have been implemented correctly, using the test procedures described below. This includes the parsing of the DKM, the generation of MACdata and the calculation of MACtag.

Function Test

The Function test verifies the ability of the TOE to implement the key agreement schemes correctly. To conduct this test the evaluator shall generate or obtain test vectors from a known good implementation of the TOE supported schemes. For each supported key agreement scheme-key agreement role combination, KDF type, and, if supported, key confirmation role- key confirmation type combination, the tester shall generate 10 sets of test vectors. The data set consists of one set of domain parameter values (FFC) or the NIST approved curve (ECC) per 10 sets of public keys. These keys are static, ephemeral or both depending on the scheme being tested.

The evaluator shall obtain the DKM, the corresponding TOE's public keys (static and/or ephemeral), the MAC tag(s), and any inputs used in the KDF, such as the Other Information field OI and TOE id fields. If the TOE does not use a KDF defined in SP 800-56A, the evaluator shall obtain only the public keys and the hashed value of the shared secret. The evaluator shall verify the correctness of the TSF's implementation of a given scheme by using a known good implementation to calculate the shared secret value, derive the keying material DKM, and compare hashes or MAC tags generated from these values.

If key confirmation is supported, the TSF shall perform the above for each implemented approved MAC algorithm.

Validity Test

The Validity test verifies the ability of the TOE to recognize another party's valid and invalid key agreement results with or without key confirmation. To conduct this test, the evaluator shall obtain a list of the supporting cryptographic functions included in the SP800-56A key agreement implementation to determine which errors the TOE should be able to recognize. The evaluator generates a set of 24 (FFC) or 30 (ECC) test vectors consisting of data sets including domain parameter values or NIST approved curves, the evaluator's public keys, the TOE's public/private key pairs, MACTag, and any inputs used in the KDF, such as the other info and TOE id fields.

The evaluator shall inject an error in some of the test vectors to test that the TOE recognizes invalid key agreement results caused by the following fields being incorrect: the shared secret value Z, the DKM, the other information field OI, the data to be MACed, or the generated MACTag. If the TOE contains the full or partial (only ECC) public key validation, the evaluator will also individually inject errors in both parties' static public keys, both parties' ephemeral public keys and the TOE's static private key to assure the TOE detects errors in the public key validation function and/or the partial key validation function (in ECC only). At least two of the test vectors shall remain unmodified and therefore should result in valid key agreement results (they should pass). The TOE

shall use these modified test vectors to emulate the key agreement scheme using the corresponding parameters. The evaluator shall compare the TOE's results with the results using a known good implementation verifying that the TOE detects these errors.

RSA-based key establishment

The evaluator shall verify the correctness of the TSF's implementation of RSAES-PKCS1-v1_5 by using a known good implementation for each protocol selected in FTP_ITC.1 that uses RSAES-PKCS1-v1_5.

FFC Schemes using "safe-prime" groups

The evaluator shall verify the correctness of the TSF's implementation of safe-prime groups by using a known good implementation for each protocol selected in FTP_ITC.1 that uses safe-prime groups. This test must be performed for each safe-prime group that each protocol uses.

4.1.3. Cryptographic Operation (FCS_COP)

4.1.3.1. FCS_COP.1/DataEncryption

4.1.3.1.1. TSS

No activities specified.

4.1.3.1.2. Guidance Documentation

No activities specified.

4.1.3.1.3. Tests

AES-CBC Known Answer Tests

There are four Known Answer Tests (KATs), described below. In all KATs, the plaintext, ciphertext, and IV values shall be 128-bit blocks. The results from each test may either be obtained by the evaluator directly or by supplying the inputs to the implementer and receiving the results in response. To determine correctness, the evaluator shall compare the resulting values to those obtained by submitting the same inputs to a known good implementation.

KAT-1. To test the encrypt functionality of AES-CBC, the evaluator shall supply a set of 10 plaintext values and obtain the ciphertext value that results from AES-CBC encryption of the given plaintext using a key value of all zeros and an IV of all zeros. Five plaintext values shall be encrypted with a 128-bit all-zeros key, and the other five shall be encrypted with a 256-bit all-zeros key.

To test the decrypt functionality of AES-CBC, the evaluator shall perform the same test as for encrypt, using 10 ciphertext values as input and AES-CBC decryption.

KAT-2. To test the encrypt functionality of AES-CBC, the evaluator shall supply a set of 10 key values and obtain the ciphertext value that results from AES-CBC encryption of an all-zeros plaintext using the given key value and an IV of all zeros. Five of the keys shall be 128-bit keys, and the other five shall be 256-bit keys.

To test the decrypt functionality of AES-CBC, the evaluator shall perform the same test as for encrypt, using an all-zero ciphertext value as input and AES-CBC decryption.

KAT-3. To test the encrypt functionality of AES-CBC, the evaluator shall supply the two sets of key values described below and obtain the ciphertext value that results from AES encryption of an all-zeros plaintext using the given key value and an IV of all zeros. The first set of keys shall have 128 128-bit keys, and the second set shall have 256 256-bit keys. Key i in each set shall have the leftmost i bits be ones and the rightmost N-i bits be zeros, for i in [1,N].

To test the decrypt functionality of AES-CBC, the evaluator shall supply the two sets of key and ciphertext value pairs described below and obtain the plaintext value that results from AES-CBC decryption of the given ciphertext using the given key and an IV of all zeros. The first set of key/ciphertext pairs shall have 128 128-bit key/ciphertext pairs, and the second set of key/ciphertext pairs shall have 256 256-bit key/ciphertext pairs. Key i in each set shall have the leftmost i bits be ones and the rightmost N-i bits be zeros, for i in [1,N]. The ciphertext value in each pair shall be the value that results in an all-zeros plaintext when decrypted with its corresponding key.

KAT-4. To test the encrypt functionality of AES-CBC, the evaluator shall supply the set of 128 plaintext values described below and obtain the two ciphertext values that result from AES-CBC encryption of the given plaintext using a 128-bit key value of all zeros with an IV of all zeros and using a 256-bit key value of all zeros with an IV of all zeros, respectively. Plaintext value i in each set shall have the leftmost i bits be ones and the rightmost 128-i bits be zeros, for i in [1,128].

To test the decrypt functionality of AES-CBC, the evaluator shall perform the same test as for encrypt, using ciphertext values of the same form as the plaintext in the encrypt test as input and AES-CBC decryption.

AES-CBC Multi-Block Message Test

The evaluator shall test the encrypt functionality by encrypting an i-block message where 1 < i ⇐10. The evaluator shall choose a key, an IV and plaintext message of length i blocks and encrypt the message, using the mode to be tested, with the chosen key and IV. The ciphertext shall be compared to the result of encrypting the same plaintext message with the same key and IV using a known good implementation.

The evaluator shall also test the decrypt functionality for each mode by decrypting an i-block message where 1 < i ⇐10. The evaluator shall choose a key, an IV and a ciphertext message of length i blocks and decrypt the message, using the mode to be tested, with the chosen key and IV. The plaintext shall be compared to the result of decrypting the same ciphertext message with the same key and IV using a known good implementation.

AES-CBC Monte Carlo Tests

The evaluator shall test the encrypt functionality using a set of 200 plaintext, IV, and key 3-tuples. 100 of these shall use 128 bit keys, and 100 shall use 256 bit keys. The plaintext and IV values shall be 128-bit blocks. For each 3-tuple, 1000 iterations shall be run as follows:

```
# Input: PT, IV, Key
for i = 1 to 1000:
        if i == 1:
            CT[1] = AES-CBC-Encrypt(Key, IV, PT)
            PT = IV
        else:
            CT[i] = AES-CBC-Encrypt(Key, PT)
            PT = CT[i-1]
```

The ciphertext computed in the 1000th iteration (i.e., CT[1000]) is the result for that trial. This result shall be compared to the result of running 1000 iterations with the same values using a known good implementation.

The evaluator shall test the decrypt functionality using the same test as for encrypt, exchanging CT and PT and replacing AES-CBC-Encrypt with AES-CBC-Decrypt.

AES-GCM Test The evaluator shall test the authenticated encrypt functionality of AES-GCM for each combination of the following input parameter lengths:

128 bit and 256 bit keys

a. **Two plaintext lengths**. One of the plaintext lengths shall be a non-zero integer multiple of 128 bits, if supported. The other plaintext length shall not be an integer multiple of 128 bits, if supported.
b. **Three AAD lengths**. One AAD length shall be 0, if supported. One AAD length shall be a non-zero integer multiple of 128 bits, if supported. One AAD length shall not be an integer multiple of 128 bits, if supported.
c. **Two IV lengths**. If 96 bit IV is supported, 96 bits shall be one of the two IV lengths tested. The evaluator shall test the encrypt functionality using a set of 10 key, plaintext, AAD, and IV tuples for each combination of parameter lengths above and obtain the ciphertext value and tag that results from AES-GCM authenticated encrypt. Each supported tag length shall be tested at least once per set of 10. The IV value may be supplied by the evaluator or the implementation being tested, as long as it is known.

The evaluator shall test the decrypt functionality using a set of 10 key, ciphertext, tag, AAD, and IV 5-tuples for each combination of parameter lengths above and obtain a Pass/Fail result on authentication and the decrypted plaintext if Pass. The set shall include five tuples that Pass and five that Fail.

The results from each test may either be obtained by the evaluator directly or by supplying the inputs to the implementer and receiving the results in response. To determine correctness, the evaluator shall compare the resulting values to those obtained by submitting the same inputs to a known good implementation.

