Protection Profile for General-Purpose Computing Platforms

Version: 1.0
2021-02-17
National Information Assurance Partnership

Revision History

Version	Date	Comment
Round 1	2020-11-09	Started
1.0	2020-11-09	Initial release

Contents

1 Introduction

1.1 Overview

The scope of this Protection Profile (PP) is to describe the security functionality of General-Purpose Computing Platforms in terms of and to define functional and assurance requirements for such products.

A platform is a collection of hardware devices and firmware that provide the functional capabilities and services needed by tenant software. Such components typically include embedded controllers, trusted platform modules, management controllers, host processors, network interface controllers, graphics processing units, flash memory, storage controllers, storage devices, boot firmware, runtime firmware, human interface devices, and a power supply.

1.2 Terms

The following sections list Common Criteria and technology terms used in this document.

1.2.1 Common Criteria Terms

Assurance	Grounds for confidence that a TOE meets the SFRs [CC].
Base Protection Profile (Base-PP)	Protection Profile used as a basis to build a PP-Configuration.
Common Criteria (CC)	Common Criteria for Information Technology Security Evaluation (International Standard ISO/IEC 15408).
Common Criteria Testing Laboratory	Within the context of the Common Criteria Evaluation and Validation Scheme (CCEVS), an IT security evaluation facility, accredited by the National Voluntary Laboratory Accreditation Program (NVLAP) and approved by the NIAP Validation Body to conduct Common Criteria-based evaluations.
Common Evaluation Methodology (CEM)	Common Evaluation Methodology for Information Technology Security Evaluation.
Distributed TOE	A TOE composed of multiple components operating as a logical whole.
Operational Environment (OE)	Hardware and software that are outside the TOE boundary that support the TOE functionality and security policy.
Protection Profile (PP)	An implementation-independent set of security requirements for a category of products.
Protection Profile Configuration (PP-Configuration)	A comprehensive set of security requirements for a product type that consists of at least one Base-PP and at least one PP-Module.
Protection Profile Module (PP-Module)	An implementation-independent statement of security needs for a TOE type complementary to one or more Base Protection Profiles.
Security Assurance Requirement (SAR)	A requirement to assure the security of the TOE.
Security Functional Requirement (SFR)	A requirement for security enforcement by the TOE.
Security Target (ST)	A set of implementation-dependent security requirements for a specific product.
TOE Security Functionality (TSF)	The security functionality of the product under evaluation.
TOE Summary Specification (TSS)	A description of how a TOE satisfies the SFRs in an ST.
Target of Evaluation (TOE)	The product under evaluation.

1.2.2 Technical Terms

Address Space Layout Randomization (ASLR)	An anti-exploitation feature which loads memory mappings into unpredictable locations. ASLR makes it more difficult for an attacker to redirect control to code that they have introduced into the address space of a process.
Administrator	An administrator is responsible for management activities, including setting policies that are applied by the enterprise on the operating system. This administrator could be acting remotely through a management server, from which the system receives configuration policies. An administrator can enforce settings on the system which cannot be overridden by non-administrator users.
Application (app)	Software that runs on a platform and performs tasks on behalf of the user or owner of the platform, as well as its supporting documentation.
Application Programming Interface (API)	A specification of routines, data structures, object classes, and variables that allows an application to make use of services provided by another software component, such as a library. APIs are often provided for a set of libraries included with the platform.
Credential	Data that establishes the identity of a user, e.g. a cryptographic key or password.
Critical Security Parameters (CSP)	Information that is either user or system defined and is used to operate a cryptographic module in processing encryption functions including cryptographic keys and authentication data, such as passwords, the disclosure or modification of which can compromise the security of a cryptographic module or the security of the information protected by the module.
DAR Protection	Countermeasures that prevent attackers, even those with physical access, from extracting data from non-volatile storage. Common techniques include data encryption and wiping.
Data Execution Prevention (DEP)	An anti-exploitation feature of modern operating systems executing on modern computer hardware, which enforces a non-execute permission on pages of memory. DEP prevents pages of memory from containing both data and instructions, which makes it more difficult for an attacker to introduce and execute code.
Developer	An entity that writes OS software. For the purposes of this document, vendors and developers are the same.
General Purpose Operating System	A class of OSes designed to support a wide-variety of workloads consisting of many concurrent applications or services. Typical characteristics for OSes in this class include support for third-party applications, support for multiple users, and security separation between users and their respective resources. General Purpose Operating Systems also lack the real-time constraint that defines Real Time Operating Systems (RTOS). RTOSes typically power routers, switches, and embedded devices.
Host-based Firewall	A software-based firewall implementation running on the OS for filtering inbound and outbound network traffic to and from processes running on the OS.
Operating System (OS)	Software that manages physical and logical resources and provides services for applications. The terms *TOE* and *OS* are interchangeable in this document.
Personally Identifiable Information (PII)	Any information about an individual maintained by an agency, including, but not limited to, education, financial transactions, medical history, and criminal or employment history and information which can be used to distinguish or trace an individual's identity, such as their name, social security number, date and place of birth, mother's maiden name, biometric records, etc., including any other personal information which is linked or linkable to an individual.
Sensitive Data	Sensitive data may include all user or enterprise data or may be specific application data such as PII, emails, messaging, documents, calendar items, and contacts. Sensitive data must minimally include credentials and keys. Sensitive data shall be identified in the OS's TSS by the ST author.
User	A user is subject to configuration policies applied to the operating system by administrators. On some systems under certain configurations, a normal user can temporarily elevate privileges to that of an administrator. At that time, such a user should be considered an administrator.
Virtual Machine (VM)	Blah Blah Blah

1.3 Compliant Targets of Evaluation

A general-purpose computing platform is a hardware device that it capable of hosting more than one different operating system, virtualization system, or bare-metal application. Typical platform implementations include--but are not limited to--servers, PC clients, laptops, and tablets.

1.3.1 TOE Boundary

Figure 1: General TOE

TODO: Add a description of what is in the TOE ahd what is outside. E.g. Tenant software is outside. Is everything else inside?

1.3.2 TOE Operational Environment

The TOE has no platform since it is itself a platform. But the TOE does have an operational environment. The OE consists of the physical environment in which the TOE operates (e.g., data center, vehicle, outdoors) and any networks to which the TOE may be connected.

1.4 Use Cases

TODO: Requirements in this Protection Profile are designed to address the security problems in at least the following use cases. Is the major distinction going to be related to physical protections?

[USE CASE 1] Server-Class Platform, Data Center-based
Server-class hardware in a data center is assumed to be physically protected by the operational environment.

[USE CASE 2] Server-Class Platform, Field-based (Edge servers)
Field-based (edge) servers may be physically protected or not.

[USE CASE 3] Thin/Zero Clients
Client platforms that run thin client operating systems. It would be great if we could fold this into one or both of the client platform use cases. At the boot firmware level, are they really any different.

[USE CASE 4] Portable Clients, (laptopns, tablets)
High-assurance, and normal. THe difference between high-assurance and normal would likely be the degree of physical protection implemented by the platform.

[USE CASE 5] Desktop clients
High-assurance, and normal

[USE CASE 6] IoT Devices
IoT devices are field-located devices without human interfaces when in normal operation. In order to qualify for evaluation under this PP, the device must meet the basic criteria for a general-purpose platform.

[USE CASE 7] Network Devices
Network devices are not addressed by this PP. They are addressed by the collaborative Protection Profile for Network Devices. Server-class platforms that run virtualized network devices fall under one of the server-class platform use cases.

[USE CASE 8] Mobile Devices
Mobile devices, as defined by the Protection Profile for Mobile Device Fundamentals, are not addressed by this PP.

2 Conformance Claims

Conformance Statement

An ST must claim exact conformance to this PP, as defined in the CC and CEM addenda for Exact Conformance, Selection-Based SFRs, and Optional SFRs (dated May 2017).

CC Conformance Claims

This PP is conformant to Parts 2 (extended) and 3 (conformant) of Common Criteria Version 3.1, Revision 5.

PP Claim

This PP does not claim conformance to any Protection Profile.

Package Claim

This PP does not claim conformance to any packages.

3 Security Problem Description

The security problem is described in terms of the threats that the GPCP is expected to address, assumptions about the operational environment, and any organizational security policies that the GPCP is expected to enforce.

