Workgroup: TLS
Internet-Draft:
draft-segers-tls-cert-validation-ext-00
Published: 24 August 2022
Intended Status: Standards Track
Expires: 25 February 2023
Authors: R. Segers A. Kopman
Federal Aviation Administration Concepts Beyond

Transport Layer Security (TLS) Extension: Validation Request

Abstract

This document describes the Server-based Certificate Validation Protocol (SCVP) Validation Request extension to the Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) protocols.

The Validation Request Extension provides a new protocol for TLS/DTLS allowing inclusion of SCVP certificate path validation information in the TLS/DTLS handshake.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction
  1.1. Requirements Language
2. Validation Request Extension
  2.1. Server Authentication
  2.2. Client Authentication
3. SCVP Validation Request
  3.1. Responder URIs
  3.2. Trust Anchors
  3.3. Validation Extensions
  3.4. CVRequest
    3.4.1. cvRequestVersion
    3.4.2. Query
    3.4.3. requestorRef
    3.4.4. requestNonce
    3.4.5. requestorName
    3.4.6. responderName
    3.4.7. requestExtensions
    3.4.8. signatureAlgorithm
    3.4.9. hashAlgorithm
    3.4.10. requestorText
4. SCVP Validation Response
  4.1. SCVP Response Processing by TLS Peer
  4.2. SCVP Response Processing by TLS Initiator
  4.3. SCVP Validation Response Cache
5. Error Alerts
6. IANA Considerations
  6.1. Reference for TLS Alerts and ExtensionTypes
7. Security Considerations
  7.1. Support for Extension
  7.2. Replay Attacks
  7.3. Extension Modifications
  7.4. Unrelated SCVP Validation Response
  7.5. Trust Anchor Maintenance
8. References
  8.1. Normative References
Authors' Addresses

1. Introduction

This document describes an extension to TLS 1.3 [RFC8446] and DTLS 1.3 [RFC9147] for the inclusion of Server-based Certificate Validation Protocol (SCVP) [RFC5055] certificate path validation information in the TLS/DTLS handshake.

This extension is defined for TLS and DTLS protocols. For convenience, the protocol will be referred to as TLS for the rest of the document. DTLS will only be specifically mentioned in cases where the protocols differ.

The TLS standard specifies that certificates should always be verified to ensure proper signing by a trusted Certificate Authority (CA) in Part Appendix C.2 of TLS 1.3 [RFC8446]. The establishment of trust requires construction and validation of a trust path from the end-entity certificate to a trust anchor. This validation can be a complex process of chaining certificates, validating revocation information, and enforcing organizational policies. Therefore, constrained clients may wish to delegate certificate path construction and validation to a trusted server. Additionally, to ensure that policies are consistently enforced throughout an ecosystem, centralization of certificate validation may be needed. The Server-based Certificate Validation Protocol (SCVP) allows simplification of client implementations and consistent application of validation policies by delegating validation to a server.

The extension described here can be used to signal the return of a SCVP certificate path validation corresponding to the certificate. Whenever it is sent by the client as a client hello message extension it indicates a request for SCVP validation of the server certificate. Whenever it is sent by the server as a certificate request extension it indicates a request for SCVP validation of the client certificate. If the peer supports this extension, it performs the appropriate certificate validation queries and returns the SCVP response. The response is returned as an extension to the Certificate message. Since path building and validation has been performed, only the end-entity certificate is needed for authentication, no supporting certificates need to be returned. This further reduces the bandwidth consumption. A previously cached validation response can be used, but periodic updating of the cached response will be needed as described in Section 4.3. Upon receipt of the Certificate message with the path validation extension the initiator examines the returned validation response and the response signature using a local trust anchor.

TLS clients and servers MAY use the extension described in this document. The extension is designed to be backwards compatible, meaning that TLS clients that support the extension can talk to TLS servers that do not support the extension, and vice versa.