AES-CTR Known Answer Tests

There are four Known Answer Tests (KATs) described below. For all KATs, the plaintext, IV, and ciphertext values shall be 128-bit blocks. The results from each test may either be obtained by the

validator directly or by supplying the inputs to the implementer and receiving the results in response. To determine correctness, the evaluator shall compare the resulting values to those obtained by submitting the same inputs to a known good implementation.

KAT-1 To test the encrypt functionality, the evaluator shall supply a set of 10 plaintext values and obtain the ciphertext value that results from encryption of the given plaintext using a key value of all zeros and an IV of all zeros. Five plaintext values shall be encrypted with a 128-bit all zeros key, and the other five shall be encrypted with a 256-bit all zeros key. To test the decrypt functionality, the evaluator shall perform the same test as for encrypt, using 10 ciphertext values as input.

KAT-2 To test the encrypt functionality, the evaluator shall supply a set of 10 key values and obtain the ciphertext value that results from encryption of an all zeros plaintext using the given key value and an IV of all zeros. Five of the key values shall be 128-bit keys, and the other five shall be 256-bit keys. To test the decrypt functionality, the evaluator shall perform the same test as for encrypt, using an all zero ciphertext value as input.

KAT-3 To test the encrypt functionality, the evaluator shall supply the two sets of key values described below and obtain the ciphertext values that result from AES encryption of an all zeros plaintext using the given key values an IV of all zeros. The first set of keys shall have 128 128-bit keys, and the second shall have 256 256-bit keys. Key_i in each set shall have the leftmost i bits be ones and the rightmost N-i bits be zeros, for i in [1, N]. To test the decrypt functionality, the evaluator shall supply the two sets of key and ciphertext value pairs described below and obtain the plaintext value that results from decryption of the given ciphertext using the given key values and an IV of all zeros. The first set of key/ciphertext pairs shall have 128 128-bit key/ciphertext pairs, and the second set of key/ciphertext pairs shall have 256 256-bit pairs. Key_i in each set shall have the leftmost i bits be ones and the rightmost N-i bits be zeros for i in [1, N]. The ciphertext value in each pair shall be the value that results in an all zeros plaintext when decrypted with its corresponding key.

KAT-4 To test the encrypt functionality, the evaluator shall supply the set of 128 plaintext values described below and obtain the two ciphertext values that result from encryption of the given plaintext using a 128-bit key value of all zeros and using a 256 bit key value of all zeros, respectively, and an IV of all zeros. Plaintext value i in each set shall have the leftmost bits be ones and the rightmost 128-i bits be zeros, for i in [1, 128]. To test the decrypt functionality, the evaluator shall perform the same test as for encrypt, using ciphertext values of the same form as the plaintext in the encrypt test as input.

AES-CTR Multi-Block Message Test

The evaluator shall test the encrypt functionality by encrypting an i-block message where 1 less-than i less-than-or-equal to 10. For each i the evaluator shall choose a key, IV, and plaintext message of length i blocks and encrypt the message, using the mode to be tested, with the chosen key. The ciphertext shall be compared to the result of encrypting the same plaintext message with the same key and IV using a known good implementation. The evaluator shall also test the decrypt functionality by decrypting an i-block message where 1 less-than i less-than-or-equal to 10. For each i the evaluator shall choose a key and a ciphertext message of length i blocks and decrypt the message, using the mode to be tested, with the chosen key. The plaintext shall be compared to the result of decrypting the same ciphertext message with the same key using a known good implementation.

AES-CTR Monte-Carlo Test

The evaluator shall test the encrypt functionality using 200 plaintext/key pairs. 100 of these shall use 128 bit keys, and 100 of these shall use 256 bit keys. The plaintext values shall be 128-bit blocks. For each pair, 1000 iterations shall be run as follows:

```
# Input: PT, Key
for i = 1 to 1000:
CT[i] = AES-CTR-Encrypt(Key, PT) PT = CT[i]
```

The ciphertext computed in the 1000th iteration is the result for that trial. This result shall be compared to the result of running 1000 iterations with the same values using a known good implementation.

There is no need to test the decryption engine.

4.1.3.2. FCS_COP.1.1/SigGen

4.1.3.2.1. TSS

No activities specified.

4.1.3.2.2. Guidance Documentation

No activities specified.

4.1.3.2.3. Tests

ECDSA Algorithm Tests

ECDSA FIPS 186-4 Signature Generation Test

For each supported NIST curve (i.e., P-256, P-384 and P-521) and SHA function pair, the evaluator shall generate 10 1024-bit long messages and obtain for each message a public key and the resulting signature values R and S. To determine correctness, the evaluator shall use the signature verification function of a known good implementation.

ECDSA FIPS 186-4 Signature Verification Test

For each supported NIST curve (i.e., P-256, P-384 and P-521) and SHA function pair, the evaluator shall generate a set of 10 1024-bit message, public key and signature tuples and modify one of the values (message, public key or signature) in five of the 10 tuples. The evaluator shall obtain in response a set of 10 PASS/FAIL values.

RSA Signature Algorithm Tests

Signature Generation Test

The evaluator generates or obtains 10 messages for each modulus size/SHA combination supported by the TOE. The TOE generates and returns the corresponding signatures.

The evaluator shall verify the correctness of the TOE's signature using a trusted reference implementation of the signature verification algorithm and the associated public keys to verify the signatures.

Signature Verification Test

For each modulus size/hash algorithm selected, the evaluator generates a modulus and three associated key pairs, (d, e). Each private key d is used to sign six pseudorandom messages each of 1024 bits using a trusted reference implementation of the signature generation algorithm. Some of the public keys, e, messages, or signatures are altered so that signature verification should fail. For both the set of original messages and the set of altered messages: the modulus, hash algorithm, public key e values, messages, and signatures are forwarded to the TOE, which then attempts to verify the signatures and returns the verification results.

The evaluator verifies that the TOE confirms correct signatures on the original messages and detects the errors introduced in the altered messages.

4.1.3.3. FCS_COP.1/Hash

4.1.3.3.1. TSS

The evaluator shall check that the association of the hash function with other TSF cryptographic functions (for example, the digital signature verification function) is documented in the TSS.

4.1.3.3.2. Guidance Documentation

The evaluator checks the AGD documents to determine that any configuration that is required to configure the required hash sizes is present.

4.1.3.3.3. Tests

The TSF hashing functions can be implemented in one of two modes. The first mode is the byteoriented mode. In this mode the TSF only hashes messages that are an integral number of bytes in length; i.e., the length (in bits) of the message to be hashed is divisible by 8. The second mode is the bitoriented mode. In this mode the TSF hashes messages of arbitrary length. As there are different tests for each mode, an indication is given in the following sections for the bitoriented vs. the byteoriented testmacs.

The evaluator shall perform all of the following tests for each hash algorithm implemented by the TSF and used to satisfy the requirements of this PP.

Short Messages Test Bitoriented Mode

The evaluators devise an input set consisting of m+1 messages, where m is the block length of the hash algorithm. The length of the messages range sequentially from 0 to m bits. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

Short Messages Test Byteoriented Mode

The evaluators devise an input set consisting of m/8+1 messages, where m is the block length of the hash algorithm. The length of the messages range sequentially from 0 to m/8 bytes, with each message being an integral number of bytes. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

Selected Long Messages Test Bitoriented Mode

The evaluators devise an input set consisting of m messages, where m is the block length of the hash algorithm (e.g. 512 bits for SHA-256). The length of the ith message is m + 99*i, where 1 ≤ i ≤ m. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

Selected Long Messages Test Byteoriented Mode

The evaluators devise an input set consisting of m/8 messages, where m is the block length of the hash algorithm (e.g. 512 bits for SHA-256). The length of the ith message is m + 8*99*i, where 1 ≤ i ≤ m/8. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

Pseudorandomly Generated Messages Test

This test is for byteoriented implementations only. The evaluators randomly generate a seed that is n bits long, where n is the length of the message digest produced by the hash function to be tested. The evaluators then formulate a set of 100 messages and associated digests by following the algorithm provided in Figure 1 of [SHAVS]. The evaluators then ensure that the correct result is produced when the messages are provided to the TSF.

4.1.3.4. FCS_COP.1/KeyedHash

4.1.3.4.1. TSS

The evaluator shall examine the TSS to ensure that it specifies the following values used by the HMAC function: key length, hash function used, block size, and output MAC length used.

4.1.3.4.2. Guidance Documentation

No activities specified.

4.1.3.4.3. Tests

For each of the supported parameter sets, the evaluator shall compose 15 sets of test data. Each set shall consist of a key and message data. The evaluator shall have the TSF generate HMAC tags for these sets of test data. The resulting MAC tags shall be compared to the result of generating HMAC tags with the same key and message data using a known good implementation.

4.1.4. Cryptographic Protocols (FCS_HTTPS_EXT)

4.1.4.1. FCS_HTTPS_EXT.1 HTTPS Protocol

4.1.4.1.1. TSS

The evaluator shall examine the TSS and determine that enough detail is provided to explain how the implementation complies with RFC 2818.

4.1.4.1.2. Guidance Documentation

No activities specified.

4.1.4.1.3. Tests

The evaluator shall perform the following tests:

- Test 1: The evaluator shall attempt to establish each trusted path or channel that utilizes HTTPS, observe the traffic with a packet analyser, verify that the connection succeeds, and verify that the traffic is identified as TLS or HTTPS.

Other tests are performed in conjunction with the TLS evaluation activities.

