Internet Draft
draft-ietf-pkix-scvp-04.txt
November 2000
Expires in six months

Ambarish Malpani
ValiCert
Paul Hoffman
VPN Consortium
Russ Housley
SPYRUS

Simple Certificate Validation Protocol (SCVP)

Status of this memo

Abstract

The SCVP protocol allows a client to offload certificate handling to a server. The server can give a variety of valuable information about the certificate, such as whether or not the certificate is valid, a chain to a trusted certificate, and so on. SCVP has many purposes, including simplifying client implementations and allowing companies to centralize their trust and policy managment.

1. Introduction

Certificate validation is a difficult problem. If certificate handling is to be widely deployed in a variety of applications and environments, the amount of processing an application needs to perform before it can accept a certificate must be reduced. There are a variety of applications that can use public key certificates but are burdened by the overhead of validating the certificates when all the application really wants is the public key and name from the certificate, and a

determination of whether or not the certificate may be used for a particular purpose. There are other applications that can perform certificate path validation but have no reliable method of obtaining a current chain to a trusted certificate.

1.1 SCVP overview and requirements

The primary goals of SCVP are to make it easier for applications to deploy systems using a PKI and to allow central administration of PKI policies. Parts of SCVP can be used by clients that do much of the PKI processing themselves and simply want a useful but untrusted server that will collect information for them. Other parts can be used by clients that have complete trust in the server to both offload the work of certificate validation and to ensure that policies are enforced in a consistent fashion across an enterprise.

Untrusted SCVP servers can give clients the certificate chains needed for path validation. They can also give clients revocation information such as CRLs and OCSP responses that the client can use in the client's path validation. These services can be valuable to client systems that do not include the protocols needed to find and download all of the intermediate certificates, CRLs, and OCSP responses needed for the client to perform complete path validation.

Trusted SCVP servers can perform full certificate validation for the client. If a client uses these services, it inherently trusts the SCVP server as much as it would its own path validation software (if it contained such software). There are two main reasons that a client may want to trust such an SCVP server:

- The client does not want to incur the overhead of including path validation software and running it for each certificate it receives.

- The client is in an enterprise that wants to centralize its PKI validation policies, such as which root certificates are trusted and which types of policy checking are performed during path validation.

1.2 Terminology

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 [MUSTSHOULD].

1.3 Open Issues

The following is a list of issues that were raised on earlier versions of this document that have not been fully dealt with here. Comments on these issues are particularly welcome.

- Extensions can be marked as critical. The usefulness and problems of criticality have been long debated and there has not been a great deal

of consensus. In SCVP, marking a request extension as critical says to the server "don't give me an answer unless you understand this extension", and marking a response extension as critical says "don't use this response unless you understand this extension". Without the critical bit in the extensions, either the semantics of extensions would have to be changed to essentially say "all extensions are critical" (which is overkill for some extensions that might really be optional), or the semantics would have to be changed to say "you can never rely on the other party understanding an extension", which would limit the usefulness of some extensions.

- Need to deal with certificate URLs, where a client doesn't have the certificate, but just a pointer to where the certificate is located. Should we even try and deal with this?

- Is there any value to an "unvalidated path"?

- Need to specify how client asks for and gets back parsed pieces of a certificate. Is this important? What fields do people want?

- Should CMS (RFC 2630) be used for signed ASN.1 messages?

- Should SCVP support validation of attribute certificates?

2. Protocol

The SCVP protocol uses a simple request-response model. That is, a SCVP client creates a single request and sends it to the server; the server creates a single response and sends it to the client. Typical use of SCVP is expected to be over HTTP, and possibly email. This document registers MIME types for SCVP requests and responses.

3. Requests

A SCVP client's request to the server MUST be a single FullRequest item. The FullRequest item contains the entire request. A FullRequest item is carried in an application/scvp-request MIME body part.

3.1 FullRequest

The FullRequest item encapsulates the client's request. The FullRequest item contains a PSRequest item, and an optional RequestSignature item.

3.2 PSRequest

The PSRequest item contains the part of the client's request. The PSRequest item contains a Version item, a Query item, a TypesOfCheck item, and a WantBack item. It can also contain an optional RequestNonce item and an optional ReqExtensions item. (The "PS" in PSRequest means "possibly signed".)

A signed request can be used to authenticate the client to the server and for non-repudiation of the client's request, such as for accounting purposes. A server might require all requests to be signed if the server did not want to respond to request unless they were from authenticated clients. A server might want to allow unsigned requests if the server is authenticating in some other fashion (such as by IP address).