In the future, extensions may be added to the TLS protocol which request other forms of certificate validation. Care should be taken to avoid duplicate validations for the same handshake. If an equivalent validation is preformed the peer should not also perform SCVP validation.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2. Validation Request Extension

A new extension type (validation_request (TBD)) is added to the extensions used in the client hello, certificate and certificate request handshake messages. The extension type is specified as follows.

```
enum {
     validation_request(TBD), (65535)
} ExtensionType;
```

2.1. Server Authentication

To indicate their desire to receive SCVP certificate path validation information, TLS clients MAY include an extension of type validation_request in the (extended) client hello. The extension_data field of the validation_request extension MUST contain a SCVPValidationRequest.

Servers that receive a client hello containing the validation_request extension MAY return a suitable SCVP certificate path validation response to the client along with their certificate.

Servers return a certificate path validation response along with their certificate by adding the validation_request extension to the extension block of the TLS server certificate. The Certificate message containing the TLS server end-entity certificate SHOULD contain the extension with extension_type validation_request and extension_data of SCVPValidationResponse. Severs that send the SCVPValidationResponse extension data MUST have received a validation_request in the extended client hello.

Servers that send the SCVPValidationResponse extension SHOULD only include the end-entity authentication certificate in the Certificate message. The server MAY include supporting certificates. The client MAY ignore supporting certificates if the SCVPValidationResponse is found to be satisfactory.

Clients that receive a SCVPValidationResponse extension without sending a validation_request extension in the client hello MUST abort the connection.

Servers that do not support (or are not configured to enable the use of) this extension SHOULD NOT include the validation_request extension in the Certificate message.

A server MAY also choose not to send a SCVPValidationResponse extension, even if has received a validation_request extension in the client hello message.

If the client sent a validation_request in the client hello extension but did not receive a validation_request extension in the server Certificate message MAY choose to use alternative means to validate the server certificate or MAY choose to abort the connection.

Clients requesting a certificate path validation and receiving validation_request Certificate extension MUST check the SCVPPValidationResponse message and abort the handshake if the response is not satisfactory with bad_certificate_validation_response (TBD) alert. This alert is always fatal.

2.2. Client Authentication

To indicate their desire to receive SCVP certificate path validation information, TLS servers MAY include an extension of type validation_request in the certificate request message. The extension_data field of the validation_request extension MUST contain a SCVPValidationRequest.

Clients that receive a certificate request containing the validation_request extension MAY return a suitable SCVP certificate path validation response to the server along with their certificate.

Clients return a certificate path validation response along with their certificate by adding the validation_request extension to the extension block of the TLS client certificate. The Certificate message containing the TLS client end-entity certificate SHOULD contain the extension with extension_type validation_request and extension_data of SCVPValidationResponse. Clients that send the SCVPValidationResponse extension data MUST have received a validation_request in the certificate request.

Clients that send the SCVPValidationResponse extension SHOULD only include the end-entity authentication certificate in the Certificate message. The client MAY include supporting certificates. The server MAY ignore supporting certificates if the SCVPValidationResponse is found to be satisfactory.

Servers that receive a SCVPValidationResponse extension without sending a validation_request extension in the certificate request MUST abort the connection.

Clients that do not support (or are not configured to enable the use of) this extension SHOULD NOT include the validation_request extension in the Certificate message.

A client MAY also choose not to send a SCVPValidationResponse extension, even if has received a validation_request extension in the certificate request message.

If the server sent a validation_request in the certificate request but did not receive a validation_request extension in the client Certificate message MAY choose to use alternative means to validate the client certificate or MAY choose to abort the connection.

Servers requesting a certificate path validation and receiving validation_request Certificate extension MUST check the SCVPPValidationResponse message and abort the handshake if the response is not satisfactory with bad_certificate_validation_response (TBD) alert. This alert is always fatal.

The SCVPValidationRequest and SCVPValidationResponse types are further defined in Section 3 and Section 4 of this document.

3. SCVP Validation Request

Deployment of Public Key Infrastructure (PKI) enabled applications can be simplified by delegating path validation processing to a server. Additionally, for organizations wishing to centralize administration of validation policies, delegation to a server ensures consistent policy validation across clients in an ecosystem. SCVP provides a standards-based client-server protocol for delegated path construction and validation defined in RFC 5055 [RFC5055].

Constrained clients wishing to delegate path validation also face challenges such as bandwidth and latency limitations. This extension allows for such information to be sent in the TLS handshake, saving round trips and resources.

