Protection Profile for General Purpose Operating Systems

This page is best viewed with JavaScript enabled!

Version: 4.3-DRAFT
2019-07-01
National Information Assurance Partnership

Revision History

Version	Date	Comment
4.2.1	2019-04-22	Formatting changes as a result of PP evaluation
4.2	2018-05-22	Multiple Technical Decisions applied
4.1	2016-03-09	Minor updates - cryptographic modes
4.0	2015-08-14	Release - significant revision

Contents

1 Introduction

1.1 Overview

The scope of this Protection Profile (PP) is to describe the security functionality of operating systems in terms of [CC] and to define functional and assurance requirements for such products. An operating system is software that manages computer hardware and software resources, and provides common services for application programs. The hardware it manages may be physical or virtual.

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-	An implementation-independent statement of security needs for a TOE type complementary to one or

Module)	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.[OMB]
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.

1.3 Compliant Targets of Evaluation

1.3.1 TOE Boundary

The TOE boundary encompasses the OS kernel and its drivers, shared software libraries, and some application software included with the OS. The applications considered within the TOE are those that provide essential security services, many of which run with elevated privileges. Applications which are covered by more-specific Protection Profiles cannot claim evaluation as part of the OS evaluation, even when it is necessary to evaluate some of their functionality as it relates to their role as part of the OS.

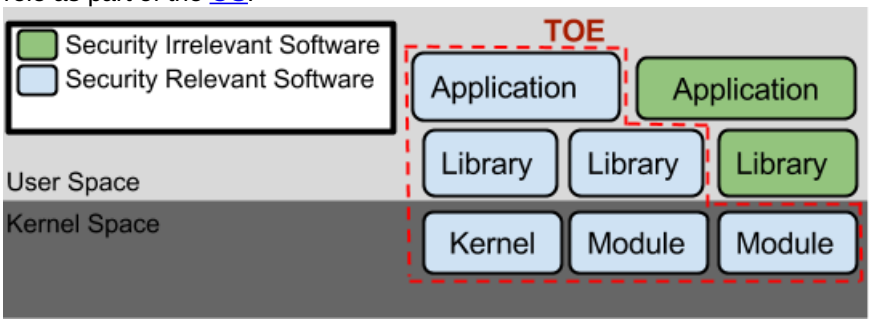

Figure 1: General TOE

1.3.2 TOE Platform

The TOE platform, which consists of the physical or virtual hardware on which the TOE executes, is outside the scope of evaluation. At the same time, the security of the TOE relies upon it. Other hardware components which independently run their own software and are relevant to overall system security are also outside the scope of evaluation.

1.4 Use Cases

Requirements in this Protection Profile are designed to address the security problems in at least the following use cases. These use cases are intentionally very broad, as many specific use cases exist for an operating system. These use cases may also overlap with one another. An operating system's functionality may even be effectively extended by privileged applications installed onto it. However, these are out of scope of this PP.

[USE CASE 1] End User Devices

The OS provides a platform for end user devices such as desktops, laptops, convertibles, and tablets. These devices may optionally be bound to a directory server or management server.

As this Protection Profile does not address threats against data-at-rest, enterprises deploying operating systems in mobile scenarios should ensure that these systems include data-at-rest protection spelled out in other Protection Profiles. Specifically, this includes the Protection Profiles for *Full Drive Encryption - Encryption Engine*, *Full Drive Encryption - Authorization Acquisition*, and *Software File Encryption*. The *Protection Profile for Mobile Device Fundamentals* includes requirements for data-at-rest protection and is appropriate for many mobile devices.

[USE CASE 2] Server Systems

The OS provides a platform for server-side services, either on physical or virtual hardware. Many specific examples exist in which the OS acts as a platform for such services, including file servers, mail servers, and web servers.

[USE CASE 3] Cloud Systems

The OS provides a platform for providing cloud services running on physical or virtual hardware. An OS is typically part of offerings identified as Infrastructure as a Service (IaaS), Software as a Service (SaaS), and Platform as a Service (PaaS).

This use case typically involves the use of virtualization technology which should be evaluated against the *Protection Profile for Server Virtualization*.

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 is Network Encryption (TLS) Package, Version 1.1 Conformant .

3 Security Problem Description

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

3.1 Threats

T.NETWORK_ATTACK
: An attacker is positioned on a communications channel or elsewhere on the network infrastructure. Attackers may engage in communications with applications and services running on or part of the OS with the intent of compromise. Engagement may consist of altering existing legitimate communications.

T.NETWORK_EAVESDROP
: An attacker is positioned on a communications channel or elsewhere on the network infrastructure. Attackers may monitor and gain access to data exchanged between applications and services that are running on or part of the OS.

T.LOCAL_ATTACK
: An attacker may compromise applications running on the OS. The compromised application may provide maliciously formatted input to the OS through a variety of channels including unprivileged system calls and messaging via the file system.

T.LIMITED_PHYSICAL_ACCESS
: An attacker may attempt to access data on the OS while having a limited amount of time with the physical device.

3.2 Assumptions

A.PLATFORM
: The OS relies upon a trustworthy computing platform for its execution. This underlying platform is out of scope of this PP.

A.PROPER_USER
: The user of the OS is not willfully negligent or hostile, and uses the software in compliance with the applied enterprise security policy. At the same time, malicious software could act *as* the user, so requirements which confine malicious subjects are still in scope.

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

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 theOS 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.

Threat, Assumption, or OSP	Security Objectives	Rationale
T.NETWORK_ATTACK	O.PROTECTED_COMMS	The threat T.NETWORK_ATTACK is countered by O.PROTECTED_COMMS as this provides for integrity of transmitted data.
	O.INTEGRITY	The threat T.NETWORK_ATTACK is countered by O.INTEGRITY as this provides for integrity of software that is installed onto the system from the network.
	O.MANAGEMENT	The threat T.NETWORK_ATTACK is countered by O.MANAGEMENT as this provides for the ability to configure the OS to defend against network attack.
	O.ACCOUNTABILITY	The threat T.NETWORK_ATTACK is countered by O.ACCOUNTABILITY as this provides a mechanism for the OS to report behavior that may indicate a network attack has occurred.
T.NETWORK_EAVESDROP	O.PROTECTED_COMMS	The threat T.NETWORK_EAVESDROP is countered by O.PROTECTED_COMMS as this provides for confidentiality of transmitted data.
	O.MANAGEMENT	The threat T.NETWORK_EAVESDROP is countered by O.MANAGEMENT as this provides for the ability to configure the OS to protect the confidentiality of its transmitted data.
T.LOCAL_ATTACK	O.INTEGRITY	The objective O.INTEGRITY protects against the use of mechanisms that weaken the TOE with regard to attack by other software on the platform.
	O.ACCOUNTABILITY	The objective O.ACCOUNTABILITY protects against local attacks by providing a mechanism to report behavior that may indicate a local attack is occurring or has occurred.
T.LIMITED_PHYSICAL_ACCESS	O.PROTECTED_STORAGE	The objective O.PROTECTED_STORAGE protects against unauthorized attempts to access physical storage used by the TOE.
A.PLATFORM	OE.PLATFORM	The operational environment objective OE.PLATFORM is realized through A.PLATFORM.
A.PROPER_USER	OE.PROPER_USER	The operational environment objective OE.PROPER_USER is realized through A.PROPER_USER.
A.PROPER_ADMIN	OE.PROPER_ADMIN	The operational environment objective OE.PROPER_ADMIN is realized through A.PROPER_ADMIN.

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 Cryptographic Support (FCS)

FCS_CKM.1 Cryptographic Key Generation (Refined)

FCS_CKM.1.1
The **OS** shall generate **asymmetric** cryptographic keys in accordance with a specified cryptographic key generation algorithm **[selection:**

- ***RSA schemes using cryptographic key sizes of 2048-bit or greater that f-component meet the following:FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.3*** ,
- ***ECC 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)", Appendix B.4*** ,
- ***FFC schemes using cryptographic key sizes of 2048-bit or greater that meet the following:FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.1***

] ~~and specified cryptographic key sizes [assignment: cryptographic key sizes] that meet the following: [assignment: list of standards]~~ .
Application Note:

The ST author shall select all key generation schemes used for key establishment and entity authentication. When key generation is used for key establishment, the schemes in FCS_CKM.2 and selected cryptographic protocols must match the selection. When key generation is used for entity authentication, the public key is expected to be associated with an X.509v3 certificate.

If the OS acts only as a receiver in the RSA key establishment scheme, theOS does not need to implement RSA key generation.

Evaluation Activity
Tests

The evaluator will ensure that the TSS identifies the key sizes supported by theOS. If the ST specifies more than one scheme, the evaluator will examine the TSS to verify that it identifies the usage for each scheme.

The evaluator will verify that the AGD guidance instructs the administrator how to configure the OS to use the selected key generation scheme(s) and key size(s) for all uses defined in this PP.

Evaluation Activity Note: The following tests may require the vendor to furnish a developer environment and developer tools that are typically not available to end-users of the OS.

Key Generation for FIPS PUB 186-4 RSA Schemes
The evaluator will verify the implementation of RSA Key Generation by the OS 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:

1. *Random Primes:*
   - *Provable primes*
   - *Probable primes*
2. *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 will verify the correctness of theTSF's implementation by comparing values generated by the TSF with those generated from a known good implementation.

If possible, the Random Probable primes method should also be verified against a known good implementation as described above. Otherwise, the evaluator will have the TSF generate 10 keys pairs for each supported key length nlen and verify:

- $n = p \cdot q$,
- *p and q are probably prime according to Miller-Rabin tests,*
- $GCD(p-1,e) = 1$,
- $GCD(q-1,e) = 1$,
- $2^{16} \leq e \leq 2^{256}$ *and e is an odd integer,*

- $|p-q| > 2^{nlen/2 - 100}$,
- $p \geq 2^{nlen/2 -1/2}$,
- $q \geq 2^{nlen/2 -1/2}$,
- $2^{(nlen/2)} < d < LCM(p-1,q-1)$,
- $e \cdot d = 1 \bmod LCM(p-1,q-1)$.

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 will 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 will 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 will 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 will obtain in response a set of 10 PASS/FAIL values.

Key Generation for Finite-Field Cryptography (FFC)
The evaluator will 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:

- *Cryptographic and Field Primes:*
  - *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:
- *Cryptographic Group Generator:*
  - *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:
- *Private Key:*
  - *len(q) bit output of RBG where $1 \leq x \leq q-1$*
  - *len(q) + 64 bit output of RBG, followed by a mod q-1 operation where $1 \leq x \leq q-1$*

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 will have the TSF generate 25 parameter sets and key pairs. The evaluator will 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*
- $g^q \bmod p = 1$
- $g^x \bmod p = y$

for each FFC parameter set and key pair.

FCS_CKM.2 Cryptographic Key Establishment (Refined)

FCS_CKM.2.1
The OS shall **implement functionality to perform cryptographic key establishment** in accordance with a specified cryptographic key **establishment** method:

[*RSA-based key establishment schemes*] that meets the following: [*NIST Special Publication 800-56B, "Recommendation for Pair-Wise Key Establishment Schemes Using Integer Factorization Cryptography"*]

and [selection:

- ***Elliptic curve-based key establishment schemes that meets the following:NIST Special Publication 800-56A, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography"*,**

- ***Finite field-based key establishment schemes that meets the following:NIST Special Publication 800-56A, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography"***,
- ***No other schemes***

] ~~that meets the following: [assignment: list of standards]~~ .
Application Note:

The ST author shall select all key establishment schemes used for the selected cryptographic protocols.

The RSA-based key establishment schemes are described in Section 9 of NIST SP 800-56B; however, Section 9 relies on implementation of other sections in SP 800-56B. If the OS acts as a receiver in the RSA key establishment scheme, theOS does not need to implement RSA key generation.