The platform has three major security responsibilities:

- ensuring the integrity of its own firmware
- ensuring that it is resilient
- providing security services to tenant workloads

These responsibilities manifest as protecting:

- Platform firmware
- Platform firmware updates
- Tenant initialization (boot)

3.1 Threats

T.PHYSICAL
An attacker with physical access might be able to compromise TOE integrity, subvert TOE protections, or access tenant data through hardware attacks such as probing, physical manipulation, fault-injection, side-channel analysis, environmental stress, or activating disabled features or pre-delivery services.

T.SIDE_CHANNEL_LEAKAGE
An attacker running in a tenant context might be able to leverage physical effects caused by the operation of the TOE to derive sensitive information about other tenants or the TOE.

T.PERSISTENCE
An attacker might be able to establish a permanent presence on the TOE in firmware. This could result in permanent compromise of tenant information, as well as TOE updates. This threat does not encompass attacker presence in tenant software, as tenant software is not part of the TOE.

T.UPDATE_COMPROMISE
An attacker may attempt to provide a compromised update of TOE firmware. Such updates can undermine the security functionality of the device if they are unauthorized, unauthenticated, or are improperly validated using non-secure or weak cryptography.

T.SECURITY_FUNCTIONALITY_FAILURE
An attacker could leverage failed or compromised security functionality to access, change, or modify tenant data, TOE data, or other security functionality of the device.

T.TENANT-BASED_ATTACK
An attacker running software as a tenant can attempt to access or modify TOE firmware or functionality. Note that direct tenant attacks against other tenants are not encompassed by this threat as they out of scope.

T.REMOTE_ATTACK
An attacker from off the TOE can attempt to compromise the TOE through a network interface connected to an active TOE component, such as a management subsystem.

T.UNAUTHORIZED_RECONFIGURATION
An attacker might be able to modify the configuration of the TOE and alter its functionality. This might include, activating dormant subsystems, disabling hardware assists, or altering boot-time behaviors.

T.UNAUTHORIZED_PLATFORM_ADMINISTRATOR
An attacker might be able to attain platform administrator status by defeating or bypassing authentication measures.

3.2 Assumptions

A.PHYSICAL_PROTECTION
The TOE is assumed to be physically protected in its operational environment and thus is not subject to physical attacks that could compromise its security or its ability to support the security of tenant workloads.

A.ROT_INTEGRITY
The TOE includes one or more Roots of Trust composed of TOE firmware, hardware, and pre-installed credentials. Roots of Trust are assumed to be free of malicious capabilities as their integrity cannot be verified.

A.TRUSTED_ADMIN
The administrator of the TOE is not careless, willfully negligent or hostile.

3.3 Organizational Security Policies

If the OS is bound to a directory or management server, the configuration of the OS software must be capable of adhering to the enterprise security policies distributed by them.

4 Security Objectives

4.1 Security Objectives for the TOE

O.ACCOUNTABILITY
Conformant OSes ensure that information exists that allows administrators to discover unintentional issues with the configuration and operation of the operating system and discover its cause. Gathering event information and immediately transmitting it to another system can also enable incident response in the event of system compromise.

O.INTEGRITY
Conformant OSes ensure the integrity of their update packages. OSes are seldom if ever shipped without errors, and the ability to deploy patches and updates with integrity is critical to enterprise network security. Conformant OSes provide execution environment-based mitigations that increase the cost to attackers by adding complexity to the task of compromising systems.

O.MANAGEMENT
To facilitate management by users and the enterprise, conformant OSes provide consistent and supported interfaces for their security-relevant configuration and maintenance. This includes the deployment of applications and application updates through the use of platform-supported deployment mechanisms and formats, as well as providing mechanisms for configuration and application execution control.

O.PROTECTED_STORAGE
To address the issue of loss of confidentiality of credentials in the event of loss of physical control of the storage medium, conformant OSes provide data-at-rest protection for credentials. Conformant OSes also provide access controls which allow users to keep their files private from other users of the same system.

O.PROTECTED_COMMS
To address both passive (eavesdropping) and active (packet modification) network attack threats, conformant OSes provide mechanisms to create trusted channels for CSP and sensitive data. Both CSP and sensitive data should not be exposed outside of the platform.

4.2 Security Objectives for the Operational Environment

The following security objectives for the operational environment assist the OS in correctly providing its security functionality. These track with the assumptions about the environment.

OE.PLATFORM
The OS relies on being installed on trusted hardware.

OE.PROPER_USER
The user of the OS is not willfully negligent or hostile, and uses the software within compliance of the applied enterprise security policy. Standard user accounts are provisioned in accordance with the least privilege model. Users requiring higher levels of access should have a separate account dedicated for that use.

OE.PROPER_ADMIN
The administrator of the OS is not careless, willfully negligent or hostile, and administers the OS within compliance of the applied enterprise security policy.

4.3 Security Objectives Rationale

This section describes how the assumptions, threats, and organization security policies map to the security objectives.

Table 1: Security Objectives Rationale

Threat, Assumption, or OSP	Security Objectives	Rationale
T.PHYSICAL	O.OBJECTIVE	The threat T.PHYSICAL is countered by O.OBJECTIVE as this provides for integrit transmitted data.
T.SIDE_CHANNEL_LEAKAGE	O.OBJECTIVE	The threat T.SIDE_CHANNEL_LEAKAGE countered by O.OBJECTIVE as this provid integrity of transmitted data.
T.PERSISTENCE	O.OBJECTIVE	The threat T.PERSISTENCE is countered O.OBJECTIVE as this provides for integrit transmitted data.
T.UPDATE_COMPROMISE	O.OBJECTIVE	The threat T.UPDATE_COMPROMISE is c by O.OBJECTIVE as this provides for inte transmitted data.
T.SECURITY_FUNCTIONALITY_FAILURE	O.OBJECTIVE	The threat T.SECURITY_FUNCTIONALITY_FAILURE countered by O.OBJECTIVE as this provid integrity of transmitted data.
T.TENANT-BASED_ATTACK	O.OBJECTIVE	The threat T.TENANT-BASED_ATTACK is countered by O.OBJECTIVE as this provid integrity of transmitted data.
T.REMOTE_ATTACK	O.OBJECTIVE	The threat T.REMOTE_ATTACK is counte O.OBJECTIVE as this provides for integrit

		transmitted data.
T.UNAUTHORIZED_RECONFIGURATION	O.OBJECTIVE	The threat T.UNAUTHORIZED_RECONFIGURATION countered by O.OBJECTIVE as this provid integrity of transmitted data.
T.UNAUTHORIZED_PLATFORM_ADMINISTRATOR	O.OBJECTIVE	The threat T.UNAUTHORIZED_PLATFORM_ADMINI is countered by O.OBJECTIVE as this prov integrity of transmitted data.
A.PHYSICAL_PROTECTION	OE.PHYSICAL_PROTECTION	The operational environment objective OE.PHYSICAL_PROTECTION is realized t A.PHYSICAL_PROTECTION.
A.ROT_INTEGRITY	OE.ROT_INTEGRITY	The operational environment objective OE.SUPPLY_CHAIN is realized through A.ROT_INTEGRITY.
A.TRUSTED_ADMIN	OE.TRUSTED_ADMIN	The operational environment objective OE.TRUSTED_ADMIN is realized through A.TRUSTED_ADMIN.
	O.MANAGEMENT	The organizational security policy P.ENTI is enforced through the objective O.MANAGEMENT as this objective repres the enterprise and user assert manageme the OS.

5 Security Requirements

This chapter describes the security requirements which have to be fulfilled by the product under evaluation. Those requirements comprise functional components from Part 2 and assurance components from Part 3 of [CC]. The following conventions are used for the completion of operations:

- **Refinement** operation (denoted by **bold text** or ~~strikethrough text~~): is used to add details to a requirement (including replacing an assignment with a more restrictive selection) or to remove part of the requirement that is made irrelevant through the completion of another operation, and thus further restricts a requirement.
- **Selection** (denoted by *italicized text*): is used to select one or more options provided by the [CC] in stating a requirement.
- **Assignment** operation (denoted by *italicized text*): is used to assign a specific value to an unspecified parameter, such as the length of a password. Showing the value in square brackets indicates assignment.
- **Iteration** operation: is indicated by appending the SFR name with a slash and unique identifier suggesting the purpose of the operation, e.g. "/EXAMPLE1."

