Requirements for Vetting Mobile Apps from the

Protection Profile for Application Software

Version: 1.2

2016-04-22

National Information Assurance Partnership

Revision History

Version	Date	Comment
v 1.2	2016-04-22	Added server-side TLS requirements (selection-based) Multiple clarifications based on NIAP TRRT inquiries Refactored FDP_DEC_EXT.1 into separate components
v 1.1	2014-11-05	Addition to TLS cipher suite selections
v 1.0	2014-10-20	Initial release

Introduction

Purpose. This document presents functional and assurance requirements found in the *Protection Profile for Application Software* which are appropriate for vetting mobile application software ("apps") **outside** formal Common Criteria (ISO/IEC 15408) evaluations. 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. Such evaluations, including those for mobile apps, must use the complete Protection Profile. However, even apps without IA functionality may impose some security risks, and concern about these risks has motivated the vetting of such apps in government and industry.

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.

In addition to providing these security requirements for vetting apps, this document provides a basis for discussion and consideration of the vetting provided by commercially-available app stores. This document does not imply to Authorizing Officials that the vetting provided by commercially-available app stores is either adequate or inadequate for the context in which

they must weigh risks. Rather, it is intended to help inform and support decision-making with regard to investment in app vetting processes.

Security Functional Requirements

Random Bit Generation Services

FCS_RBG_EXT.1.1 The application shall [**selection**:

use no DRBG functionality,

invoke platform-provided DRBG functionality,

implement DRBG functionality

] for its cryptographic operations.

Application Note: If *implement DRBG functionality* is chosen, then additional FCS_RBG_EXT.2 elements shall be included in the ST. In this requirement, cryptographic operations include all cryptographic key generation/derivation/agreement, IVs (for certain modes), as well as protocol-specific random values.

Storage of Credentials

FCS_STO_EXT.1.1 The application shall [**selection**:

not store any credentials,

*invoke the functionality provided by the platform to securely store [**assignment**: list of credentials]* ,

*implement functionality to securely store [**assignment**: list of credentials]*

] to non-volatile memory.

Application Note: This requirement ensures that persistent credentials (secret keys, PKI private keys, or passwords) are stored securely.
The assurance activity implicitly restricts which selections can be made, on per-platform basis. For example, if a platform provides hardware-backed protection for credential storage, then the third selection cannot be indicated.
If *implement functionality to securely store* credentials is selected, then the following components must be included in the ST: FCS_COP.1(1). If other cryptographic operations are used to implement the secure storage of credentials, the corresponding requirements must be included in the ST.

Access to Platform Resources

FDP_DEC_EXT.1.1 The application shall restrict its access to [**selection**:

no hardware resources,

network connectivity,

camera,

microphone,

location services,

NFC,

USB,

Bluetooth,

*[**assignment**: list of additional hardware resources]*

] .

Application Note: The intent is for the evaluator to ensure that the selection captures all hardware resources which the application accesses, and that these are restricted to those which are justified. On some platforms, the application must explicitly solicit permission in order to access hardware resources. Seeking such permissions, even if the application does not later make use of the hardware resource, should still be considered access. Selections should be expressed in a manner consistent with how the application expresses its access needs to the underlying platform. For example, the platform may provide *location services* which implies the potential use of a variety of hardware resources (e.g. satellite receivers, WiFi, cellular radio) yet *location services* is the proper selection. This is because use of these resources can be inferred, but also because the actual usage may vary based on the particular platform. Resources that do not need to be explicitly identified are

those which are ordinarily used by any application such as central processing units, main memory, displays, input devices (e.g. keyboards, mice), and persistent storage devices provided by the platform.

FDP_DEC_EXT.1.2 The application shall restrict its access to [**selection**:

no sensitive information repositories,

address book,

calendar,

call lists,

system logs,

*[**assignment**: list of additional sensitive information repositories]*

] .

Application Note: *Sensitive information repositories* are defined as those collections of sensitive data that could be expected to be shared among some applications, users, or user roles, but to which not all of these would ordinarily require access.

