Functional Package for Transport Layer Security (TLS)

Version: 2.0-draft
2022-08-24
National Information Assurance Partnership

Revision History

Version	Date	Comment
1.0	2018-12-17	First publication
1.1	2019-03-01	Clarifications regarding override for invalid certificates, renegotiation_info extension, DTLS versions, and named Diffie-Hellman groups in DTLS contexts
2.0	2022-08-24	Added audit events, added TLS 1.3 support, deprecated TLS 1.0 and 1.1, updated algorithms/ciphersuites in accordance with CNSA suite RFC and to consider PSK, restructured SFRs for clarity

Contents

1 Introduction

1.1 Overview

Transport Layer Security (TLS) and the closely-related Datagram TLS (DTLS) are cryptographic protocols designed to provide communications security over IP networks. Several versions of the protocol are in widespread use in software that provides functionality such as web browsing, email, instant messaging, and voice-over-IP (VoIP). Major websites use TLS to protect communications to and from their servers. TLS is also used to protect communications between hosts and network infrastructure devices for administration. The underlying platform, such as an operating system, often provides the actual TLS implementation. The primary goal of the TLS protocol is to provide confidentiality and integrity of data transmitted between two communicating endpoints, as well as authentication of at least the server endpoint.

TLS supports many different methods for exchanging keys, encrypting data, and authenticating message integrity. These methods are dynamically negotiated between the client and server when the TLS connection is established. As a result, evaluating the implementation of both endpoints is typically necessary to provide assurance for the operating environment.

This "Functional Package for Transport Layer Security" (short name "TLS-PKG") defines functional requirements for the implementation of the TLS and DTLS protocols. The requirements are intended to improve the security of products by enabling their evaluation.

1.2 Terms

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

1.2.1 Common Criteria Terms

Assurance	Grounds for confidence that a TOE meets the SFRs [CC].
Base Protection Profile (Base-PP)	Protection Profile used as a basis to build a PP-Configuration.
Collaborative Protection Profile (cPP)	A Protection Profile developed by international technical communities and approved by multiple schemes.
Common Criteria (CC)	Common Criteria for Information Technology Security Evaluation (International Standard ISO/IEC 15408).
Common Criteria Testing Laboratory	Within the context of the Common Criteria Evaluation and Validation Scheme (CCEVS), an IT security evaluation facility accredited by the National Voluntary Laboratory Accreditation Program (NVLAP) and approved by the NIAP Validation Body to conduct Common Criteria-based evaluations.
Common Evaluation Methodology (CEM)	Common Evaluation Methodology for Information Technology Security Evaluation.
Distributed TOE	A TOE composed of multiple components operating as a logical whole.
Extended Package (EP)	A deprecated document form for collecting SFRs that implement a particular protocol, technology, or functionality. See Functional Packages.
Functional Package (FP)	A document that collects SFRs for a particular protocol, technology, or functionality.
Operational Environment (OE)	Hardware and software that are outside the TOE boundary that support the TOE functionality and security policy.
Protection Profile (PP)	An implementation-independent set of security requirements for a category of products.
Protection Profile Configuration (PP-Configuration)	A comprehensive set of security requirements for a product type that consists of at least one Base-PP and at least one PP-Module.

Protection Profile Module (PP-Module)	An implementation-independent statement of security needs for a TOE type complementary to one or more Base-PPs.
Security Assurance Requirement (SAR)	A requirement to assure the security of the TOE.
Security Functional Requirement (SFR)	A requirement for security enforcement by the TOE.
Security Target (ST)	A set of implementation-dependent security requirements for a specific product.
Target of Evaluation (TOE)	The product under evaluation.
TOE Security Functionality (TSF)	The security functionality of the product under evaluation.
TOE Summary Specification (TSS)	A description of how a TOE satisfies the SFRs in an ST.

1.2.2 Technical Terms

Certificate Authority (CA)	Issuer of digital certificates.
Datagram Transport Layer Security (DTLS)	Cryptographic network protocol, based on TLS, which provides communications security for datagram protocols.
Transport Layer Security (TLS)	Cryptographic network protocol for providing communications security over a TCP/IP network.