In this specification, the item(s) in the Query item are certificates. The TypesOfCheck item tells the server what types of checking it should do on the item(s) in the Query item. The WantBack item tells the server what the client wants to know about the item(s). ReqExtensions in the PSRequest item are used to extend the request, such as to request a different type of item.

3.3 Version

The Version item tells the version of SCVP used in a request or a response. The value of the Version item for this specification is 1.

3.4 Query

The Query item specifies the object of the request. One type of object is defined in this specification: CertsQuery. (Other types of queries might be specified in the future.) The CertsQuery is a request for information on one or more certificates. A CertsQuery contains a list of certificates, and can also optionally contain each of the following items: ValidityTime, IntermediateCerts, TrustedCerts, RevocationInfo, PolicyID, ConfigurationIdentifier, and QueryExtensions.

The list of certificates in the Query item tells the server the certificate(s) the client wants a reply for. The optional ValidityTime item tells the time at which the client wants to know about. The optional IntermediateCerts, TrustedCerts, RevocationInfo, PolicyID, and ConfigurationIdentifier items tell the server how to process the request.

[[[Is it valuable to add a URL to a certificate in the query (for wireless applications)? If so, how should that be indicated and how does it change the fields that should be returned?]]]

3.5 CertBundle

The CertBundle item contains one or more Certs. The order of the Cert(s) in the bundle is not important.

3.6 Cert

The Cert item contains a complete certificate. The Cert item contains an identifier for the type of certificate and the octets of

the certificate itself. One type of certificate, for PKIX [PKIX], is defined, but other types of certificates (such as for OpenPGP [OpenPGP]) may be defined in the future.

3.8 QueryExtensions

The QueryExtensions item specifies a list of extensions to the SCVP protocol. For example to request additional information about the certificate(s) in the CertsQuery. The QueryExtensions item contains a sequence of Extension items, each of which contain an ExtnID item, a Critical item, and an ExtnValue item.

3.9 ExtnID

The ExtnID item is an identifier for the extension. It contains the OID of the extension.

3.10 Critical

The Critical item tells whether the extension is critical. The values for the item are:

```
False  Not critical
True   Critical
```

In a request, if the Critical item is true, the server MUST NOT process the request unless it understands the extension. In a reply, if the Critical item is true, the client MUST NOT process the response unless it understands the extension.

3.11 ExtnValue

The ExtnValue item gives the value of an extension. It contains a sequence of octets.

3.12 IntermediateCerts

The IntermediateCerts item specifies to the server the intermediate certificates that the server MAY use when forming a certificate chain. The certificates in the IntermediateCerts item can be used by the server in addition to any other certificates that the server knows of when building chains. The IntermediateCerts item contains a list of certificates. The certificates in the IntermediateCerts MUST NOT be self-signed certificates.

The purpose of the IntermediateCerts item is to help the server create validation chains.

3.13 TrustedCerts

The TrustedCerts item specifies to the server the trusted certificates

that the server MUST use. If a TrustedCerts item is included in a CertsQuery item, the server MUST NOT use any certificate chain anchors other than the certificates in the TrustedCerts item when forming a certificate chain for validation. The TrustedCerts item contains a CertBundle item.

3.14 RevocationInfo

The RevocationInfo item specifies to the server revocation information such as CRLs and OCSP [OCSP] responses that the server MAY use when validating certificate chains. The purpose of the RevocationInfo item is to provide revocation information to the server that the server may not have access to, such as an OCSP response that the client received along with the certificate. Note that the information in the RevocationInfo item might not be used by the server, such as if the information is for certificates that the server does not use in chain building.

The types of revocation proof that can be provided are:
- CRL
- OCSP response

[[[Need to specify the format of the extensions for both CRLs and for OCSP responses.]]]

3.15 PolicyID

The PolicyID item specifies to the server the policy ID that the server MUST use when forming a certificate chain. The PolicyID item contains a URL that, when resolved, defines the policy.

3.16 ConfigurationIdentifier

The ConfigurationIdentifier item tells the server the SCVP options that the client wants the server to use. The client can use this option instead of specifying other SCVP configuration such as PolicyID, TrustedCerts, RevocationInfo, and so on. The value of this item is determined by private agreement between the client and the server and is not specified in this document. The server might want to have identifiers that indicate that some settings are used in addition to others given in the request; in this way, the configuration identifier might be a shorthand for some SCVP options.