If the TOE is an HTTPS client or an HTTPS server utilizing X.509 client authentication, then the certificate validity shall be tested in accordance with testing performed for FIA_X509_EXT.1.

TLS Protocol

If the TOE implements the TLS protocol, the evaluator shall test the TOE as per evaluation activities from [TLS Package]

4.2. Identification and Authentication (FIA)

4.2.1. External Identity Provider (FIA_EIP)

4.2.1.1. FIA_EIP_EXT.1 (External Identity Provider)

4.2.1.1.1. TSS

The evaluator shall examine the TSS and verify that a description is provided for the secure channel used to communicate with the external identity provider, including the supported security parameters (e.g. AES, SHA, RSA, ECDSA claims).

The evaluator shall examine the TSS and verify that a description is provided for how the TOE enrolls with the external identity provider.

4.2.1.1.2. Guidance Documentation

The evaluator shall review the guidance documentation and verify that procedures are provided for configuring the channel between the TOE and external identity provider.

The evaluator shall review the guidance documentation to ensure that a list of the maintained attributes are provided.

4.2.1.1.3. Test

The evaluator shall perform the following test activities:

- Test 1: The evaluator shall follow the guidance documentation to establish the secure communication channel between the TOE and external identity provider. The evaluator shall review packet captures and/or log information and verify that the supported security parameters are used, and no other options are offered during the session establishment.
- Test 2: The evaluator shall follow the guidance documentation to enroll with the external identify provider. The evaluator shall complete the enrollment process and attempt a successful authentication attempt to the TOE. If any additional steps are required that are not documented within the AGD, this test fails.
- Test 3: (Conditional) If the TSF can locally configure any of the maintained attributes listed within the guidance documentation, the evaluator shall attempt to modify the attribute prior to authenticating to the TSF. The evaluator shall verify that the attempt fails. The evaluator shall then successfully authenticate to the TOE and repeat the modification of the attribute, verifying the attempt now succeeds.
- Test 4: The evaluator shall interrupt the connection between the TOE and External Identity Provider. The evaluator shall attempt to authenticate to the administrative interface and verify that the attempt fails. The evaluator shall then re-establish the connection to the EIP and attempt to authenticate to the admin interface. The evaluator shall verify that the authentication attempt is successful and the connection to the EIP is securely established.

4.2.2. User Authentication (FIA_UAU)

4.2.2.1. FIA_UAU_EXT.5 (User Authentication Mechanisms)

4.2.2.1.1. TSS

The evaluator shall examine the TSS to ensure that it describes each mechanism provided to support user authentication and the rules describing how the authentication mechanism(s) provide authentication.

4.2.2.1.2. Guidance Documentation

(Conditional - provide an authentication mechanism) The evaluator will review the Guidance documentation and verify that steps are provided that allow for a Security Administrator to configure and store user credentials in accordance with FCS_STO_EXT.1.

(Conditional - integrate with an external identity provider) The evaluator will review the Guidance documentation and verify that steps are provided that allow for the TOE to successfully connect and enroll with the remote external identity provider for authentication to the TOE.

4.2.2.1.3. Test

The evaluator shall perform the following test cases:

- Test 1: (Conditional - provide an authentication mechanism) – For each authentication mechanism claimed, the evaluator shall follow the guidance documentation to configure the

TOE to use the listed mechanism. The evaluator shall attempt to authenticate to the TOE using a known credential and verify that the authentication attempt is successful. The evaluator shall then repeat the test using an invalid credential and verify the attempt fails.

- Test 2: (Conditional - integrate with an external identity provider) - The evaluator will follow the guidance documentation and verify that following completion of the configuration steps a connection is established with the EIP. The evaluator shall attempt to authenticate to the TOE using a known credential and verify that the authentication attempt is successful. The evaluator shall then repeat the test using an invalid credential and verify the attempt fails.

4.2.2.2. FIA_UAU_EXT.2 (User Authentication)

Evaluation Activities for this requirement are covered under those for FIA_UIA_EXT.1. If other authentication mechanisms are specified, the evaluator shall include those methods in the activities for FIA_UIA_EXT.1.

4.2.2.3. FIA_UAU.7 (User Authentication Obfuscation)

4.2.2.3.1. TSS

No additional assurance activities.

4.2.2.3.2. Guidance Documentation

The evaluator shall examine the guidance documentation to determine that any necessary preparatory steps to ensure authentication data is not revealed while entering for each login allowed.

4.2.2.3.3. Test

The evaluator shall perform the following test for each method of login allowed:

- Test 1: The evaluator shall authenticate to the TOE. While making this attempt, the evaluator shall verify that at most obscured feedback is provided while entering the authentication information.

Application Note : Temporary display of the last character typed during input is permitted.

4.2.3. User Identification (FIA_UIA)

4.2.3.1. FIA_UIA_EXT.1 (User Identification)

4.2.3.1.1. TSS

The evaluator shall examine the TSS to determine that it describes which actions are allowed before user identification and authentication. The description shall cover authentication and identification for TOE administration.

4.2.3.1.2. Guidance Documentation

If configuration is necessary to ensure the services provided before login are limited, the evaluator shall determine that the guidance documentation provides sufficient instruction on limiting the

allowed services.

4.2.3.1.3. Test

- Test 1: The evaluator shall configure the services allowed (if any) according to the guidance documentation. The evaluator shall attempt to perform an operation that is not claimed within the requirement and verify that the operation fails. The evaluator shall then authenticate to the TOE and verify the operation is now successful

4.2.4. User Authentication Failure (FIA_AFL)

4.2.4.1. FIA_AFL.1 (Authentication Failure)

4.2.4.1.1. TSS

The evaluator shall examine the TSS to determine that it contains a description, for each supported method for authentication, of how successive unsuccessful authentication attempts are detected and tracked. The TSS shall also describe the method by which the administrator is prevented from successfully logging on to the TOE, and the actions necessary to restore this ability.

The evaluator shall examine the TSS to confirm that the TOE ensures that authentication failures by administrators cannot lead to a situation where no administrator access is available, either permanently or temporarily (e.g. by providing local logon which is not subject to blocking).

4.2.4.1.2. Guidance Documentation

The evaluator shall examine the guidance documentation to ensure that instructions for configuring the number of successive unsuccessful authentication attempts and time period (if implemented) are provided, and that the process of allowing the administrator to once again successfully log on is described for each “action” specified (if that option is chosen). If different actions or mechanisms are implemented, all must be described.

4.2.4.1.3. Test

The evaluator shall perform the following tests for each method by which administrators access the TOE:

- Test 1: The evaluator shall use the operational guidance to configure the number of successive unsuccessful authentication attempts allowed by the TOE (and, if the time period selection in FIA_AFL.1.2 is included in the ST, then the evaluator shall also use the operational guidance to configure the time period after which access is re-enabled). The evaluator shall test that once the authentication attempts limit is reached, authentication attempts with valid credentials are no longer successful.
- Test 2: After reaching the limit for unsuccessful authentication attempts as in Test 1 above, the evaluator shall proceed as follows. If the administrator action selection in FIA_AFL.1.2 is included in the ST, then the evaluator shall confirm by testing that following the operational guidance and performing each action specified in the ST to re-enable the administrator’s access results in successful access (when using valid credentials for that administrator).

  If the time period selection in FIA_AFL.1.2 is included in the ST, then the evaluator shall wait for

just less than the time period configured in Test 1 and show that an authorisation attempt using valid credentials does not result in successful access. The evaluator shall then wait until just after the time period configured in Test 1 and show that an authorisation attempt using valid credentials results in successful access.

4.2.5. Certificate Validation (FIA_X509)

4.2.5.1. FIA_X509_EXT.1.1/Rev (X.509 Certificate Validation)

4.2.5.1.1. TSS

The evaluator shall ensure the TSS describes where the check of validity of the certificates takes place, and that the TSS identifies any of the rules for extendedKeyUsage fields (in FIA_X509_EXT.1.1/Rev) that are not supported by the TOE (i.e. where the ST is therefore claiming that they are trivially satisfied). It is expected that revocation checking is performed when a certificate is used in an authentication step and when performing trusted updates (if selected). It is not necessary to verify the revocation status of X.509 certificates during power-up self-tests (if the option for using X.509 certificates for self-testing is selected).

4.2.5.1.2. Guidance Documentation

No activities specified.

4.2.5.1.3. Test

The evaluator shall demonstrate that checking the validity of a certificate is performed when a certificate is used in an authentication step or when performing trusted updates (if FPT_TUD_EXT.1 is selected). It is not sufficient to verify the status of a X.509 certificate only when it is loaded onto the TOE. It is not necessary to verify the revocation status of X.509 certificates during power-up self-tests (if the option for using X.509 certificates for self-testing is selected). The evaluator shall perform the following tests for:

Test 1a: The evaluator shall present the TOE with a valid chain of certificates (terminating in a trusted CA certificate) as needed to validate the leaf certificate to be used in the function, and shall use this chain to demonstrate that the function succeeds. Test 1a shall be designed in a way that the chain can be 'broken' in Test 1b by either being able to remove the trust anchor from the TOEs trust store, or by setting up the trust store in a way that at least one intermediate CA certificate needs to be provided, together with the leaf certificate from outside the TOE, to complete the chain (e.g. by storing only the root CA certificate in the trust store.

Test 1b: The evaluator shall then 'break' the chain used in Test 1a by either removing the trust anchor in the TOE’s trust store used to terminate the chain, or by removing one of the intermediate CA certificates (provided together with the leaf certificate in Test 1a) to complete the chain. The evaluator shall show that an attempt to validate this broken chain fails.