1.3 Compliant Targets of Evaluation

The Target of Evaluation (TOE) in this Package is a product which acts as a (D)TLS client, a (D)TLS server, or both. This Package describes the security functionality of TLS and DTLS in terms of [CC].

The contents of this Package must be appropriately combined with a PP or PP-Module. When this Package is instantiated by a PP or PP-Module, the Package must include selection-based requirements in accordance with the selections or assignments indicated in the PP or PP-Module. These may be expanded by the the ST author.

The PP or PP-Module which instantiates this Package must typically include the following components in order to satisfy dependencies of this Package. It is the responsibility of the PP or PP-Module author who instantiates this Package to ensure that dependence on these components is satisfied:

Component	Explanation
FCS_CKM.1	To support TLS ciphersuites that use RSA, DHE or ECDHE for key exchange, the PP or PP-Module must include FCS_CKM.1 and specify the corresponding key generation algorithm.
FCS_CKM.2	To support TLS ciphersuites that use RSA, DHE or ECDHE for key exchange, the PP or PP-Module must include FCS_CKM.2 and specify the corresponding algorithm.
FCS_COP.1	To support TLS ciphersuites that use AES for encryption and decryption, the PP or PP-Module must include FCS_COP.1 (iterating as needed) and specify AES with corresponding key sizes and modes. To support TLS ciphersuites that use SHA for hashing, the PP or PP-Module must include FCS_COP.1 (iterating as needed) and specify SHA with corresponding digest sizes.
FCS_RBG_EXT.1	To support random bit generation needed for the TLS handshake, the PP or PP-Module must include FCS_RBG_EXT.1.
FIA_X509_EXT.1	To support validation of certificates needed during TLS connection setup, the PP or PP-Module must include FIA_X509_EXT.1.
FIA_X509_EXT.2	To support the use of X509 certificates for authentication in TLS connection setup, the PP

or PP-Module must include FIA_X509_EXT.2.

An ST must identify the applicable version of the PP or PP-Module and this Package in its conformance claims.

2 Conformance Claims

Conformance Statement

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

CC Conformance Claims

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

PP Claim

This Package does not claim conformance to any Protection Profile.

Package Claim

This Package does not claim conformance to any packages.

Conformance Statement

This Package serves to provide Protection Profiles with additional SFRs and associated Evaluation Activities specific to TLS clients and servers.

This Package conforms to Common Criteria [CC] for Information Technology Security Evaluation, Version 3.1, Revision 5. It is CC Part 2 extended conformant.

In accordance with CC Part 1, dependencies are not included when they are addressed by other SFRs. The evaluation activities provide adequate proof that any dependencies are also satisfied.

3 Security Functional Requirements

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

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

3.1 Auditable Events for Mandatory SFRs

The auditable events specified in this Functional Package are included in a Security Target if the incorporating PP or PP-Module supports audit event reporting through FAU_GEN.1 and all other criteria in the incorporating PP or PP-Module are met.

Table 1: Auditable Events for Mandatory Requirements

Requirement	Auditable Events	Additional Audit Record Contents
FCS_TLS_EXT.1	No events specified	N/A

3.2 Cryptographic Support (FCS)

FCS_TLS_EXT.1 TLS Protocol

FCS_TLS_EXT.1.1

The TSF shall implement [**selection**:

- *TLS as a client*
- *TLS as a server*
- *DTLS as a client*
- *DTLS as a server*

].

Application Note: If *TLS as a client* is selected, then the ST must include the requirements from FCS_TLSC_EXT.1.
If *TLS as a server* is selected, then the ST must include the requirements from FCS_TLSS_EXT.1.

If *DTLS as a client* is selected, then the ST must include the requirements from FCS_DTLSC_EXT.1.
If *DTLS as a server* is selected, then the ST must include the requirements from FCS_DTLSS_EXT.1.

Evaluation Activities

FCS_TLS_EXT.1
TSS
The evaluator shall examine the TSS to verify that the TLS and DTLS claims are consistent with those selected in the SFR.

Guidance
The evaluator shall ensure that the selections indicated in the ST are consistent with selections in the dependent components.

Tests
There are no test activities for this SFR; the following information is provided as an overview of the expected functionality and test environment for all subsequent SFRs.