5.1 Security Functional Requirements

5.1.1 Class: Protection of the TSF (FPT)

FPT_ROT_EXT.1 Platform Integrity Root

FPT_ROT_EXT.1.1

The integrity of platform firmware shall be rooted in [**selection**:

- *code or data written to immutable memory or storage,*
- *credentials held in immutable storage or protected off-platform,*
- *a separate management controller that is itself rooted in a mechanism that meets this requirement,*
- *integrity measurements held securely in an on-platform dedicated security component,*
- *integrity measurements held securely by an off-platform third-party,*
- *a combination of the above*

].

Application Note: The integrity of the first platform firmware that executes must be established by one or more of the selections in FPT_ROT_EXT.1.1. The integrity of subsequently executed platform firmware must be traceable back to this root or to some other root. This SFR should be iterated for additional roots (for example, a management controller or firmware executed from an add-in card).

If a platform does not have immutable storage, firmware integrity can be established through cryptographic means implemented in code that has had its integrity ensured by a NIST SP800-147/B-conformant update mechanism as defined in FPT_TUD_EXT.1.

Alternately, integrity can be established through secure storage of known-good measurements in an on-platform dedicated security component, such as a Trusted Platform Module (TPM), or through a trusted measurement certification process involving an off-platform third-party, such as those defined in NIST SP 800-155.

FPT_ROT_EXT.1.2

The TOE shall take the following actions if an integrity check specified in FPT_ROT_EXT.1.1 fails: [**selection**:

- *Reboot the platform and try again,*
- *Stop all execution and shut down,*
- *Restore the Integrity Root to a known-good state through a Recovery process as specified in FPT_RVR_EXT.1,*
- *Log an audit event and notify an administrator,*
- *Do nothing and continue booting the system*

]

Application Note: The ability to generate audit events would generally be limited to server platforms with management controllers.

Evaluation Activities

FPT_ROT_EXT.1:
TSS
The evaluator shall verify that the TSS describes the means by which initial integrity of platform firmware is established, and that the means are consistent with the selection(s) made in FPT_ROT_EXT.1.
Guidance
The evaluator shall examine the operational guidance to ensure that it describes the actions taken in case of failure to establish the integrity of the platform firmware root. If the actions are configurable, the guidance shall explain how they are configured.

FPT_ROT_EXT.2 Platform Integrity Extension

FPT_ROT_EXT.2.1

The integrity of all platform firmware outside of the platform integrity root specified in FPT_ROT_EXT.1 shall be verified prior to execution or use through:

[**selection**:

- *the code or data being executed/used from immutable storage,*
- *computation and verification of a hash using previously verified code/data,*
- *verification of a digital signature using previously verified code/data,*
- *measurement and verification using previously verified platform code/data,*
- *measurement and verification by a 3rd-party off-platform component*

].

Application Note: This requirement specifies the means for extending the initial integrity of platform firmware established by FPT_ROT_EXT.1.1 to subsequently executed platform firmware.

Integrity of code and data written to immutable storage is assured. Otherwise, integrity must be extended through cryptograpihic means: either through hashes or digitial signatures computed and verified by firmware that has itself been previously had its integrity verified or ensured. Verificcation can be performed by TOE components such as management controllers or off-TOE 3rd-party entities.

FPT_ROT_EXT.2.2

The TOE shall take the following actions if an integrity check specified in FPT_ROT_EXT.2.1 fails: [**selection**:

- *Reboot the platform and try again,*
- *Stop all execution and shut down,*
- *Restore the firmware image to a known-good state through a Recovery Process specified in FPT_RVR_EXT.1,*
- *Log an audit event and notofy an administrator,*
- *Do nothing*

]

Application Note: The ability to generate audit events would generally be limited to server platforms with management controllers.

Evaluation Activities

FPT_ROT_EXT.2:
TSS
The evaluator shall verify that the TSS describes the means by which initial integrity of platform firmware is extended to other platform components, and that the means are consistent with the selection(s) made in FPT_ROT_EXT.2.
Guidance
The evaluator shall examine the operational guidance to ensure that it describes the actions taken in case of failure to establish the integrity of the platform firmware root. If the actions are configurable, the guidance shall explain how they are configured.
Tests
Is there a reasonable way to test this?

FPT_PPF_EXT.1

FPT_PPF_EXT.1.1

The TSF shall allow modification of platform firmware only through the update mechanisms described in FPT_TUD_EXT.1.

Application Note: Platform firmware must be modifiable only through one of the secure update mechanisms specified in FPT_TUD_EXT.1. If the update mechanism itself is implemented in platform firmware, then naturally, it must itself also be modifiable only through the secure update mechanism.

Configuration data used by platform firmware that is stored in nonvolatile memory is not included in these protections.

Software portions of TSF and data critical for ensuring the integrity of the TSF are included in these protections. Specifically, this includes the key store and the signature verification algorithm used by the update mechanisms.

Evaluation Activities

FPT_PPF_EXT.1:
TSS
The evaluator shall examine the TSS to ensure that it explains how the various areas of platform firmware and critical data are protected from modification outside of the platform firmware update mechanism described in FPT_TUD_EXT.1. If the TOE implements a secure update mechanism as specified in FPT_TUD_EXT.2, then the evauator shall ensure that the TSS describes specically how the signature verification code and key store is protected from update outside of the secure platform firmware update mechanism.
Guidance
The evaluator shall check the operational guidance to ensure that there are instructions for how to securely modify the platform firmware and critical data using a mechanism specified in FPT_TUD_EXT.2 or FPT_TUD_EXT.3.
Tests

- ***Test 1:*** *Using the information contained in the TSS and in other evidences (AGD, interface specifications), the evaluator shall attempt to overwrite or modify the platform firmware in the system while bypassing the authenticated update (e.g., using a modified Linux boot loader such as GRUB that attempts to write to the memory where the platform firmware is stored). The test succeeds if the attempts to overwrite platform firmware fail.*

- ***Test 2:*** *If the TOE implements an update mechanism, using the information contained in the TSS and in other evidences (AGD, interface specifications), the evaluator shall attempt to overwrite or modify the TSF components (e.g., key store, cryptographic algorithms). The test succeeds if the attempts to overwrite platform firmware fail.* ***QQQQ: I don't like that these tests are so open-ended. Is there a way to tighen them up?***

FPT_TUD_EXT.1

FPT_TUD_EXT.1.1

The TSF shall [**selection**:

- *make no provision for platform firmware update,*
- *implement a secure platform firmware update mechanism as described in FPT_TUD_EXT.2,*
- *implement a secure local platform firmware update mechanism described in FPT_TUD_EXT.3*

].

Application Note: The secure platform firmware update mechanism is used for verifying the authenticity and integrity of platform firmware updates. If platform firmware is immutable (not updateable by any means) then the ST author must select "make no provision for platform firmware update." If platform firmware is modifiable only through a local update requiring physical presence at the platform, then the ST author must select "implement a secure local update process..." and include FPT_TUD_EXT.3 in the ST. If the platform implements an update mechanism that does not require physical presence at the platform, then the ST author selects "implement a secure platform update mechanism..." and include FPT_TUD_EXT.2 in the ST.

Evaluation Activities

FPT_TUD_EXT.1:
TSS
If the ST author selects "make no provision for platform firmware update," then the evaluator shall examine the TSS to ensure that it explains all ways of modifying platform firmware in the absence of any provided mechanism. For example, breaking open the case and prying a chip off the motherboard and then reprogramming the chip. The purpose of this activity is to ensure that the TOE does not implement a local update mechanism that does not meet the requirements of FPT_TUD_EXT.3.

This requirement is met if the platform implements no means for updating platform firmware and the TSS describes a method for updating or replacing platform firmware that involves potentially destroying or damaging the TOE or some of its components. ***(QQQQ: The point here is to distinguish between secure local update and unauthorized physical update.)***

If the ST author selects "implement a secure platform update mechanism..." then this requirement is satisfied if FPT_TUD_EXT.2 is satisfied.

If the ST author selects "implement a secure local platform update mechanism..." then this requirement is satisfied if FPT_TUD_EXT.3 is satisfied.