The elliptic curves used for the key establishment scheme shall correlate with the curves specified in FCS_CKM.1.1. The domain parameters used for the finite field-based key establishment scheme are specified by the key generation according to FCS_CKM.1.1.

Evaluation Activity
Tests

The evaluator will ensure that the supported key establishment schemes correspond to the key generation schemes identified in FCS_CKM.1.1 If the ST specifies more than one scheme, the evaluator will examine theTSS to verify that it identifies the usage for each scheme.

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

Evaluation 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.

Key Establishment Schemes

The evaluator will verify the implementation of the key establishment schemes supported by the OS using the applicable tests below.

SP800-56A Key Establishment Schemes

The evaluator will verify the OS'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 the OS has implemented the components of the key agreement scheme according to the specifications in the Recommendation. These components include the calculation of the discrete logarithm cryptography (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 will 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 MAC data and the calculation of MAC tag.

> ***Function Test***
>
> *The Function test verifies the ability of the OS to implement the key agreement schemes correctly. To conduct this test the evaluator will generate or obtain test vectors from a known good implementation of the OS's 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 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 will obtain the DKM, the corresponding OS'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 OS id fields.*
>
> *If the OS does not use a KDF defined in SP 800-56A, the evaluator will obtain only the public keys and the hashed value of the shared secret.*
>
> *The evaluator will 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 OS shall perform the above for each implemented approved MAC algorithm.*
>
> ***Validity Test***
>
> *The Validity test verifies the ability of the OS to recognize another party's valid and invalid key agreement results with or without key confirmation. To conduct this test, the evaluator will obtain a list of the supporting cryptographic functions included in the SP800-56A key agreement implementation to determine which errors the OS should be able to recognize. The evaluator generates a set of 30 test vectors consisting of data sets including domain parameter values or NIST approved curves, the evaluator's public keys, theOS's public/private key pairs, MAC tag, and any inputs used in the KDF, such as the other info and OS id fields.*

The evaluator will inject an error in some of the test vectors to test that the OS 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 MAC'd, or the generated MAC tag. If the OS 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 OS's static private key to assure theOS 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 OS shall use these modified test vectors to emulate the key agreement scheme using the corresponding parameters. The evaluator will compare the OS's results with the results using a known good implementation verifying that the OS detects these errors.

SP800-56B Key Establishment Schemes

The evaluator will verify that the TSS describes whether the OS acts as a sender, a recipient, or both for RSA-based key establishment schemes.

If the OS acts as a sender, the following assurance activity shall be performed to ensure the proper operation of everyOS supported combination of RSA-based key establishment scheme:

To conduct this test the evaluator will generate or obtain test vectors from a known good implementation of the OS's supported schemes. For each combination of supported key establishment scheme and its options (with or without key confirmation if supported, for each supported key confirmation MAC function if key confirmation is supported, and for each supported mask generation function if KTS-OAEP is supported), the tester shall generate 10 sets of test vectors. Each test vector shall include the RSA public key, the plaintext keying material, any additional input parameters if applicable, the MAC key and MAC tag if key confirmation is incorporated, and the outputted ciphertext. For each test vector, the evaluator shall perform a key establishment encryption operation on the OS with the same inputs (in cases where key confirmation is incorporated, the test shall use the MAC key from the test vector instead of the randomly generated MAC key used in normal operation) and ensure that the outputted ciphertext is equivalent to the ciphertext in the test vector.

If the OS acts as a receiver, the following evaluation activities shall be performed to ensure the proper operation of everyOS supported combination of RSA-based key establishment scheme:

To conduct this test the evaluator will generate or obtain test vectors from a known good implementation of the OS's supported schemes. For each combination of supported key establishment scheme and its options (with our without key confirmation if supported, for each supported key confirmation MAC function if key confirmation is supported, and for each supported mask generation function if KTS-OAEP is supported), the tester shall generate 10 sets of test vectors. Each test vector shall include the RSA private key, the plaintext keying material, any additional input parameters if applicable, the MAC tag in cases where key confirmation is incorporated, and the outputted ciphertext. For each test vector, the evaluator will perform the key establishment decryption operation on the OS and ensure that the outputted plaintext keying material is equivalent to the plaintext keying material in the test vector. In cases where key confirmation is incorporated, the evaluator will perform the key confirmation steps and ensure that the outputted MAC tag is equivalent to the MAC tag in the test vector.

The evaluator will ensure that the TSS describes how theOS handles decryption errors. In accordance withNIST Special Publication 800-56B, the OS must not reveal the particular error that occurred, either through the contents of any outputted or logged error message or through timing variations. If KTS-OAEP is supported, the evaluator will create separate contrived ciphertext values that trigger each of the three decryption error checks described in NIST Special Publication 800-56B section 7.2.2.3, ensure that each decryption attempt results in an error, and ensure that any outputted or logged error message is identical for each. If KTS-KEM-KWS is supported, the evaluator will create separate contrived ciphertext values that trigger each of the three decryption error checks described in NIST Special Publication 800-56B section 7.2.3.3, ensure that each decryption attempt results in an error, and ensure that any outputted or logged error message is identical for each.

FCS_CKM_EXT.4 Cryptographic Key Destruction

FCS_CKM_EXT.4.1
The OS shall destroy cryptographic keys and key material in accordance with a specified cryptographic key destruction method [**selection**:

- *For volatile memory, the destruction shall be executed by a **selection**:*
  - *single overwrite consisting of [**selection**: a pseudo-random pattern using theTSF's RBG, zeroes, ones, a new value of a key, [**assignment**: any value that does not contain any CSP]],*
  - *removal of power to the memory,*
  - *destruction of reference to the key directly followed by a request for garbage collection*

  ],
- *For non-volatile memory that consists of [**selection**:*
  - *destruction of all key encrypting keys protecting the target key according toFCS_CKM_EXT.4.1, where none of the KEKs protecting the target key are derived ,*
  - *the invocation of an interface provided by the underlying platform that **selection**:*
    - *logically addresses the storage location of the key and performs a **selection**: single, [**assignment**: ST author defined multi-pass]]overwrite consisting of [**selection**: zeroes, ones, pseudo-random pattern, a new value of a key of the same size, [**assignment**: any value that does not contain any CSP]],*

- *instructs the underlying platform to destroy the abstraction that represents the key*

]

]

] .
Application Note:

The interface referenced in the requirement could take different forms, the most likely of which is an application programming interface to an OS kernel. There may be various levels of abstraction visible. For instance, in a given implementation that overwrites a key stored in non-volatile memory, the application may have access to the file system details and may be able to logically address specific memory locations. In another implementation, that instructs the underlying platform to destroy the representation of a key stored in non-volatile memory, the application may simply have a handle to a resource and can only ask the platform to delete the resource, as may be the case with a platforms secure key store. The latter implementation should only be used for the most restricted access. The level of detail to which the TOE has access will be reflected in the TSS section of the ST.

Several selections allow assignment of a 'value that does not contain any CSP'. This means that the TOE uses some other specified data not drawn from a source that may contain key material or reveal information about key material, and not being any of the particular values listed as other selection options. The point of the phrase 'does not contain any CSP' is to ensure that the overwritten data is carefully selected, and not taken from a general 'pool' that might contain current or residual data that itself requires confidentiality protection.

For the selection *destruction of all key encrypting keys protecting target key according to FCS_CKM_EXT.4.1, where none of the KEKs protecting the target key are derived*, a key can be considered destroyed by destroying the key that protects the key. If a key is wrapped or encrypted it is not necessary to “overwrite” that key, overwriting the key that is used to wrap or encrypt the key used to encrypt/decrypt data, using the appropriate method for the memory type involved, will suffice. For example, if a product uses a Key Encryption Key (KEK) to encrypt a Data Encryption Key (DEK), destroying the KEK using one of the methods in FCS_CKM_EXT.4 is sufficient, since the DEK would no longer be usable (of course, presumes the DEK is still encrypted and the KEK cannot be recovered or re-derived.).

FCS_CKM_EXT.4.2
The OS shall destroy all keys and key material when no longer needed.
Application Note:

For the purposes of this requirement, key material refers to authentication data, passwords, secret/private symmetric keys, private asymmetric keys, data used to derive keys, values derived from passwords, etc.

Key destruction procedures are performed in accordance with FCS_CKM_EXT.4.1.

Evaluation Activity
TSS

The evaluator examines the TSS to ensure it describes how the keys are managed in volatile memory. This description includes details of how each identified key is introduced into volatile memory (e.g. by derivation from user input, or by unwrapping a wrapped key stored in non-volatile memory) and how they are overwritten.

The evaluator will check to ensure the TSS lists each type of key that is stored in in non-volatile memory, and identifies how the TOE interacts with the underlying platform to manage keys (e.g., store, retrieve, destroy). The description includes details on the method of how the TOE interacts with the platform, including an identification and description of the interfaces it uses to manage keys (e.g., file system APIs, platform key store APIs).

If the ST makes use of the open assignment and fills in the type of pattern that is used, the evaluator examines the TSS to ensure it describes how that pattern is obtained and used. The evaluator will verify that the pattern does not contain any CSPs.

The evaluator will check that the TSS identifies any configurations or circumstances that may not strictly conform to the key destruction requirement.

If the selection “destruction of all key encrypting keys protecting target key according to FCS_CKM_EXT.4.1, where none of the KEKs protecting the target key are derived” is included the evaluator shall examine the TOE’s keychain in the TSS and identify each instance when a key is destroyed by this method. In each instance the evaluator shall verify all keys capable of decrypting the target key are destroyed in accordance with a specified key destruction method in FCS_CKM_EXT.4.1. The evaluator shall verify that all of the keys capable of decrypting the target key are not able to be derived to reestabish the keychain after their destruction.

Guidance

Operational Guidance

There are a variety of concerns that may prevent or delay key destruction in some cases. The evaluator will check that the guidance documentation identifies configurations or circumstances that may not strictly conform to the key destruction requirement, and that this description is consistent with the relevant parts of the TSS and any other relevant Required Supplementary Information. The evaluator will check that the guidance documentation provides guidance on situations where key destruction may be delayed at the physical layer and how such situations can be avoided or mitigated if possible.

Some examples of what is expected to be in the documentation are provided here.

When the TOE does not have full access to the physical memory, it is possible that the storage may be implementing wear-leveling and garbage collection. This may create additional copies of the key that are logically inaccessible but persist physically. In this case, to mitigate this the drive should support the TRIM command and implements garbage collection to destroy these persistent copies when not actively engaged in other tasks.

Drive vendors implement garbage collection in a variety of different ways, as such there is a variable amount of time until data is truly removed from these solutions. There is a risk that data may persist for a longer amount of time if it is contained in a block with other data not ready for erasure. To reduce this risk, the operating system and file system of the OE should support TRIM, instructing the non-volatile memory to erase copies via garbage collection upon their deletion. If a RAID array is being used, only set-ups that support TRIM are utilized. If the drive is connected via PCI-Express, the operating system supports TRIM over that channel.

The drive should be healthy and contains minimal corrupted data and should be end-of-lifed before a significant amount of damage to drive health occurs, this minimizes the risk that small amounts of potentially recoverable data may remain in damaged areas of the drive.

Tests

- ***Test 1:** Applied to each key held as in volatile memory and subject to destruction by overwrite by theTOE (whether or not the value is subsequently encrypted for storage in volatile or non-volatile memory). In the case where the only selection made for the destruction method key was removal of power, then this test is unnecessary. The evaluator will:*
  1. *Record the value of the key in the TOE subject to clearing.*
  2. *Cause the TOE to perform a normal cryptographic processing with the key from Step #1.*
  3. *Cause the TOE to clear the key.*
  4. *Cause the TOE to stop the execution but not exit.*
  5. *Cause the TOE to dump the entire memory of theTOE into a binary file.*
  6. *Search the content of the binary file created in Step #5 for instances of the known key value from Step #1.*

  Steps 1-6 ensure that the complete key does not exist anywhere in volatile memory. If a copy is found, then the test fails.