3.17 TypesOfCheck

The TypesOfCheck item describes the kind of checking that the client wants the server to do on the certificate(s) in the Query item. If the TypesOfCheck item is given in a request, it can contain one or more types of checks. For each type of check specified in the request, the server MUST perform all the checks requested, or return an error.

The types of checks are:
- Build a path to a trusted root
- Build a validated path to a trusted root
- Do revocation status checks on the path
Note that revocation status check inherently includes path construction.

3.18 WantBack

The WantBack item describes the kind of information the client wants from the server for the certificate(s) in the Query item. If the WantBack item is given in a request, it can contain one or more types of information. For each type of information specified in the request, the server MUST return information on what it found during the check (in a successful response).

The types of information that can be requested are:
- Certificate chain built for the certificate
- Proof of revocation status

For example, a request might include a TypesOfCheck item that only specifies path building, and include a WantBack item that specifies the certificate chain built. The response would not include a status for the validation, but would include a certificate chain that the server thinks might validate. This set of options might be used by a client that wants to do its own path validation.

3.19 ValidityTime

The ValidityTime indicates the time for which the client wants the information to be relevant. Not specifying a ValidityTime means that the server should use the current time. For example, when asking for validation of a certificate, the client might ask "was this certificate valid at this time". The information in the CertReply item in the response MUST be formatted as if the server created the response at the time indicated in the ValidityTime, if the server doesn't have historical information about that time, it MAY either return an error or return information for a later time. A client MUST be able to handle responses that have ThisUpdate items that are later than the requested ValidityTime.

3.20 RequestNonce

The RequestNonce item is an identifier generated by the client for the request; the server MUST return the same RequestNonce in the signed part of the server's response. The RequestNonce item is simply a sequence of octets. The client SHOULD include a RequestNonce item in every request to prevent an attacker acting as a man-in-the-middle from replaying old responses from the server. The value of the nonce SHOULD change with every request sent from the client.

3.22 RequestSignature

The RequestSignature item is the signature of the PSRequest item. The details of how a RequestSignature is computed is defined in the specific sections which describe how a request/response is represented in various formats.

4. Responses

A SCVP server's response to the client MUST be a single FullResponse item. The FullResponse item contains the entire response. A FullResponse item is carried in an application/scvp-response MIME body part.

4.1 FullResponse

The FullResponse item encapsulates the server's response. The FullResponse item contains a PSResponse item and an optional ResponseSignature item.

4.2 PSResponse

The PSResponse item contains the part of the server's response that is signed by the ResponseSignature item. The item contains a Version item, a ProducedAt item, a ResponseStatus item, and a RequestHash item. The item can also contain an optional ReplyObjects item, an optional RequestNonce item, and an optional RespExtensions item. The PSResponse item MUST contain exactly one CertReply item for each certificate requested in the request. The RequestNonce item MUST be included if the request had a RequestNonce item.

4.3 ProducedAt

The ProducedAt item tells the time at which the whole response was produced. The ProducedAt item represents the date at UTC.

4.4 ResponseStatus

The ResponseStatus item gives status information to the client about its request. The ResponseStatus item has a numeric status code and an optional string that is a sequence of characters from the ISO/IEC 10646-1 character set encoded with the UTF-8 transformation format defined in [UTF8].

The optional string may be used to transmit status information, but it is optional. The client MAY choose to display the string to the client. However, because there is no way to know the languages understood by the user, the string may be of little or no use to them.

The complete list of status codes for the ResponseStatus item is:

0 The request was fully processable
1 The request included unrecognized items; continuing

10 Too busy; try again later

20 The structure of the request was wrong
21 The version of request is not supported by this server
22 The request included unrecognized items; aborting
23 The key given in the RequestSignature is not recognized
24 The signature did not match the body of the request
25 The encoding was not understood
26 The request was not authorized
27 The request included unsupported items; continuing
28 The request included unsupported items; aborting

4.4a RequestHash

The RequestHash item is the SHA-1 hash of the PSRequest item. The RequestHash item serves the following purposes:

- It helps a client know that the request was not maliciously modified when the client gets the response back

- It allows the client to associate a response with a request when using connectionless protocols

The purpose of the RequestHash is not for authentication of the client.

The server MUST return the RequestHash item in the response.