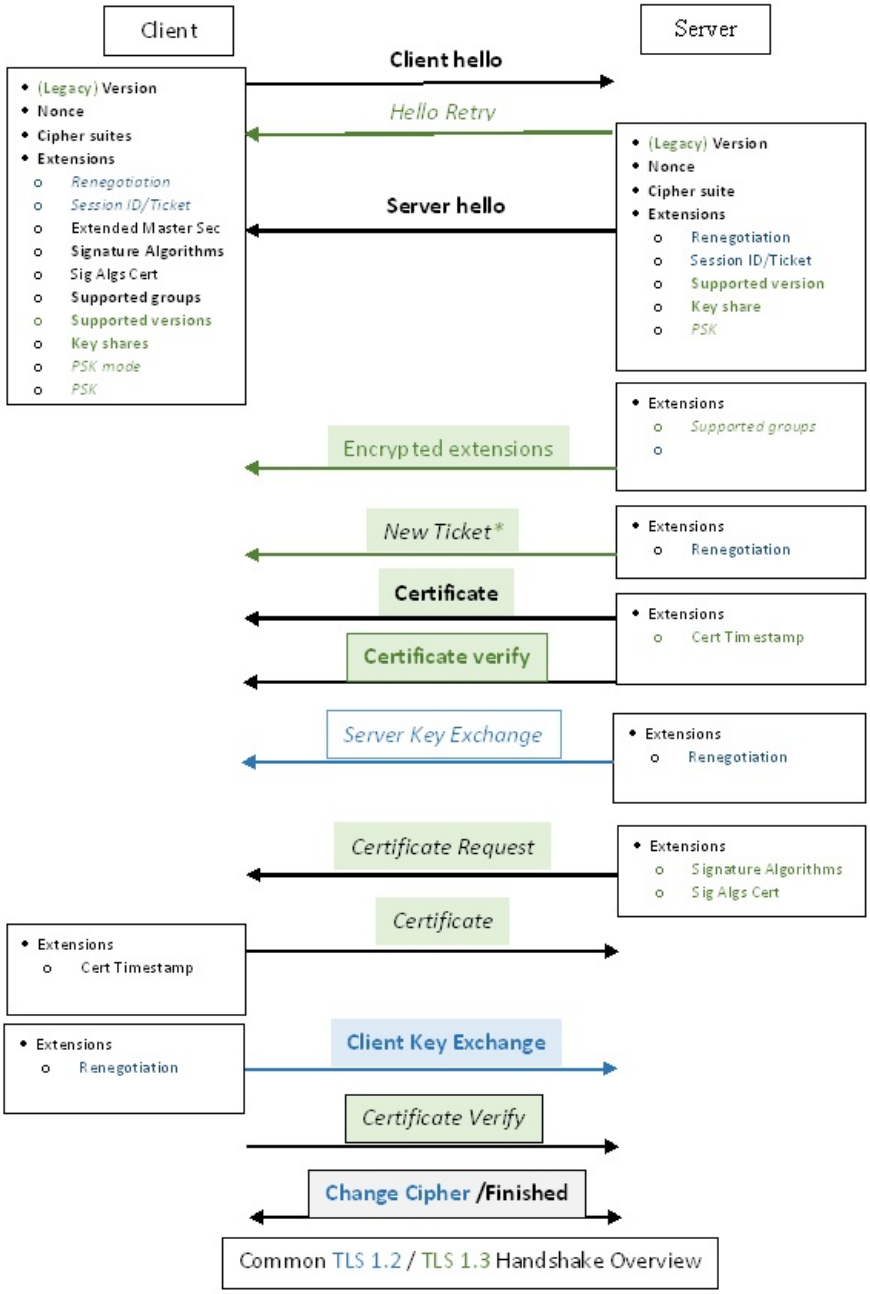

Figure 1: TLS Hello

The chart above provides an overview of the TLS hello messages, the content and protections, and the establishment of cryptographic keys in support of the protections.