5.1.2 TOE Security Functional Requirements Rationale

The following rationale provides justification for each security objective for the TOE, showing that the SFRs are suitable to meet and achieve the security objectives:

Table 2: SFR Rationale

OBJECTIVE	ADDRESSED BY	RATIONALE
O.ACCOUNTABILITY	FAU_GEN.1	'cause FAU_GEN.1 is awesome
	FTP_ITC_EXT.1	Cause FTP reasons
O.INTEGRITY	FPT_SBOP_EXT.1	For reasons
	FPT_ASLR_EXT.1	ASLR For reasons
	FPT_TUD_EXT.1	For reasons
	FPT_TUD_EXT.2	For reasons
	FCS_COP.1/HASH	For reasons
	FCS_COP.1/SIGN	For reasons
	FCS_COP.1/KEYHMAC	For reasons
	FPT_ACF_EXT.1	For reasons
	FPT_SRP_EXT.1	For reasons
	FIA_X509_EXT.1	For reasons
	FPT_TST_EXT.1	For reasons

	FTP_ITC_EXT.1	For reasons
	FPT_W^X_EXT.1	For reasons
	FIA_AFL.1	For reasons
	FIA_UAU.5	For reasons
O.MANAGEMENT	FMT_MOF_EXT.1	For reasons
	FMT_SMF_EXT.1	For reasons
	FTA_TAB.1	For reasons
	FTP_TRP.1	For reasons
O.PROTECTED_STORAGE	FCS_STO_EXT.1, FCS_RBG_EXT.1, FCS_COP.1/ENCRYPT, FDP_ACF_EXT.1	Rationale for a big chunk
O.PROTECTED_COMMS	FCS_RBG_EXT.1, FCS_CKM.1, FCS_CKM.2, FCS_CKM_EXT.4, FCS_COP.1/ENCRYPT, FCS_COP.1/HASH, FCS_COP.1/SIGN, FCS_COP.1/HMAC, FDP_IFC_EXT.1, FIA_X509_EXT.1, FIA_X509_EXT.2, FTP_ITC_EXT.1	Rationale for a big chunk

5.2 Security Assurance Requirements

The Security Objectives in Section 4 Security Objectives were constructed to address threats identified in Section 3.1 Threats. The Security Functional Requirements (SFRs) in Section 5.1 Security Functional Requirements are a formal instantiation of the Security Objectives. The PP identifies the Security Assurance Requirements (SARs) to frame the extent to which the evaluator assesses the documentation applicable for the evaluation and performs independent testing.
This section lists the set of SARs from CC part 3 that are required in evaluations against this PP. Individual Assurance Activities to be performed are specified both in Section 5 Security Requirements as well as in this section.
The general model for evaluation of OSs against STs written to conform to this PP is as follows:
After the ST has been approved for evaluation, the ITSEF will obtain the OS, supporting environmental IT, and the administrative/user guides for the OS. The ITSEF is expected to perform actions mandated by the Common Evaluation Methodology (CEM) for the ASE and ALC SARs. The ITSEF also performs the Assurance Activities contained within Section 5 Security Requirements, which are intended to be an interpretation of the other CEM assurance requirements as they apply to the specific technology instantiated in the OS. The Assurance Activities that are captured in Section 5 Security Requirements also provide clarification as to what the developer needs to provide to demonstrate the OS is compliant with the PP.

5.2.1 Class ASE: Security Target

As per ASE activities defined in .

5.2.2 Class ADV: Development

The information about the OS is contained in the guidance documentation available to the end user as well as the TSS portion of the ST. The OS developer must concur with the description of the product that is contained in the TSS as it relates to the functional requirements. The Assurance Activities contained in Section 5.1 Security Functional Requirements should provide the ST authors with sufficient information to determine the appropriate content for the TSS section.

ADV_FSP.1 Basic Functional Specification (ADV_FSP.1)

The functional specification describes the TSFIs. It is not necessary to have a formal or complete specification of these interfaces. Additionally, because OSs conforming to this PP will necessarily have interfaces to the Operational Environment that are not directly invokable by OS users, there is little point specifying that such interfaces be described in and of themselves since only indirect testing of such interfaces may be possible. For this PP, the activities for this family should focus on understanding the interfaces presented in the TSS in response to the functional requirements and the interfaces presented in the AGD documentation. No additional “functional specification” documentation is necessary to satisfy the assurance activities specified. The interfaces that need to be evaluated are characterized through the information needed to perform the assurance activities listed, rather than as an independent, abstract list.

Developer action elements:

ADV_FSP.1.1D

The developer shall provide a functional specification.

Content and presentation elements:

ADV_FSP.1.2C

The developer shall provide a tracing from the functional specification to the SFRs.

Application Note: As indicated in the introduction to this section, the functional specification is comprised of the information contained in the AGD_OPE and AGD_PRE documentation. The developer may reference a website accessible to application developers and the evaluator. The assurance activities in the functional requirements point to evidence that should exist in the documentation and TSS section; since these are directly associated with the SFRs, the tracing in element ADV_FSP.1.2D is implicitly already done and no additional documentation is necessary.

ADV_FSP.1.3C

The functional specification shall describe the purpose and method of use for

each SFR-enforcing and SFR-supporting TSFI.

ADV_FSP.1.4C

The functional specification shall identify all parameters associated with each SFR-enforcing and SFR-supporting TSFI.

ADV_FSP.1.5C

The functional specification shall provide rationale for the implicit categorization of interfaces as SFR-non-interfering.

ADV_FSP.1.6C

The tracing shall demonstrate that the SFRs trace to TSFIs in the functional specification.

Evaluator action elements:

ADV_FSP.1.7E

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_FSP.1.8E

The evaluator shall determine that the functional specification is an accurate and complete instantiation of the SFRs.

Evaluation Activities

ADV_FSP.1:
There are no specific assurance activities associated with these SARs, except ensuring the information is provided. The functional specification documentation is provided to support the evaluation activities described in Section 5.1 Security Functional Requirements, and other activities described for AGD, ATE, and AVA SARs. The requirements on the content of the functional specification information is implicitly assessed by virtue of the other assurance activities being performed; if the evaluator is unable to perform an activity because there is insufficient interface information, then an adequate functional specification has not been provided.

5.2.3 Class AGD: Guidance Documentation

The guidance documents will be provided with the ST. Guidance must include a description of how the IT personnel verifies that the Operational Environment can fulfill its role for the security functionality. The documentation should be in an informal style and readable by the IT personnel. Guidance must be provided for every operational environment that the product supports as claimed in the ST. This guidance includes instructions to successfully install the TSF in that environment; and Instructions to manage the security of the TSF as a product and as a component of the larger operational environment. Guidance pertaining to particular security functionality is also provided; requirements on such guidance are contained in the assurance activities specified with each requirement.

AGD_OPE.1 Operational User Guidance (AGD_OPE.1)

Developer action elements:

AGD_OPE.1.1D

The developer shall provide operational user guidance.

Application Note: The operational user guidance does not have to be contained in a single document. Guidance to users, administrators and application developers can be spread among documents or web pages. Rather than repeat information here, the developer should review the assurance activities for this component to ascertain the specifics of the guidance that the evaluator will be checking for. This will provide the necessary information for the preparation of acceptable guidance.

Content and presentation elements:

AGD_OPE.1.2C

The operational user guidance shall describe, for each user role, the user-accessible functions and privileges that should be controlled in a secure processing environment, including appropriate warnings.

Application Note: User and administrator are to be considered in the definition of user role.

AGD_OPE.1.3C

The operational user guidance shall describe, for each user role, how to use the available interfaces provided by the OS in a secure manner.

AGD_OPE.1.4C

The operational user guidance shall describe, for each user role, the available functions and interfaces, in particular all security parameters under the control of the user, indicating secure values as appropriate.

Application Note: This portion of the operational user guidance should be presented in the form of a checklist that can be quickly executed by IT personnel (or end-users, when necessary) and suitable for use in compliance activities. When possible, this guidance is to be expressed in the eXtensible Configuration Checklist Description Format (XCCDF) to support security automation. Minimally, it should be presented in a structured format which includes a title for each configuration item, instructions for achieving the secure configuration, and any relevant rationale.

AGD_OPE.1.5C

The operational user guidance shall, for each user role, clearly present each type of security-relevant event relative to the user-accessible functions that need to be performed, including changing the security characteristics of entities under the control of the TSF.

AGD_OPE.1.6C

The operational user guidance shall identify all possible modes of operation of the OS (including operation following failure or operational error), their consequences, and implications for maintaining secure operation.