4.5 ReplyObjects

The ReplyObjects item returns objects to the client. In this specification, the ReplyObjects item is always a CertReply, which tells the client about a single certificate from the request. The CertReply item contains a Cert item identifying the certificate, a ReplyStatus item, a ThisUpdate item, and an optional NextUpdate item. There may also be the following optional items: ValidationStatus, RevocationStatus, PublicKey, CertSubject, ValidationChain, RevocationProof, and SingleReplyExtensions.

The presence or absence of the ValidationStatus, RevocationStatus, PublicKey, CertSubject, ValidationChain, and RevocationProof items in the CertReply item is controlled by the TypesOfCheck, and WantBack items in the request. A server MUST include one of the above items for each related item requested in the TypesOfCheck, and WantBack items.

4.6 ReplyStatus

The ReplyStatus item gives status information to the client about the request for the specific certificate. Note that the ResponseStatus item is different than the ReplyStatus item. The ResponseStatus item is the status of the whole request, while the ReplyStatus item is the status for the individual certificate.

The complete list of status codes for the ReplyStatus item is:

0 Success: a definitive answer follows
1 Failure: the certificate type is not recognized
2 Failure: an item wanted in TypesOfCheck is not recognized
3 Failure: an item wanted in WantBack is not recognized
4 Failure: the certificate was malformed
5 Failure: the mandatory PolicyID is not recognized
6 Failure: the ConfigurationIdentifier is not recognized
7 Failure: unauthorized request

Status code 4 is used to tell the client that the request was properly formed but the certificate in question was not. This is useful to clients that cannot parse a certificate.

4.7 ThisUpdate

The ThisUpdate item tells the time at which the information in the CertReply was correct. The ThisUpdate item represents the date as UTC.

4.8 NextUpdate

The NextUpdate item tells the time until which the server expects the information in the CertReply to be valid. The NextUpdate item represents the date at UTC. [[[Is there a desire for another item that says "the server takes liability for this response up to this particular time?]]]

4.9 ReplyTypesOfCheck

The ReplyTypesOfCheck contains the responses to the client's TypesOfCheck item in the request. It has the same form as the Extensions item, and the OIDs in the ReplyTypesOfCheck item MUST match the OIDS in the TypesOfCheck item. The criticality bit MUST NOT be set.

The value for path building to a trusted root, {type-arc 0}, can be one of the following:

Value	Meaning
0	Built a path
1	Could not build a path

The value for path validation to a trusted root, {type-arc 1}, can be

one of the following:

Value	Meaning
0	Valid
1	Not valid

The value for the revocation status, {type-arc 2}, can be one of the following:

Value	Meaning
0	Good
1	Revoked
2	Unknown

4.10 ReplyWantBack

The ReplyWantBack contains the responses to the client's WantBack item in the request. It has the same form as the Extensions item, and the OIDs in the ReplyWantBack item MUST match the OIDS in the WantBack item. The criticality bit MUST NOT be set.

The value for the certificate chain used to verify the certificate in the request, {want-arc 0}, is a CertBundle item.

The value for the proof of revocation status, {want-arc 1}, is a RevocationProof item.

4.11 RevocationProof

The RevocationProof item gives the client the proof that the server used to check revocation. The structure of the RevocationProof item is the same as an Extensions item. The OIDs in the RevocationProof item are the same as those in the RevocationInfo item.

4.12 ResponseSignature

The ResponseSignature item is the signature of the PSResponse item.

The client SHOULD check the signature on every signed message it receives from the server. In order to check the signature, the client MUST know and rely on the public signing key of the server. The client could have obtained the server's public key through an out-of-band mechanism of direct trust or through a certificate that chains to a root that the client trusts to delegate this type of authority.

5. ASN.1 Syntax for SCVP

This section defines the syntax for SCVP messages. The semantics for

the messages are defined in sections 2, 3, and 4.

5.1 Signatures in ASN.1

Signatures in ASN.1 are done over the DER encoding of the PSRequest/PSResponse item. The Name is the distinguished name of the signer. The SignatureAlgorithm is the algorithm used to sign the request, and a SignatureBits item that is the signature itself. The signature may also contain an optional CertBundle that represents a chain of certs to verify the key used to sign the request.

5.1.1 SignatureAlgorithm

The SignatureAlgorithm identifies the algorithm used to sign a request or response. The SigningAlgorithm item contains the OID of the algorithm and any necessary parameters for the algorithm.

5.1.2 SignatureBits

The SignatureBits item holds the octets of a signature. The structure of the SignatureBits item is determined by the value of the SignatureAlgorithm item.