- *Blue text indicates a message or content unique to TLS 1.2.*
- *Green text indicates uniqueness to TLS 1.3.*
- *Black text indicates features common to both TLS 1.2 and TLS 1.3.*
- *Bold text indicates mandatory features.*
- *Italics emphasizes optional features.*
- *A shaded text box indicates that the message is encrypted for TLS 1.2 (blue), TLS 1.3 (green) or both TLS 1.2 and TLS 1.3 (grey).*
- *An outlined text box indicates that the content in the message is signed, and/or provides authentication of the handshake to that point.*

Test Environment:

Tests for TLS 1.2 and TLS 1.3 include examination of the handshake messages and behavior of the TSF when presented with unexpected or invalid messages. For TLS 1.2 and below, previous versions of this Functional Package only required visibility of network traffic and the ability to modify a valid handshake message sent to the TSF.

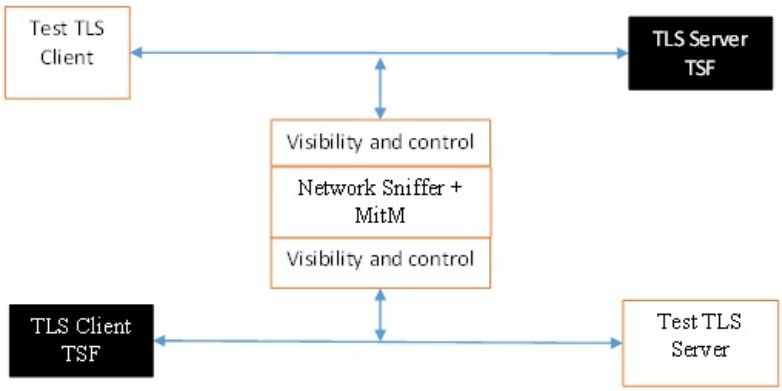

Figure 2: Test environment for TLS 1.2 using network traffic visibility and control tools

TLS 1.3 introduces the encryption of handshake messages subsequent to the server hello exchange which prevents visibility and control using midpoint capabilities. To achieve equivalent validation of TLS 1.3 requires the ability to modify the traffic underlying the encryption applied after the server hello message. This can be achieved by introducing additional control of the messages sent, and visibility of messages received by the test TLS client, when validating TLS server functionality or test server, when validating TLS client functionality.

Figure 3: Test environment for TLS 1.3 using custom endpoint capabilities for visibility and control

Typically, a compliant TLS 1.3 library modified to provide visibility and control of the handshake messages prior to encryption suffices for all tests. Such modification will require the test client and/or server to be validated.

Since validations of products supporting only TLS 1.2 are still expected under this Package, the test environment for TLS 1.2-only validations may include network sniffers and man-in-the-middle products that do not require such modifications to a compliant TLS 1.2 library. For consistency, a compliant TLS client (or TLS server) together with the network sniffers and man-in-the-middle capabilities will also be referred to as a test TLS client (or test TLS server, respectively) in the following evaluation activities.

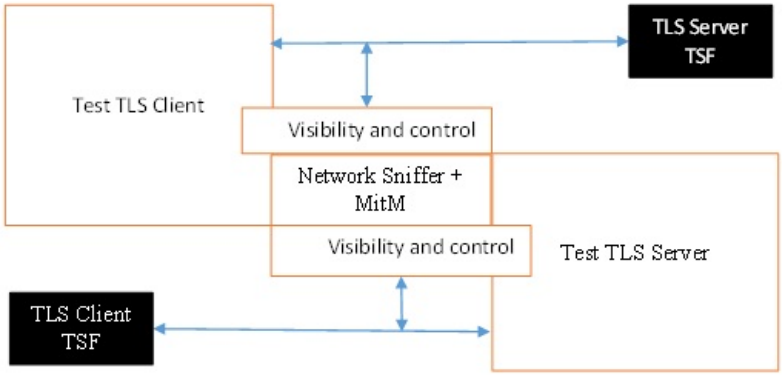

Figure 4: Combined test environment for TLS 1.2 and TLS 1.3 using both network tools and custom endpoint capabilities

Appendix A - Optional Requirements

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

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

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

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

A.1 Strictly Optional Requirements

This Package does not define any Strictly Optional requirements.

A.2 Objective Requirements

This Package does not define any Objective requirements.