Network Communications

FDP_NET_EXT.1.1 The application shall restrict network communication to [**selection**:

no network communication,

*user-initiated communication for [**assignment**: list of functions for which the user can initiate network communication]* ,

*respond to [**assignment**: list of remotely initiated communication]* ,

*[**assignment**: list of application-initiated network communication]*

] .

Application Note: This requirement is intended to restrict both inbound and outbound network communications to only those required, or to network communications that are user initiated. It does not apply to network communications in which the application may generically access the filesystem which may result in the platform accessing remotely mounted drives/shares.

Encryption Of Sensitive Application Data

FDP_DAR_EXT.1.1 The application shall [**selection**:

leverage platform-provided functionality to encrypt sensitive data,

implement functionality to encrypt sensitive data,

not store any sensitive data

] in non-volatile memory.

Application Note: If *implement functionality to encrypt sensitive data* is selected, then evaluation is required against the *Application Software Protection Profile Extended Package: File Encryption*. Any file that may potentially contain sensitive data (to include temporary files) shall be protected. The only exception is if the user intentionally exports the sensitive data to non-protected files.

Supported Configuration Mechanism

FMT_MEC_EXT.1.1 The application shall invoke the mechanisms recommended by the platform vendor for storing and setting configuration options.

Application Note: Configuration options that are stored remotely are not subject to this requirement.

Secure by Default Configuration

FMT_CFG_EXT.1.1 The application shall provide only enough functionality to set new credentials when configured with default credentials or no credentials.

Application Note: Default credentials are credentials (e.g., passwords, keys) that are automatically (without user interaction) loaded onto the platform during application installation. Credentials that are generated during installation using requirements laid out in FCS_RBG_EXT.1 are not by definition default credentials.

FMT_CFG_EXT.1.2 The application shall be configured by default with file permissions which protect it and its data from unauthorized access.

Application Note: The precise expectations for file permissions vary per platform but the general intention is that a trust boundary protects the application and its data.

Specification of Management Functions

FMT_SMF.1.1 The TSF shall be capable of performing the following management functions [**selection**:

no management functions,

enable/disable the transmission of any information describing the system's hardware, software, or configuration ,

enable/disable the transmission of any PII ,

enable/disable transmission of any application state (e.g. crashdump) information,

*enable/disable network backup functionality to [**assignment**: list of enterprise or commercial cloud backup systems]* ,

*[**assignment**: list of other management functions to be provided by the TSF]*

] .

Application Note: this requirement stipulates that an application needs to provide the ability to enable/disable only those functions that it actually implements. the application is not responsible for controlling the behavior of the platform or other applications.

User Consent for Transmission of Personally Identifiable Information

FPR_ANO_EXT.1.1 The application shall [**selection**:

not transmit PII over a network ,

*require user approval before executing [**assignment**: list of functions that transmit PII over a network]*

] .

Application Note: This requirement applies only to PII that is specifically requested by the application; it does not apply if the user volunteers PII without prompting from the application into a general (or inappropriate) data field. A dialog box that declares intent to send PII presented to the user at the time the application is started is sufficient to meet this requirement.

Use of Supported Services and APIs

FPT_API_EXT.1.1 The application shall use only documented platform APIs.

Application Note: The definition of *documented* may vary depending upon whether the application is provided by a third party (who relies upon documented platform APIs) or by a platform vendor who may be able to guarantee support for platform APIs.

Anti-Exploitation Capabilities

FPT_AEX_EXT.1.1 The application shall not request to map memory at an explicit address except for [**assignment**: *list of explicit exceptions]* .

Application Note: Requesting a memory mapping at an explicit address subverts address space layout randomization (ASLR).

FPT_AEX_EXT.1.2 The application shall [**selection**:

not allocate any memory region with both write and execute permissions ,

*allocate memory regions with write and execute permissions for only [**assignment**: list of functions performing just-in-time compilation]*

] .