To indicate their desire to receive certificate validation information, TLS clients MAY include an extension of type validation_request in the extended client hello and TLS servers may include an extension of type validation_request in the certificate request. Handshake messages containing the validation_request extension SHALL contain a SCVPValidationRequest in the extension_data field of this extension. The TLS client or server sending the SCVPValidationRequest and receiving the

SCVPValidationResponse is the will be referred to in this document as the TLS initiator. The TLS client or server that receives the SCVPValidationRequest and sends the SCVPValidationResponse will be referred to as the TLS peer.

The SCVPValidationRequest consists of three optional lists: a list of SCVP responder URIs, a list of trust anchors, and a list of validation extensions. SCVPValidationRequest is defined as follows:

```
struct {
     ResponderURIs responder_uri_list<0..2^16-1>;
     TrustAnchors trust_anchor_list<0..2^16-1>;
     ValidationExtensions validation_extensions_list<0..2^16-1>;
} SCVPValidationRequest;
```

3.1. Responder URIs

The ResponderURIs provides a list of SCVP responders that the initiator trusts. A zero-length responder_uri_list sequence has special meaning that the responders are implicitly known to the peer, e.g. by prior arrangement. The ResponderURIs list is in the initiator's preferred order. The peer SHOULD process the responder URI list in order and return a response from the first reachable URI with an acceptable response. What constitutes an acceptable response is discussed in Section 4.1. The ResponderURIs list is represented as a DER encoded SEQUENCE OF ASN.1 IA5String objects.

3.2. Trust Anchors

Zero or more trust anchors MAY be provided in the SCVPValidationRequest to specify the trust anchors at which the certification path must terminate if the path is to be considered valid. The TrustAnchors type is an ASN.1 SEQUENCE OF PKCReference. The SCVP Server usage is defined in Section 3.2.4.7 of RFC 5055 [RFC5055]. If a TLS peer receives a SCVPValidationRequest which contains TrustAnchors it SHOULD include the TrustAnchors in the SCVP Request Validation Policy. A non-zero length TrustAnchors sequence combined with a zero length Responder URI sequence indicates that the peer's default SCVP Responder SHOULD be used to construct a certification path which terminates at a specified certificate. If a non-zero length TrustAnchors is provided and the the validation_policy validation extension is included as defined in Section 3.3, the ValidationPolicy TrustAnchors MUST be equivalent to the SCVPValidationRequest TrustAnchors.

As defined in RFC 5055, the trust anchor PKCReference MAY be either an SCVPCertID or a Certificate. To minimize the size of the SCVPValidationRequest, TrustAnchors SHOULD be included by SCVPCertID.

3.3. Validation Extensions

This document defines nine optional ValidationExtensionTypes. These validation extensions allow the initiator to specify values in the CV Request as described in Section 3.4. Inclusion of the validation extensions will increase the size of the request and response. Therefore, the extensions should be included only when necessary.

The validation_extension_data values are DER-encoded ASN.1 types that can be directly mapped to the SCVP CVRequest as defined by RFC 5055 [RFC5055].

```
struct {
     ValidationExtensionType validation_extension_type;
     select (validation_extension_type {
          case want_back: WantBack;
          case validation_policy: ValidationPolicy;
          case cached_response: BOOLEAN;
          case query_extensions: Extensions;
          case request_nonce: OCTET STRING;
          case request_extensions: Extensions;
          case signature_algorithm: AlgorithmIdentifier;
          case hash_algorithm: OBJECT IDENTIFIER;
          case requestor_text: UTF8String (SIZE (1..256));
     } validation_extension_data;
} ValidationExtension
```

3.4. CVRequest

Peers that receive a client hello or certificate request containing the validation_request extension MAY return a suitable path validation response with their certificate. If SCVP is requested, the TLS peer SHOULD use the information contained in the extension when selecting an SCVP responder and Trust Anchor. The TLS peer SHOULD map the validation extension types to the SCVP request as follows.

```
want_back : CVRequest.query.wantBack
validation_policy : CVRequest.query.validationPolicy
cached_response : CVRequest.query.responseFlags.cachedResponse
query_extensions : CVRequest.query.queryExtensions
request_nonce : CVRequest.requestNonce
request_extensions : CVRequest.requestExtensions
signature_algorithm : CVRequest.signatureAlgorithm
hash_algorithm : CVRequest.hashAlgorithm
requestor_text : CVRequest.requestorText
```

Conforming TLS Peers MUST construct a CVRequest with a cvRequestVersion and query.