A.3 Implementation-dependent Requirements

This Package does not define any Implementation-dependent requirements.

Appendix B - Selection-based Requirements

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

B.1 Auditable Events for Selection-based Requirements

The auditable events in the table below are included in a Security Target if both the associated requirement is included and the incorporating PP or PP-Module supports audit event reporting through FAU_GEN.1 and any other criteria in the incorporating PP or PP-Module are met.

Table 2: Auditable Events for Selection-based Requirements

Requirement	Auditable Events	Additional Audit Record Contents
FCS_TLSC_EXT.2	No events specified	N/A
FCS_DTLSC_EXT.1	**[selection:** *Failure of the certificate validity check, None*]	Issuer Name and Subject Name of certificate.
FCS_DTLSC_EXT.2	No events specified	N/A
FCS_DTLSS_EXT.1	**[selection:** *Failure of the certificate validity check, None*]	Issuer Name and Subject Name of certificate
FCS_DTLSS_EXT.2	No events specified	N/A

B.2 Cryptographic Support (FCS)

FCS_TLSC_EXT.2 TLS Client Support for Mutual Authentication

FCS_TLSC_EXT.2.1

The TSF shall support mutual authentication using X.509v3 certificates during the handshake and [**selection**: *in support of post-handshake authentication requests, at no other time*], in accordance with [**selection**: *RFC 5246, section 7.4.4, RFC 8446, section 4.3.2*].

Application Note: Clients that support TLS 1.3 and post-handshake authentication claim ‘in support of post-handshake authentication requests’ in the first selection. The ‘at no other time’ selection is claimed for clients only supporting TLS 1.2 or for TLS 1.3 clients that do not support post-handshake authentication.

The certificate request sent by the server specifies the signature algorithms and certification authorities supported by the server. If the client does not possess a matching certificate, it sends an empty certificate message. The structure of the certificate request message is changed in TLS 1.3 to use the signature_algorithm, signature_algorithms_cert, and certificate_authorities extensions, and RFC 8446 allows for TLS 1.2 implementations to use the new message structure. The "RFC 8446, section 4.3.2" option is claimed in the second selection if TLS 1.3 is supported or if the RFC 8446 method is supported for TLS 1.2 servers. The "RFC 5246, section 7.4.4" option is claimed if the RFC 5246 method is supported for interoperability with TLS 1.2 servers that do not adopt the RFC 8446 method. When mutual authentication is supported, at least one of these methods must be claimed, per the selection.

This SFR is claimed if "mutual authentication" is selected in FCS_TLSC_EXT.1.1.

Evaluation Activities

FCS_TLSC_EXT.2
TSS
The evaluator shall ensure that the TSS description required per FIA_X509_EXT.2.1 includes the use of client-side certificates for TLS mutual authentication. The evaluator shall also ensure that the TSS describes any factors beyond configuration that are necessary in order for the client to

engage in mutual authentication using X.509v3 certificates.

Guidance
The evaluator shall ensure that the operational guidance includes any instructions necessary to configure the TOE to perform mutual authentication. The evaluator also shall verify that the operational guidance required per FIA_X509_EXT.2.1 includes instructions for configuring the client-side certificates for TLS mutual authentication.

Tests
For each supported TLS version, the evaluator shall perform the following tests:

- ***Test 1.1:*** *The evaluator shall establish a TLS connection from the TSF to a test TLS server that negotiates the tested version and which is not configured for mutual authentication (i.e., does not send a Server's Certificate Request (type 13) message). The evaluator observes negotiation of a TLS channel and confirms that the TOE did not send a Client's Certificate message (type 11) during handshake.*
- ***Test 1.2:*** *The evaluator shall establish a connection to a test TLS server with a shared trusted root that is configured for mutual authentication (i.e., it sends a Server's Certificate Request (type 13) message). The evaluator observes negotiation of a TLS channel and confirms that the TOE responds with a non-empty Client's Certificate message (type 11) and Certificate Verify (type 15) message.*
- ***Test 1.3:*** *[conditional] If the TSF supports post-handshake authentication, the evaluator shall establish a pre-shared key between the TSF and a test TLS 1.3 server. The evaluator shall initiate a TLS session using the pre-shared key and confirm the TSF and test TLS 1.3 server successfully complete the TLS handshake and both support post-handshake authentication. After the session is successfully established, the evaluator shall initiate a certificate request message from the test TLS 1.3 server. The evaluator shall observe that the TSF receives that authentication request and shall take necessary actions, in accordance with the operational guidance, to complete the authentication request. The evaluator shall confirm that the test TLS 1.3 server receives certificate and certificate verification messages from the TSF over the channel that authenticates the client.*