- ***Test 2:** Applied to each key help in non-volatile memory and subject to destruction by theTOE. The evaluator will use special tools (as needed), provided by the TOE developer if necessary, to ensure the tests function as intended.*
  1. *Identify the purpose of the key and what access should fail when it is deleted. (e.g. the data encryption key being deleted would cause data decryption to fail.)*
  2. *Cause the TOE to clear the key.*
  3. *Have the TOE attempt the functionality that the cleared key would be necessary for.*

  The test succeeds if step 3 fails.

- ***Test 3:***

  Tests 3 and 4 do not apply for the selection instructing the underlying platform to destroy the representation of the key as the TOE has no visibility into the inner workings and completely relies on the underlying platform.

  The following tests are used to determine if the TOE is able to request the platform to overwrite the key with aTOE supplied pattern.

  Applied to each key held in non-volatile memory and subject to destruction by overwrite by the TOE. The evaluator will use a tool that provides a logical view of the media (e.g., MBR file system):

  1. *Record the value of the key in the TOE subject to clearing.*
  2. *Cause the TOE to perform a normal cryptographic processing with the key from Step #1.*
  3. *Cause the TOE to clear the key.*
  4. *Search the logical view that the key was stored in for instances of the known key value from Step #1. If a copy is found, then the test fails.*

- ***Test 4:** Applied to each key held as non-volatile memory and subject to destruction by overwrite by the TOE. The evaluator will use a tool that provides a logical view of the media:*
  1. *Record the logical storage location of the key in the TOE subject to clearing.*
  2. *Cause the TOE to perform a normal cryptographic processing with the key from Step #1.*
  3. *Cause the TOE to clear the key.*
  4. *Read the logical storage location in Step #1 of non-volatile memory to ensure the appropriate pattern is utilized.*

  The test succeeds if correct pattern is used to overwrite the key in the memory location. If the pattern is not found the test fails.

FCS_COP.1/ENCRYPT Cryptographic Operation - Encryption/Decryption (Refined)

FCS_COP.1.1/ENCRYPT
The **OS** shall perform [*encryption/decryption services for data*] in accordance with a specified cryptographic algorithm **[selection:**

- ***AES*-XTS (as defined in *NIST* SP 800-38E)**,
- ***AES*-CBC (as defined in *NIST* SP 800-38A)**

] and [selection:

- ***AES*-CCMP (as defined in *FIPS* PUB 197, *NIST* SP 800-38C and IEEE 802.11-2012)**
- ***AES* Key Wrap (KW) (as defined in *NIST* SP 800-38F)**,
- ***AES* Key Wrap with Padding (KWP) (as defined in *NIST* SP 800-38F)**,
- ***AES*-GCM (as defined in *NIST* SP 800-38D)**,
- ***AES*-CCM (as defined in *NIST* SP 800-38C)**,
- ***AES*-CCMP-256 (as defined in *NIST* SP800-38C and IEEE 802.11ac-2013)**
- ***AES*-GCMP-256 (as defined in *NIST* SP800-38D and IEEE 802.11ac-2013)**
- ***no other modes***

] and cryptographic key sizes [**selection**: *128-bit*, *256-bit*] ~~that meet the following: [assignment: list of standards]~~.
Application Note:

AES CCMP (which uses AES in CCM as specified in SP 800-38C) becomes mandatory and must be selected if the ST includes the WLAN Client EP.

For the second selection, the ST author should choose the mode or modes in which AES operates. For the third selection, the ST author should choose the key sizes that are supported by this functionality.

Evaluation Activity
Tests

The evaluator will verify that the AGD documents contains instructions required to configure the OS to use the required modes and key sizes. The evaluator will execute all instructions as specified to configure the OS to the appropriate state. The evaluator will perform all of the following tests for each algorithm implemented by the OS and used to satisfy the requirements of this PP:

***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 will 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 will 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 will 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 will 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 will 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 will 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 will 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 will 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 will 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 will test the encrypt functionality by encrypting an i-block message where 1 < i ≤ 10. The evaluator will 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 will also test the decrypt functionality for each mode by

decrypting an i-block message where 1 < i ≤10. The evaluator will 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 will 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 will 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 Monte Carlo Tests**

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

- *128 bit and 256 bit keys*
- *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.*
- *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.*
- *Two IV lengths. If 96 bit IV is supported, 96 bits shall be one of the two IV lengths tested.*

The evaluator will 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 will 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 will compare the resulting values to those obtained by submitting the same inputs to a known good implementation.

***AES*-CCM Tests**

The evaluator will test the generation-encryption and decryption-verification functionality of AES-CCM for the following input parameter and tag lengths:

- *128 bit and 256 bit keys*
- *Two payload lengths. One payload length shall be the shortest supported payload length, greater than or equal to zero bytes. The other payload length shall be the longest supported payload length, less than or equal to 32 bytes (256 bits).*
- *Two or three associated data lengths. One associated data length shall be 0, if supported. One associated data length shall be the shortest supported payload length, greater than or equal to zero bytes. One associated data length shall be the longest supported payload length, less than or equal to 32 bytes (256 bits). If the implementation supports an associated data length of 2 16 bytes, an associated data length of 216 bytes shall be tested.*
- *Nonce lengths. All supported nonce lengths between 7 and 13 bytes, inclusive, shall be tested.*
- *Tag lengths. All supported tag lengths of 4, 6, 8, 10, 12, 14 and 16 bytes shall be tested.*

To test the generation-encryption functionality of AES-CCM, the evaluator will perform the following four tests:

- ***Test 1:** For EACH supported key and associated data length and ANY supported payload, nonce and tag length, the evaluator will supply one key value, one nonce value and 10 pairs of associated data and payload values and obtain the resulting ciphertext.*
- ***Test 2:** For EACH supported key and payload length and ANY supported associated data, nonce and tag length, the evaluator will supply one key value, one nonce value and 10 pairs of associated data and payload values and obtain the resulting ciphertext.*

- ***Test 3:*** *For EACH supported key and nonce length and ANY supported associated data, payload and tag length, the evaluator will supply one key value and 10 associated data, payload and nonce value 3-tuples and obtain the resulting ciphertext.*
- ***Test 4:*** *For EACH supported key and tag length and ANY supported associated data, payload and nonce length, the evaluator will supply one key value, one nonce value and 10 pairs of associated data and payload values and obtain the resulting ciphertext.*

To determine correctness in each of the above tests, the evaluator will compare the ciphertext with the result of generation-encryption of the same inputs with a known good implementation.

To test the decryption-verification functionality of AES-CCM, for EACH combination of supported associated data length, payload length, nonce length and tag length, the evaluator shall supply a key value and 15 nonce, associated data and ciphertext 3-tuples and obtain either a FAIL result or a PASS result with the decrypted payload. The evaluator will supply 10 tuples that should FAIL and 5 that should PASS per set of 15.

Additionally, the evaluator will use tests from the IEEE 802.11-02/362r6 document "Proposed Test vectors for IEEE 802.11 TGi", dated September 10, 2002, Section 2.1 AESCCMP Encapsulation Example and Section 2.2 Additional AES CCMP Test Vectors to further verify the IEEE 802.11-2007 implementation of AES-CCMP.

AES-GCM Test

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

- *128 bit and 256 bit keys*
- *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.*
- *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.*
- *Two IV lengths. If 96 bit IV is supported, 96 bits shall be one of the two IV lengths tested.*

The evaluator will 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 will 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 will compare the resulting values to those obtained by submitting the same inputs to a known good implementation.

XTS-AES Test

The evaluator will test the encrypt functionality of XTS-AES for each combination of the following input parameter lengths:

- *256 bit (for AES-128) and 512 bit (for AES-256) keys*
- *Three data unit (i.e., plaintext) lengths. One of the data unit lengths shall be a nonzero integer multiple of 128 bits, if supported. One of the data unit lengths shall be an integer multiple of 128 bits, if supported. The third data unit length shall be either the longest supported data unit length or 216 bits, whichever is smaller.*

using a set of 100 (key, plaintext and 128-bit random tweak value) 3-tuples and obtain the ciphertext that results from XTS-AES encrypt.

The evaluator may supply a data unit sequence number instead of the tweak value if the implementation supports it. The data unit sequence number is a base-10 number ranging between 0 and 255 that implementations convert to a tweak value internally.

The evaluator will test the decrypt functionality of XTS-AES using the same test as for encrypt, replacing plaintext values with ciphertext values and XTS-AES encrypt with XTSAES decrypt.

AES Key Wrap (AES-KW) and Key Wrap with Padding (AES-KWP) Test

The evaluator will test the authenticated encryption functionality of AES-KW for EACH combination of the following input parameter lengths:

- *128 and 256 bit key encryption keys (KEKs)*
- *Three plaintext lengths. One of the plaintext lengths shall be two semi-blocks (128 bits). One of the plaintext lengths shall be three semi-blocks (192 bits). The third data unit length shall be the longest supported plaintext length less than or equal to 64 semi-blocks (4096 bits).*

using a set of 100 key and plaintext pairs and obtain the ciphertext that results from AES-KW authenticated encryption. To

determine correctness, the evaluator will use the AES-KW authenticated-encryption function of a known good implementation.

The evaluator will test the authenticated-decryption functionality of AES-KW using the same test as for authenticated-encryption, replacing plaintext values with ciphertext values and AES-KW authenticated-encryption with AES-KW authenticated-decryption.

The evaluator will test the authenticated-encryption functionality of AES-KWP using the same test as forAES-KW authenticated-encryption with the following change in the three plaintext lengths:

- *One plaintext length shall be one octet. One plaintext length shall be 20 octets (160 bits).*
- *One plaintext length shall be the longest supported plaintext length less than or equal to 512 octets (4096 bits).*

The evaluator will test the authenticated-decryption functionality of AES-KWP using the same test as forAES-KWP authenticated-encryption, replacing plaintext values with ciphertext values and AES-KWP authenticated-encryption with AES-KWP authenticated-decryption.

FCS_COP.1/HASH Cryptographic Operation - Hashing (Refined)

FCS_COP.1.1/HASH
The **OS** shall perform [*cryptographic hashing services*] in accordance with a specified cryptographic algorithm**[selection:**

- ***SHA-1*,**
- ***SHA-256*,**
- ***SHA-384*,**
- ***SHA-512*,**
- ***no other algorithms***

] and message digest sizes 160 bits and [selection:

- ***256 bits*,**
- ***384 bits*,**
- ***512 bits*,**
- ***no other sizes***

] that meet the following: [*FIPS Pub 180-4*].
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 is risk in accepting these signatures.

Evaluation of SHA-1 is currently permitted in order to allow for evaluation ofTLS or DTLS ciphersuites that may be selected in the TLS Package. Vendors are strongly encouraged to implement updated protocols that support theSHA-2 family; until updated protocols are supported, this PP allows support for SHA-1 implementations in compliance with SP 800-131A.

The intent of this requirement is to specify the hashing function. The hash selection must support the message digest size selection. The hash selection should be consistent with the overall strength of the algorithm used.

Evaluation Activity
Tests

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

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

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.

- ***Test 1:** Short Messages Test (Bit oriented Mode) - The evaluator will generate 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 evaluator will 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.*
- ***Test 2:** Short Messages Test (Byte oriented Mode) - The evaluator will generate 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 evaluator will 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.*
- ***Test 3:** Selected Long Messages Test (Bit oriented Mode) - The evaluator will generate 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 pseudorandomly generated. The evaluator will 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.