AGD_OPE.1.7C

The operational user guidance shall, for each user role, describe the security measures to be followed in order to fulfill the security objectives for the operational environment as described in the ST.

AGD_OPE.1.8C

The operational user guidance shall be clear and reasonable.

Evaluator action elements:

AGD_OPE.1.9E

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

Evaluation Activities

AGD_OPE.1:
Some of the contents of the operational guidance are verified by the assurance activities in Section 5.1 Security Functional Requirements and evaluation of the OS according to the . The following additional information is also required. If cryptographic functions are provided by the OS, the operational guidance shall contain instructions for configuring the cryptographic engine associated with the evaluated configuration of the OS. 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 OS. The documentation must describe the process for verifying updates to the OS by verifying a digital signature – this may be done by the OS or the underlying platform. The evaluator will 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 OS (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 generation of the hash/digital signature. The OS will likely contain security functionality that does not fall in the scope of evaluation under this PP. The operational guidance shall make it clear to an administrator which security functionality is covered by the evaluation activities.

AGD_PRE.1 Preparative Procedures (AGD_PRE.1)

Developer action elements:

AGD_PRE.1.1D

The developer shall provide the OS, including its preparative procedures.

Application Note: As with the operational guidance, the developer should look to the assurance activities to determine the required content with respect to preparative procedures.

Content and presentation elements:

AGD_PRE.1.2C

The preparative procedures shall describe all the steps necessary for secure acceptance of the delivered OS in accordance with the developer's delivery procedures.

AGD_PRE.1.3C

The preparative procedures shall describe all the steps necessary for secure installation of the OS and for the secure preparation of the operational environment in accordance with the security objectives for the operational environment as described in the ST.

Evaluator action elements:

AGD_PRE.1.4E

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AGD_PRE.1.5E

The evaluator shall apply the preparative procedures to confirm that the OS can be prepared securely for operation.

Evaluation Activities

AGD_PRE.1:
As indicated in the introduction above, there are significant expectations with respect to the documentation—especially when configuring the operational environment to support OS functional requirements. The evaluator shall check to ensure that the guidance provided for the OS adequately addresses all platforms claimed for the OS in the ST.

5.2.4 Class ALC: Life-cycle Support

At the assurance level provided for OSs conformant to this PP, life-cycle support is limited to end-user-visible aspects of the life-cycle, rather than an examination of the OS vendor's development and configuration management process. This is not meant to diminish the critical role that a developer's practices play in contributing to the overall trustworthiness of a product; rather, it is a reflection on the information to be made available for evaluation at this assurance level.

ALC_CMC.1 Labeling of the TOE (ALC_CMC.1)

This component is targeted at identifying the OS such that it can be distinguished from other products or versions from the same vendor and can be easily specified when being procured by an end user.

Developer action elements:

ALC_CMC.1.1D
The developer shall provide the OS and a reference for the OS.

Content and presentation elements:

ALC_CMC.1.2C
The OS shall be labeled with a unique reference.

Application Note: Unique reference information includes:
- OS Name
- OS Version
- OS Description
- Software Identification (SWID) tags, if available

Evaluator action elements:

ALC_CMC.1.3E
The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

Evaluation Activities

ALC_CMC.1:
The evaluator will check the ST to ensure that it contains an identifier (such as a product name/version number) that specifically identifies the version that meets the requirements of the ST. Further, the evaluator will check the AGD guidance and OS samples received for testing to ensure that the version number is consistent with that in the ST. If the vendor maintains a web site advertising the OS, the evaluator will examine the information on the web site to ensure that the information in the ST is sufficient to distinguish the product.

ALC_CMS.1 TOE CM Coverage (ALC_CMS.1)

Given the scope of the OS and its associated evaluation evidence requirements, this component's assurance activities are covered by the assurance activities listed for ALC_CMC.1.

Developer action elements:

ALC_CMS.1.1D
The developer shall provide a configuration list for the OS.

Content and presentation elements:

ALC_CMS.1.2C
The configuration list shall include the following: the OS itself; and the evaluation evidence required by the SARs.

ALC_CMS.1.3C
The configuration list shall uniquely identify the configuration items.

Evaluator action elements:

ALC_CMS.1.4E
The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

Evaluation Activities

ALC_CMS.1:
The "evaluation evidence required by the SARs" in this PP is limited to the information in the ST coupled with the guidance provided to administrators and users under the AGD requirements. By ensuring that the OS is specifically identified and that this identification is consistent in the ST and in the AGD guidance (as done in the assurance activity for ALC_CMC.1), the evaluator implicitly confirms the information required by this component. Life-cycle support is targeted aspects of the developer's life-cycle and instructions to providers of applications for the developer's devices, rather than an in-depth examination of the TSF manufacturer's development and configuration management process. This is not meant to diminish the critical role that a developer's practices play in contributing to the overall trustworthiness of a product; rather, it's a reflection on the information to be made available for evaluation.
The evaluator will ensure that the developer has identified (in guidance documentation for application developers concerning the targeted platform) one or more development environments appropriate for use in developing applications for the developer's platform. For each of these development environments, the developer shall provide information on how to configure the environment to ensure that buffer overflow protection mechanisms in the environment(s) are invoked (e.g., compiler and linker flags). The evaluator will ensure that this

documentation also includes an indication of whether such protections are on by default, or have to be specifically enabled. The evaluator will ensure that the TSF is uniquely identified (with respect to other products from the TSF vendor), and that documentation provided by the developer in association with the requirements in the ST is associated with the TSF using this unique identification.

ALC_TSU_EXT.1 Timely Security Updates

This component requires the OS developer, in conjunction with any other necessary parties, to provide information as to how the end-user devices are updated to address security issues in a timely manner. The documentation describes the process of providing updates to the public from the time a security flaw is reported/discovered, to the time an update is released. This description includes the parties involved (e.g., the developer, carriers(s)) and the steps that are performed (e.g., developer testing, carrier testing), including worst case time periods, before an update is made available to the public.

Developer action elements:

ALC_TSU_EXT.1.1D
: The developer shall provide a description in the TSS of how timely security updates are made to the OS.

ALC_TSU_EXT.1.2D
: The developer shall provide a description in the TSS of how users are notified when updates change security properties or the configuration of the product.

Content and presentation elements:

ALC_TSU_EXT.1.3C
: The description shall include the process for creating and deploying security updates for the OS software.

ALC_TSU_EXT.1.4C
: The description shall include the mechanisms publicly available for reporting security issues pertaining to the OS.

  Note: The reporting mechanism could include web sites, email addresses, as well as a means to protect the sensitive nature of the report (e.g., public keys that could be used to encrypt the details of a proof-of-concept exploit).

Evaluator action elements:

ALC_TSU_EXT.1.5E
: The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

Evaluation Activities

ALC_TSU_EXT.1:
The evaluator will verify that the TSS contains a description of the timely security update process used by the developer to create and deploy security updates. The evaluator will verify that this description addresses the entire application. The evaluator will also verify that, in addition to the OS developer's process, any third-party processes are also addressed in the description. The evaluator will also verify that each mechanism for deployment of security updates is described.
The evaluator will verify that, for each deployment mechanism described for the update process, the TSS lists a time between public disclosure of a vulnerability and public availability of the security update to the OS patching this vulnerability, to include any third-party or carrier delays in deployment. The evaluator will verify that this time is expressed in a number or range of days.
The evaluator will verify that this description includes the publicly available mechanisms (including either an email address or website) for reporting security issues related to the OS.
The evaluator shall verify that the description of this mechanism includes a method for protecting the report either using a public key for encrypting email or a trusted channel for a website.

5.2.5 Class ATE: Tests

Testing is specified for functional aspects of the system as well as aspects that take advantage of design or implementation weaknesses. The former is done through the ATE_IND family, while the latter is through the AVA_VAN family. At the assurance level specified in this PP, testing is based on advertised functionality and interfaces with dependency on the availability of design information. One of the primary outputs of the evaluation process is the test report as specified in the following requirements.

ATE_IND.1 Independent Testing – Conformance (ATE_IND.1)

Testing is performed to confirm the functionality described in the TSS as well as the administrative (including configuration and operational) documentation provided. The focus of the testing is to confirm that the requirements specified in Section 5.1 Security Functional Requirements being met, although some additional testing is specified for SARs in Section 5.2 Security Assurance Requirements. The Assurance Activities identify the additional testing activities associated with these components. The evaluator produces a test report documenting the plan for and results of testing, as well as coverage arguments focused on the platform/OS combinations that are claiming conformance to this PP. Given the scope of the OS and its associated evaluation evidence requirements, this component's assurance activities are covered by the assurance activities listed for ALC_CMC.1.