Application Note: Requesting a memory mapping with both write and execute permissions subverts the platform protection provided by DEP. If the application performs no just-in-time compiling, then the first selection must be chosen.

FPT_AEX_EXT.1.3 The application shall be compatible with security features provided by the platform vendor.

Application Note: This requirement is designed to ensure that platform security features do not need to be disabled in order for the application to run.

FPT_AEX_EXT.1.4 The application shall not write user-modifiable files to directories that contain executable files unless explicitly directed by the user to do so.

Application Note: Executables and user-modifiable files may not share the same parent directory, but may share directories above the parent.

FPT_AEX_EXT.1.5 The application shall be compiled with stack-based buffer overflow protection enabled.

Integrity for Installation and Update

FPT_TUD_EXT.1.1 The application shall [**selection**: *provide the ability*, *leverage the platform*] to check for updates and patches to the application software.

Application Note: This requirement is about the ability to "check" for updates. The actual installation of any updates should be done by the platform. This requirement is intended to ensure that the application can check for updates provided by the vendor, as updates provided by another source may contain malicious code.

FPT_TUD_EXT.1.2 The application shall be distributed using the format of the platform-supported package manager.

FPT_TUD_EXT.1.3 The application shall be packaged such that its removal results in the deletion of all traces of the application, with the exception of configuration settings, output files, and audit/log events.

Application Note: Applications bundled with the system/firmware image are not subject to this requirement if the user is unable to remove the application through means provided by the OS.

FPT_TUD_EXT.1.4 The application shall not download, modify, replace or update its own binary code.

Application Note: This requirement applies to the code of the application; it does not apply to mobile code technologies that are designed for download and execution by the application.

FPT_TUD_EXT.1.5 The application shall [**selection**, at least one of: *provide the ability*, *leverage the platform*] to query the current version of the application software.

FPT_TUD_EXT.1.6 The application installation package and its updates shall be digitally signed such that its platform can cryptographically verify them prior to installation.

Application Note: The specifics of the verification of installation packages and updates involves requirements on the platform (and not the application), so these are not fully specified here.

Use of Third Party Libraries

FPT_LIB_EXT.1.1 The application shall be packaged with only [**assignment**: *list of third-party libraries]* .

Application Note: The intention of this requirement is for the evaluator to discover and document whether the application is including unnecessary or unexpected third-party libraries. This includes adware libraries which could present a privacy threat, as well as ensuring documentation of such libraries in case vulnerabilities are later discovered.

Protection of Data in Transit

FTP_DIT_EXT.1.1 The application shall [**selection**:

not transmit any data,

not transmit any sensitive data,

*encrypt all transmitted sensitive data with [**selection**, at least one of: HTTPS, TLS, DTLS, SSH as conforming to the Extended Package for Secure Shell]* ,

*encrypt all transmitted data with [**selection**, at least one of: HTTPS, TLS, DTLS, SSH]*

] between itself and another trusted IT product.

Application Note: Extended packages may override this requirement to provide for other protocols. Encryption is not required for applications transmitting data that is not sensitive.

If *TLS* is selected, then evaluation of elements from FCS_TLSC_EXT.1 is required.

If *HTTPS* is selected, then evaluation of elements from FCS_HTTPS_EXT.1 is required.

If *DTLS* is selected, then evaluation of elements from FCS_DTLS_EXT.1 is required.

If *SSH* is selected, the TSF shall be validated against the *Extended Package for Secure Shell*.

Security Assurance Requirements

ALC_CMC.1.1C The application shall be labeled with a unique reference.

Application Note: Unique reference information includes:

- Application Name
- Application Version
- Application Description
- Platform on which Application Runs
- Software Identification (SWID) tags, if available

ATE_IND.1.2E The evaluator shall test a subset of the TSF to confirm that the TSF operates as specified.

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

AVA_VAN.1.1C The application shall be suitable for testing.

Application Note: Suitability for testing means not being obfuscated or packaged in such a way as to disrupt either static or dynamic analysis by the evaluator.