- ***Test 4:*** *Selected Long Messages Test (Byte oriented Mode) - The evaluator will generate 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 pseudorandomly generated. The evaluator will 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.*
- ***Test 5:*** *Pseudorandomly Generated Messages Test - This test is for byte-oriented implementations only. The evaluator will 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 evaluator will then formulate a set of 100 messages and associated digests by following the algorithm provided in Figure 1 of [SHAVS]. The evaluator will then ensure that the correct result is produced when the messages are provided to the TSF.*

FCS_COP.1/SIGN Cryptographic Operation - Signing (Refined)

FCS_COP.1.1/SIGN
The **OS** shall perform [*cryptographic signature services (generation and verification)*] in accordance with a specified cryptographic algorithm **[selection:**

- ***RSA schemes using cryptographic key sizes of 2048-bit 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***

] ~~and cryptographic key sizes [assignment: cryptographic algorithm] that meet the following: [assignment: list of standards]~~
Application Note: The ST Author should choose the algorithm implemented to perform 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.
Evaluation Activity
Tests

The evaluator will perform the following activities based on the selections in the ST.

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 supportedNIST curve (i.e., P-256, P-384 and P-521) and SHA function pair, the evaluator will 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 will use the signature verification function of a known good implementation.*
- ***Test 2:*** *ECDSA FIPS 186-4 Signature Verification Test. For each supportedNIST curve (i.e., P-256, P-384 and P-521) and SHA function pair, the evaluator will 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 will verify that 5 responses indicate success and 5 responses indicate failure.*

RSA Signature Algorithm Tests

- ***Test 1:*** *Signature Generation Test. The evaluator will verify the implementation of RSA Signature Generation by the OS 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 will have the OS use its private key and modulus value to sign these messages. The evaluator will 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 will perform the Signature Verification test to verify the ability of the OS to recognize another party's valid and invalid signatures. The evaluator will 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 evaluator will verify that the OS returns failure when validating each signature.*

FCS_COP.1/KEYHMAC Cryptographic Operation - Keyed-Hash Message Authentication (Refined)

FCS_COP.1.1/KEYHMAC
The **OS** shall perform [*keyed-hash message authentication services*] in accordance with a specified cryptographic algorithm **[selection: *SHA-1*, *SHA-256*, *SHA-384*, *SHA-512*] with key sizes [assignment: key size (in bits) used in HMAC] and message digest** sizes [**selection**: *160 bits*, *256 bits*, *384 bits*, *512 bits*] that meet the following: [*FIPS Pub 198-1 The Keyed-Hash Message Authentication Code and FIPS Pub 180-4 Secure Hash Standard*].
Application Note:

The intent of this requirement is to specify the keyed-hash message authentication function used for key establishment purposes for the various cryptographic protocols used by the OS (e.g., trusted channel). The hash selection must support the message digest size selection. The hash selection should be consistent with the overall strength of the algorithm used for FCS_COP.1/HASH.

Evaluation of SHA-1 is currently permitted in order to allow for evaluation of TLS or DTLS ciphersuites that may be selected in the TLS Package.

Evaluation Activity
Tests

The evaluator will perform the following activities based on the selections in the ST.

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

FCS_RBG_EXT.1 Random Bit Generation

FCS_RBG_EXT.1.1
The OS shall perform all deterministic random bit generation (DRBG) services in accordance with NIST Special Publication 800-90A using [**selection**:

- *Hash_DRBG (any)*,
- *HMAC_DRBG (any)*,
- *CTR_DRBG (AES)*

] .
Application Note: NIST SP 800-90A contains three different methods of generating random numbers; each of these, in turn, depends on underlying cryptographic primitives (hash functions/ciphers). The ST author will select the function used and include the specific underlying cryptographic primitives used in the requirement or in the TSS. While any of the identified hash functions (SHA-1, SHA-224, SHA-256, SHA-384, SHA-512) are allowed for Hash_DRBG or HMAC_DRBG, only AES-based implementations for CTR_DRBG are allowed.
Evaluation Activity
Tests

The evaluator will perform the following tests:

The evaluator will perform 15 trials for the RNG implementation. If the RNG is configurable, the evaluator will perform 15 trials for each configuration. The evaluator will also confirm that the operational guidance contains appropriate instructions for configuring the RNG functionality.

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 will 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 SP 800-90A).

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 will 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 list contains 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 less than or equal to 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 will 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.*

FCS_RBG_EXT.1.2
The deterministic RBG used by the OS shall be seeded by an entropy source that accumulates entropy from a [**selection**:

- *software-based noise source*,
- *platform-based noise source*

] with a minimum of [**selection**:

- *128 bits*,

- *256 bits*

] of entropy at least equal to the greatest security strength (according to NIST SP 800-57) of the keys and hashes that it will generate.
Application Note:

For the first selection in this requirement, the ST author selects 'software-based noise source' if any additional noise sources are used as input to the DRBG.

In the second selection in this requirement, the ST author selects the appropriate number of bits of entropy that corresponds to the greatest security strength of the algorithms included in the ST. Security strength is defined in Tables 2 and 3 ofNIST SP 800-57A. For example, if the implementation includes 2048-bit RSA (security strength of 112 bits), AES 128 (security strength 128 bits), and HMAC-SHA-256 (security strength 256 bits), then theST author would select 256 bits.

Evaluation Activity
Tests

Documentation shall be produced - and the evaluator will perform the activities - in accordance with Appendix D - Entropy Documentation and Assessment and the Clarification to the Entropy Documentation and Assessment Annex

In the future, specific statistical testing (in line with NIST SP 800-90B) will be required to verify the entropy estimates.

FCS_STO_EXT.1 Storage of Sensitive Data

FCS_STO_EXT.1.1
The OS shall implement functionality to encrypt sensitive data stored in non-volatile storage and provide interfaces to applications to invoke this functionality.
Application Note: Sensitive data shall be identified in theTSS by the ST author, and minimally includes credentials and keys. The interface for invoking the functionality could take a variety of forms: it could consist of an API, or simply well-documented conventions for accessing credentials stored as files.
Evaluation Activity
TSS

The evaluator will check the TSS to ensure that it lists all persistent sensitive data for which theOS provides a storage capability. For each of these items, the evaluator will confirm that the TSS lists for what purpose it can be used, and how it is stored. The evaluator will confirm that cryptographic operations used to protect the data occur as specified in FCS_COP.1/HASH.

The evaluator will also consult the developer documentation to verify that an interface exists for applications to securely store credentials.

5.1.2 User Data Protection (FDP)

FDP_ACF_EXT.1 Access Controls for Protecting User Data

FDP_ACF_EXT.1.1
The OS shall implement access controls which can prohibit unprivileged users from accessing files and directories owned by other users.
Application Note: Effective protection by access controls may also depend upon system configuration. This requirement is designed to ensure that, for example, files and directories owned by one user in a multi user system can be protected from access by another user in that system.
Evaluation Activity
TSS

The evaluator will confirm that the TSS comprehensively describes the access control policy enforced by theOS. The description must include the rules by which accesses to particular files and directories are determined for particular users. The evaluator will inspect the TSS to ensure that it describes the access control rules in such detail that given any possible scenario between a user and a file governed by the OS the access control decision is unambiguous.

Tests

The evaluator will create two new standard user accounts on the system and conduct the following tests:

- ***Test 1:** The evaluator will authenticate to the system as the first user and create a file within that user's home directory. The evaluator will then log off the system and log in as the second user. The evaluator will then attempt to read the file created in the first user's home directory. The evaluator will ensure that the read attempt is denied.*
- ***Test 2:** The evaluator will authenticate to the system as the first user and create a file within that user's home directory. The evaluator will then log off the system and log in as the second user. The evaluator will then attempt to modify the file created in the first user's home directory. The evaluator will ensure that the modification is denied.*
- ***Test 3:** The evaluator will authenticate to the system as the first user and create a file within that user's user directory. The evaluator will then log off the system and log in as the second user. The evaluator will then attempt to delete the file created in the first user's home directory. The evaluator will ensure that the deletion is denied.*
- ***Test 4:** The evaluator will authenticate to the system as the first user. The evaluator will attempt to create a file in the*

second user's home directory. The evaluator will ensure that the creation of the file is denied.

- ***Test 5:** The evaluator will authenticate to the system as the first user and attempt to modify the file created in the first user's home directory. The evaluator will ensure that the modification of the file is accepted.*
- ***Test 6:** The evaluator will authenticate to the system as the first user and attempt to delete the file created in the first user's directory. The evaluator will ensure that the deletion of the file is accepted.*

5.1.3 Security Management (FMT)

FMT_MOF_EXT.1 Management of security functions behavior

FMT_MOF_EXT.1.1
The OS shall restrict the ability to perform the function indicated in the "Administrator" column in FMT_SMF_EXT.1.1 to the administrator.
Application Note: The functions with an "X" in the "Administrator" column must be restricted to (or overridden by) the administrator in the TOE. The functions with an "O" in the "Administrator" column may be restricted to (or overridden by) the administrator when implemented in the TOE at the discretion of the ST author. For such functions, the ST author indicates this by replacing an "O" with an "X" in the ST.
Evaluation Activity
TSS

The evaluator will verify that the TSS describes those management functions that are restricted to Administrators, including how the user is prevented from performing those functions, or not able to use any interfaces that allow access to that function.

Tests
The evaluator will also perform the following test.

- ***Test 1:** For each function that is indicated as restricted to the administrator, the evaluation shall perform the function as an administrator, as specified in the Operational Guidance, and determine that it has the expected effect as outlined by the Operational Guidance and the SFR. The evaluator will then perform the function (or otherwise attempt to access the function) as a non-administrator and observe that they are unable to invoke that functionality.*

FMT_SMF_EXT.1 Specification of Management Functions

FMT_SMF_EXT.1.1
The OS shall be capable of performing the following management functions:

#	Management Function	Administrator	User
1	Enable/disable [**selection**: *screen lock, session timeout*]	MMandatory	OOptional
2	Configure [**selection**: *screen lock, session*] inactivity timeout	MMandatory	OOptional
3	Configure local audit storage capacity	OOptional	OOptional
4	Configure minimum password length	OOptional	OOptional
5	Configure minimum number of special characters in password	OOptional	OOptional
6	Configure minimum number of numeric characters in password	OOptional	OOptional
7	Configure minimum number of uppercase characters in password	OOptional	OOptional
8	Configure minimum number of lowercase characters in password	OOptional	OOptional
9	Configure lockout policy for unsuccessful authentication attempts through [**selection**: *timeouts between attempts, limiting number of attempts during a time period*]	OOptional	OOptional
10	Configure host-based firewall	OOptional	OOptional
11	Configure name/address of directory server with which to bind	OOptional	OOptional
12	Configure name/address of remote management server from which to receive management settings	OOptional	OOptional
13	Configure name/address of audit/logging server to which to send audit/logging records	OOptional	OOptional
14	Configure audit rules	OOptional	OOptional
15	Configure name/address of network time server	OOptional	OOptional
16	Enable/disable automatic software update	OOptional	OOptional
17	Configure WiFi interface	OOptional	OOptional
18	Enable/disable Bluetooth interface	OOptional	OOptional
19	Enable/disable [**assignment**: list of other external interfaces]	OOptional	OOptional
20	[**assignment**: list of other management functions to be provided by the TSF]	OOptional	OOptional

.
Application Note:

The ST should indicate which of the optional management functions are implemented in the TOE. This can be done by copying the above table into the ST and adjusting the "Administrator" and "User" columns to "X" according to which capabilities are present or not present, and for which privilege level. The Application Note for FMT_MOF_EXT.1 explains how to indicate Administrator or User capability.

The terms "Administrator" and "User" are defined in 1.2 Terms. The intent of this requirement is to ensure that theST is populated with the relevant management functions that are provided by the OS.