Developer action elements:

ATE_IND.1.1D

The developer shall provide the OS for testing.

Content and presentation elements:

ATE_IND.1.2C

The OS shall be suitable for testing.

Evaluator action elements:

ATE_IND.1.3E

The evaluator *shall confirm* that the information provided meets all requirements for content and presentation of evidence.

ATE_IND.1.4E

The evaluator shall test a subset of the TSF to confirm that the TSF operates as specified.

Application Note: The evaluator will test the OS on the most current fully patched version of the platform.

Evaluation Activities

ATE_IND.1:
The evaluator will prepare a test plan and report documenting the testing aspects of the system, including any application crashes during testing. The evaluator shall determine the root cause of any application crashes and include that information in the report. The test plan covers all of the testing actions contained in the and the body of this PP's Assurance Activities.
While it is not necessary to have one test case per test listed in an Assurance Activity, the evaluator must document in the test plan that each applicable testing requirement in the ST is covered. The test plan identifies the platforms to be tested, and for those platforms not included in the test plan but included in the ST, the test plan provides a justification for not testing the platforms. This justification must address the differences between the tested platforms and the untested platforms, and make an argument that the differences do not affect the testing to be performed. It is not sufficient to merely assert that the differences have no affect; rationale must be provided. If all platforms claimed in the ST are tested, then no rationale is necessary. The test plan describes the composition of each platform to be tested, and any setup that is necessary beyond what is contained in the AGD documentation. It should be noted that the evaluator is expected to follow the AGD documentation for installation and setup of each platform either as part of a test or as a standard pre-test condition. This may include special test drivers or tools. For each driver or tool, an argument (not just an assertion) should be provided that the driver or tool will not adversely affect the performance of the functionality by the OS and its platform. This also includes the configuration of the cryptographic engine to be used. The cryptographic algorithms implemented by this engine are those specified by this PP and used by the cryptographic protocols being evaluated (IPsec, TLS). The test plan identifies high-level test objectives as well as the test procedures to be followed to achieve those objectives. These procedures include expected results.
The test report (which could just be an annotated version of the test plan) details the activities that took place when the test procedures were executed, and includes the actual results of the tests. This shall be a cumulative account, so if there was a test run that resulted in a failure; a fix installed; and then a successful re-run of the test, the report would show a "fail" and "pass" result (and the supporting details), and not just the "pass" result.

5.2.6 Class AVA: Vulnerability Assessment

For the first generation of this protection profile, the evaluation lab is expected to survey open sources to discover what vulnerabilities have been discovered in these types of products. In most cases, these vulnerabilities will require sophistication beyond that of a basic attacker. Until penetration tools are created and uniformly distributed to the evaluation labs, the evaluator will not be expected to test for these vulnerabilities in the OS. The labs will be expected to comment on the likelihood of these vulnerabilities given the documentation provided by the vendor. This information will be used in the development of penetration testing tools and for the development of future protection profiles.

AVA_VAN.1 Vulnerability Survey (AVA_VAN.1)

Developer action elements:

AVA_VAN.1.1D

The developer shall provide the OS for testing.

Content and presentation elements:

AVA_VAN.1.2C

The OS shall be suitable for testing.

Evaluator action elements:

AVA_VAN.1.3E

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AVA_VAN.1.4E

The evaluator shall perform a search of public domain sources to identify potential vulnerabilities in the OS.

Application Note: Public domain sources include the Common Vulnerabilities and Exposures (CVE) dictionary for publicly-known vulnerabilities. Public domain sources also include sites which provide free checking of files for viruses.

AVA_VAN.1.5E

The evaluator shall conduct penetration testing, based on the identified potential vulnerabilities, to determine that the OS is resistant to attacks performed by an attacker possessing Basic attack potential.

Evaluation Activities

AVA_VAN.1:
The evaluator will generate a report to document their findings with respect to this requirement. This report could physically be part of the overall test report mentioned in ATE_IND, or a separate document. The evaluator performs a search of public information to find vulnerabilities that have been found in similar applications with a particular focus on network protocols the application uses and document formats it parses. The evaluator documents the sources consulted and the vulnerabilities found in the report.
For each vulnerability found, the evaluator either provides a rationale with respect to its non-applicability, or the evaluator formulates a test (using the guidelines provided in ATE_IND) to confirm the vulnerability, if suitable. Suitability is determined by assessing the attack vector needed to take advantage of the vulnerability. If exploiting the vulnerability requires expert skills and an electron microscope, for instance, then a test would not be suitable and an appropriate justification would be formulated.

Appendix A - Optional Requirements

As indicated in the introduction to this PP, the baseline requirements (those that must be performed by the TOE) are contained in the body of this PP. This appendix contains three other types of optional requirements that may be included in the ST, but are not required in order to conform to this PP. However, applied modules, packages and/or use cases may refine specific requirements as mandatory.

The first type (A.1 Strictly Optional Requirements) are strictly optional requirements that are independent of the TOE implementing any function. If the TOE fulfills any of these requirements or supports a certain functionality, the vendor is encouraged to included the SFRs in the ST, but are not required in order to conform to this PP.

The second type (A.2 Objective Requirements) are objective requirements that describe security functionality not yet widely available in commercial technology. The requirements are not currently mandated in the body of this PP, but will be included in the baseline requirements in future versions of this PP. Adoption by vendors is encouraged and expected as soon as possible.

The third type (A.3 Implementation-based Requirements) are dependent on the TOE implementing a particular function. If the TOE fulfills any of these requirements, the vendor must either add the related SFR or disable the functionality for the evaluated configuration.

A.1 Strictly Optional Requirements

This PP does not define any Strictly Optional requirements.

A.2 Objective Requirements

This PP does not define any Objective requirements.

A.3 Implementation-based Requirements

This PP does not define any Implementation-based requirements.

Appendix B - Selection-based Requirements

As indicated in the introduction to this PP, the baseline requirements (those that must be performed by the TOE or its underlying platform) are contained in the body of this PP. There are additional requirements based on selections in the body of the PP: if certain selections are made, then additional requirements below must be included.

B.1 Auditable Events for Selection-based Requirements

Table 3: Audit Events for Sel-based Requirements

Requirement	Auditable Events	Additional Audit Record Contents
FAU_GEN.1	No events specified	
FCS_COP.1/Hash	No events specified	
FCS_COP.1/SigVer	No events specified	
FPT_RVR_EXT.1	No events specified	
FPT_TUD_EXT.2	**[selection:** *Failure of update authentication, None***]**	Version numbers of the current firmware and of the attempted update
FPT_TUD_EXT.2	**[selection:** *Failure of update operation, None***]**	Version numbers of the current firmware and of the attempted update
FPT_TUD_EXT.2	**[selection:** *Success of update operation, None***]**	Version numbers of the new and old firmware images.
FPT_TUD_EXT.3	No events specified	

B.2 Security Audit (FAU)

FAU_GEN.1

The inclusion of this selection-based component depends upon a selection in FPT_ROT_EXT.1.2, FPT_ROT_EXT.2.2.

FAU_GEN.1.1

The TSF shall be able to generate an audit record of the following auditable events:

1. Start-up and shutdown of the audit functions
2. All auditable events for the [not selected] level of audit
3. All administrative actions
4. Start-up, shutdown, and reboot of the platform
5. Specifically defined auditable events in Table 1
6. [selection: Audit records reaching [assignment: integer value less than 100] percentage of audit capacity, [assignment: other auditable events derived from this profile]]
7. [selection: Specifically defined auditable event in Table 2, no additional auditable events]

Application Note: The ST Author should include this SFR in the ST if the TOE generates audit events for integrity verification or boot failures as indicated by the appropriate selections in FPT_ROT_EXT.1.2 and FPR_ROT_EXT.2.2.

Evaluation Activities

B.3 Cryptographic Support (FCS)

FCS_COP.1/Hash Cryptographic Operation (Hashing)

The inclusion of this selection-based component depends upon a selection in FPT_TUD_EXT.2.1.