Test 2: The evaluator shall demonstrate that validating an expired certificate results in the function failing.

Test 3: The evaluator shall test that the TOE can properly handle revoked certificates-–conditional on whether CRL or OCSP is selected; if both are selected, then a test shall be performed for each

method. The evaluator shall test revocation of the peer certificate and revocation of the peer intermediate CA certificate i.e. the intermediate CA certificate should be revoked by the root CA. The evaluator shall ensure that a valid certificate is used, and that the validation function succeeds. The evaluator then attempts the test with a certificate that has been revoked (for each method chosen in the selection) to ensure when the certificate is no longer valid that the validation function fails. Revocation checking is only applied to certificates that are not designated as trust anchors. Therefore the revoked certificate(s) used for testing shall not be a trust anchor.

Test 4: If any OCSP option is selected, the evaluator shall ensure the TSF has no other source of revocation information available and configure the OCSP server or use a man-in-the-middle tool to present an OCSP response signed by a certificate that does not have the OCSP signing purpose and which is the only source of revocation status information advertised by the CA issuing the certificate being validated. The evaluator shall verify that validation of the OCSP response fails and that the TOE treats the certificate being checked as invalid and rejects the connection. If CRL is selected, the evaluator shall likewise configure the CA to be the only source of revocation status information, and sign a CRL with a certificate that does not have the cRLsign key usage bit set. The evaluator shall verify that validation of the CRL fails and that the TOE treats the certificate being checked as invalid and rejects the connection.

Note: The intent of this test is to ensure a TSF does not trust invalid revocation status information. A TSF receiving invalid revocation status information from the only advertised certificate status provider should treat the certificate whose status is being checked as invalid. This should generally be treated differently from the case where the TSF is not able to establish a connection to check revocation status information,but it is acceptable that the TSF ignore any invalid information and attempt to find another source of revocation status (another advertised provider, a locally configured provider, or cached information) and treat this situation as not having a connection to a valid certificate status provider.

Test 5: The evaluator shall modify any byte in the first eight bytes of the certificate and demonstrate that the certificate fails to validate. (The certificate will fail to parse correctly.)

Test 6: The evaluator shall modify any byte in the last byte of the certificate and demonstrate that the certificate fails to validate. (The signature on the certificate will not validate.)

Test 7: The evaluator shall modify any byte in the public key of the certificate and demonstrate that the certificate fails to validate. (The hash of the certificate will not validate.)

Test 8: Create a valid certificate chain from root certificate designated as a trust anchor. This chain must contain at least one Elliptic Curve certificate, that has a public key information field where the EC parameters uses an explicit format version of the Elliptic Curve parameters in the public key information field. Initiate validation of this chain by the TOE. The evaluator shall confirm the TOE treats the certificate chain as invalid.

4.2.5.2. FIA_X509_EXT.1.2/Rev (X.509 Certificate Validation)

4.2.5.2.1. TSS

No activities specified.

4.2.5.2.2. Guidance Documentation

No activities specified.

4.2.5.2.3. Test

The evaluator shall perform the following tests for FIA_X509_EXT.1.2/Rev. The tests described must be performed in conjunction with the other certificate services assurance activities, including the functions in FIA_X509_EXT.2.1. The tests for the extendedKeyUsage rules are performed in conjunction with the uses that require those rules. Where the TSS identifies any of the rules for extendedKeyUsage fields (in FIA_X509_EXT.1.1) that are not supported by the TOE (i.e. where the ST is therefore claiming that they are trivially satisfied) then the associated extendedKeyUsage rule testing may be omitted.

For each of the following tests the evaluator shall create a chain of at least three certificates: a self-signed root CA certificate, an intermediate CA certificate and a leaf (node) certificate. The properties of the certificates in the chain are adjusted as described in each individual test below (and this modification shall be the only invalid aspect of the relevant certificate chain).

Test 1: The evaluator shall ensure that at least one of the CAs in the chain does not contain the basicConstraints extension. The evaluator confirms that the TOE rejects such a certificate at one (or both) of the following points: (i) as part of the validation of the leaf certificate belonging to this chain; (ii) when attempting to add a CA certificate without the basicConstraints extension to the TOE's trust store (i.e. when attempting to install the CA certificate as one which will be retrieved from the TOE itself when validating future certificate chains).

Test 2: The evaluator shall ensure that at least one of the CA certificates in the chain has a basicConstraints extension in which the CA flag is set to FALSE. The evaluator confirms that the TOE rejects such a certificate at one (or both) of the following points: (i) as part of the validation of the leaf certificate belonging to this chain; (ii) when attempting to add a CA certificate with the CA flag set to FALSE to the TOE's trust store (i.e. when attempting to install the CA certificate as one which will be retrieved from the TOE itself when validating future certificate chains).

The evaluator shall repeat these tests for each distinct use of certificates. Thus, for example, use of certificates for TLS connection is distinct from use of certificates for trusted updates so both of these uses would be tested.

4.2.5.3. FIA_X509.EXT.2 X.509 Certificate Authentication

4.2.5.3.1. TSS

The evaluator shall check the TSS to ensure that it describes how the TOE chooses which certificates to use, and any necessary instructions in the administrative guidance for configuring the operating environment so that the TOE can use the certificates.

The evaluator shall examine the TSS to confirm that it describes the behavior of the TOE when a connection cannot be established during the validity check of a certificate used in establishing a trusted channel. The evaluator shall verify that any distinctions between trusted channels are described. If the requirement that the administrator is able to specify the default action, then the evaluator shall ensure that the guidance documentation contains instructions on how this

configuration action is performed.

4.2.5.3.2. Guidance Documentation

No activities specified.

4.2.5.3.3. Test

The evaluator shall perform the following test for each trusted channel:

The evaluator shall demonstrate that using a valid certificate that requires certificate validation checking to be performed in at least some part by communicating with a non-TOE IT entity. The evaluator shall then manipulate the environment so that the TOE is unable to verify the validity of the certificate, and observe that the action selected in FIA_X509_EXT.2.2 is performed. If the selected action is administrator-configurable, then the evaluator shall follow the guidance documentation to determine that all supported administrator-configurable options behave in their documented manner.

4.3. TOE Access (FTA)

4.3.1. TOE Login Banner (FTA_TAB)

4.3.1.1. FTA_TAB.1 (TOE Login Banner)

4.3.1.1.1. TSS

No assurance activities.

4.3.1.1.2. Guidance Documentation

The evaluator shall check the guidance documentation to ensure that it describes how to configure the banner message.

4.3.1.1.3. Test

The evaluator shall also perform the following test:

- Test 1: The evaluator follows the guidance documentation to configure a notice and consent warning message. The evaluator shall then attempt to establish an administrative session with the TOE and verify that the notice and consent warning message is displayed prior to or during the authentication step.

5. Evaluation Activities for Objective Requirements

There are no Objective Requirements present in collaborative Protection Profile for Application Software

6. Evaluation Activities for SARs

The sections below specify EAs for the Security Assurance Requirements (SARs) included in the related cPPs. The EAs in Section 2, “Evaluation Activities for Mandatory SFRs”, Section 4, “Evaluation Activities for Selection-Based Requirements”, and Section 3, “Evaluation Activities for Optional Requirements” are an interpretation of the more general CEM assurance requirements as they apply to the specific technology area of the TOE.

In this section, each SAR that is contained in the cPP is listed, and the EAs that are not associated with an SFR are captured here, or a reference is made to the CEM, and the evaluator is expected to perform the CEM work units.

6.1. Class ASE: Security Target

When evaluating a Security Target, the evaluator performs the work units as presented in the CEM. In addition, the evaluator ensures the content of the TSS in the ST satisfies the EAs specified in Section 2, “Evaluation Activities for Mandatory SFRs”.

6.2. Class ADV: Development

6.2.1. Basic Functional Specification (ADV_FSP.1)

The EAs for this assurance component focus on understanding the interfaces (e.g., application programing interfaces, command line interfaces, graphical user interfaces, network interfaces) described in the AGD documentation, and possibly identified in the TOE Summary Specification (TSS) in response to the SFRs. Specific evaluator actions to be performed against this documentation are identified (where relevant) for each SFR in Section 2, “Evaluation Activities for Mandatory SFRs”, and in EAs for AGD, ATE and AVA SARs in other parts of Section 5.

The EAs presented in this section address the CEM work units ADV_FSP.1-1, ADV_FSP.1-2, ADV_FSP.1-3, and ADV_FSP.1-5.

The EAs are reworded for clarity and interpret the CEM work units such that they will result in more objective and repeatable actions by the evaluator. The EAs in this SD are intended to ensure the evaluators are consistently performing equivalent actions.

The documents to be examined for this assurance component in an evaluation are therefore the Security Target, AGD documentation, and any required supplementary information required by the cPP: no additional "functional specification" documentation is necessary to satisfy the EAs. The interfaces that need to be evaluated are also identified by reference to the EAs listed for each SFR, and are expected to be identified in the context of the Security Target, AGD documentation, and any required supplementary information defined in the cPP rather than as a separate list specifically for the purposes of CC evaluation. The direct identification of documentation requirements and their assessment as part of the EAs for each SFR also means that the tracing required in ADV_FSP.1.2D (work units ADV_FSP.1-4, ADV_FSP.1-6 and ADV_FSP.1-7 is treated as implicit and no separate mapping information is required for this element.