Sophisticated account management policies, such as intricate password complexity requirements and handling of temporary accounts, are a function of directory servers. The OS can enroll in such account management and enable the overall information system to achieve such policies by binding to a directory server.

Evaluation Activity
Guidance
The evaluator will verify that every management function captured in the ST is described in the operational guidance and that the description contains the information required to perform the management duties associated with the management function.
Tests
The evaluator will test the OS's ability to provide the management functions by configuring the operating system and testing each option selected from above. 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.

5.1.4 Protection of the TSF (FPT)

FPT_ACF_EXT.1 Access controls

FPT_ACF_EXT.1.1
The OS shall implement access controls which prohibit unprivileged users from modifying:

- Kernel and its drivers/modules
- Security audit logs
- Shared libraries
- System executables
- System configuration files
- [**assignment**: other objects]

.
Evaluation Activity
TSS
The evaluator will confirm that the TSS specifies the locations of kernel drivers/modules, security audit logs, shared libraries, system executables, and system configuration files. Every file does not need to be individually identified, but the system's conventions for storing and protecting such files must be specified.
Tests
The evaluator will create an unprivileged user account. Using this account, the evaluator will ensure that the following tests result in a negative outcome (i.e., the action results in the OS denying the evaluator permission to complete the action):

- ***Test 1:** The evaluator will attempt to modify all kernel drivers and modules.*
- ***Test 2:** The evaluator will attempt to modify all security audit logs generated by the logging subsystem.*
- ***Test 3:** The evaluator will attempt to modify all shared libraries that are used throughout the system.*
- ***Test 4:** The evaluator will attempt to modify all system executables.*
- ***Test 5:** The evaluator will attempt to modify all system configuration files.*
- ***Test 6:** The evaluator will attempt to modify any additional components selected.*

FPT_ACF_EXT.1.2
The OS shall implement access controls which prohibit unprivileged users from reading:

- Security audit logs
- System-wide credential repositories
- [**assignment**: list of other objects]

.
Application Note: "Credential repositories" refer, in this case, to structures containing cryptographic keys or passwords.
Evaluation Activity
Tests
The evaluator will create an unprivileged user account. Using this account, the evaluator will ensure that the following tests result in a negative outcome (i.e., the action results in the OS denying the evaluator permission to complete the action):

- ***Test 1:** The evaluator will attempt to read security audit logs generated by the auditing subsystem*
- ***Test 2:** The evaluator will attempt to read system-wide credential repositories*
- ***Test 3:** The evaluator will attempt to read any other object specified in the assignment*

FPT_ASLR_EXT.1 Address Space Layout Randomization

FPT_ASLR_EXT.1.1
The OS shall always randomize process address space memory locations with [**selection**: *8*, [***assignment**: number greater than 8*]] bits of entropy except for [**assignment**: list of explicit exceptions].
Evaluation Activity

Tests
The evaluator will select 3 executables included with the TSF. If the TSF includes a web browser it must be selected. If the TSF includes a mail client it must be selected. For each of these apps, the evaluator will launch the same executables on two separate instances of the OS on identical hardware and compare all memory mapping locations. The evaluator will ensure that no memory mappings are placed in the same location. If the rare chance occurs that two mappings are the same for a single executable and not the same for the other two, the evaluator will repeat the test with that executable to verify that in the second test the mappings are different. This test can also be completed on the same hardware and rebooting between application launches.

FPT_SBOP_EXT.1 Stack Buffer Overflow Protection

FPT_SBOP_EXT.1.1
The OS shall [**selection**: *employ stack-based buffer overflow protections*, *not store parameters/variables in the same data structures as control flow values*].
Application Note: Many OSes store control flow values (i.e. return addresses) in stack data structures that also contain parameters and variables. For these OSes, it is expected that most of the OS, to include the kernel, libraries, and application software from the OS vendor be compiled with stack-based buffer overflow protection enabled. OSes that store parameters and variables separately from control flow values do not need additional stack protections.
Evaluation Activity
Tests

For stack-based OSes, the evaluator will determine that the TSS contains a description of stack-based buffer overflow protections used by the OS. These are referred to by a variety of terms, such as stack cookie, stack guard, and stack canaries. The TSS must include a rationale for any binaries that are not protected in this manner. The evaluator will also preform the following test:

- ***Test 1:*** *The evaluator will inventory the kernel, libraries, and application binaries to determine those that do not implement stack-based buffer overflow protections. This list should match up with the list provided in the TSS.*

For OSes that store parameters/variables separately from control flow values, the evaluator will verify that the TSS describes what data structures control values, parameters, and variables are stored. The evaluator will also ensure that the TSS includes a description of the safeguards that ensure parameters and variables do not intermix with control flow values.

FPT_TST_EXT.1 Boot Integrity

FPT_TST_EXT.1.1
The OS shall verify the integrity of the bootchain up through theOS kernel and [**selection**:

- *all executable code stored in mutable media*
- *[**assignment**: list of other executable code]*,
- *no other executable code*

] prior to its execution through the use of [**selection**:

- *a digital signature using a hardware-protected asymmetric key and X.509 certificates*
- *a hardware-protected hash*

] .
Application Note:

The bootchain of the OS is the sequence of software, to include theOS loader, the kernel, system drivers or modules, and system files, which ultimately result in loading the OS. The first part of the OS, usually referred to as the first-stage bootloader, must be loaded by the platform. Assessing its integrity, while critical, is the platform's responsibility; and therefore outside the scope of this PP. All software loaded after this stage is potentially within the control of theOS and is in scope.

The verification may be transitive in nature: a hardware-protected public key or hash may be used to verify the mutable bootloader code which contains a key or hash used by the bootloader to verify the mutable OS kernel code, which contains a key or hash to verify the next layer of executable code, and so on. However, the way in which the hardware stores and protects these keys is out of scope.

If all executable code (including bootloader(s), kernel, device drivers, pre-loaded applications, user-loaded applications, and libraries) is verified, "all executable code stored in mutable media" should be selected.

Evaluation Activity
TSS

The evaluator will verify that the TSS section of the ST includes a comprehensive description of the boot procedures, including a description of the entire bootchain, for the TSF. The evaluator will ensure that the OS cryptographically verifies each piece of software it loads in the bootchain to include bootloaders and the kernel. Software loaded for execution directly by the platform (e.g. first-stage bootloaders) is out of scope. For each additional category of executable code verified before execution, the evaluator will verify that the description in the TSS describes how that software is cryptographically verified.

The evaluator will verify that the TSS contains a description of the protection afforded to the mechanism performing the cryptographic verification.

Tests

The evaluator will also perform the following tests:

- ***Test 1:** The evaluator will perform actions to cause TSF software to load and observe that the integrity mechanism does not flag any executables as containing integrity errors and that the OS properly boots.*
- ***Test 2:** The evaluator will modify a TSF executable that is part of the bootchain verified by theTSF (i.e. Not the first-stage bootloader) and attempt to boot. The evaluator will ensure that an integrity violation is triggered and the OS does not boot (Care must be taken so that the integrity violation is determined to be the cause of the failure to load the module, and not the fact that in such a way to invalidate the structure of the module.).*
- ***Test 3:** If the ST author indicates that the integrity verification is performed using a public key, the evaluator will verify that the update mechanism includes a certificate validation according to FIA_X509_EXT.1.*

FPT_TUD_EXT.1 Trusted Update

FPT_TUD_EXT.1.1
The OS shall provide the ability to check for updates to theOS software itself.
Application Note: This requirement is about the ability to check for the availability of authentic updates, while the installation of authentic updates is covered by FPT_TUD_EXT.1.2.
Evaluation Activity
Tests
The evaluator will check for an update using procedures described in the documentation and verify that the OS provides a list of available updates. Testing this capability may require installing and temporarily placing the system into a configuration in conflict with secure configuration guidance which specifies automatic update. (The evaluator is also to ensure that this query occurs over a trusted channel as described in FTP_ITC_EXT.1.)
FPT_TUD_EXT.1.2
The OS shall [**selection**: *cryptographically verify*, *invoke platform-provided functionality to cryptographically verify*] updates to itself using a digital signature prior to installation using schemes specified in FCS_COP.1/SIGN.
Application Note: The intent of the requirement is to ensure that only digitally signed and verifiedTOE updates are applied to the TOE.
Evaluation Activity
Tests
For the following tests, the evaluator will initiate the download of an update and capture the update prior to installation. The download could originate from the vendor's website, an enterprise-hosted update repository, or another system (e.g. network peer). All supported origins for the update must be indicated in the TSS and evaluated.

- ***Test 1:** The evaluator will ensure that the update has a digital signature belonging to the vendor prior to its installation. The evaluator will modify the downloaded update in such a way that the digital signature is no longer valid. The evaluator will then attempt to install the modified update. The evaluator will ensure that the OS does not install the modified update.*
- ***Test 2:** The evaluator will ensure that the update has a digital signature belonging to the vendor. The evaluator will then attempt to install the update (or permit installation to continue). The evaluator will ensure that the OS successfully installs the update.*

FPT_TUD_EXT.2 Trusted Update for Application Software

FPT_TUD_EXT.2.1
The OS shall provide the ability to check for updates to application software.
Application Note: This requirement is about the ability to check for authentic updates, while the actual installation of such updates is covered by FPT_TUD_EXT.2.2.
Evaluation Activity
Tests
The evaluator will check for updates to application software using procedures described in the documentation and verify that the OS provides a list of available updates. Testing this capability may require temporarily placing the system into a configuration in conflict with secure configuration guidance which specifies automatic update. (The evaluator is also to ensure that this query occurs over a trusted channel as described in FTP_ITC_EXT.1.)
FPT_TUD_EXT.2.2
The OS shall cryptographically verify the integrity of updates to applications using a digital signature specified by FCS_COP.1/SIGN prior to installation.
Evaluation Activity
Tests
The evaluator will initiate an update to an application. This may vary depending on the application, but it could be through the application vendor's website, a commercial app store, or another system. All origins supported by theOS must be indicated in the TSS and evaluated. However, this only includes those mechanisms for which theOS is providing a trusted installation and update functionality. It does not include user or administrator-driven download and installation of arbitrary files.

- ***Test 1:** The evaluator will ensure that the update has a digital signature which chains to the OS vendor or another trusted root managed through the OS. The evaluator will modify the downloaded update in such a way that the digital*

signature is no longer valid. The evaluator will then attempt to install the modified update. The evaluator will ensure that the OS does not install the modified update.

- ***Test 2:*** *The evaluator will ensure that the update has a digital signature belonging to the OS vendor or another trusted root managed through the OS. The evaluator will then attempt to install the update. The evaluator will ensure that the OS successfully installs the update.*

5.1.5 Audit Data Generation (FAU)

FAU_GEN.1 Audit Data Generation (Refined)

FAU_GEN.1.1
The **OS** shall be able to generate an audit record of the following auditable events:

a. Start-up and shut-down of the audit functions;
b. All auditable events for the [*not specified*] level of audit; and [
c.
   - *Authentication events (Success/Failure);*
   - *Use of privileged/special rights events (Successful and unsuccessful security, audit, and configuration changes);*
   - *Privilege or role escalation events (Success/Failure);*
   - *[**selection**:*
     - *File and object events (Successful and unsuccessful attempts to create, access, delete, modify, modify permissions),*
     - *User and Group management events (Successful and unsuccessful add, delete, modify, disable, enable, and credential change),*
     - *Audit and log data access events (Success/Failure)*
     - *Cryptographic verification of software (Success/Failure)*
     - *Attempted application invocation with arguments (Success/Failure e.g. due to software restriction policy)*
     - *System reboot, restart, and shutdown events (Success/Failure)*
     - *Kernel module loading and unloading events (Success/Failure)*
     - *Administrator or root-level access events (Success/Failure)*
     - *[**assignment**: other specifically defined auditable events].*

   ]

].

