Requirements from the

Protection Profile for General Purpose Operating Systems

Version: 4.2

2018-05-22

National Information Assurance Partnership

Revision History

Version	Date	Comment

Introduction

Purpose. This document presents the functional and assurance requirements found in the *Protection Profile for General Purpose Operating Systems*. Common Criteria evaluation, facilitated in the U.S. by the National Information Assurance Partnership (NIAP), is required for IA and IA-enabled products in National Security Systems according to CNSS Policy #11.

Using this document. This representation of the Protection Profile includes:

- *Security Functional Requirements* for use in evaluation. These are featured without the formal Assurance Activities specified in the Protection Profile, to assist the reader who is interested only in the requirements.

  It also includes, in tables shown later, particular types of security functional requirements that are not strictly required in all cases. These are:

  - *Selection-based Security Functional Requirements* which become required when certain selections are made inside the regular Security Functionality Requirements (as indicated by the **[selection:]** construct).
  - *Objective Security Functional Requirements* which are highly desired but not yet widely-available in commercial technology.
  - *Optional Security Functional Requirements* which are available for evaluation and which some customers may insist upon.

- *Security Assurance Requirements* which relate to developer support for the product under evaluation, development processes, and other non-functionality security relevant requirements.

Security Functional Requirements

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 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, the OS does not need to implement RSA key generation.

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. FCS_TLSC_EXT.1 requires cipher suites that use RSA-based key establishment schemes.

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, the OS 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.

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 the TSF'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 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, selection a, the application may have access to the file system details and may be able to logically address specific memory locations. In another implementation, selection b, 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. Selection b 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.

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.

Cryptographic Operation - Encryption/Decryption (Refined)

FCS_COP.1.1(1) 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. 128-bit key size is required in order to comply with FCS_TLSC_EXT.1 and FCS_CKM.1, if those are selected.

Cryptographic Operation - Hashing (Refined)

FCS_COP.1.1(2) The **OS** shall perform cryptographic hashing services in accordance with a specified cryptographic algorithm SHA-1 and [**selection**:

- *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 may be risk in accepting these signatures.

SHA-1 is currently required in order to comply with FCS_TLSC_EXT.1 and, depending on selections, FCS_DTLS_EXT.1. Vendors are strongly encouraged to implement updated protocols that support the SHA-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.

Cryptographic Operation - Signing (Refined)

FCS_COP.1.1(3) 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. RSA signature generation and verification is currently required in order to comply with FCS_TLSC_EXT.1.

Cryptographic Operation - Keyed-Hash Message Authentication (Refined)

FCS_COP.1.1(4) 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(1). Though HMAC with SHA-256 (HMAC-SHA-256) is listed as a selectable cipher suite in this requirement, FCS_TLSC_EXT.1 effectively makes its implementation mandatory for compliant OSes.

SHA-1 is currently required in order to comply with FCS_TLSC_EXT.1 and, depending on selections, FCS_DTLS_EXT.1. SHA-1 is currently required in order to comply with FCS_TLSC_EXT.1, but has been deprecated and should not be used for any other purpose beyond TLS and DTLS

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.

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 of NIST 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 the ST author would select 256 bits.

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 the TSS 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.

TLS Client Protocol

FCS_TLSC_EXT.1.1 The OS shall implement TLS 1.2 (RFC 5246) supporting the following cipher suites: [**selection**:

- *TLS_RSA_WITH_AES_128_CBC_SHA as defined in RFC 5246*,
- *TLS_RSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5246*,
- *TLS_RSA_WITH_AES_256_CBC_SHA256 as defined in RFC 5246*,
- *TLS_RSA_WITH_AES_256_CBC_SHA384 as defined in RFC 5288*,
- *TLS_DHE_RSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5246*,
- *TLS_DHE_RSA_WITH_AES_256_CBC_SHA256 as defined in RFC 5246*,
- *TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5288*,
- *TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5289*,
- *TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289*,
- *TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 as defined in RFC 5289*,
- *TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289*,
- *TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5289*,
- *TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289*,
- *TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 as defined in RFC 5289*,
- *TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289*

].

Application Note: The cipher suites to be tested in the evaluated configuration are limited by this requirement. The ST author should select the optional cipher suites that are supported; if there are no cipher suites supported other than the mandatory suites, then "No other cipher suite" should be selected. It is necessary to limit the cipher suites that can be used in an evaluated configuration administratively on the server in the test environment. The Suite B algorithms listed above (RFC 6460) are the preferred algorithms for implementation. TLS_RSA_WITH_AES_128_CBC_SHA is required in order to ensure compliance with RFC 5246.