Table 3. Mapping of ADV_FSP.1 CEM Work Units to Evaluation Activities

CEM ADV_FSP.1 Work Units	Evaluator Activities
ADV_FSP.1-1 The evaluator ***shall examine*** the functional specification to determine that it states the purpose of each SFR-supporting and SFR-enforcing TSFI.	Section 6.2.2, "ADV_FSP.1-1 Evaluation Activity": *The evaluator shall examine the interface documentation to ensure it describes the purpose and method of use for each TSFI that is identified as being security relevant.*
ADV_FSP.1-2 The evaluator ***shall examine*** the functional specification to determine that the method of use for each SFR-supporting and SFR-enforcing TSFI is given.	Section 6.2.3, "ADV_FSP.1-2 Evaluation Activity": *The evaluator shall examine the interface documentation to ensure it describes the purpose and method of use for each TSFI that is identified as being security relevant.*
ADV_FSP.1-3 The evaluator ***shall examine*** the presentation of the TSFI to determine that it identifies all parameters associated with each SFR-enforcing and SFR supporting TSFI.	Section 6.2.4, "ADV_FSP.1-3 Evaluation Activity": *The evaluator shall check the interface documentation to ensure it identifies and describes the parameters for each TSFI that is identified as being security relevant.*
ADV_FSP.1-4 The evaluator shall examine the rationale provided by the developer for the implicit categorisation of interfaces as SFR-non-interfering to determine that it is accurate.	Paragraph 561 from the CEM: "In the case where the developer has provided adequate documentation to perform the analysis called for by the rest of the work units for this component without explicitly identifying SFR-enforcing and SFR-supporting interfaces, this work unit should be considered satisfied." Since the rest of the ADV_FSP.1 work units will have been satisfied upon completion of the EAs, it follows that this work unit is satisfied as well.
ADV_FSP.1-5 The evaluator ***shall check*** that the tracing links the SFRs to the corresponding TSFIs.	[ADV_FSP.1-5 Evaluation Activity]: *The evaluator shall examine the interface documentation to develop a mapping of the interfaces to SFRs.*
ADV_FSP.1-6 The evaluator ***shall examine*** the functional specification to determine that it is a complete instantiation of the SFRs.	EAs that are associated with the SFRs in Section 2, "Evaluation Activities for Mandatory SFRs", and, if applicable, Section 4, "Evaluation Activities for Selection-Based Requirements" and Section 3, "Evaluation Activities for Optional Requirements", are performed to ensure that all the SFRs where the security functionality is externally visible (i.e., at the TSFI) are covered. Therefore, the intent of this work unit is covered.

CEM ADV_FSP.1 Work Units	Evaluator Activities
ADV_FSP.1-7 The evaluator ***shall examine*** the functional specification to determine that it is an accurate instantiation of the SFRs.	EAs that are associated with the SFRs in Section 2, "Evaluation Activities for Mandatory SFRs", and, if applicable, Section 4, "Evaluation Activities for Selection-Based Requirements" and Section 3, "Evaluation Activities for Optional Requirements", are performed to ensure that all the SFRs where the security functionality is externally visible (i.e., at the TSFI) are addressed, and that the description of the interfaces is accurate with respect to the specification captured in the SFRs. Therefore, the intent of this work unit is covered.

6.2.2. ADV_FSP.1-1 Evaluation Activity

The evaluator shall examine the interface documentation to ensure it describes the purpose and method of use for each TSFI that is identified as being security relevant.

In this context, TSFI are deemed security relevant if they are used by the administrator to configure the TOE, or to perform other administrative functions (e.g., audit review or performing updates). Additionally, those interfaces that are identified in the ST, or guidance documentation, as adhering to the security policies (as presented in the SFRs), are also considered security relevant. The intent, is that these interfaces will be adequately tested, and having an understanding of how these interfaces are used in the TOE is necessary to ensure proper test coverage is applied.

The set of TSFI that are provided as evaluation evidence are contained in the Administrative Guidance and User Guidance.

6.2.3. ADV_FSP.1-2 Evaluation Activity

The evaluator shall check the interface documentation to ensure it identifies and describes the parameters for each TSFI that is identified as being security relevant.

6.2.4. ADV_FSP.1-3 Evaluation Activity

The evaluator shall examine the interface documentation to develop a mapping of the interfaces to SFRs.

The evaluator uses the provided documentation and first identifies, and then examines a representative set of interfaces to perform the EAs presented in Section 2, "Evaluation Activities for Mandatory SFRs", including the EAs associated with testing of the interfaces.

It should be noted that there may be some SFRs that do not have an interface that is explicitly "mapped" to invoke the desired functionality. For example, generating a random bit string, destroying a cryptographic key that is no longer needed, or the TSF failing to a secure state, are capabilities that may be specified in SFRs, but are not invoked by an interface.

However, if the evaluator is unable to perform some other required EA because there is insufficient design and interface information, then the evaluator is entitled to conclude that an adequate functional specification has not been provided, and hence that the verdict for the ADV_FSP.1

assurance component is a 'fail'.

6.3. Class AGD: Guidance Documentation

It is not necessary for a TOE to provide separate documentation to meet the individual requirements of AGD_OPE and AGD_PRE. Although the EAs in this section are described under the traditionally separate AGD families, the mapping between the documentation provided by the developer and the AGD_OPE and AGD_PRE requirements may be many-to-many, as long as all requirements are met in documentation that is delivered to administrators and users (as appropriate) as part of the TOE.

6.3.1. Operational User Guidance (AGD_OPE.1)

The evaluator performs the CEM work units associated with the AGD_OPE.1 SAR. Specific requirements and EAs on the guidance documentation are identified (where relevant) in the individual EAs for each SFR.

In addition, the evaluator performs the EAs specified below.

6.3.1.1. Evaluation Activity

The evaluator shall ensure the Operational guidance documentation is distributed to administrators and users (as appropriate) as part of the TOE, so that there is a reasonable guarantee that administrators and users are aware of the existence and role of the documentation in establishing and maintaining the evaluated configuration.

6.3.1.2. Evaluation Activity

The evaluator shall ensure that the Operational guidance is provided for every Operational Environment that the product supports as claimed in the Security Target and shall adequately address all platforms claimed for the TOE in the Security Target.

6.3.1.3. Evaluation Activity

The evaluator shall ensure that the Operational guidance contains instructions for configuring any cryptographic engine associated with the evaluated configuration of the TOE. It shall provide a warning to the administrator that use of other cryptographic engines was not evaluated nor tested during the CC evaluation of the TOE.

6.3.1.4. Evaluation Activity

The evaluator shall ensure the Operational guidance makes it clear to an administrator which security functionality and interfaces have been assessed and tested by the EAs.

6.3.1.5. Evaluation Activity

In addition the evaluator shall ensure that the following requirements are also met.

- The guidance documentation shall contain instructions for configuring any cryptographic engine associated with the evaluated configuration of the TOE. It shall provide a warning to the

administrator that use of other cryptographic engines was not evaluated nor tested during the CC evaluation of the TOE.

- The documentation must describe the process for verifying updates to the TOE by verifying a digital signature. The evaluator shall verify that this process includes the following steps:
  - Instructions for obtaining the update itself. This should include instructions for making the update accessible to the TOE (e.g., placement in a specific directory).
  - Instructions for initiating the update process, as well as discerning whether the process was successful or unsuccessful. This includes instructions that describe at least one method of validating the hash/digital signature.
- The TOE will likely contain security functionality that does not fall in the scope of evaluation under this cPP. The guidance documentation shall make it clear to an administrator which security functionality is covered by the Evaluation Activities.

6.3.1.6. Evaluation Activity

The evaluator shall examine the Operational Guidance to ensure that the following ALC_FLR.3 information is included:

- A means for the TOE user to provide reports of suspected security flaws or requests for corrections to such flaws.
- A means of enabling the TOE users to register with the developer.
- Specific points of contact for user reports and enquiries about security issues involving the TOE.

6.3.2. Preparative Procedures (AGD_PRE.1)

The evaluator performs the CEM work units associated with the AGD_PRE.1 SAR. Specific requirements and EAs on the preparative documentation are identified (and where relevant are captured in the Guidance Documentation portions of the EAs) in the individual EAs for each SFR.

Preparative procedures are distributed to administrators and users (as appropriate) as part of the TOE, so that there is a reasonable guarantee that administrators and users are aware of the existence and role of the documentation in establishing and maintaining the evaluated configuration.

In addition, the evaluator performs the EAs specified below.

6.3.2.1. Evaluation Activity

The evaluator shall examine the Preparative procedures to ensure they include a description of how the administrator verifies that the operational environment can fulfil its role to support the security functionality (including the requirements of the Security Objectives for the Operational Environment specified in the Security Target).

The documentation should be in an informal style and should be written with sufficient detail and explanation that they can be understood and used by the target audience (which will typically include IT staff who have general IT experience but not necessarily experience with the TOE product itself).

6.3.2.2. Evaluation Activity

The evaluator shall examine the Preparative procedures to ensure they are provided for every Operational Environment that the product supports as claimed in the Security Target and shall adequately address all platforms claimed for the TOE in the Security Target.

6.3.2.3. Evaluation Activity

The evaluator shall examine the preparative procedures to ensure they include instructions to successfully install the TSF in each Operational Environment.

6.3.2.4. Evaluation Activity

The evaluator shall examine the preparative procedures to ensure they include instructions to manage the security of the TSF as a product and as a component of the larger operational environment.

6.3.2.5. Evaluation Activity

In addition the evaluator shall ensure that the following requirements are also met.