AVA_VAN.1.2E The evaluator shall perform a search of public domain sources to identify potential vulnerabilities in the TOE.

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.

Selection-Based Security Functional Requirements

Random Bit Generation from Application

FCS_RBG_EXT.2.1 The application 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: This requirement shall be included in STs in which *implement DRBG functionality* is chosen in FCS_RBG_EXT.1.1. The ST author should select the standard to which the RBG services comply (either SP 800-90A or FIPS 140-2 Annex C).
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 (if SP 800-90A is selected), 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.2.2 The deterministic RBG shall be seeded by an entropy source that accumulates entropy from a platform-based DRBG and [**selection**:

a software-based noise source,

no other 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: This requirement shall be included in STs in which *implement DRBG functionality* is chosen in FCS_RBG_EXT.1.1. 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 application's DRBG. Note that the application must use the platform's DRBG to seed its 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) and AES 256 (security strength 256 bits), then the ST author would select 256 bits.

Cryptographic Key Generation Services

FCS_CKM_EXT.1.1 The application shall [**selection**:

generate no asymmetric cryptographic keys,

invoke platform-provided functionality for asymmetric key generation,

implement asymmetric key generation

] .

Application Note: If *implement asymmetric key generation* or *invoke platform-provided functionality for asymmetric key generation* is chosen, then additional FCS_CKM.1(1) elements shall be included in the ST.

Cryptographic Asymmetric Key Generation

FCS_CKM.1.1(1) The application 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: [****selection****:*

FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.3 *,*

ANSI X9.31-1998, Section 4.1

] ,

[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]***

] .

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.2.1 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 TOE acts as a receiver in the RSA key establishment scheme, the TOE does not need to implement RSA key generation.
The ANSI X9.31-1998 option will be removed from the selection in a future publication of this document. Presently, the selection is not exclusively limited to the FIPS PUB 186-4 options in order to allow industry some further time to complete the transition to the modern FIPS PUB 186-4 standard.

Cryptographic Key Establishment

FCS_CKM.2.1 The application shall [**selection**: *invoke platform-provided functionality* , *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

] .

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 TOE acts as a receiver in the RSA key establishment scheme, the TOE 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(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(1).

Cryptographic Operation - Encryption/Decryption

FCS_COP.1.1(1) The application shall perform encryption/decryption in accordance with a specified cryptographic algorithm

- AES-CBC (as defined in NIST SP 800-38A) mode;

and [**selection**:

AES-GCM (as defined in NIST SP 800-38D),

no other modes

] and cryptographic key sizes 256-bit and [**selection**: *128-bit*, *no other key sizes*] .

Application Note: For the first selection, the ST author should choose the mode or modes in which AES operates. For the second 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(1), if those are selected.

Cryptographic Operation - Hashing

FCS_COP.1.1(2) The application 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 [**selection**:

160,

256,

384,

512,

no other message digest sizes

] bits 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. If FCS_TLSC_EXT.1.1 is included in the ST, the hashing algorithms selection for FCS_COP.1(2) must match the hashing algorithms used in the mandatory and selected cipher suites of FCS_TLSC_EXT.1.1. 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 (for example, SHA 256 for 128-bit keys).

Cryptographic Operation - Signing

FCS_COP.1.1(3) The application 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*

] .

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

FCS_COP.1.1(4) The application shall perform keyed-hash message authentication in accordance with a specified cryptographic algorithm

- HMAC-SHA-256

and [**selection**:

SHA-1,

SHA-384,

SHA-512,

no other algorithms

] with key sizes [**assignment**: *key size (in bits) used in HMAC]* and message digest sizes 256 and [**selection**: *160*, *384*, *512*, *no other size*] 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 application (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). HMAC-SHA256 is required in order to comply with the required cipher suites in FCS_TLSC_EXT.1.