These requirements will be revisited as new TLS versions are standardized by the IETF.

If any cipher suites are selected using ECDHE, then FCS_TLSC_EXT.2 is required.

FCS_TLSC_EXT.1.2 The OS shall verify that the presented identifier matches the reference identifier according to RFC 6125.

Application Note: The rules for verification of identity are described in Section 6 of RFC 6125. The reference identifier is established by the user (e.g. entering a URL into a web browser or clicking a link), by configuration (e.g. configuring the name of a mail server or authentication server), or by an application (e.g. a parameter of an API) depending on the OS service. Based on a singular reference identifier's source domain and application service type (e.g. HTTP, SIP, LDAP), the client establishes all reference identifiers which are acceptable, such as a Common Name for the Subject Name field of the certificate and a (case-insensitive) DNS name, URI name, and Service Name for the Subject Alternative Name field. The client then compares this list of all acceptable reference identifiers to the presented identifiers in the TLS server's certificate.

The preferred method for verification is the Subject Alternative Name using DNS names, URI names, or Service Names. Verification using the Common Name is required for the purposes of backwards compatibility. Additionally, support for use of IP addresses in the Subject Name or Subject Alternative name is discouraged, as against best practices, but may be implemented. Finally, the client should avoid constructing reference identifiers using wildcards. However, if the presented identifiers include wildcards, the client must follow the best practices regarding matching; these best practices are captured in the evaluation activity.

FCS_TLSC_EXT.1.3 The OS shall only establish a trusted channel if the peer certificate is valid.

Application Note: Validity is determined by the identifier verification, certificate path, the expiration date, and the revocation status in accordance with RFC 5280. Certificate validity shall be tested in accordance with testing performed for FIA_X509_EXT.1.

For TLS connections, this channel shall not be established if the peer certificate is invalid.

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.

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.

Specification of Management Functions

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

Management Function	Administrator	User
Enable/disable [**selection**: *screen lock*, *session timeout*]	X	O
Configure [**selection**: *screen lock*, *session*] inactivity timeout	X	O
Configure local audit storage capacity	O	O
Configure minimum password length	O	O

Configure minimum number of special characters in password	O	O
Configure minimum number of numeric characters in password	O	O
Configure minimum number of uppercase characters in password	O	O
Configure minimum number of lowercase characters in password	O	O
Configure lockout policy for unsuccessful authentication attempts through [**selection**: *timeouts between attempts*, *limiting number of attempts during a time period*]	O	O
Configure host-based firewall	O	O
Configure name/address of directory server with which to bind	O	O
Configure name/address of remote management server from which to receive management settings	O	O
Configure name/address of audit/logging server to which to send audit/logging records	O	O
Configure audit rules	O	O
Configure name/address of network time server	O	O
Enable/disable automatic software update	O	O
Configure WiFi interface	O	O
Enable/disable Bluetooth interface	O	O
Enable/disable [**assignment**: *list of other external interfaces*]	O	O
[**assignment**: *list of other management functions to be provided by the TSF*]	O	O

.

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 . The intent of this requirement is to ensure that the ST 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.

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*]

.

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.

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*].

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.

Boot Integrity

FPT_TST_EXT.1.1 The OS shall verify the integrity of the bootchain up through the OS 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 the OS 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 the OS 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.

Trusted Update

FPT_TUD_EXT.1.1 The OS shall provide the ability to check for updates to the OS 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.

FPT_TUD_EXT.1.2 The OS shall cryptographically verify updates to itself using a digital signature prior to installation using schemes specified in FCS_COP.1(3).

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.

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(3) prior to installation.

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].***

]

.

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.

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***

] .

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.

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".

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.

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.

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.

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.

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 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*.

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 and 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. 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 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.

Security Assurance Requirements

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.
ADV_FSP.1.1C	The functional specification shall describe the purpose and method of use for each SFR-enforcing and SFR-supporting TSFI.
ADV_FSP.1.2C	The functional specification shall identify all parameters associated with each SFR-enforcing and SFR-supporting TSFI.
ADV_FSP.1.3C	The functional specification shall provide rationale for the implicit categorization of interfaces as SFR-non-interfering.
ADV_FSP.1.4C	The tracing shall demonstrate that the SFRs trace to TSFIs in the functional specification.
ADV_FSP.1.1E	The evaluator will confirm that the information provided meets all requirements for content and presentation of evidence.
ADV_FSP.1.2E	The evaluator will determine that the functional specification is an accurate and complete instantiation of the SFRs.
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.
AGD_OPE.1.1C	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.1.2C	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.1.3C	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.1.4C	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.1.5C	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.1.6C	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.1.7C	The operational user guidance shall be clear and reasonable.