The preparative procedures must

- Include instructions to provide a protected administrative capability; and
- Identify TOE passwords that have default values associated with them and instructions shall be provided for how these can be changed.

6.4. Class ALC: Life-cycle Support

6.4.1. Labelling of the TOE (ALC_CMC.1)

When evaluating that the TOE has been provided and is labelled with a unique reference, the evaluator performs the work units as presented in the CEM.

6.4.2. TOE CM coverage (ALC_CMS.1)

When evaluating the developer's coverage of the TOE in their CM system, the evaluator performs the work units as presented in the CEM.

6.4.3. Systematic Flaw Remediation (ALC_FLR.3)

It is not uncommon for software applications needing to be updated due security flaws. Therefore, the response to potential security flaws must be clearly established, and comprehensive. There must be a means of providing information and solutions to users in a timely manner, using automated means. ALC_FLR.3 has been mandated to meet these requirements.

ALC_FLR.3 documentation is split into two inputs: guidance for TOE users and procedures for TOE developers. The ALC_FLR.3 guidance must be a public facing document. The ALC_FLR.3 procedures may be a public or private document.

The following table indicates, for each work unit in ALC_FLR.3, whether the [CEM] work unit is to be performed as written, or if it has been clarified by an Evaluation Activity. If clarification has been provided, a reference to this clarification is provided in the table.

Table 4. Mapping of ALC_FLR.3 CEM Work Units to Evaluation Activities

CEM ALC_FLR.3 Work Units	**Evaluator Activities**
ALC_FLR.3-1 The evaluator shall examine the flaw remediation procedures documentation to determine that it describes the procedures used to track all reported security flaws in each release of the TOE.	The evaluator shall perform the [CEM] activity as specified.
ALC_FLR.3-2 The evaluator shall examine the flaw remediation procedures to determine that the application of these procedures would produce a description of each security flaw in terms of its nature and effects.	The evaluator shall perform the [CEM] activity as specified.
ALC_FLR.3-3 The evaluator shall examine the flaw remediation procedures to determine that the application of these procedures would identify the status of finding a correction to each security flaw.	The evaluator shall perform the [CEM] activity as specified.
ALC_FLR.3-4 The evaluator shall check the flaw remediation procedures to determine that the application of these procedures would identify the corrective action for each security flaw.	The evaluator shall perform the [CEM] activity as specified.
ALC_FLR.3-5 The evaluator shall examine the flaw remediation procedures documentation to determine that it describes a means of providing the TOE users with the necessary information on each security flaw.	The evaluator shall perform the [CEM] activity as specified.

CEM ALC_FLR.3 Work Units	Evaluator Activities
ALC_FLR.3-6 The evaluator shall examine the flaw remediation procedures to determine that the application of these procedures would result in a means for the developer to receive from TOE user reports of suspected security flaws or requests for corrections to such flaws.	The evaluator shall perform the [CEM] activity as specified.
ALC_FLR.3-7 The evaluator shall examine the flaw remediation procedures to determine that the application of these procedures would result in a timely means of providing the registered TOE users who might be affected with reports about, and associated corrections to, each security flaw.	The evaluator shall perform the [CEM] activity as specified. The evaluator must ensure that the vendor has a defined set of timeframes for response to vulnerabilities. The evaluator must ensure that the vendor has rationale for those timeframes.
ALC_FLR.3-8 The evaluator shall examine the flaw remediation procedures to determine that the application of these procedures would result in automatic distribution of the reports and associated corrections to the registered TOE users who might be affected.	The evaluator shall perform the [CEM] activity as specified.
ALC_FLR.3-9 The evaluator shall examine the flaw remediation procedures to determine that the application of these procedures would help to ensure that every reported flaw is corrected.	The evaluator shall perform the [CEM] activity as specified.

CEM ALC_FLR.3 Work Units	Evaluator Activities
ALC_FLR.3-10 The evaluator shall examine the flaw remediation procedures to determine that the application of these procedures would help to ensure that the TOE users are issued remediation procedures for each security flaw.	The evaluator shall perform the [CEM] activity as specified.
ALC_FLR.3-11 The evaluator shall examine the flaw remediation procedures to determine that the application of these procedures would result in safeguards that the potential correction contains no adverse effects.	The evaluator shall perform the [CEM] activity as specified.
ALC_FLR.3-12 The evaluator shall examine the flaw remediation guidance to determine that the application of these procedures would result in a means for the TOE user to provide reports of suspected security flaws or requests for corrections to such flaws.	The evaluator shall perform the [CEM] activity as specified.
ALC_FLR.3-13 The evaluator shall examine the flaw remediation guidance to determine that it describes a means of enabling the TOE users to register with the developer.	The evaluator shall perform the [CEM] activity as specified.
ALC_FLR.3-14 The evaluator shall examine the flaw remediation guidance to determine that it identifies specific points of contact for user reports and enquiries about security issues involving the TOE.	The evaluator shall perform the [CEM] activity as specified.

6.5. Class ATE: Tests

6.5.1. Independent Testing - Conformance (ATE_IND.1)

The focus of the testing is to confirm that the requirements specified in the SFRs are being met. Additionally, testing is performed to confirm the functionality described in the TSS, as well as the dependencies on the Operational guidance documentation is accurate.

The evaluator performs the CEM work units associated with the ATE_IND.1 SAR. Specific testing requirements and EAs are captured for each SFR in Section 2, “Evaluation Activities for Mandatory SFRs”, Section 4, “Evaluation Activities for Selection-Based Requirements” and Section 3, “Evaluation Activities for Optional Requirements”.

6.6. Class AVA: Vulnerability Assessment

6.6.1. Vulnerability Survey (AVA_VAN.1)

While vulnerability analysis is inherently a subjective activity, a minimum level of analysis can be defined and some measure of objectivity and repeatability (or at least comparability) can be imposed on the vulnerability analysis process. In order to achieve such objectivity and repeatability it is important that the evaluator follows a set of well-defined activities, and documents their findings so others can follow their arguments and come to the same conclusions as the evaluator. While this does not guarantee that different evaluation facilities will identify exactly the same type of vulnerabilities or come to exactly the same conclusions, the approach defines the minimum level of analysis and the scope of that analysis, and provides Certification Bodies a measure of assurance that the minimum level of analysis is being performed by the evaluation facilities.

In order to meet these goals some refinement of the AVA_VAN.1 CEM work units is needed. The following table indicates, for each work unit in AVA_VAN.1, whether the CEM work unit is to be performed as written, or if it has been clarified by an Evaluation Activity. If clarification has been provided, a reference to this clarification is provided in the table.

Table 5. Mapping of AVA_VAN.1 CEM Work Units to Evaluation Activities

CEM AVA_VAN.1 Work Units	Evaluator Activities
AVA_VAN.1-1 The evaluator ***shall examine*** the TOE to determine that the test configuration is consistent with the configuration under evaluation as specified in the ST.	The evaluator shall perform the CEM activity as specified. The calibration of test resources specified in paragraph 1418 of the CEM applies to the tools listed in Section A.1.4. *If the iTC specifies any tools to be used in performing this analysis in section A.3.4, the following text is also included in this cell: "The calibration of test resources specified in paragraph 1418 of the CEM applies to the tools listed in Appendix A, Section A.1.4."*

CEM AVA_VAN.1 Work Units	Evaluator Activities
AVA_VAN.1-2 The evaluator ***shall examine*** the TOE to determine that it has been installed properly and is in a known state The evaluator shall perform the CEM activity as specified.	The evaluator shall perform the CEM activity as specified.
AVA_VAN.1-3 The evaluator ***shall examine*** sources of information publicly available to identify potential vulnerabilities in the TOE. Replace CEM work unit with activities outlined in Appendix A, Section A.1	Replace CEM work unit with activities outlined in Appendix A, Section A.1
AVA_VAN.1-4 The evaluator ***shall record*** in the ETR the identified potential vulnerabilities that are candidates for testing and applicable to the TOE in its operational environment.	Replace the CEM work unit with the analysis activities on the list of potential vulnerabilities in Appendix A, section A.1, and documentation as specified in Appendix A, Section A.3.
AVA_VAN.1-5 The evaluator ***shall devise*** penetration tests, based on the independent search for potential vulnerabilities.	Replace the CEM work unit with the activities specified in Appendix A, section A.2.

CEM AVA_VAN.1 Work Units	Evaluator Activities
AVA_VAN.1-6 The evaluator ***shall produce*** penetration test documentation for the tests based on the list of potential vulnerabilities in sufficient detail to enable the tests to be repeatable. The test documentation shall include: [loweralpha] * Identification of the potential vulnerability the TOE is being tested for; * Instructions to connect and setup all required test equipment as required to conduct the penetration test; * Instructions to establish all penetration test prerequisite initial conditions; * Instructions to stimulate the TSF; * Instructions for observing the behaviour of the TSF; * Descriptions of all expected results and the necessary analysis to be performed on the observed behaviour for comparison against expected results; * Instructions to conclude the test and establish the necessary post-test state for the TOE.	The CEM work unit is captured in Appendix A, Section A.3; there are no substantive differences.
AVA_VAN.1-7 The evaluator shall conduct penetration testing.	The evaluator shall perform the CEM activity as specified. See Appendix A, Section A.3, paragraph 110 for guidance related to attack potential for confirmed flaws.
AVA_VAN.1-8 The evaluator shall record the actual results of the penetration tests.	The evaluator shall perform the CEM activity as specified.
AVA_VAN.1-9 The evaluator shall report in the ETR the evaluator penetration testing effort, outlining the testing approach, configuration, depth and results.	Replace the CEM work unit with the reporting called for in Appendix A, Section A.3.