TLS Client Protocol

FCS_TLSC_EXT.1.1 The application shall [**selection**: *invoke platform-provided TLS 1.2*, *implement TLS 1.2 (RFC 5246)*] supporting the following cipher suites:

Mandatory Cipher Suites: TLS_RSA_WITH_AES_128_CBC_SHA as defined in RFC 5246

Optional Cipher Suites: [**selection**:

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_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,

TLS_RSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5246,

TLS_RSA_WITH_AES_256_CBC_SHA256 as defined in RFC 5246,

no other cipher suite

] .

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 "None" 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.4 is required.
If *implement TLS 1.2 (RFC 5246)* is selected, then FCS_CKM.2, FCS_CKM_EXT.1, FCS_COP.1(1), FCS_COP.1(2), FCS_COP.1(3), and FCS_COP.1(4) are required.

FCS_TLSC_EXT.1.2 The application 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 application 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 assurance activity.

FCS_TLSC_EXT.1.3 The application shall establish a trusted channel only 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. The HTTPS protocol (FCS_HTTPS_EXT.1) requires different behavior, though HTTPS is implemented over TLS. This element addresses non-HTTPS TLS connections.

TLS Client Protocol

FCS_TLSC_EXT.4.1 The application shall present the supported Elliptic Curves Extension in the Client Hello with the following NIST curves: [**selection**: *secp256r1*, *secp384r1*, *secp521r1*] and no other curves.

Application Note: This requirement limits the elliptic curves allowed for authentication and key agreement to the NIST curves from FCS_COP.1(3) and FCS_CKM.1(1) and FCS_CKM.2. This extension is required for clients supporting Elliptic Curve cipher suites.

TLS Server Protocol

FCS_TLSS_EXT.1.1 The application shall [**selection**: *invoke platform-provided TLS 1.2*, *implement TLS 1.2 (RFC 5246)*] supporting the following cipher suites:

Mandatory Cipher Suites: TLS_RSA_WITH_AES_128_CBC_SHA as defined in RFC 5246

Optional Cipher Suites: [**selection**:

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_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,

TLS_RSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5246,

TLS_RSA_WITH_AES_256_CBC_SHA256 as defined in RFC 5246,

no other cipher suite

] .

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 “None” 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.4 is required.
If *implement TLS 1.2 (RFC 5246)* is selected, then FCS_CKM.2.1, FCS_COP.1.1(1), FCS_COP.1.1(2), FCS_COP.1.1(3), and FCS_COP.1.1(4) are required.

FCS_TLSS_EXT.1.2 The application shall deny connections from clients requesting SSL 2.0, SSL 3.0, TLS 1.0, TLS 1.1, and [**selection**: *TLS 1.2*, *none*] .

Application Note: All SSL versions and TLS 1.0 and 1.1 are denied. Any TLS version not selected in FCS_TLSS_EXT.1.1 should be selected here.

FCS_TLSS_EXT.1.3 The application shall generate key establishment parameters using RSA with size 2048 bits and [**selection**: *3072 bits*, *4096 bits*, *no other sizes*] and [**selection**: *over NIST curves [**selection**: secp256r, secp384r] and no other curves*, *Diffe-Hellman parameters of size 2048 and [**selection**: 3072 bits, no other size] , no other*]

Application Note: If the ST lists a DHE ciphersuite in FCS_TLSS_EXT.1.1, the ST must include the Diffie-Hellman selection in the requirement

FCS_TLSS_EXT.1.4 The application shall support mutual authentication of TLS clients using X.509v3 certificates.

FCS_TLSS_EXT.1.5 The application shall not establish a trusted channel if the peer certificate is invalid.

Application Note: The use of X.509v3 certificates for TLS is addressed in FIA_X509_EXT.2.1 This requirement adds that this use must include support for client-side certificates for TLS mutual authentication. Validity is determined by the 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.

FCS_TLSS_EXT.1.6 The application shall not establish a trusted channel if the distinguished name (DN) or Subject Alternative Name (SAN) contained in a certificate does not match the expected identifier for the peer.

Application Note: The peer identifier may be in the Subject field or the Subject Alternative Name extension of the certificate. The expected identifier may either be configured, may be compared to the Domain Name, IP address, username, or email address used by the peer, or may be passed to a directory server for comparison. Matching should be performed by a bit-wise comparison.