AGD_OPE.1.1E	The evaluator will confirm that the information provided meets all requirements for content and presentation of evidence.
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.
AGD_PRE.1.1C	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.1.2C	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.
AGD_PRE.1.1E	The evaluator will confirm that the information provided meets all requirements for content and presentation of evidence.
AGD_PRE.1.2E	The evaluator will apply the preparative procedures to confirm that the OS can be prepared securely for operation.
ALC_CMC.1.1D	The developer shall provide the OS and a reference for the OS.
ALC_CMC.1.1C	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
ALC_CMC.1.1E	The evaluator will confirm that the information provided meets all requirements for content and presentation of evidence.
ALC_CMS.1.1D	The developer shall provide a configuration list for the OS.
ALC_CMS.1.1C	The configuration list shall include the following: the OS itself; and the evaluation evidence required by the SARs.
ALC_CMS.1.2C	The configuration list shall uniquely identify the configuration items.
ALC_CMS.1.1E	The evaluator will confirm that the information provided meets all requirements for content and presentation of evidence.
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.
ALC_TSU_EXT.1.1C	The description shall include the process for creating and deploying security updates for the OS software.
ALC_TSU_EXT.1.2C	The description shall include the mechanisms publicly available for reporting security issues pertaining to the OS.
ALC_TSU_EXT.1.1E	The evaluator will confirm that the information provided meets all requirements for content and presentation of evidence.
ATE_IND.1.1D	The developer shall provide the OS for testing.
ATE_IND.1.1C	The OS shall be suitable for testing.
ATE_IND.1.1E	The evaluator *shall confirm* that the information provided meets all requirements for content and presentation of evidence.
ATE_IND.1.2E	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.
AVA_VAN.1.1D	The developer shall provide the OS for testing.

AVA_VAN.1.1C	The OS shall be suitable for testing.
AVA_VAN.1.1E	The evaluator will confirm that the information provided meets all requirements for content and presentation of evidence.
AVA_VAN.1.2E	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.1.3E	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.

Selection-Based Security Functional Requirements

DTLS Implementation

FCS_DTLS_EXT.1.1	The OS shall implement the DTLS protocol in accordance with [**selection**: *DTLS 1.0 (RFC 4347)*, *DTLS 1.2 (RFC 6347)*].
FCS_DTLS_EXT.1.2	The OS shall implement the requirements in TLS (FCS_TLSC_EXT.1) for the DTLS implementation, except where variations are allowed according to DTLS 1.2 (RFC 6347). **Application Note:** Differences between DTLS 1.2 and TLS 1.2 are outlined in RFC 6347; otherwise the protocols are the same. In particular, for the applicable security characteristics defined for the TSF, the two protocols do not differ. Therefore, all application notes and evaluation activities that are listed for TLS apply to the DTLS implementation.

TLS Client Protocol

FCS_TLSC_EXT.2.1	The OS shall present the Supported Elliptic Curves Extension in the Client Hello with the following NIST curves: [**selection**: *secp256r1*, *secp384r1*, *secp521r1*]. **Application Note:** This requirement does not limit the elliptic curves the client may propose for authentication and key agreement. Rather, it asks the ST author to define which of the NIST curves from FCS_COP.1(3) and FCS_CKM.1 and FCS_CKM.2 the TOE supports. This extension is required for clients supporting Elliptic Curve cipher suites.

Objective Security Functional Requirements

TLS Client Protocol

FCS_TLSC_EXT.3.1	The OS shall present the signature_algorithms extension in the Client Hello with the supported_signature_algorithms value containing the following hash algorithms: [**selection**: *SHA256*, *SHA384*, *SHA512*] and no other hash algorithms. **Application Note:** This requirement limits the hashing algorithms supported for the purpose of digital signature verification by the client and limits the server to the supported hashes for the purpose of digital signature generation by the server. The signature_algorithm extension is only supported by TLS 1.2.

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.

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.

Optional Security Functional Requirements

TLS Client Protocol

FCS_TLSC_EXT.4.1 The OS shall support mutual authentication using X.509v3 certificates.

Application Note: The use of X.509v3 certificates for TLS is addressed in FIA_X509_EXT.2.1. This requirement adds that a client must be capable of presenting a certificate to a TLS server for TLS mutual authentication.

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 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.

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.