5.2 ASN.1 Module definition

```
SCVP DEFINITIONS EXPLICIT TAGS ::=

BEGIN

IMPORTS

      -- Directory Authentication Framework (X.509)
            Certificate, AlgorithmIdentifier
            FROM AuthenticationFramework { joint-iso-itu-t ds(5)
                     module(1) authenticationFramework(7) 3 }

      -- PKIX Imports
          Name, Extensions,
            FROM PKIX1Explicit88 {iso(1) identified-organization(3)
                 dod(6) internet(1) security(5) mechanisms(5) pkix(7)
                 id-mod(0) id-pkix1-explicit-88(1)};

FullRequest ::= SEQUENCE {
    psRequest            PSRequest,
    requestSignature     [0] Signature OPTIONAL
}

PSRequest ::= SEQUENCE {
    version                 INTEGER,
    query                   Query,
```

```
    typesOfCheck            TypesOfCheck,
    wantBack                WantBack,
    requestNonce        [1] OCTET STRING OPTIONAL,
    reqExtensions       [2] Extensions OPTIONAL
}

Query ::= CHOICE {
    certsQuery       [0] CertsQuery
}

CertsQuery ::= SEQUENCE {
    queriedCerts          SEQUENCE SIZE (1..MAX) OF Cert,
    validityTime      [0] GeneralizedTime OPTIONAL,
    intermediateCerts [1] SEQUENCE SIZE (1..MAX) OF Cert OPTIONAL,
    trustedCerts      [2] CertBundle OPTIONAL,
    revocationInfo    [3] Extensions OPTIONAL,
    policyID          [4] UTF8String OPTIONAL,
    configurationIdentifier [5] OBJECT IDENTIFIER OPTIONAL,
    queryExtensions        [6] Extensions OPTIONAL
}

CertBundle ::= SEQUENCE SIZE (1..MAX) OF Cert

Cert ::= CHOICE {
    pkixCert        [0] Certificate
}

TypesOfCheck ::= SEQUENCE SIZE (1..MAX) OF OBJECT IDENTIFIER

WantBack ::= SEQUENCE SIZE (1..MAX) OF OBJECT IDENTIFIER


Signature ::= SEQUENCE {
    signerName              Name,
    signatureAlgorithm      AlgorithmIdentifier,
    signatureBits           BIT STRING,
    certs                [0] CertBundle OPTIONAL
}

FullResponse ::= SEQUENCE {
    psResponse           PSResponse,
    responseSignature    [0] Signature OPTIONAL
}

PSResponse ::= SEQUENCE {
    version              INTEGER,
    producedAt           GeneralizedTime,
    responseStatus       ResponseStatus,
    requestHash          OCTET STRING,
    replyObjects      [0] ReplyObjects OPTIONAL,
    requestNonce      [1] OCTET STRING OPTIONAL,
```

```
    respExtensions    [2] Extensions OPTIONAL
}

ResponseStatus ::= SEQUENCE {
    statusCode          INTEGER,
    errorMessage    [0] UTF8String OPTIONAL
}

ReplyObjects ::= CHOICE {
    certReplies     [0] SEQUENCE SIZE (1..MAX) OF CertReply
}

CertReply ::= SEQUENCE {
    cert                    Cert,
    replyStatus             ReplyStatus,
    thisUpdate              GeneralizedTime,
    nextUpdate          [0] GeneralizedTime OPTIONAL,
    replyTypesOfCheck   [1] Extensions OPTIONAL,
    replyWantBack       [2] Extensions OPTIONAL,
    singleReplyExtensions       [3] Extensions OPTIONAL
}

-- The encoding of the value for path validation and revocation status
-- will be as an INTEGER


ReplyStatus ::= ENUMERATED {
    success                   (0),
    certTypeUnrecognized      (1),
    typeOfCheckUnrecognized   (2),
    wantBackUnrecognized      (3),
    certMalformed             (4),
    policyIDUnrecognized      (5),
    configInfoUnrecognized    (6),
    unauthorizedRequest       (7)
}

-- Need to include type-arc, want-arc, and revinfo-arc

END
```

6. XML Syntax for SCVP

This section defines the syntax for SCVP messages. The semantics for the messages are defined in sections 2, 3, and 4.

TODO: We need to import the XML DSig data into our DTD. We also need to provide more information about the format of the elements which map to PCDATA.

Note: this is the second attempt at XML for SCVP. We invite any comments

on it.