Evaluation Activity
Guidance

The evaluator will check the administrative guide and ensure that it lists all of the auditable events. The evaluator will check to make sure that every audit event type selected in the ST is included.

Tests

The evaluator will test the OS's ability to correctly generate audit records by having theTOE generate audit records for the events listed in the ST. This should include all instance types of an event specified. When verifying the test results, the evaluator will ensure the audit records generated during testing match the format specified in the administrative guide, and that the fields in each audit record have the proper entries.

FAU_GEN.1.2
The **OS** shall record within each audit record at least the following information:

a. Date and time of the event, type of event, subject identity (if applicable), and outcome (success or failure) of the event; and
b. For each audit event type, based on the auditable event definitions of the functional components included in the PP/ST, [**assignment**: other audit relevant information]

.
Application Note: The term *subject* here is understood to be the user that the process is acting on behalf of. If no auditable event definitions of functional components are provided, then no additional audit-relevant information is required.
Evaluation Activity
Guidance

The evaluator will check the administrative guide and ensure that it provides a format for audit records. Each audit record format type must be covered, along with a brief description of each field. The evaluator will ensure that the fields contains the information required.

Tests

The evaluator shall test the OS's ability to correctly generate audit records by having theTOE generate audit records for the events listed in the ST. The evaluator will ensure the audit records generated during testing match the format specified in the administrative guide, and that the fields in each audit record provide the required information.

5.1.6 Identification and Authentication (FIA)

FIA_AFL.1 Authentication failure handling (Refined)

FIA_AFL.1.1
The **OS** shall detect when [**selection**:

- *[**assignment**: positive integer number]*,
- *an administrator configurable positive integer within [**assignment**: range of acceptable values]*

] unsuccessful authentication attempts occur related to **events with [selection:**

- ***authentication based on user name and password***,
- ***authentication based on user name and a PIN that releases an asymmetric key stored in OE-protected storage***,
- ***authentication based on X.509 certificates***

] .
Evaluation Activity
Tests
The evaluator will set an administrator-configurable threshold for failed attempts, or note the ST-specified assignment. The evaluator will then (per selection) repeatedly attempt to authenticate with an incorrect password, PIN, or certificate until the number of attempts reaches the threshold. Note that the authentication attempts and lockouts must also be logged as specified in FAU_GEN.1.
FIA_AFL.1.2
When the defined number of unsuccessful authentication attempts for an account has been **met**, the **OS** shall: **[selection:** ***Account Lockout***, ***Account Disablement***, ***Mandatory Credential Reset***, ***[assignment: list of actions]*]** .
Application Note: The action to be taken shall be populated in the assignment of the ST and defined in the administrator guidance.
Evaluation Activity
Tests

- ***Test 1:*** *The evaluator will attempt to authenticate repeatedly to the system with a known bad password. Once the defined number of failed authentication attempts has been reached the evaluator will ensure that the account that was being used for testing has had the actions detailed in the assignment list above applied to it. The evaluator will ensure that an event has been logged to the security event log detailing that the account has had these actions applied.*
- ***Test 2:*** *The evaluator will attempt to authenticate repeatedly to the system with a known bad certificate. Once the defined number of failed authentication attempts has been reached the evaluator will ensure that the account that was being used for testing has had the actions detailed in the assignment list above applied to it. The evaluator will ensure that an event has been logged to the security event log detailing that the account has had these actions applied.*
- ***Test 3:*** *The evaluator will attempt to authenticate repeatedly to the system using both a bad password and a bad certificate. Once the defined number of failed authentication attempts has been reached the evaluator will ensure that the account that was being used for testing has had the actions detailed in the assignment list above applied to it. The evaluator will ensure that an event has been logged to the security event log detailing that the account has had these actions applied.*

FIA_UAU.5 Multiple Authentication Mechanisms (Refined)

FIA_UAU.5.1
The **OS** shall provide the following authentication mechanisms **[selection:**

- ***authentication based on user name and password***,
- ***authentication based on user name and a PIN that releases an asymmetric key stored in OE-protected storage***,
- ***authentication based on X.509 certificates***,
- ***for use in SSH only, SSH public key-based authentication as specified by the EP for Secure Shell***

] to support user authentication.
Application Note: The "for use in SSH only, SSH public key-based authentication as specified by the EP for Secure Shell" selection can only be included, and must be included, if FTP_ITC_EXT.1.1 selects "SSH as conforming to the EP for Secure Shell".
Evaluation Activity
Tests
If user name and password authentication is selected, the evaluator will configure the OS with a known user name and password and conduct the following tests:

- ***Test 1:*** *The evaluator will attempt to authenticate to the OS using the known user name and password. The evaluator will ensure that the authentication attempt is successful.*
- ***Test 2:*** *The evaluator will attempt to authenticate to the OS using the known user name but an incorrect password. The evaluator will ensure that the authentication attempt is unsuccessful.*

If user name and PIN that releases an asymmetric key is selected, the evaluator will examine the TSS for guidance on supported protected storage and will then configure the TOE or OE to establish a PIN which enables release of the asymmetric key from the protected storage (such as a TPM, a hardware token, or isolated execution environment) with which the OS can interface. The evaluator will then conduct the following tests:

- ***Test 1:*** *The evaluator will attempt to authenticate to the OS using the known user name and PIN. The evaluator will ensure that the authentication attempt is successful.*

- ***Test 2:*** *The evaluator will attempt to authenticate to the OS using the known user name but an incorrect PIN. The evaluator will ensure that the authentication attempt is unsuccessful.*

If X.509 certificate authentication is selected, the evaluator will generate an X.509v3 certificate for a user with the Client Authentication Enhanced Key Usage field set. The evaluator will provision the OS for authentication with the X.509v3 certificate. The evaluator will ensure that the certificates are validated by the OS as per FIA_X509_EXT.1.1 and then conduct the following tests:

- ***Test 1:*** *The evaluator will attempt to authenticate to the OS using the X.509v3 certificate. The evaluator will ensure that the authentication attempt is successful.*
- ***Test 2:*** *The evaluator will generate a second certificate identical to the first except for the public key and any values derived from the public key. The evaluator will attempt to authenticate to the OS with this certificate. The evaluator will ensure that the authentication attempt is unsuccessful.*

FIA_UAU.5.2
The **OS** shall authenticate any user's claimed identity according to the [**assignment**: rules describing how the multiple authentication mechanisms provide authentication].
Application Note: For all authentication mechanisms specified in FIA_UAU.5.1, the TSS shall describe the rules as to how each authentication mechanism is used. Example rules are how the authentication mechanism authenticates the user (i.e. how does the TSF verify that the correct password or authentication factor is used), the result of a successful authentication (i.e. is the user input used to derive or unlock a key) and which authentication mechanism can be used at which authentication factor interfaces (i.e. if there are times, for example, after a reboot, that only specific authentication mechanisms can be used). Rules regarding how the authentication factors interact in terms of unsuccessful authentication are covered in FIA_AFL.1.
Evaluation Activity
TSS

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

Guidance

The evaluator will verify that configuration guidance for each authentication mechanism is addressed in the AGD guidance.

Tests

- ***Test 1:*** *For each authentication mechanism selected, the evaluator will enable that mechanism and verify that it can be used to authenticate the user at the specified authentication factor interfaces.*
- ***Test 2:*** *For each authentication mechanism rule, the evaluator will ensure that the authentication mechanism(s) behave as documented in the TSS.*

FIA_X509_EXT.1 X.509 Certificate Validation

FIA_X509_EXT.1.1
The OS shall implement functionality to validate certificates in accordance with the following rules:

- RFC 5280 certificate validation and certificate path validation.
- The certificate path must terminate with a trusted CA certificate.
- The OS shall validate a certificate path by ensuring the presence of the *basicConstraints* extension and that the *CA* flag is set to TRUE for all CA certificates.
- The OS shall validate the revocation status of the certificate using [**selection**: *the Online Certificate Status Protocol (OCSP) as specified in RFC 2560*, *a Certificate Revocation List (CRL) as specified in RFC 5759*, *an OCSP TLS Status Request Extension (i.e., OCSP stapling) as specified in RFC 6066*].
- The OS shall validate the *extendedKeyUsage* field according to the following rules:
  - Certificates used for trusted updates and executable code integrity verification shall have the Code Signing purpose (id-kp 3 with OID 1.3.6.1.5.5.7.3.3) in the *extendedKeyUsage* field.
  - Server certificates presented for TLS shall have the Server Authentication purpose (id-kp 1 with OID 1.3.6.1.5.5.7.3.1) in the *extendedKeyUsage* field.
  - Client certificates presented for TLS shall have the Client Authentication purpose (id-kp 2 with OID 1.3.6.1.5.5.7.3.2) in the *extendedKeyUsage* field.
  - S/MIME certificates presented for email encryption and signature shall have the Email Protection purpose (id-kp 4 with OID 1.3.6.1.5.5.7.3.4) in the *extendedKeyUsage* field.
  - OCSP certificates presented for OCSP responses shall have the OCSP Signing purpose (id-kp 9 with OID 1.3.6.1.5.5.7.3.9) in the *extendedKeyUsage* field.
  - (Conditional) Server certificates presented for EST shall have the CMC Registration Authority (RA) purpose (id-kp-cmcRA with OID 1.3.6.1.5.5.7.3.28) in the *extendedKeyUsage* field.

.
Application Note: FIA_X509_EXT.1.1 lists the rules for validating certificates. The ST author shall select whether revocation status is verified using OCSP or CRLs. FIA_X509_EXT.2 requires that certificates are used for HTTPS, TLS and DTLS; this use requires that the *extendedKeyUsage* rules are verified.
Evaluation Activity
TSS

The evaluator will ensure the TSS describes where the check of validity of the certificates takes place. The evaluator ensures the TSS also provides a description of the certificate path validation algorithm.

Tests

The tests described must be performed in conjunction with the other certificate services evaluation 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. The evaluator will create a chain of at least four certificates: the node certificate to be tested, two Intermediate CAs, and the self-signed Root CA.

- ***Test 1:** The evaluator will demonstrate that validating a certificate without a valid certification path results in the function failing. The evaluator will then load a certificate or certificates as trusted CAs needed to validate the certificate to be used in the function, and demonstrate that the function succeeds. The evaluator shall then delete one of the certificates, and show that the function fails.*
- ***Test 2:** The evaluator will demonstrate that validating an expired certificate results in the function failing.*
- ***Test 3:** The evaluator will test that the OS can properly handle revoked certificates--conditional on whether CRL, OCSP, or OCSP stapling is selected; if multiple methods are selected, then a test shall be performed for each method. The evaluator will test revocation of the node certificate and revocation of the intermediate CA certificate (i.e. the intermediate CA certificate should be revoked by the root CA). The evaluator will 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.*
- ***Test 4:** If either OCSP option is selected, the evaluator will configure the OCSP server or use a man-in-the-middle tool to present a certificate that does not have the OCSP signing purpose and verify that validation of the OCSP response fails. If CRL is selected, the evaluator will configure the CA to sign a CRL with a certificate that does not have the cRLsign key usage bit set, and verify that validation of the CRL fails.*
- ***Test 5:** The evaluator will modify any byte in the first eight bytes of the certificate and demonstrate that the certificate fails to validate. (The certificate should fail to parse correctly.)*
- ***Test 6:** The evaluator will modify any byte in the last byte of the certificate and demonstrate that the certificate fails to validate. (The signature on the certificate should not validate.)*
- ***Test 7:** The evaluator will modify any byte in the public key of the certificate and demonstrate that the certificate fails to validate. (The signature on the certificate should not validate.)*

FIA_X509_EXT.1.2
The OS shall only treat a certificate as a CA certificate if the *basicConstraints* extension is present and the CA flag is set to TRUE.
Application Note: This requirement applies to certificates that are used and processed by the TSF and restricts the certificates that may be added as trusted CA certificates.
Evaluation Activity
Tests
The tests described must be performed in conjunction with the other certificate services evaluation activities, including the functions in FIA_X509_EXT.2.1. The evaluator will create a chain of at least four certificates: the node certificate to be tested, two Intermediate CAs, and the self-signed Root CA.