3.4.1. cvRequestVersion

TLS peers conforming to RFC 5055 [RFC5055] MUST set the value of the cvRequestVersion item to one (1).

3.4.2. Query

The query item is defined in Section 3.2 of RFC 5055 [RFC5055]

TLS peers conforming to RFC 5055 [RFC5055] SHALL include the query item. For processing the TLS Validation Request the TLS peer SHALL include the queriedCerts, certChecks and validationPolicy fields, MAY include wantBack, responseFlags, intermediateCerts, revInfos, and queryExtension fields, SHOULD NOT include producedAt and serverContextInfo fields, and MUST NOT include the validationTime field.

The TLS peer MUST populate the CertReferences item. The CertReferences sequence MUST be of length one and specify the TLS peer's X.509v3 Certificate. This certificate MUST correspond to the TLS Certificate sent in the Certificate handshake message.

The TLS peer MUST populate the CertChecks item. The item MUST be set to id-stc-build-status-checked-pkc-path (id-stc 3).

The TLS peer SHOULD NOT include want back unless specified in the validation extension want_back item. If the TLS peer receives a SCVPValidationRequest with a want_back validation extension, the TLS peer MAY set the wantBack item in the CVRequest to the value of the want_back. Want backs can significantly increase the size of the CVResponse and should be used only when specifically required.

The TLS peer MUST include the ValidationPolicy item in the CVRequest query item as follows:

If a validation_policy was specified in the SCVPValidationRequest validation extensions:

* The TLS peer SHOULD include that ValidationPolicy in the CVRequest.

If a a validation_policy was not specified in the SCVPValidaitonRequest validation extensions:

* If the ResponderURIs is a zero-length list, indicating that the TLS peer should query a pre-configured SCVP responder, the TLS peer SHOULD set the ValidationPolicy validationPolRef to either a pre-configured ValidationPolicy or the default validation policy OID id-svp-defaultValPolicy (id-svp 1).

* If the ResponderURIs is a non-zero-length list, indicating that the TLS peer should query a specified SCVP responder, the TLS peer SHOULD set the ValidationPolicy validationPolRef item to the default validation policy OID id-svp-defaultValPolicy (id-svp 1).

* If the TrustAnchors is a non-zero length list the TLS peer MUST include the provided TrustAnchors in the ValidationPolicy TrustAnchors.

The TLS peer MAY only include the ResponseFlags item in the CVRequest if requesting non-default values. If default values are used for all flags, the responseFlags item MUST NOT be included in the request. To enable the initiator to trust the CVResponse, the TLS peer MUST use the default value for the protectResponse flag (TRUE). To minimize the size of the CVResponse, the TLS peer SHOULD use the default values for the response flags fullRequestInResponse (FALSE) and responseValidationPolicyByRef (TRUE). If the TLS peer included the cached_response validation extension with a value of FALSE, the TLS peer SHOULD NOT use its cache as described in Section 4.2 and MAY include the cachedResponse set to FALSE in the CVRequest responseFlags item. If the TLS peer sets the cachedResponse flag to FALSE the request_nonce MUST be set. If the TLS peer received the request_nonce validation extension, the requestNonce in the CVRequest MAY be set to the value of the request_nonce extension. Otherwise, the TLS peer MUST generate and set a requestNonce in the CVRequest.

The TLS peer SHOULD NOT include the serverContextInfo item in the CVRequest.

The TLS peer MUST NOT include the validationTime item in the CVRequest.

The TLS peer MAY include the intermediateCerts item in the CVRequst to help the SCVP server create a valid certification path as defined in Section 3.2.8 of RFC 5055 [RFC5055].

The TLS peer MAY include the revInfos item in the CV Request which MAY be used by the SCVP Server when validating certification paths as defined in Section 3.2.9 of RFC 5055 [RFC5055].

The TLS peer SHOULD NOT include the producedAt item in the CVRequest.