  Note: *TLS 1.3 certificate requests from the test server and client certificate and certificate verify messages are encrypted. The evaluator confirms that the TSF sends the appropriate messages by examining the messages received at the test TLS 1.3 server and by inspecting any relevant server logs. The evaluator may also take advantage of the calling application to demonstrate that the TOE receives data configured at the test TLS server.*

FCS_DTLSC_EXT.1 DTLS Client Protocol

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

FCS_DTLSC_EXT.1.1

The product shall implement DTLS 1.2 (RFC 6347) and [**selection**: *DTLS 1.0 (RFC 4347), no earlier DTLS versions*] as a client that supports the ciphersuites [**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_GCM_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*

] and also supports functionality for [**selection**:

- *mutual authentication*
- *none*

].

Application Note: If any ECDHE or DHE ciphersuites are selected, then

FCS_TLSC_EXT.5 is required.

If *mutual authentication* is selected, then the ST must additionally include the requirements from FCS_DTLSC_EXT.2. If the TOE implements mutual authentication, this selection must be made.

Differences between DTLS 1.2 and TLS 1.2 are outlined in RFC 6347; otherwise the protocols are the same. All application notes listed for that are relevant to DTLS apply to this requirement.

FCS_DTLSC_EXT.1.2

The product shall verify that the presented identifier matches the reference identifier according to RFC 6125.

Application Note: All application notes listed for that are relevant to DTLS apply to this requirement.

FCS_DTLSC_EXT.1.3

The product shall not establish a trusted channel if the server certificate is invalid [**selection**: *with no exceptions*, *except when override is authorized*].

Application Note: All application notes listed for that are relevant to DTLS apply to this requirement.

FCS_DTLSC_EXT.1.4

The product shall [**selection, choose one of**: *terminate the DTLS session*, *silently discard the record*] if a message received contains an invalid MAC or if decryption fails in the case of GCM and other AEAD ciphersuites.

Evaluation Activities

FCS_DTLSC_EXT.1
Tests
The evaluator shall perform the evaluation activities listed for .

FCS_DTLSC_EXT.1.1
Tests
The evaluator shall perform the evaluation activities listed for , but ensuring that DTLS (and not TLS) is used in each evaluation activity.

For tests which involve version numbers, note that in DTLS the on-the-wire representation is the 1's complement of the corresponding textual DTLS version numbers. This is described in Section 4.1 of RFC 6347 and RFC 4347. For example, DTLS 1.0 is represented by the bytes 0xfe 0xff, while the undefined DTLS 1.4 would be represented by the bytes 0xfe 0xfb.

FCS_DTLSC_EXT.1.2
Tests
The evaluator shall perform the evaluation activities listed for .

FCS_DTLSC_EXT.1.4
TSS
The evaluator shall verify that the TSS describes the actions that take place if a message received from the DTLS Server fails the MAC integrity check.

Tests
The evaluator shall establish a connection using a server. The evaluator will then modify at least one byte in a record message, and verify that the client discards the record or terminates the DTLS session.

FCS_DTLSC_EXT.2 DTLS Client Support for Mutual Authentication

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

FCS_DTLSC_EXT.2.1

The product shall support mutual authentication using X.509v3 certificates.

Application Note: All application notes listed for FCS_TLSC_EXT.2.1 that are relevant to DTLS apply to this requirement.

Evaluation Activities

FCS_DTLSC_EXT.2
Tests
The evaluator shall perform the evaluation activities listed for FCS_TLSC_EXT.2.1.