- ***Test 1:** The evaluator will construct a certificate path, such that the certificate of the CA issuing the OS's certificate does not contain the basicConstraints extension. The validation of the certificate path fails.*
- ***Test 2:** The evaluator will construct a certificate path, such that the certificate of the CA issuing the OS's certificate has the CA flag in the basicConstraints extension not set. The validation of the certificate path fails.*
- ***Test 3:** The evaluator will construct a certificate path, such that the certificate of the CA issuing the OS's certificate has the CA flag in the basicConstraints extension set to TRUE. The validation of the certificate path succeeds.*

FIA_X509_EXT.2 X.509 Certificate Authentication

FIA_X509_EXT.2.1
The OS shall use X.509v3 certificates as defined by RFC 5280 to support authentication for TLS and [**selection**: *DTLS*, *HTTPS*, [***assignment**: other protocols*], *no other protocols*] connections.
Evaluation Activity
Tests

The evaluator will acquire or develop an application that uses the OS TLS mechanism with an X.509v3 certificate. The evaluator will then run the application and ensure that the provided certificate is used to authenticate the connection.

The evaluator will repeat the activity for any other selections listed.

5.1.7 Trusted Path/Channels (FTP)

FTP_ITC_EXT.1 Trusted channel communication

FTP_ITC_EXT.1.1
The OS shall use [**selection**:

- *TLS as conforming to the Package for Transport Layer Security*,

- *DTLS as conforming to the Package for Transport Layer Security*,
- *IPsec as conforming to the EP for IPsec VPN Clients*,
- *SSH as conforming to the EP for Secure Shell*

] to provide a trusted communication channel between itself and authorized IT entities supporting the following capabilities: [**selection**: *audit server*, *authentication server*, *management server*, *[**assignment**: other capabilities]*] that is logically distinct from other communication channels and provides assured identification of its end points and protection of the channel data from disclosure and detection of modification of the channel data.
Application Note: The ST author must include the security functional requirements for the trusted channel protocol selected in FTP_ITC_EXT.1.1 in the main body of the ST.
If the ST author selects TLS or DTLS, the TSF shall be validated against requirements from the *Package for Transport Layer Security*. The TSF can act as TLS client or server, or both.
If the ST author selects IPsec, the TSF shall be validated against the *EP for IPsec Virtual Private Network (VPN) Clients*.
If the ST author selects SSH, the TSF shall be validated against the *EP for Secure Shell*.
Evaluation Activity
Tests
The evaluator will configure the OS to communicate with another trusted IT product as identified in the second selection. The evaluator will monitor network traffic while the OS performs communication with each of the servers identified in the second selection. The evaluator will ensure that for each session a trusted channel was established in conformance with the protocols identified in the first selection.

FTP_TRP.1 Trusted Path

FTP_TRP.1.1
The **OS** shall provide a communication path between itself and [**selection**: *remote*, *local*] users that is logically distinct from other communication paths and provides assured identification of its endpoints and protection of the communicated data from [modification, disclosure].
Application Note:

This requirement ensures that all remote administrative actions are protected. Authorized remote administrators must initiate all communication with the OS via a trusted path and all communication with the OS by remote administrators must be performed over this path. The data passed in this trusted communication channel is encrypted as defined in FTP_ITC_EXT.1.1. If local users access is selected and no unprotected traffic is sent to remote users, then this requirement is met. If remote users access is selected, the ST author must include the security functional requirements for the trusted channel protocol selected in FTP_ITC_EXT.1.1 in the main body of the ST.

This requirement is tested with the evaluation activities for FTP_TRP.1.3.

FTP_TRP.1.2
The **OS** shall permit [**selection**: *the TSF*, *local users*, *remote users*] to initiate communication via the trusted path.
Application Note:

This requirement is tested with the evaluation activities for FTP_TRP.1.3.

FTP_TRP.1.3
The **OS** shall require use of the trusted path for [[*all remote administrative actions*]].
Application Note:

This requirement ensures that authorized remote administrators initiate all communication with the OS via a trusted path, and that all communication with the OS by remote administrators is performed over this path. The data passed in this trusted communication channel is encrypted as defined in FTP_ITC_EXT.1.

The evaluation activities for this requirement also test requirements FTP_TRP.1.1 and FTP_TRP.1.2

Evaluation Activity
TSS
The evaluator will examine the TSS to determine that the methods of remote OS administration are indicated, along with how those communications are protected. The evaluator will also confirm that all protocols listed in the TSS in support of OS administration are consistent with those specified in the requirement, and are included in the requirements in the ST.
Guidance
The evaluator will confirm that the operational guidance contains instructions for establishing the remote administrative sessions for each supported method.
Tests
The evaluator will also perform the following tests:

- ***Test 1:** The evaluator will ensure that communications using each remote administration method is tested during the course of the evaluation, setting up the connections as described in the operational guidance and ensuring that communication is successful.*
- ***Test 2:** For each method of remote administration supported, the evaluator will follow the operational guidance to ensure that there is no available interface that can be used by a remote user to establish a remote administrative sessions without invoking the trusted path.*
- ***Test 3:** The evaluator will ensure, for each method of remote administration, the channel data is not sent in plaintext.*
- ***Test 4:** The evaluator will ensure, for each method of remote administration, modification of the channel data is*

detected by the OS.

5.1.8 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:

OBJECTIVE	ADDRESSED BY	RATIONALE
O.ACCOUNTABILITY	FAU_GEN.1, FTP_ITC_EXT.1	QQQ
O.INTEGRITY	FPT_SBOP_EXT.1, FPT_ASLR_EXT.1, FPT_TUD_EXT.1, FPT_TUD_EXT.2, FCS_COP.1/HASH, FCS_COP.1/SIGN, FCS_COP.1/KEYHMAC, FPT_ACF_EXT.1, FPT_SRP_EXT.1, FIA_X509_EXT.1, FPT_TST_EXT.1, FTP_ITC_EXT.1, FPT_W^X_EXT.1, FIA_AFL.1, FIA_UAU.5	QQQ
O.MANAGEMENT	FMT_MOF_EXT.1, FMT_SMF_EXT.1, FTA_TAB.1, FTP_TRP.1	QQQ
O.PROTECTED_STORAGE	FCS_STO_EXT.1, FCS_RBG_EXT.1, FCS_COP.1/ENCRYPT, FDP_ACF_EXT.1	QQQ
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/KEYHMAC, FDP_IFC_EXT.1, FIA_X509_EXT.1, FIA_X509_EXT.2, FTP_ITC_EXT.1	QQQ

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 evaluation activities to be performed are specified both in Section 5 Security Requirements as well as in this section.

The general model for evaluation of TOEs 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 evaluation 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 evaluation 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

The following ASE components as defined in [CEM] are required:

- Conformance claims (ASE_CCL.1)
- Extended components definition (ASE_ECD.1)
- ST introduction (ASE_INT.1)
- Security objectives (ASE_OBJ.2)
- Derived security requirements (ASE_REQ.2)
- Security Problem Definition (ASE_SPD.1)
- TOE summary specification (ASE_TSS.1)

The requirements for exact conformance of the Security Target are described in Section 2 Conformance Claims.

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 evaluation 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 OSes 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 evaluation 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.
ADV_FSP.1.2D
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 evaluation 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.

Content and presentation elements:

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 will confirm that the information provided meets all requirements for content and presentation of evidence.
ADV_FSP.1.8E
The evaluator will determine that the functional specification is an accurate and complete instantiation of the SFRs.
Evaluation Activity
There are no specific evaluation 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 evaluation 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 theIT 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 theTSF 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 Evaluation 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 evaluation 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 will confirm that the information provided meets all requirements for content and presentation of evidence.
Evaluation Activity
Some of the contents of the operational guidance are verified by the evaluation activities in Section 5.1 Security Functional Requirements and evaluation of the OS according to the [CEM]. 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 evaluation 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 will confirm that the information provided meets all requirements for content and presentation of evidence.
AGD_PRE.1.5E
The evaluator will apply the preparative procedures to confirm that the OS can be prepared securely for operation.
Evaluation Activity
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 OSes 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 will confirm that the information provided meets all requirements for content and presentation of evidence.
Evaluation Activity
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 evaluation activities are covered by the evaluation 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 will confirm that the information provided meets all requirements for content and presentation of evidence.
Evaluation Activity

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 evaluation 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 will confirm that the information provided meets all requirements for content and presentation of evidence.
Evaluation Activity

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 evaluation 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 evaluation activities are covered by the evaluation 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 will 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 Activity

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 [CEM] and the body of this PP's evaluation activities.

While it is not necessary to have one test case per test listed in an evaluation 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 will confirm that the information provided meets all requirements for content and presentation of evidence.
AVA_VAN.1.4E
The evaluator will 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 will 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 Activity

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

A.1.1 User Data Protection (FDP)

FDP_IFC_EXT.1 Information flow control

FDP_IFC_EXT.1.1
The OS shall [**selection**:

- *provide an interface which allows a VPN client to protect all IP traffic using IPsec*,
- *provide a VPN client which can protects all IP traffic using IPsec*

] with the exception of IP traffic required to establish the VPN connection and [**selection**: *signed updates directly from the OS vendor*, *no other traffic*] .
Application Note:

Typically, the traffic required to establish the VPN connection is referred to as "Control Plane" traffic, whereas the IP traffic protected by the IPsec VPN is referred to as "Data Plane" traffic. All Data Plane traffic must flow through the VPN connection and the VPN must not split-tunnel.

If no native IPsec client is validated or third-party VPN clients may also implement the required Information Flow Control, the first option shall be selected. In these cases, the TOE provides an API to third-party VPN clients that allows them to configure the TOE's network stack to perform the required Information Flow Control.

The ST author shall select the second option if the TSF implements a native VPN client (IPsec is selected in FTP_ITC_EXT.1). If the native VPN client is to be validated (IPsec is selected in FTP_ITC_EXT.1.1 and the TSF is validated against the *EP for IPsec Virtual Private Network (VPN) Clients*), the ST author shall also include FDP_IFC_EXT.1 from this package. In the future, this requirement may also make a distinction between the current requirement (which requires that when the IPsec trusted channel is enabled, all traffic from the TSF is routed through that channel) and having an option to force the establishment of an IPsec trusted channel to allow any communication by the TSF.