6.1 Signatures in XML

Signatures are done using [XMLDSIG].

6.2 Namespaces

The XML namespace [XML-ns] URI that MUST be used by implementations of this (dated) specification is:

```
   xmlns="http://www.ietf.org/pkixwg/01/scvp"
```

6.3 XML Request/Response syntax

```
<!-- Need to specify the namespace for SCVP. Also, need to include
        the DTD for XMLDSIG -->


<?xml version="1.0" encoding="UTF-8"?>

<!ELEMENT FullRequest ( PSRequest,
                 RequestSignature? )>

<!ELEMENT RequestSignature (Signature)>
<!ELEMENT PSRequest ( Version,
                        Query,
                        TypesOfCheck,
                        WantBack,
                        RequestNonce?,
                        ReqExtensions? )>
<!ATTLIST PSRequest
    Id     ID      #IMPLIED >

<!ELEMENT ReqExtensions (Extension+)>

<!ELEMENT Query ( CertsQuery ) >

<!ELEMENT CertsQuery ( QueriedCerts,
                        ValidityTime?,
                        IntermediateCerts?,
                        TrustedCerts?,
                        RevocationInfo?,
                        PolicyID?,
                        ConfigurationIdentifier?,
                        QueryExtensions? )>

<!ELEMENT QueryExtensions (Extension+)>

<!ELEMENT QueriedCerts ( Cert+ )>
```

```
<!ELEMENT ValidityTime ( GeneralizedTime )>

<!ELEMENT IntermediateCerts ( Cert+ )>

<!ELEMENT TrustedCerts ( CertBundle )>

<!ELEMENT RevocationInfo ( Extension+ )>

<!ELEMENT PolicyID ( #PCDATA )>

<!ELEMENT Version ( #PCDATA )>

<!ELEMENT ConfigurationIdentifier ( ObjID )>

<!ELEMENT CertBundle ( Cert+ )>

<!ELEMENT Cert ( X509Certificate )>

<!ELEMENT ObjID ( #PCDATA )>

<!ELEMENT GeneralizedTime ( #PCDATA )>

<!ELEMENT TypesOfCheck ( ObjID+ )>

<!ELEMENT WantBack ( ObjID+ )>

<!ELEMENT RequestNonce ( #PCDATA )>

<!ELEMENT Extensions ( Extension+ )>

<!ELEMENT Extension ( ExtnID,
                      ExtnValue )>
<!ATTLIST Extension critical ( true | false ) "false">

<!ELEMENT ExtnID ( ObjID )>


<!ELEMENT ExtnValue ( #PCDATA )>

<!ELEMENT FullResponse ( PSResponse,
                     ResponseSignature? )>

<!ELEMENT ResponseSignature (Signature)>

<!ELEMENT PSResponse ( Version,
                       ProducedAt,
                       ResponseStatus,
                       RequestHash,
                       ReplyObjects?,
                       RequestNonce?,
```

```
                          RespExtensions? )>

<!ATTLIST PSResponse
    Id     ID      #IMPLIED >

<!ELEMENT ProducedAt ( GeneralizedTime )>

<!ELEMENT ResponseStatus ( StatusCode,
                           ErrorMessage? )>

<!ELEMENT RequestHash ( #PCDATA )>

<!ELEMENT StatusCode ( #PCDATA )>

<!ELEMENT ErrorMessage ( #PCDATA )>

<!ELEMENT ReplyObjects ( CertReplies )>

<!ELEMENT RespExtensions (Extension+)>

<!ELEMENT CertReplies ( CertReply+ )>

<!ELEMENT CertReply ( Cert,
                      ReplyStatus,
                      ThisUpdate,
                      NextUpdate?,
                      ReplyTypesOfCheck?,
                      ReplyWantBack?,
                      SingleReplyExtensions? )>

<!ELEMENT ThisUpdate ( GeneralizedTime )>

<!ELEMENT NextUpdate ( GeneralizedTime )>

<!ELEMENT ReplyTypesOfCheck ( Extensions )>

<!ELEMENT ReplyWantBack ( Extensions )>

<!ELEMENT ReplyStatus EMPTY>
<!ATTLIST ReplyStatus value ( success |
                  certTypeUnrecognized |
                  typeOfCheckUnrecognized |
                  WantBackUnrecognized |
                  CertMalformed |
                  PolicyIDUnrecogized |
                  ConfigInfoUnrecognized |
                  UnauthorizedRequest)
                  "success" >


<!ELEMENT SingleReplyExtensions (Extension+)>
```