FCS_DTLSS_EXT.1 DTLS Server Protocol

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

FCS_DTLSS_EXT.1.1

The product shall implement DTLS 1.2 (RFC 6347) and [**selection**: *DTLS 1.0 (RFC 4347), no earlier DTLS versions*] as a server that supports the ciphersuites [**selection**:

- *TLS_RSA_WITH_AES_128_CBC_SHA as defined in RFC 5246*
- *TLS_RSA_WITH_AES_256_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_128_GCM_SHA256 as defined in RFC 5288*
- *TLS_RSA_WITH_AES_256_GCM_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_128_GCM_SHA256 as defined in RFC 5288*
- *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*

] and no other ciphersuites, and also supports functionality for [**selection**:

- *mutual authentication*
- *none*

].

Application Note: If *mutual authentication* is selected, then the ST must additionally include the requirements from FCS_DTLSS_EXT.2. If the TOE implements mutual authentication, this selection must be made.

All application notes listed for that are relevant to DTLS apply to this requirement.

FCS_DTLSS_EXT.1.2

The product shall deny connections from clients requesting [**assignment**: *list of DTLS protocol versions*].

Application Note: Any specific DTLS version not selected in FCS_DTLSS_EXT.1.1 should be assigned here. This version of the FP does not require the server to deny DTLS 1.0, and if the TOE supports DTLS 1.0 then "none" can be assigned. In a future version of this FP, DTLS 1.0 will be required to be denied.

FCS_DTLSS_EXT.1.3

The product shall not proceed with a connection handshake attempt if the DTLS Client fails validation.

Application Note: The process to validate the IP address of a DTLS client is specified in section 4.2.1 of RFC 6347 (DTLS 1.2) and RFC 4347 (DTLS 1.0). The server validates the DTLS client during Connection Establishment (Handshaking) and prior to sending a Server Hello message. After receiving a ClientHello, the DTLS Server sends a HelloVerifyRequest along with a cookie. The cookie is a signed message using a keyed hash function. The DTLS Client then sends another ClientHello with the cookie attached. If the DTLS server successfully verifies the signed cookie, the Client is not using a spoofed IP address.

FCS_DTLSS_EXT.1.4

The product shall perform key establishment for DTLS using [**selection**:

- *RSA with size [**selection**: 2048 bits, 3072 bits, 4096 bits, no other sizes]*
- *Diffie-Hellman parameters with size [**selection**: 2048 bits, 3072 bits, 4096 bits, 6144 bits, 8192 bits, no other size]*
- *Diffie-Hellman groups [**selection**: ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192, no other groups]*
- *ECDHE parameters using elliptic curves [**selection**: secp256r1, secp384r1, secp521r1] and no other curves*
- *no other key establishment methods*

].

Application Note: If the ST lists an RSA ciphersuite in FCS_DTLSS_EXT.1.1, the ST must include the RSA selection in the requirement.
If the ST lists a DHE ciphersuite in FCS_DTLSS_EXT.1.1, the ST must include either the Diffie-Hellman selection for parameters of a certain size, or for particular Diffie-Hellman groups.
If the ST lists an ECDHE ciphersuite in FCS_DTLSS_EXT.1.1, the ST must include the NIST curves selection in the requirement.

FCS_DTLSS_EXT.1.5

The product shall [**selection, choose one of**: *terminate the DTLS session*, *silently discard the record*] if a message received contains an invalid MAC or if decryption fails in the case of GCM and other AEAD ciphersuites.

Evaluation Activities

FCS_DTLSS_EXT.1.1
Tests
The evaluator shall perform the evaluation activities listed for , but ensuring that DTLS (and not TLS) is used in each stage of the evaluation activities.

For tests which involve version numbers, note that in DTLS the on-the-wire representation is the 1's complement of the corresponding textual DTLS version numbers. This is described in Section 4.1 of RFC 6347 and RFC 4347. For example, DTLS 1.0 is represented by the bytes 0xfe 0xff, while the undefined DTLS 1.4 would be represented by the bytes 0xfe 0xfb.

FCS_DTLSS_EXT.1.2
The following evaluation activities shall be conducted unless "none" is assigned.
TSS
The evaluator shall verify that the TSS contains a description of the denial of old DTLS versions consistent relative to selections in FCS_DTLSS_EXT.1.2.