FCS_COP.1.1/Hash

The TSF shall perform [*cryptographic hashing*] in accordance with a specified cryptographic algorithm [**selection**: *SHA-1*, *SHA-256*, *SHA-384*, *SHA-512*, *SHA-3-224*, *SHA-3-256*, *SHA-3-384*, *SHA-3-512*] and message digest sizes [**selection**: *160*, *256*, *384*, *512 bits*] that meet the following: [**selection**: *FIPS PUB 180-4* "*Secure Hash Standard*", *ISO/IEC 10118-3:2018*]

Application Note: Per NIST SP 800-131A, SHA-1 for generating digital signatures is no longer allowed, and SHA-1 for verification of digital signatures is strongly discouraged as there may be risk in accepting these signatures. It is expected that vendors will implement SHA-2 algorithms in accordance with SP 800-131A.

The intent of this requirement is to specify the hashing function. The hash selection shall support the message digest size selection. The hash selection should be consistent with the overall strength of the algorithm used (for example, SHA 256 for 128-bit keys).

Evaluation Activities

FCS_COP.1/Hash:
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.

Guidance
The evaluator checks the AGD documents to determine that any configuration that is required to be done to configure the functionality for the required hash sizes is present.

Tests
SHA-1 and SHA-2 Tests
The TSF hashing functions can be implemented in one of two modes. The first mode is the byte-oriented 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 bit-oriented 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 bit-oriented vs. the byte-oriented 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.

Assurance Activity 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.

Short Messages Test Bit-oriented 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 pseudo-randomly 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 Byte-oriented 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 pseudo-randomly 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 Bit-oriented Mode
*The evaluators devise an input set consisting of m messages, where m is the block length of the hash algorithm. The length of the ith message is 512 + 99*i, where 1 <= i <= m. The message text shall be pseudo-randomly 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 Byte-oriented Mode
*The evaluators devise an input set consisting of m/8 messages, where m is the block length of the hash algorithm. The length of the ith message is 512 + 8*99*i, where 1 <= i <= m/8. The message text shall be pseudo-randomly 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.*

Pseudo-randomly Generated Messages Test
This test is for byte-oriented 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.

SHA-3 Tests
The tests below are derived from the The Secure Hash Algorithm-3 Validation System (SHA3VS), Updated: April 7, 2016, from the National Institute of Standards and Technology.

For each SHA-3-XXX implementation, XXX represents d, the digest length in bits. The capacity, c, is equal to 2d bits. The rate is equal to 1600-c bits.

The TSF hashing functions can be implemented with one of two orientations. The first is a bit-oriented mode that hashes messages of arbitrary length. The second is a byte-oriented mode that 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). Separate tests for each orientation are given below.

The evaluator shall perform all of the following tests for each hash algorithm and orientation implemented by the TSF and used to satisfy the requirements of this PP. The evaluator shall compare digest values produced by a known-good SHA-3 implementation against those generated by running the same values through the TSF.

Short Messages Test, Bit-oriented Mode
The evaluators devise an input set consisting of rate+1 short messages. The length of the messages ranges sequentially from 0 to rate bits. The message text shall be pseudo-randomly 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. The message of length 0 is omitted if the TOE does not support zero-length messages.

Short Messages Test, Byte-oriented Mode

The evaluators devise an input set consisting of rate/8+1 short messages. The length of the messages ranges sequentially from 0 to rate/8 bytes, with each message being an integral number of bytes. The message text shall be pseudo-randomly 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. The message of length 0 is omitted if the TOE does not support zero-length messages.

Selected Long Messages Test, Bit-oriented Mode
The evaluators devise an input set consisting of 100 long messages ranging in size from rate+(rate+1) to rate+(100(rate+1)), incrementing by rate+1. (For example, SHA-3-256 has a rate of 1088 bits. Therefore, 100 messages will be generated with lengths 2177, 3266, ..., 109988 bits.) The message text shall be pseudo-randomly 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, Byte-oriented Mode
The evaluators devise an input set consisting of 100 messages ranging in size from (rate+(rate+8)) to (rate+100(rate+8)), incrementing by rate+8. (For example, SHA-3-256 has a rate of 1088 bits. Therefore 100 messages will be generated of lengths 2184, 3280, 4376, ..., 110688 bits.) The message text shall be pseudo-randomly 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.*

Pseudo-randomly Generated Messages Monte Carlo) Test, Byte-oriented Mode

The evaluators supply a seed of d bits (where d is the length of the message digest produced by the hash function to be tested. This seed is used by a pseudorandom function to generate 100,000 message digests. One hundred of the digests (every 1000th digest) are recorded as checkpoints. The TOE then uses the same procedure to generate the same 100,000 message digests and 100 checkpoint values. The evaluators then compare the results generated ensure that the correct result is produced when the messages are generated by the TSF.

FCS_COP.1/SigVer

The inclusion of this selection-based component depends upon a selection in FPT_TUD_EXT.1.1.

FCS_COP.1.1/SigVer

Refinement: The TSF shall perform **cryptographic signature verification for a platform firmware update image** in accordance with a specified cryptographic algorithm [**selection**:

- ***RSA Schemes*** *using cryptographic key sizes of 2048-bits or greater that meet the following: FIPS PUB 186-4, "Digital Signature Standard (DSS)", Section 4,*
- ***ECDSA Schemes*** *using "NIST Curves" P-256, P-384 and [**selection**: P-521, no other curves] that meet the following: FIPS PUB 186-4, "Digital Signature Standard (DSS)", Section 5*

]

Application Note: The ST author should choose the algorithm implemented to perform vertification of digital signatures. if more than one algorithm is available, this requirement should be iterated to specify the functionality. For the algorithm chosen, the ST author should make the appropriate assignments/selections to specify the parameters that are implemented for that algorithm. In particular, if ECDSA is selected as one of the signature algorithms, the key size specified must match the selection for the curve used in the algorithm.

For elliptic curve-based schemes, the key size refers to the binary logarithm (log2) of the order of the base point. As the preferred approach for digital signatures, elliptic curves will be required after all the necessary standards and other supporting information are fully established.

Evaluation Activities

FCS_COP.1/SigVer:
Tests
The following tests require the developer to provide access to a test application that provides the evaluator with tools that are typically not found in the production application.

ECDSA Algorithm Tests

- ***Test 1:*** *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 ten 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.*
- ***Test 2:*** *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 ten 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

- ***Test 1:*** *Signature Generation Test. The evaluator shall verify the implementation of RSA Signature Generation by the TOE using the Signature Generation Test. To conduct this test*

the evaluator must generate or obtain 10 messages from a trusted reference implementation for each modulus size/SHA combination supported by the TSF. The evaluator shall have the TOE use their private key and modulus value to sign these messages. The evaluator shall verify the correctness of the TSF's signature using a known good implementation and the associated public keys to verify the signatures.

- ***Test 2:*** *Signature Verification Test. The evaluator shall perform the Signature Verification test to verify the ability of the TOE to recognize another party's valid and invalid signatures. The evaluator shall inject errors into the test vectors produced during the Signature Verification Test by introducing errors in some of the public keys, e, messages, IR format, and/or signatures. The TOE attempts to verify the signatures and returns success or failure.*

B.4 Class: Protection of the TSF (FPT)

FPT_RVR_EXT.1

The inclusion of this selection-based component depends upon a selection in FPT_ROT_EXT.1.2, FPT_ROT_EXT.2.2.

FPT_RVR_EXT.1.1

Something about recovery.

Application Note: Something in here describing the permissible ways to recover from a boot or integrity failure. Could this be merged with the secure local update requyirement.

Evaluation Activities

FPT_RVR_EXT.1:
Nothing to see here.

FPT_TUD_EXT.2

The inclusion of this selection-based component depends upon a selection in FPT_TUD_EXT.1.1.

FPT_TUD_EXT.2.1

The TSF shall authenticate the source of all platform firmware updates using a digital signature algorithm specified in FCS_COP.1/SigVer and using a key store that contains [**selection**: *the public key*, *hash value of the public key*].

FPT_TUD_EXT.2.2

The TSF shall allow installation of updates only if the digital signature has been successfully verified as specified in FCS_COP.1/SigVer and [**selection**: *the version number of the platform firmware update is more recent than the version number of the current installed platform firmware*, *no other reasons*].