6.3 Example of XML syntax

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE FullRequest SYSTEM "scvp.dtd">

    <FullRequest xmlns="http://www.ietf.org/pkixwg/01/scvp">
        <PSRequest Id="SCVPReq">
            <Version>1</Version>
            <Query>
                <CertsQuery>
                    <QueriedCerts>
                        <Cert>
                            <X509Certificate>
                            MIICEzCCAb0CAgfYMA0GCSqGSIb3D
                            QEBBAUAMIGTMQswCQYDVQQGEwJLTz
                            . . .
                            oeedsN6iA4IhpA4Ev2rWiM92OoKag
                            UvVGaQoBuDkz7JfYNw==
                            </X509Certificate>
                        </Cert>
                    </QueriedCerts>

                    <ValidityTime>
                        <GeneralizedTime>19991232235959
                        </GeneralizedTime>
                    </ValidityTime>

                    <IntermediateCerts>
                        <Cert>
                            <X509Certificate>
                            MIICEzCCAb0CAgfYMA0GCSqGSIb3D
                            QEBBAUAMIGTMQswCQYDVQQGEwJLTz
                            . . .
                            oeedsN6iA4IhpA4Ev2rWiM92OoKag
                            UvVGaQoBuDkz7JfYNw==
                            </X509Certificate>
                        </Cert>

                    </IntermediateCerts>

                </CertsQuery>
            </Query>

            <TypesOfCheck>
                <ObjID>1.3.5.5.5.2.5.2</ObjID>
            </TypesOfCheck>
```

```
        <WantBack>
            <ObjID>1.3.5.5.5.2.5.2</ObjID>
        </WantBack>
        <RequestNonce>2888475218934</RequestNonce>
        <ReqExtensions>
            <Extension critical="true">
                <ExtnID><ObjID>1.5.4.5.9.12.1</ObjID></ExtnID>
                <ExtnValue>192812</ExtnValue>
            </Extension>
        </ReqExtensions>