DTLS Implementation

FCS_DTLS_EXT.1.1 The application shall implement the DTLS protocol in accordance with DTLS 1.2 (RFC 6347).

FCS_DTLS_EXT.1.2 The application 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 assurance activities that are listed for TLS apply to the DTLS implementation.

FCS_DTLS_EXT.1.3 The application shall not establish a trusted communication channel if the peer certificate is deemed invalid.

Application Note: Validity is determined by the certificate path, the expiration date, and the revocation status in accordance with RFC 5280.

HTTPS Protocol

FCS_HTTPS_EXT.1.1 The application shall implement the HTTPS protocol that complies with RFC 2818.

FCS_HTTPS_EXT.1.2 The application shall implement HTTPS using TLS (FCS_TLSC_EXT.1).

FCS_HTTPS_EXT.1.3 The application shall notify the user and [**selection**: *not establish the connection* , *request application authorization to establish the connection* , *no other action*] if the peer certificate is deemed invalid.

Application Note: Validity is determined by the certificate path, the expiration date, and the revocation status in accordance with RFC 5280.

X.509 Certificate Validation

FIA_X509_EXT.1.1 The application shall [**selection**: *invoked platform-provided functionality* , *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 application 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 application 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 application 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.
  - 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.
Regardless of the selection of *implement functionality* or *invoke platform-provided functionality*, the validation is expected to end in a trusted root CA certificate in a root store managed by the platform.

FIA_X509_EXT.1.2 The application shall treat a certificate as a CA certificate only 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 application shall use X.509v3 certificates as defined by RFC 5280 to support authentication for [**selection**: *HTTPS* , *TLS* , *DTLS*] .

Application Note: The ST author's selection shall match the selection in FTP_DIT_EXT.1.1.

FIA_X509_EXT.2.2 When the application cannot establish a connection to determine the validity of a certificate, the application shall [**selection**: *allow the administrator to choose whether to accept the certificate in these cases* , *accept the certificate* , *not accept the certificate*] .

Application Note: Often a connection must be established to perform a verification of the revocation status of a certificate - either to download a CRL or to perform OCSP. The selection is used to describe the behavior in the event that such a connection cannot be established (for example, due to a network error). If the TOE has determined the certificate valid according to all

other rules in FIA_X509_EXT.1, the behavior indicated in the selection shall determine the validity. The TOE must not accept the certificate if it fails any of the other validation rules in FIA_X509_EXT.1.

Objective Security Functional Requirements

TLS Client Protocol

FCS_TLSC_EXT.3.1 The application 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.

Use of Supported Services and APIs

FPT_API_EXT.2.1 The application [**selection**: *shall use platform-provided libraries*, *does not implement functionality*] for parsing [**assignment**: *list of formats parsed that are included in the IANA MIME media types]* .

Application Note: The IANA MIME types are listed at http://www.iana.org/assignments/media-types and include many image, audio, video, and content file formats. This requirement does not apply if providing parsing services is the purpose of the application.

Software Identification and Versions

FPT_IDV_EXT.1.1 The application shall include SWID tags that comply with the minimum requirements for SWID tag from ISO/IEC 19770-2:2009 standard.

Application Note: Valid SWID tags must contain a SoftwareIdentity element and an Entity element as defined in the ISO/IEC 19770-2:2009 standard. SWID tags must be stored with a .swidtag file extensions as defined in the ISO/IEC 19770-2:2009.

Optional Security Functional Requirements

Cryptographic Symmetric Key Generation

FCS_CKM.1.1(2) The application shall generate symmetric cryptographic keys using a Random Bit Generator as specified in FCS_RBG_EXT.1 and specified cryptographic key sizes [**selection**:

128 bit,

256 bit

] .

Application Note: Symmetric keys may be used to generate keys along the key chain.

TLS Client Protocol

FCS_TLSC_EXT.2.1 The application 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.