Evaluation Activity

TSS
The evaluator will verify that the TSS section of the ST describes the routing of IP traffic when a VPN client is enabled. The evaluator will ensure that the description indicates which traffic does not go through the VPN and which traffic does, and that a configuration exists for each in which only the traffic identified by the ST author as necessary for establishing the VPN connection (IKE traffic and perhaps HTTPS or DNS traffic) is not encapsulated by the VPN protocol (IPsec).
Tests
The evaluator will perform the following test:

- ***Test 1:***
  - ***Step 1:*** *The evaluator will enable a network connection. The evaluator will sniff packets while performing running applications that use the network such as web browsers and email clients. The evaluator will verify that the sniffer captures the traffic generated by these actions, turn off the sniffing tool, and save the session data.*
  - ***Step 2:*** *The evaluator will configure an IPsec VPN client that supports the routing specified in this requirement. The evaluator will turn on the sniffing tool, establish the VPN connection, and perform the same actions with the device as performed in the first step. The evaluator will verify that the sniffing tool captures traffic generated by these actions, turn off the sniffing tool, and save the session data.*
  - ***Step 3:*** *The evaluator will examine the traffic from both step one and step two to verify that all non-excepted Data Plane traffic in Step 2 is encapsulated by IPsec. The evaluator will examine the Security Parameter Index (SPI) value present in the encapsulated packets captured in Step 2 from the TOE to the Gateway and shall verify this value is the same for all actions used to generate traffic through the VPN. Note that it is expected that the SPI value for packets from the Gateway to the TOE is different than the SPI value for packets from theTOE to the Gateway.*
  - ***Step 4:*** *The evaluator will perform a ping on the TOE host on the local network and verify that no packets sent are captured with the sniffer. The evaluator will attempt to send packets to the TOE outside the VPN tunnel (i.e. not through the VPN gateway), including from the local network, and verify that the TOE discards them.*

A.1.2 TOE Access (FTA)

FTA_TAB.1 Default TOE access banners

FTA_TAB.1.1
Before establishing a user session, the **OS** shall display an advisory warning message regarding unauthorized use of the OS.
Evaluation Activity
Tests
The evaluator will configure the OS, per instructions in the OS manual, to display the advisory warning message "TEST TEST Warning Message TEST TEST". The evaluator will then log out and confirm that the advisory message is displayed before logging in can occur.

A.2 Objective Requirements

A.2.1 Protection of the TSF (FPT)

FPT_SRP_EXT.1 Software Restriction Policies

FPT_SRP_EXT.1.1
The OS shall restrict execution to only programs which match an administrator-specified **selection**:

- *file path*,
- *file digital signature*,
- *version*,
- *hash*,
- *[**assignment**: other characteristics]*

] .
Application Note: The assignment permits implementations which provide a low level of granularity such as a volume. The restriction is only against direct execution of executable programs. It does not forbid interpreters which may take data as an input, even if this data can subsequently result in arbitrary computation.
Evaluation Activity
Tests
For each selection specified in the ST, the evaluator will ensure that the corresponding tests result in a negative outcome (i.e., the action results in the OS denying the evaluator permission to complete the action):

- ***Test 1:*** *The evaluator will configure the OS to only allow code execution from the coreOS directories. The evaluator will then attempt to execute code from a directory that is in the allowed list. The evaluator will ensure that the code they attempted to execute has been executed.*
- ***Test 2:*** *The evaluator will configure the OS to only allow code execution from the coreOS directories. The evaluator will then attempt to execute code from a directory that is not in the allowed list. The evaluator will ensure that the code they attempted to execute has not been executed.*
- ***Test 3:*** *The evaluator will configure the OS to only allow code that has been signed by theOS vendor to execute. The evaluator will then attempt to execute code signed by the OS vendor. The evaluator will ensure that the code they*

attempted to execute has been executed.

- ***Test 4:*** *The evaluator will configure the OS to only allow code that has been signed by theOS vendor to execute. The evaluator will then attempt to execute code signed by another digital authority. The evaluator will ensure that the code they attempted to execute has not been executed.*
- ***Test 5:*** *The evaluator will configure the OS to allow execution of a specific application based on version. The evaluator will then attempt to execute the same version of the application. The evaluator will ensure that the code they attempted to execute has been executed.*
- ***Test 6:*** *The evaluator will configure the OS to allow execution of a specific application based on version. The evaluator will then attempt to execute an older version of the application. The evaluator will ensure that the code they attempted to execute has not been executed.*
- ***Test 7:*** *The evaluator will configure the OS to allow execution based on the hash of the application executable. The evaluator will then attempt to execute the application with the matching hash. The evaluator will ensure that the code they attempted to execute has been executed.*
- ***Test 8:*** *The evaluator will configure the OS to allow execution based on the hash of the application executable. The evaluator will modify the application in such a way that the application hash is changed. The evaluator will then attempt to execute the application with the matching hash. The evaluator will ensure that the code they attempted to execute has not been executed.*

FPT_W^X_EXT.1 Write XOR Execute Memory Pages

FPT_W^X_EXT.1.1
The OS shall prevent allocation of any memory region with both write and execute permissions except for [**assignment**: list of exceptions].
Application Note: Requesting a memory mapping with both write and execute permissions subverts the platform protection provided by DEP. If the OS provides no exceptions (such as for just-in-time compilation), then "no exceptions" should be indicated in the assignment. Full realization of this requirement requires hardware support, but this is commonly available.
Evaluation Activity
TSS
The evaluator will inspect the vendor-provided developer documentation and verify that no memory-mapping can be made with write and execute permissions except for the cases listed in the assignment.
Tests
The evaluator will also perform the following tests.

- ***Test 1:*** *The evaluator will acquire or construct a test program which attempts to allocate memory that is both writable and executable. The evaluator will run the program and confirm that it fails to allocate memory that is both writable and executable.*
- ***Test 2:*** *The evaluator will acquire or construct a test program which allocates memory that is executable and then subsequently requests additional write/modify permissions on that memory. The evaluator will run the program and confirm that at no time during the lifetime of the process is the memory both writable and executable.*
- ***Test 3:*** *The evaluator will acquire or construct a test program which allocates memory that is writable and then subsequently requests additional execute permissions on that memory. The evaluator will run the program and confirm that at no time during the lifetime of the process is the memory both writable and executable.*

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 theTOE 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.

This PP does not define any Selection-based requirements.

Appendix C - Implicitly Satisfied Requirements

This appendix lists requirements that should be considered satisfied by products successfully evaluated against this Protection Profile. However, these requirements are not featured explicitly as SFRs and should not be included in the ST. They are not included as standalone SFRs because it would increase the time, cost, and complexity of evaluation. This approach is permitted by [CC] Part 1, **8.2 Dependencies between components**.

This information benefits systems engineering activities which call for inclusion of particular security controls. Evaluation against the Protection Profile provides evidence that these controls are present and have been evaluated.

Requirement	Rationale for Satisfaction
FIA_UAU.1 - Timing of authentication	FIA_AFL.1 implicitly requires that the OS perform all necessary actions, including those on behalf of the user who has not been authenticated, in order to authenticate; therefore it is duplicative to include these actions as a separate assignment and test.

FIA_UID.1 - Timing of identification	FIA_AFL.1 implicitly requires that the OS perform all necessary actions, including those on behalf of the user who has not been identified, in order to authenticate; therefore it is duplicative to include these actions as a separate assignment and test.
FMT_SMR.1 - Security roles	FMT_MOF_EXT.1 specifies role-based management functions that implicitly defines user and privileged accounts; therefore, it is duplicative to include separate role requirements.
FPT_STM.1 - Reliable time stamps	FAU_GEN.1.2 explicitly requires that the OS associate timestamps with audit records; therefore it is duplicative to include a separate timestamp requirement.
FTA_SSL.1 - TSF-initiated session locking	FMT_MOF_EXT.1 defines requirements for managing session locking; therefore, it is duplicative to include a separate session locking requirement.
FTA_SSL.2 - User-initiated locking	FMT_MOF_EXT.1 defines requirements for user-initiated session locking; therefore, it is duplicative to include a separate session locking requirement.
FAU_STG.1 - Protected audit trail storage	FPT_ACF_EXT.1 defines a requirement to protect audit logs; therefore, it is duplicative to include a separate protection of audit trail requirements.
FAU_GEN.2 - User identity association	FAU_GEN.1.2 explicitly requires that the OS record any user account associated with each event; therefore, it is duplicative to include a separate requirement to associate a user account with each event.
FAU_SAR.1 - Audit review	FPT_ACF_EXT.1.2 requires that audit logs (and other objects) are protected from reading by unprivileged users; therefore, it is duplicative to include a separate requirement to protect only the audit information.

Appendix D - Entropy Documentation and Assessment

This appendix describes the required supplementary information for the entropy source used by the OS.

The documentation of the entropy source should be detailed enough that, after reading, the evaluator will thoroughly understand the entropy source and why it can be relied upon to provide sufficient entropy. This documentation should include multiple detailed sections: design description, entropy justification, operating conditions, and health testing. This documentation is not required to be part of the TSS.

Appendix E - Design Description

Documentation shall include the design of the entropy source as a whole, including the interaction of all entropy source components. Any information that can be shared regarding the design should also be included for any third-party entropy sources that are included in the product.

The documentation will describe the operation of the entropy source to include, how entropy is produced, and how unprocessed (raw) data can be obtained from within the entropy source for testing purposes. The documentation should walk through the entropy source design indicating where the entropy comes from, where the entropy output is passed next, any post-processing of the raw outputs (hash, XOR, etc.), if/where it is stored, and finally, how it is output from the entropy source. Any conditions placed on the process (e.g., blocking) should also be described in the entropy source design. Diagrams and examples are encouraged.

This design must also include a description of the content of the security boundary of the entropy source and a description of how the security boundary ensures that an adversary outside the boundary cannot affect the entropy rate.

If implemented, the design description shall include a description of how third-party applications can add entropy to the RBG. A description of any RBG state saving between power-off and power-on shall be included.

Appendix F - Entropy Justification

There should be a technical argument for where the unpredictability in the source comes from and why there is confidence in the entropy source delivering sufficient entropy for the uses made of the RBG output (by this particular OS). This argument will include a description of the expected min-entropy rate (i.e. the minimum entropy (in bits) per bit or byte of source data) and explain that sufficient entropy is going into the OS randomizer seeding process. This discussion will be part of a justification for why the entropy source can be relied upon to produce bits with entropy.

The amount of information necessary to justify the expected min-entropy rate depends on the type of entropy source included in the product.

For developer provided entropy sources, in order to justify the min-entropy rate, it is expected that a large number of raw source bits will be collected, statistical tests will be performed, and the min-entropy rate determined from the statistical tests. While no particular statistical tests are required at this time, it is expected that some testing is necessary in order to determine the amount of min-entropy in each output.

For third-party provided entropy sources, in which the OS vendor has limited access to the design and raw entropy data of the source, the documentation will indicate an estimate of the amount of min-entropy obtained from this third-party source. It is acceptable for the vendor to "assume" an amount of min-entropy, however, this assumption must be clearly stated in the documentation provided. In particular, the min-entropy estimate must be specified and the assumption included in the ST.

Regardless of type of entropy source, the justification will also include how the DRBG is initialized with the entropy stated in the ST, for example by verifying that the min-entropy rate is multiplied by the amount of source data used to seed the DRBG or that the rate of entropy expected based on the amount of source data is explicitly stated and compared to the statistical rate. If the amount of source data used to seed the DRBG is not clear or the calculated rate is not explicitly related to the seed, the documentation will not be considered complete.

The entropy justification shall not include any data added from any third-party application or from any state saving between restarts.

Appendix G - Operating Conditions

The entropy rate may be affected by conditions outside the control of the entropy source itself. For example, voltage, frequency, temperature, and elapsed time after power-on are just a few of the factors that may affect the operation of the entropy source. As such, documentation will also include the range of operating conditions under which the entropy source is expected to generate random data. It will clearly describe the measures that have been taken in the system design to ensure the entropy source continues to operate under those conditions. Similarly, documentation shall describe the conditions under which the entropy source is known to malfunction or become inconsistent. Methods used to detect failure or degradation of the source shall be included.

Appendix H - Health Testing

More specifically, all entropy source health tests and their rationale will be documented. This includes a description of the health tests, the rate and conditions under which each health test is performed (e.g., at start, continuously, or on-demand), the expected results for each health test, and rationale indicating why each test is believed to be appropriate for detecting one or more failures in the entropy source.

Appendix I - References

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-2017-04-004, Version 3.1, Revision 5, April 2017.
[NCSC]	National Cyber Security Centre - End User Device (EUD) Security 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.
[x509]	Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, May 2008.

Appendix J - Acronyms

Acronym	Meaning
AES	Advanced Encryption Standard
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
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-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
XCCDF	eXtensible Configuration Checklist Description Format
XOR	Exclusive Or
app	Application