Guidance
The evaluator shall verify that the operational guidance includes any configuration necessary to meet this requirement.

Tests
- ***Test 2.1:*** *The evaluator shall send a Client Hello requesting a connection with each version of DTLS specified in the selection and verify that the server denies the connection.*

FCS_DTLSS_EXT.1.3
TSS
The evaluator shall verify that the TSS describes how the DTLS Client IP address is validated prior to issuing a ServerHello message.

Tests
Modify at least one byte in the cookie from the Server's HelloVerifyRequest message, and verify that the Server rejects the Client's handshake message.

FCS_DTLSS_EXT.1.4
Tests
The evaluator shall perform the evaluation activities listed for .

FCS_DTLSS_EXT.1.5
TSS
The evaluator shall verify that the TSS describes the actions that take place if a message received from the DTLS client fails the MAC integrity check.

Tests
The evaluator shall establish a connection using a client. The evaluator will then modify at least one byte in a record message, and verify that the server discards the record or terminates the DTLS session.

FCS_DTLSS_EXT.2 DTLS Server Support for Mutual Authentication

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

FCS_DTLSS_EXT.2.1

The product shall support mutual authentication of DTLS clients using X.509v3 certificates.

Application Note: All application notes listed for that are relevant to DTLS apply to this requirement.

FCS_DTLSS_EXT.2.2

The product shall not establish a trusted channel if the client certificate is invalid.

Application Note: All application notes listed for that are relevant to DTLS apply to this requirement.

FCS_DTLSS_EXT.2.3

The product shall not establish a trusted channel if the Distinguished Name (DN) or Subject Alternative Name (SAN) contained in a certificate does not match one of the expected identifiers for the client.

Application Note: All application notes listed for that are relevant to DTLS apply to this requirement.

Evaluation Activities

FCS_DTLSS_EXT.2.1
Tests
The evaluator shall perform the evaluation activities listed for .

FCS_DTLSS_EXT.2.2
Tests
The evaluator shall perform the evaluation activities listed for .

FCS_DTLSS_EXT.2.3
Tests
The evaluator shall perform the evaluation activities listed for .

Appendix C - Acronyms

Acronym	Meaning
AES	Advanced Encryption Standard
Base-PP	Base Protection Profile
CA	Certificate Authority
CBC	Cipher Block Chaining
CC	Common Criteria
CEM	Common Evaluation Methodology
CN	Common Name
cPP	Collaborative Protection Profile
DHE	Diffie-Hellman Ephemeral
DN	Distinguished Name
DNS	Domain Name Server
DTLS	Datagram Transport Layer Security
EAP	Extensible Authentication Protocol
ECDHE	Elliptic Curve Diffie-Hellman Ephemeral
ECDSA	Elliptic Curve Digital Signature Algorithm
EP	Extended Package
FP	Functional Package
GCM	Galois/Counter Mode
HTTP	Hypertext Transfer Protocol
IETF	Internet Engineering Task Force
IP	Internet Protocol
NIST	National Institute of Standards and Technology
OE	Operational Environment
PP	Protection Profile
PP-Configuration	Protection Profile Configuration
PP-Module	Protection Profile Module
RFC	Request for Comment (IETF)
RSA	Rivest Shamir Adelman
SAN	Subject Alternative Name
SAR	Security Assurance Requirement
SCSV	Signaling ciphersuite Value
SFR	Security Functional Requirement
SHA	Secure Hash Algorithm
ST	Security Target
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TOE	Target of Evaluation

TSF	TOE Security Functionality
TSFI	TSF Interface
TSS	TOE Summary Specification
UDP	User Datagram Protocol
URI	Uniform Resource Identifier
URL	Uniform Resource Locator

Appendix D - Bibliography

Identifier	Title
[CC]	Common Criteria for Information Technology Security Evaluation - • Part 1: Introduction and General Model, CCMB-2017-04-001, Version 3.1 Revision 5, April 2017. • Part 2: Security Functional Components, CCMB-2017-04-002, Version 3.1 Revision 5, April 2017. • Part 3: Security Assurance Components, CCMB-2017-04-003, Version 3.1 Revision 5, April 2017.