The TLS peer MAY include the queryExtensions item in the CVRequest to extend the query. If the TLS peer receives a SCVPValidationRequest with a query_extensions validation extension, the TLS peer MAY set the queryExtensions item in the CVRequest to the value of the query_extension.

3.4.3. requestorRef

The TLS peer SHOULD NOT set the requestorRef item in the CVRequest.

3.4.4. requestNonce

The TLS peer MAY include the requestNonce item in the CVRequest to indicate a preference for a non-cached response. If the TLS peer receives a SCVPValidationRequest with a request_nonce validation extension, the TLS peer MAY set the requestNonce item in the CVRequest to the value of the request_nonce validation extension. If the TLS peer sets the cachedResponse response flag to FALSE but did not receive the request_nonce validation extension the TLS peer MUST generate and set a requestNonce in the CVRequest.

3.4.5. requestorName

The TLS peer SHOULD NOT include the requestorName item in the CVRequest.

3.4.6. responderName

The TLS peer SHOULD NOT include the responderName item in the CVRequest.

3.4.7. requestExtensions

The TLS peer MAY include the requestExtensions item in the CVRequest to extend the request. If the TLS peer receives a SCVPValidationRequest with a request_extensions validation extension, the TLS peer MAY set the requestExtensions item in the CVRequest to the value of the request_extension.

3.4.8. signatureAlgorithm

The TLS peer SHOULD NOT include the signatureAlgorithm item in the CVRequest unless specified in the SCVPValidationRequest signature_algorithm validation extension. If the TLS peer receives a SCVPValidationRequest with a signature_algorithm validation extension, the TLS peer MAY set the signatureAlgorithm item in the CVRequest to the value of the signature_algorithm.

To keep the size of the request and response small, it is recommended that community define the signature algorithm rather than using the signature_algorithm extension.

3.4.9. hashAlgorithm

The TLS peer SHOULD NOT include the hashAlgorithm item in the CVRequest unless specified in the SCVPValidationRequest hash_algorithm validation extension. If the TLS peer receives a SCVPValidationRequest with a hash_algorithm validation extension, the TLS peer MAY set the hashAlgorithm item in the CVRequest to the value of the hash_algorithm.

To keep the size of the request and response small, it is recommended that community define the hash algorithm rather than using the hash_algorithm extension.

3.4.10. requestorText

The TLS peer SHOULD NOT include the requestorText item in the CVRequest unless specified in the SCVPValidationRequest requestor_text validation extension. If the TLS peer receives a SCVPValidationRequest with a requestor_text validation extension, the TLS peer MAY set the requestorText item in the CVRequest to the value of the requestor_text.

To keep the size of the request and response small, it is recommended that the requestor text be used only for debugging purposes.

4. SCVP Validation Response

Servers that receive a client hello containing the validation_request extension and Clients that receive a certificate request containing the validation_request extension MAY return a suitable path validation response along with their certificate by sending a SCVPValidationResponse as an extension to the Certificate message.

TLS peers send the SCVPValidationResponse in the extension_data of the validation_request extension to the end-entity certificate in the Certificate message. The SCVPValidationResponse data conveys whether the certificate path was successfully built and validated. Therefore, in most cases, it is unnecessary for the TLS peer to include supporting certificates in the Certificates message.

The SCVPValidationResponse is defined as follows.

```
struct {
  opaque signed_cv_response
  ValidationExtensions validation_extensions_list<0..2^16-1>;
} SCVPValidationResponse;
```

A signed_cv_response contains a complete, DER-encoded CMS SignedData object as defined in RFC 5652 [RFC5652] with an EncapsulatedContent of type CVResponse as defined in RFC 5055 [RFC5055]. Only one SCVP response may be sent.

The TLS peer SHOULD include the list of validation extensions from the SCVPValidationRequest that were used in the CVRequest to indicate which validation extensions were honored. If the TLS peer did not use a validation extension in the CVRequest, it MUST NOT be included in the SCVPValidationResponse. The SCVPValidationResponse MUST NOT include validation extensions that were not present in the SCVPValidationRequest.