CEM AVA_VAN.1 Work Units	Evaluator Activities
AVA_VAN.1-10 The evaluator shall examine the results of all penetration testing to determine that the TOE, in its operational environment, is resistant to an attacker possessing a Basic attack potential.	This work unit is not applicable for Type 1 and Type 2 flaws (as defined in Appendix A, Section A.1), as inclusion in this Supporting Document by the iTC makes any confirmed vulnerabilities stemming from these flaws subject to an attacker possessing a Basic attack potential. This work unit is replaced for Type 3 and Type 4 flaws by the activities defined in Appendix A, Section A.2, paragraph 110.
AVA_VAN.1-11 The evaluator shall report in the ETR all exploitable vulnerabilities and residual vulnerabilities, detailing for each: [loweralpha] . Its source (e.g. CEM activity being undertaken when it was conceived, known to the evaluator, read in a publication); . The SFR(s) not met; . A description; . Whether it is exploitable in its operational environment or not (i.e. exploitable or residual). . The amount of time, level of expertise, level of knowledge of the TOE, level of opportunity and the equipment required to perform the identified vulnerabilities, and the corresponding values using the tables 3 and 4 of Annex B.4.	Replace the CEM work unit with the reporting called for in Appendix A, Section A.3.

Because of the level of detail required for the evaluation activities, the bulk of the instructions are contained in Appendix A, while an "outline" of the assurance activity is provided below.

6.6.1.1. Evaluation Activity (Documentation)

In addition to the activities specified by the CEM in accordance with Table 2 above, the evaluator shall perform the following activities.

The evaluator shall examine the documentation outlined below provided by the developer to confirm that it contains all required information. This documentation is in addition to the documentation already required to be supplied in response to the EAs listed previously.

The developer shall provide documentation identifying the list of software components that compose the TOE. Software components include applications, the operating system, virtualization

or conainerization services or support, and other major components that are independently identifiable and reusable (outside the TOE) such as a web server and protocol or cryptographic libraries. This additional documentation is merely a list of the name and version number of the components for identification and/or equivalency purposes.

6.6.1.2. Evaluation Activity

The evaluator formulates hypotheses in accordance with process defined in Appendix A.1. The evaluator documents the flaw hypotheses generated for the TOE in the report in accordance with the guidelines in Appendix A.3. The evaluator shall perform vulnerability analysis in accordance with Appendix A.2. The results of the analysis shall be documented in the report according to Appendix A.3.

7. Required Supplementary Information

This Supporting Document refers in various places to the possibility that 'required supplementary information' may need to be supplied as part of the deliverables for an evaluation. This term is intended to describe information that is not necessarily included in the Security Target or operational guidance, and that may not necessarily be public. Examples of such information could be entropy analysis, or description of a cryptographic key management architecture used in (or in support of) the TOE. The requirement for any such supplementary information will be identified in the relevant cPP.

The cPPs associated with this SD require an entropy analysis as described in [collaborative Protection Profile for Application Software, Appendix D].

8. References

- [CC1] Common Criteria for Information Technology Security Evaluation, Part 1: Introduction and General Model, CCMB-2017-04-001, Version 3.1 Revision 5, April 2017.
- [CC2] Common Criteria for Information Technology Security Evaluation, Part 2: Security Functional Components, CCMB-2017-04-002, Version 3.1 Revision 5, April 2017.
- [CC3] Common Criteria for Information Technology Security Evaluation, Part 3: Security Assurance Components, CCMB-2017-04-003, Version 3.1 Revision 5, April 2017.
- [CEM] Common Methodology for Information Technology Security Evaluation, Evaluation Methodology, CCMB-2017-04-004, Version 3.1 Revision 5, April 2017.
- [addenda] CC and CEM addenda, Exact Conformance, Selection-Based SFRs, Optional SFRs, Version 0.5, May 2017
- [cPP] collaborative Protection Profile for Application Software, Version 1.1, 2023-08-16
- [TLS Package SHAVS], Functional Package for Transport Layer Security (TLS) v1.1 The Secure Hash Algorithm Validation System (SHAVS)

Appendix A: Vulnerability Analysis

A.1. Sources of Vulnerability Information

CEM Work Unit AVA_VAN.1-3 has been supplemented in this Supporting Document to provide a better-defined set of flaws to investigate and procedures to follow based on this particular technology. Terminology used is based on the flaw hypothesis methodology, where the evaluation team hypothesizes flaws and then either proves or disproves those flaws (a flaw is equivalent to a "potential vulnerability" as used in the CEM). Flaws are categorized into four "types" depending on how they are formulated:

1. A list of flaw hypotheses applicable to the technology described by the cPP derived from public sources as documented in Section A.1.1, “Type 1 Hypotheses - Public-Vulnerability-based” - this fixed set has been agreed to by the iTC. Additionally, this will be supplemented with entries for a set of public sources (as indicated below) that are directly applicable to the TOE or its identified components (as defined by the process in Section A.1.1, “Type 1 Hypotheses - Public-Vulnerability-based” below); this is to ensure that the evaluators include in their assessment applicable entries that have been discovered since the cPP was published;
2. A list of flaw hypotheses contained in this document that are derived from lessons learned specific to that technology and other iTC input (that might be derived from other open sources and vulnerability databases, for example) as documented in Section A.1.2, “Type 2 Hypotheses - iTC-sourced”;
3. A list of flaw hypotheses derived from information available to the evaluators; this includes the baseline evidence provided by the vendor described in this Supporting Document (documentation associated with EAs, documentation described in Section 6.6.1.1, “Evaluation Activity (Documentation)”, <the iTC can remove the reference to Section 6.6.1.1, “Evaluation Activity (Documentation)” if no additional documentation is defined> documentation described in Section 7, “Required Supplementary Information”), as well as other information (public and/or based on evaluator experience) as documented in Section A.1.3, “Type 3 Hypotheses - Evaluation-Team-Generated”; and. A list of flaw hypotheses that are generated through the use of iTC-defined tools (e.g., nmap, protocol testers) and their application is specified in Section A.1.4, “Type 4 Hypotheses - Tool-Generated”.(If applicable) A list of flaw hypotheses that are generated through the use of TC-defined tools (e.g., nmap, fuzz testers) and their application as specified in section A.1.4.

A.1.1. Type 1 Hypotheses - Public-Vulnerability-based

The list of public sources of vulnerability information selected by the iTC is given in Section A.4.

The evaluators shall perform a search on the sources listed in Section A.4 to determine a list of potential flaw hypotheses that are more recent that the publication date of the cPP, and those that are specific to the TOE and its components as specified by the additional documentation mentioned above. Any duplicates – either in a specific entry, or in the flaw hypothesis that is generated from an entry from the same or a different source – can be noted and removed from consideration by the evaluation team.

The search criteria to be used when searching the sources published after the publication date of

the cPP shall include:

- Any protocols not listed above supported (through an SFR) by the TOE (these will include at least one of the remote management protocols (TLS, SSH))
- The TOE name (including appropriate model information as appropriate)

It is expected that all remotely exploitable vulnerabilities present in the application shall be considered as part of vulnerability assessment ("Application" is used to refer to the entire software application and is not limited to the claimed security functionality).

As part of type 1 flaw hypothesis generation for the specific components of the TOE, the evaluator shall also search the component manufacturer's websites to determine if flaw hypotheses can be generated on this basis (for instance, if security patches have been released for the version of the component being evaluated, the subject of those patches may form the basis for a flaw hypothesis).

As the search terms can contain proprietary information and there is a possibility that this information could be used by attackers to identify potential attack surfaces, there is no expectation that search terms containing proprietary information are published in any public-facing document.

A.1.2. Type 2 Hypotheses - iTC-sourced

Section A.5 contains the list of flaw hypothesis generated by the iTC for this technology that must be considered by the evaluation team as flaw hypotheses in performing the vulnerability assessment.

If the evaluators discover a Type 3 or Type 4 flaw that they believe should be considered as a Type 2 flaw in future versions of this cPP, they should work with their Certification Body to determine the appropriate means of submitting the flaw for consideration by the iTC.

A.1.3. Type 3 Hypotheses - Evaluation-Team-Generated

Type 3 flaws are formulated by the evaluator based on information presented by the product (through on-line help, product documentation and user guides, etc.) and product behaviour during the (functional) testing activities. The evaluator is also free to formulate flaws that are based on material that is not part of the baseline evidence (e.g., information gleaned from an Internet mailing list, or reading interface documentation on interfaces not included in the set provided by the developer), although such activities have the potential to vary significantly based upon the product and evaluation facility performing the analysis.

If the evaluators discover a Type 3 flaw that they believe should be considered as a Type 2 flaw in future versions of this cPP, they should work with their Certification Body to determine the appropriate means of submitting the flaw for consideration by the iTC.

A.1.4. Type 4 Hypotheses - Tool-Generated

Automated vulnerability scanning is a method where a specialized tool is applied to a specific attack surface in order to test it for the presence of known types of vulnerabilities. If this approach is technically feasible, the expectation is that the vulnerability scanning is conducted by the evaluator within a controlled test environment. The evaluator must ensure that the TOE is directly reachable and there is no network traffic filtering device, like a firewall, between the scanner and

the TOE.

Vulnerability scanners are known to produce recommendations for hardening along with possible vulnerabilities. There is no expectation that all recommendations produced by such tool must be implemented. However, each reported vulnerability must be investigated and resolved or classified as a false positive.