FPT_TUD_EXT.2.3

The TSF shall include an observable platform firmware version identifier that is accessible by the update mechanism and includes information that enables the update mechanism to determine the intended relative order of updates.

FPT_TUD_EXT.2.4

The TSF shall provide an observable indication of the success or failure of the update operation.

FPT_TUD_EXT.2.5

On failure to authenticate a platform firmware update image, the TSF shall not install the update image and [**selection**: *Throw an audit event as described in (Table Ref)*, *Initiate auto recovery*, *Go dark*, *Struggle to reboot*]

FPT_TUD_EXT.2.6

On failure of the update process to install an authenticated platform firmware update image, the TSF shall [**selection**: *Throw an audit event as described in (Table Ref)*, *Initiate auto recovery*, *Go dark*, *Struggle to reboot*]

Application Note: The platform firmware update mechanism employs digital signatures to ensure the authenticity of the firmware update image. The TSF includes a signature verification algorithm and a key store containing the public key needed to verify the signature on the firmware update image.

A hash of the public key may be stored if a copy of the public key is provided with firmware update images. In this case, the update mechanism shall hash the public key provided with the update image, and ensure that it matches a hash which appears in the key store before using the provided public key to verify the signature of the update image. If the hash of the public key is selected, the ST author may iterate the FCS_COP.1/Hash requirement to specify the hashing functions used.

An indication of success or failure can be generation of an audit event by a management subsystem, a beep code, an updated version number on a splash screen, or simple failure to continue functioning.

If the update mechanism generates audit events, the ST author shall make the approriate selections from the audit events table ( ).

Evaluation Activities

FPT_TUD_EXT.2:
TSS
The evaluator shall ensure that the TSS includes a comprehansive description of how the authentication of platform firmware updates is implemented by the TSF. The TSS should cover the initialization process and the activities that are performed to ensure that the digital signature of the update image is verified before modification of the firmware.

The evaluator shall examine the TSF to ensure that it describes the platform firmware version identifier and explains its meaning and encoding.

The evaluator shall also ensure that the TSS describes the actions taken by the TSF is an update image fails authentication.
Guidance
The evaluator shall examine the operational guidance to ensure that it describes the process for updating the platform firmware.

The evaluator shall examine the operational guidance to ensure that it documents the observable indications of update success or failure, and that it describes how to access the platform firmware version indicators.
Tests

- ***Test 1:*** *The evaluator determines the current version of the platform firmware, and obtains or produces a valid, authentic, and permissible update image of platform firmware. The evaluator initiates an update using this image through the process described in the operational guidance. After the process is complete, the evaluator checks the current firmware version to ensure that the new firmware version matches that of the update.*
- ***Test 2:*** *The evaluator performs the same test, this time using a valid update image that is signed with an incorrect key. The update must fail.*
- ***Test 3:*** *The evaluator performs the same test, this time using an update image that is corrupted but is signed with the correct key. The update must fail.*
- ***Test 4:*** *The evaluator performs the same test, this time using a valid update image that is not signed. The update must fail.*

FPT_TUD_EXT.3

The inclusion of this selection-based component depends upon a selection in FPT_TUD_EXT.1.1.

FPT_TUD_EXT.3.1

The TSF shall provide a secure local update mechanism that requires physical access to the TOE before permitting the update of platform firmware.

FPT_TUD_EXT.3.2

The secure local update mechanism shall be used [**selection**:

- *to replace the manufacturer's original platform firmware image,*
- *to recover from a corrupted platform firmware installation or bricked machine,*
- *as the normal platform firmware update mechanism*

]

Application Note: This requirement pertains to platform firmware update mechanisms that do not use the authentication-based update mechanism described in FPT_TUD_EXT.2. The secure local update mechanism ensures the authenticity and integrity of the firmware update image by requiring a physical presence at the TOE. An assertion of physical presence can take the form, for example, of requiring entry of a password at a boot screen, unlocking of a physical lock (e.g., a motherboard jumper), or inserting a USB cable before permitting platform firmware to be updated.

This mechanism is distinguished from the lack of an update capability (as described in FPT_TUD_EXT.1) in that the local update is designed in to the system. ???

Is a chip in a socket that renders it removable a local update mechanism? Or is that not an update mechanism? Sometimes it's a fine line.

Would this potentially be subject to rollback protections?

Should we worry about update versions? If this is used for recovery only, then versions won't matter, but if it is used as the normal update mechanism, then it does matter. Probably you can have a local update with an override for rollback protections. If that's a thing.

Evaluation Activities

FPT_TUD_EXT.3:
TSS
The evaluator shall check the TSS section to confirm that it clearly and thoroughly describes how the secure local update functionality is implemented.
Guidance
The evaluator shall examine the operational guidance to ensure that it describes instructions for using the local update mechanism, and how to validate that the update was succesful.
Tests

The evaluator tests the secure local update by following the instructions provided in the operational guidance to replace the ...

Appendix C - References

ext-comp-def

Identifier	Title
[CC]	Common Criteria for Information Technology Security Evaluation - • Part 1: Introduction and General Model, CCMB-2017-04-001, Version 3.1, Revision 5, April 2017. • Part 2: Security Functional Components, CCMB-2017-04-002, Version 3.1, Revision 5, April 2017. • Part 3: Security Assurance Components, CCMB-2017-04-003, Version 3.1, Revision 5, April 2017.
[CEM]	Common Evaluation Methodology for Information Technology Security - Evaluation Methodology, CCMB-2012-09-004, Version 3.1, Revision 4, September 2012.
[CESG]	CESG - End User Devices Security and Configuration Guidance
[CSA]	Computer Security Act of 1987, H.R. 145, June 11, 1987.
[OMB]	Reporting Incidents Involving Personally Identifiable Information and Incorporating the Cost for Security in Agency Information Technology Investments, OMB M-06-19, July 12, 2006.

Appendix D - Acronyms

Acronym	Meaning
AES	Advanced Encryption Standard
API	Application Programming Interface
API	Application Programming Interface
ASLR	Address Space Layout Randomization
Base-PP	Base Protection Profile
CC	Common Criteria
CEM	Common Evaluation Methodology
CESG	Communications-Electronics Security Group
CMC	Certificate Management over CMS
CMS	Cryptographic Message Syntax
CN	Common Names
CRL	Certificate Revocation List
CSA	Computer Security Act
CSP	Critical Security Parameters
DAR	Data At Rest
DEP	Data Execution Prevention
DES	Data Encryption Standard
DHE	Diffie-Hellman Ephemeral
DNS	Domain Name System
DRBG	Deterministic Random Bit Generator
DSS	Digital Signature Standard
DSS	Digital Signature Standard
DT	Date/Time Vector
DTLS	Datagram Transport Layer Security
EAP	Extensible Authentication Protocol
ECDHE	Elliptic Curve Diffie-Hellman Ephemeral
ECDSA	Elliptic Curve Digital Signature Algorithm
EST	Enrollment over Secure Transport
FIPS	Federal Information Processing Standards
HMAC	Hash-based Message Authentication Code
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IETF	Internet Engineering Task Force
IP	Internet Protocol
ISO	International Organization for Standardization
IT	Information Technology
ITSEF	Information Technology Security Evaluation Facility
NIAP	National Information Assurance Partnership
NIST	National Institute of Standards and Technology
OCSP	Online Certificate Status Protocol
OE	Operational Environment
OID	Object Identifier
OMB	Office of Management and Budget
OS	Operating System
PII	Personally Identifiable Information

PKI	Public Key Infrastructure
PP	Protection Profile
PP	Protection Profile
PP-Configuration	Protection Profile Configuration
PP-Module	Protection Profile Module
RBG	Random Bit Generator
RFC	Request for Comment
RNG	Random Number Generator
RNGVS	Random Number Generator Validation System
S/MIME	Secure/Multi-purpose Internet Mail Extensions
SAN	Subject Alternative Name
SAR	Security Assurance Requirement
SFR	Security Functional Requirement
SHA	Secure Hash Algorithm
SIP	Session Initiation Protocol
ST	Security Target
SWID	Software Identification
TLS	Transport Layer Security
TOE	Target of Evaluation
TSF	TOE Security Functionality
TSFI	TSF Interface
TSS	TOE Summary Specification
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
USB	Universal Serial Bus
VM	Virtual Machine
XCCDF	eXtensible Configuration Checklist Description Format
XOR	Exclusive Or
app	Application