Section 9 of RFC 5055 [RFC5055] asserts that clients MUST verify that the response matches their original request and outlines the steps necessary to perform this verification. For this extension, the client responsibility of divided between the TLS initiator that sends the validation_reqeust extension and the TLS peer that sends the SCVP response. Certain values are only known by the TLS peer whereas other values require verification at the final end-point, the TLS initiator. The following two sections specify the verification of the SCVP response at the TLS peer Section 4.1 and at the TLS initiator Section 4.2.

4.1. SCVP Response Processing by TLS Peer

The TLS Peer that generates the CVRequest MUST verify the response from the SCVP server. If the TLS peer finds the response unacceptable, it MAY query another SCVP server (from the ResponderURIs or a pre-configured list) or MAY send a bad_path_validation_response alert notifying to close the connection.

The TLS peer MUST verify that the response is a protected response consisting of a CVResponse encapsulated in CMS SignedData.

The TLS peer SHOULD verify that the SignedData Message Digest is a hash of the received CVResponse.

The TLS peer MAY verify the certificate of the SCVP responder used for signing the response. In some environments, it may be left to the TLS initiator to validate.

As an SCVP client, the TLS peer MUST process the CVResponse as defined by RFC 5055 [RFC5055].

The TLS peer SHOULD verify the responseStatus code. If the code does not indicate okay(0), the TLS peer MAY choose to query another SCVP server from the Responder URIs.

As stated in RFC 5055, the requestRef item allows the SCVP client to determine that the request was not maliciously altered. The TLS peer creates the CVRequest and is therefore the only place where the full CVRequst is known. The TLS peer SHOULD compare the returned requestRef to the CVRequest.

If the TLS peer generated a requestNonce it SHOULD verify that the requestNonce in the response matches the value in the request.

4.2. SCVP Response Processing by TLS Initiator

On receipt of a SCVPValidationResponse, the TLS initiator MUST verify that the response indicates a successful path validation and can be trusted.

The TLS initiator MUST verify that the CVResponse is encapsulated in a CMS SignedData object and validate the digital signature on the response to ensure that the expected SCVP server generated the response. The TLS initiator MUST verify that the SignedData Message Digest matches a hash of the received CVResponse.

The CVResponse CertReply item MUST contain a single certificate matching the TLS peer's certificate sent in the TLS Certificate Handshake message. If the CertReply does not meet this requirement the TLS initiator MUST abort the connection with a bad_path_validation_response.

The TLS initiator MUST verify that the CVResponse indicates success. A CVResponse is successful if: the responseStatus is CVStatusCode okay(0) and the CertReply item containing the TLS peer's certificate has a replyStatus of success (0). If these conditions are not met the TLS initiator MUST abort the connection with a bad_path_validation_response.

The TLS initiator should check the SCVPValidationResponse validation_extensions against the validation_extensions sent in the SCVPValidationRequest. If the SCVPValidationResponse validation_extensions list does not match the list of sent validation_extensions, the TLS initiator MAY abort the connection.

If the initiator set validation extensions in the SCVPValidationRequest, the TLS initiator SHOULD verify that the CVResponse appropriately reflects those validation extensions. For example, if the request_nonce validation extension was set the initiator SHOULD verify that the CVResponse respNonce contains the same value.

4.3. SCVP Validation Response Cache

To improve performance and survive path validation service outages the TLS peer MAY cache SCVPValidationResponse messages. On receipt of a client hello or certificate request with a validation_request extension the TLS peer MAY check a local cache for a SCVPValidationResponse matching the SCVPValidationRequest's settings. If a matching response is found the peer MAY use this response rather than generating a fresh response.

The TLS peer should check if the SCVPValidationRequest has validation extensions before returning a cached response. If the cached_response validation extension is set to FALSE or the request_nonce validation extension is set. The TLS peer SHOULD NOT return a cached response.

The TLS peer SHOULD place a time limit on cached responses and generate a fresh SCVPValidationResponse after that time has elapsed. To improve performance, the TLS peer MAY proactively refresh cached responses before the cache time limit has been reached. SCVP servers may also perform caching to optimize response times. SCVP servers may optionally include the nextUpdate wantBack value in the CVResponse to indicate when a fresh response will be available from the SCVP server.