A.2. Process for Evaluator Vulnerability Analysis

As flaw hypotheses are generated from the activities described above, the evaluation team will disposition them; that is, attempt to prove, disprove, or determine the non-applicability of the hypotheses. This process is as follows.

The evaluator will refine each flaw hypothesis for the TOE and attempt to disprove it using the information provided by the developer or through penetration testing. During this process, the evaluator is free to interact directly with the developer to determine if the flaw exists, including requests to the developer for additional evidence (e.g., detailed design information, consultation with engineering staff); however, the CB should be included in these discussions. Should the developer object to the information being requested as being not compatible with the overall level of the evaluation activity/cPP and cannot provide evidence otherwise that the flaw is disproved, the evaluator prepares an appropriate set of materials as follows:

1. The source documents used in formulating the hypothesis, and why it represents a potential compromise against a specific TOE function;
2. An argument why the flaw hypothesis could not be proven or disproved by the evidence provided so far; and
3. The type of information required to investigate the flaw hypothesis further.

The Certification Body (CB) will then either approve or disapprove the request for additional information. If approved, the developer provides the requested evidence to disprove the flaw hypothesis (or, of course, acknowledge the flaw).

For each hypothesis, the evaluator will note whether the flaw hypothesis has been successfully disproved, successfully proven to have identified a flaw, or requires further investigation. It is important to have the results documented as outlined in Section A.3 below.

If the evaluator finds a flaw, the evaluator must report these flaws to the developer. All reported flaws must be addressed as follows:

If the developer confirms that the flaw exists and that it is exploitable at Basic Attack Potential, then a change is made by the developer, and the resulting resolution is agreed by the evaluator and noted as part of the evaluation report.

If the developer, the evaluator, and the CB agree that the flaw is exploitable only above Basic Attack Potential and does not require resolution for any other reason, then no change is made and the flaw is noted as a residual vulnerability in the CB-internal report (ETR).

If the developer and evaluator agree that the flaw is exploitable only above Basic Attack Potential, but it is deemed critical to fix because of technology-specific or cPP-specific aspects such as typical

use cases or operational environments, then a change is made by the developer, and the resulting resolution is agreed by the evaluator and noted as part of the evaluation report.

Disagreements between evaluator and vendor regarding questions of the existence of a flaw, its attack potential, or whether it should be deemed critical to fix are resolved by the CB.

Any testing performed by the evaluator shall be documented in the test report as outlined in Section A.3, "Reporting" below.

As indicated in Section A.3, "Reporting", Reporting, the public statement with respect to vulnerability analysis that is performed on TOEs conformant to the cPP is constrained to coverage of flaws associated with Types 1 and 2 (defined in Section A.1, "Sources of Vulnerability Information") flaw hypotheses only. The fact that the iTC generates these candidate hypotheses indicates these must be addressed.

For flaws of Types 3 and 4, each CB is responsible for determining what constitutes Basic Attack Potential for the purposes of determining whether a flaw is exploitable in the TOE's environment. The determination criteria shall be documented in the CB-internal report as specified in Section A.3, "Reporting". As this is a per-CB activity, no public claims are made with respect to the resistance of a particular TOE against flaws of Types 3 and 4; rather, the claim is that the activities outlined in this appendix were carried out, and the evaluation team and CB agreed that any residual vulnerabilities are not exploitable by an attacker with Basic Attack Potential.

A.3. Reporting

The evaluators shall produce two reports on the testing effort; one that is public-facing (that is, included in the non-proprietary evaluation report, which is a subset of the Evaluation Technical Report (ETR)), and the complete ETR that is delivered to the overseeing CB.

The public-facing report contains:

- The flaw identifiers returned when the procedures for searching public sources were followed according to instructions in the Supporting Document per Section A.1.1;
- A statement that the evaluators have examined the Type 1 flaw hypotheses specified in this Supporting Document in Section A.1.1 (i.e. the flaws listed in the previous bullet) and the Type 2 flaw hypotheses specified in this Supporting Document by the iTC in Section A.1.2.
- A statement that the evaluation team developed Types 3 and 4 flaw hypotheses in accordance with Section A.1.3, Section A.1.4, and Section A.2, and that no residual vulnerabilities exist that are exploitable by attackers with Basic Attack Potential as defined by the CB in accordance with the guidance in the CEM. It should be noted that this is just a statement about the "fact of" Types 3 and 4 flaw hypotheses being developed, and that no specifics about the number of flaws, the flaws themselves, or the analysis pertaining to those flaws will be included in the public-facing report.

No other information is provided in the public-facing report.

The internal CB report contains, in addition to the information in the public-facing report:

- a list of all of the flaw hypotheses generated (cf. AVA_VAN.1-4); the evaluator penetration testing

effort, outlining the testing approach, configuration, depth and results (cf. AVA_VAN.1-9);

- all documentation used to generate the flaw hypotheses (in identifying the documentation used in coming up with the flaw hypotheses, the evaluation team must characterize the documentation so that a reader can determine whether it is strictly required by this Supporting Document, and the nature of the documentation (design information, developer engineering notebooks, etc.));
- the evaluator shall report all exploitable vulnerabilities and residual vulnerabilities, detailing for each:
  - its source (e.g. CEM activity being undertaken when it was conceived, known to the evaluator, read in a publication);
  - the SFR(s) not met;
  - a description;
  - whether it is exploitable in its operational environment or not (i.e. exploitable or residual).
  - the amount of time, level of expertise, level of knowledge of the TOE, level of opportunity and the equipment required to perform the identified vulnerabilities (cf. AVA_VAN.1-11);
- how each flaw hypothesis was resolved (this includes whether the original flaw hypothesis was confirmed or disproved, and any analysis relating to whether a residual vulnerability is exploitable by an attacker with Basic Attack Potential) (cf. AVA_VAN.1-10); and
- in the case that actual testing was performed in the investigation (either as part of flaw hypothesis generation using tools specified by the iTC in Section A.1.4, or in proving/disproving a particular flaw) the steps followed in setting up the TOE (and any required test equipment); executing the test; post-test procedures; and the actual results (to a level of detail that allow repetition of the test, including the following:
  - identification of the potential vulnerability the TOE is being tested for;
  - instructions to connect and setup all required test equipment as required to conduct the penetration test;
  - instructions to establish all penetration test prerequisite initial conditions;
  - instructions to stimulate the TSF;
  - instructions for observing the behaviour of the TSF;
  - descriptions of all expected results and the necessary analysis to be performed on the observed behaviour for comparison against expected results;
  - instructions to conclude the test and establish the necessary post-test state for the TOE. (cf. AVA_VAN.1-6, AVA_VAN.1-8).

A.4. Public Vulnerability Sources

The following sources of public vulnerabilities are sources for the iTC to consider in both formulating the specific list of flaws to be investigated by the evaluators, as well as to reference in directing the evaluators to perform keyword searches during the evaluation of a specific TOE.

a. NIST National Vulnerabilities Database (can be used to access CVE and US-CERT databases

identified below): https://web.nvd.nist.gov/view/vuln/search

b. Common Vulnerabilities and Exposures: https://cve.mitre.org/cve/ https://www.cvedetails.com/vulnerability-search.php

c. US-CERT: https://www.kb.cert.org/vuls/html/search

d. Exploit / Vulnerability Search Engine: https://www.exploitsearch.net

e. SecurITeam Exploit Search: https://www.securiteam.com

f. Tenable Network Security: https://nessus.org/plugins/index.php?view=search

g. Tipping Point Zero Day Initiative: https://www.zerodayinitiative.com/advisories

h. Offensive Security Exploit Database: https://www.exploit-db.com/

i. Rapid7 Vulnerability Database: https://www.rapid7.com/db/vulnerabilities

A.5. Additional Flaw Hypothese

No entries are currently defined for this list.

Appendix B: Glossary

For definitions of standard CC terminology see [CC1].

Assurance

Grounds for confidence that a TOE meets the SFRs [CC1].

Required Supplementary Information

Information that is not necessarily included in the Security Target or operational guidance, and that may not necessarily be public. Examples of such information could be entropy analysis, or description of a cryptographic key management architecture used in (or in support of) the TOE. The requirement for any such supplementary information will be identified in the relevant cPP (see description in Section 6).

Target of Evaluation

A set of software, firmware and/or hardware possibly accompanied by guidance. [CC1]

TOE Security Functionality (TSF)

A set consisting of all hardware, software, and firmware of the TOE that must be relied upon for the correct enforcement of the SFRs. [CC1]

TSF Data

Data for the operation of the TSF upon which the enforcement of the requirements relies.

Appendix C: Acronyms

Table 6. Acronyms

Acronym	Meaning
AES	Advanced Encryption Standard
CA	Certificate Authority
CBC	Cipher Block Chaining
CCM	Counter with CBC-Message Authentication Code
cPP	Collaborative protection Profile
DSA	Digital Signature Algorithm
EA	Evaluation Activity
ECDSA	Elliptic Curve Digital Signature Algorithm
FIPS	Federal Information Processing Standards
GCM	Galois Counter Mode
HMAC	Keyed-Hash Message Authentication Code
KDF	Key Derivation Function
NIST	National Institute of Standards and Technology
PP	Protection Profile
RBG	Random Bit Generator
RSA	Rivest Shamir Adleman Algorithm
SHA	Secure Hash Algorithm
SFR	Security Functional Requirement
ST	Security Target
TOE	Target of Evaluation
TSF	TOE Security Functionality
TSS	TOE Summary Specification