5. Error Alerts

On receipt of a SCVPValidationResponse the TLS initiator MUST validate that the response indicates a successful path validation as described in Section 4. If the SCVPValidationResponse does not indicate that the certificate was successfully validated, the TLS connection MUST be aborted with a bad_path_validation_response as follows.

```
enum {
  bad_path_validation_response(TBD),
  (255)
} AlertDescription;
```

6. IANA Considerations

IANA considerations for TLS extensions and the creation of a registry are covered in Section 11 of RFC 8446 [RFC8446].

6.1. Reference for TLS Alerts and ExtensionTypes

The following values in the TLS Alert Registry have been updated to reference this document:

   TBD bad_path_validation_response

The following ExtensionType values have been updated to reference this document:

   TBD validation_request

7. Security Considerations

General security considerations for TLS extensions are covered in TLS 1.3 RFC 8446 [RFC8446].

For security considerations specific to the Cryptographic Message Syntax message formats, see RFC 5652 [RFC5652]. For security considerations specific to the process of PKI certification path validation, see RFC 5280 [RFC5280]. For security considerations specific to SCVP, see RFC 5055 [RFC5055].

This section summarizes some of the more important security aspects specific to the TLS validation_request extension, though there are many security-relevant details in the remainder of this document.

7.1. Support for Extension

If a client or server requests a path validation response, it must consider that an attacker could (and probably would) pretend not to support the extension. In this case, the initiator of the request that requires path validation of certificates SHOULD either contact the validation server directly or abort the handshake.

7.2. Replay Attacks

Use of the optional SCVP cached response flag and request nonce items in the SCVPValidationRequest validation extensions and in the CVRequest may improve security against attacks that attempt to replay SCVP responses. However, use of these properties must be balanced with the performance impact of requiring generation of a fresh SCVP response.

7.3. Extension Modifications

Values in the client hello validation_request extension and SCVPValidationRequest are passed between the TLS client to the TLS server unprotected. This makes the values vulnerable to modification. An attacker might try to influence the handshake exchange in multiple ways including to increase latency, cause parties to abort the connection or to create trust in an untrusted server.

This extension is made further vulnerable by the inclusion of validation extensions in the SCVPValidationRequest. These validation extensions have been included to support flexibility. However, to mitigate the vulnerability to modification, domains should consider limiting use of validation extensions and instead use preconfigured domain specific values. TLS client and server verification of values returned in the signed CVResponse as described in Section 4 should also be used to protect against these attacks and detect attempts to modify these values.

This extension modification consideration does not apply in the case where the TLS Server sends a validation_request in the certificate request message which is encrypted.

7.4. Unrelated SCVP Validation Response

The received SCVPValidationResponse could contain information unrelated to the request. A path for an end-entity certificate other than the TLS certificate could be returned. The first certificate in the certificate path could not match any of the provided trust anchors. Or the SCVP responder signing the response could be unknown to the TLS client or server. If any such unrelated SCVPValidationResponse is received, it MUST be discarded. An alternative means MAY be used to validate the certificate or if no alternative means is used it SHOULD abort the handshake.

7.5. Trust Anchor Maintenance

The TLS client or server sending the validation_request extension relies on a locally known trust anchor to verify the signed SCVPValidationResponse. The trust anchor may change or expire periodically. TLS clients and servers using this specification MUST implement a secure mechanism to keep their trust anchors up to date.

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC5055] Freeman, T., Housley, R., Malpani, A., Cooper, D., and W. Polk, "Server-Based Certificate Validation Protocol (SCVP)", RFC 5055, DOI 10.17487/RFC5055, December 2007, <https://www.rfc-editor.org/info/rfc5055>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/info/rfc5280>.

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, DOI 10.17487/RFC5652, September 2009, <https://www.rfc-editor.org/info/rfc5652>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC9147] Rescorla, E., Tschofenig, H., and N. Modadugu, "The Datagram Transport Layer Security (DTLS) Protocol Version

1.3", RFC 9147, DOI 10.17487/RFC9147, April 2022,
<https://www.rfc-editor.org/info/rfc9147>.

Authors' Addresses

Robert Segers
Federal Aviation Administration
800 Independence Ave. SW
Washington, DC 20591
United States of America

Email: Robert.Segers@faa.gov

Ashley Kopman
Concepts Beyond
1155 F St NW
Washington, DC 20004
United States of America

Email: akopman@conceptsbeyond.com