    </PSRequest>

    <RequestSignature> <Signature xmlns="http://www.w3.org/2000/02/xmldsig#
        <SignedInfo Id="v5">
        <CanonicalizationMethod Algorithm=
            "http://www.w3.org/1999/07/WD-xml-c14n-19990729"/>
        <SignatureMethod Algorithm=
            "http://www.w3.org/2000/01/xmldsig/dsa"/>
        <Reference URI="#SCVPReq">
        <Transforms>
            <Transform Algorithm=
                "http://www.w3.org/2000/01/xmldsig/null">
            </Transform>
        </Transforms>
        <DigestMethod Algorithm=
            "http://www.w3.org/2000/01/xmldsig/sha1"/>
        <DigestValue>a23bcd43</DigestValue>
        </Reference>
    </SignedInfo>
    <SignatureValue>dd2323dd</SignatureValue>
    <KeyInfo Id="v10">
        <X509Data>
            <X509IssuerSerial>
                <X509IssuerName>C=US, ST=Illinois, L=Chicago, O=Aromatic
                Penguin Playing Basketball, OU=Certificate
                Authority, CN=www.ceramic.com</X509IssuerName>
                <X509SerialNumber>2007 </X509SerialNumber>
            </X509IssuerSerial>
            <X509Certificate>
            MIICITCCAcsCAgfXMA0GCSqGSIb3DQEBBAUAMIGaMQswCQYDVQQG
            EwJVUzERMA8GA1UECBMISWxsaW5vaXMxEDAOBgNVBAcTB0NoaWNh
            . . .
            bD2d2MixUSENihcgGbCEikUpNrMREO/eYkyKsiqmzAxlr3Tu/eKB
            NBeu
            </X509Certificate>
        </X509Data>
    </KeyInfo>
    </Signature> </RequestSignature>
</FullRequest>
```

TODO: Need to add an example of a response

7. Security Considerations

A client that trusts a server's responses for validation of certificates inherently trusts that server as much as it would trust its own validation software. This means that if an attacker compromises a trusted SCVP server, the attacker can change the validation processing for every client that relies on that server. Thus, an SCVP server must be protected at least as well as the weakest root server that the SCVP server trusts.

If the client does not check the signature on the response, a man-in-the-middle attack could fool the client into believing modified responses from the server, or responses to questions the client did not ask. This attack does not affect the usefulness of some responses (such as a response that returns a certificate path that the client will validate itself) but does affect things such as a validation response.

If the client does not include a RequestNonce item, or if the client does not check that the RequestNonce in the reply matches that in the request, an attacker can replay previous responses from the server.

If the server does not require some sort of authorization (such as signed requests), an attacker can get the server to reply to arbitrary requests. Such responses may give the attacker information about weaknesses in the server or about the timeliness of the server's checking. This information may be valuable for a future attack.

A. References

[MUSTSHOULD] "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119.

[OCSP] "PKIX Online Certificate Status Protocol (OCSP)", RFC 2560.

[OpenPGP] "OpenPGP Message Format", RFC 2440.

[PKIX] "PKIX Certificate and CRL Profile", RFC 2459.

[SHA-1] "Secure Hash Standard", NIST FIPS publication 180-1, April 1995.

[UTF8] "UTF-8, a transformation format of ISO 10646", RFC 2279.

[XMLDSIG] NEED THE REFERENCE

B. Acknowledgments

The lively debate in the PKIX Working Group also had a significant impact on the types of items described in this protocol. Denis Pinkas suggested some additional requirements for the protocol, and Mike Myers helped point out sections that needed clarification. Frank Balluffi and Ameya Talwalkar were responsible for the first implementation and suggestions on a few deficiencies in the document.

C. Changes Between Versions of This Document

C.1 Differences between -00 and -01

1: Rewrote to both narrow focus and to explain the goals more fully.

1.1: Removed second paragraph.

2: Removed the discussion of the two syntaxes.

3: Reorganized the section to put the Extensions items after the CertsQuery items. The section numbers below are from the -00 draft. Throughout the section, made RequestHash mandatory instead of optional. Added RevocationInfo item. Changed CertID to CertHash throughout. Fixed the names of the parts of the signature to match the text.

3.1: Split the item into a TBSRequest followed by the hash and/or signature. Changed the order of the extensions item so that all the optional items were together. Changed CertsQuery into Query. Added the ValidityTime item.

3.3: Redefined Extension to be Extensions to be more similar to Extensions in PKIX. Other wording changes.

3.5: Gave more explanation for the ExtensionCritical bit, and made the values boolean. Note that this item may disappear, depending on discussion of the open issue on it.

3.7: Changed CertsQuery into Query and described the one defined instance as CertsQuery. Moved the TypesOfCheck and WantBack from the Query and up one level to the TBSRequest.

3.9: Removed OpenPGP cert, but allowed for it to be added back in the future.

3.10: Removed OpenPGP cert hash, but allowed for it to be added back in the future.

3.11 Made TypesOfCheck OIDs.

3.12: Made WantBack OIDs. Removed the public key and the names.

3.10: Added sentence about when a client might include a CertHash item in the TrustedRoots.

3.13: Clarified use of IntermediateCerts

3.18: Added wording that the RequestHash should not be used for authentication.

3.19: Changed wording to make it clear that RequestSignature was needed only for authentication of the client.

3.23: Clarified purpose of KeyID.

4: The section numbers below are from the -00 draft. Throughout the section, made returning the RequestHash mandatory because it is now mandatory in the request.

4.1: Split the item into a TBSResponse followed by the hash and/or signature. Made ResponseSignature mandatory. Made the items returned in the form of Extensions to match the fact that TypesOfCheck and WantBack are now sequences of OIDs.

4.3: Made the status code a single number.

4.4 Removed the subject names and public keys. Added NextUpdate.

4.10: Clarified that CertSubject for PKIX certs must contain both the subject name and the subjectAltName.

4.13: Made ResponseSignature mandatory; this might be changed back to optional for some types of responses in a future revision of the spec. Added a discussion of how the client can get the server's signing key.

Old 5: Removed tiny syntax, renumbered old 6 to 5.

5: Added note about semantics in 2-4.
Split FullRequest into FullRequest and TBSRequest.
Moved the extensions item in FullRequest.
Changed the certsQuery to Query.
Move TypesOfCheck and WantBack up to TBSRequest.
Made TypesOfCheck and WantBack SEQUENCE of OIDs.
Added ValidityTime.
Changed "CertID" to "CertHash".
Made the status code a single number.
Added reminder in CertItem about full certs.
Changed order of Signature items.
Split FullResponse into FullResponse and TBSResponse.
Added ReplyTypesOfChecks and ReplyWantBack items.
Added Extensions item and sub-items.

7: Updated to reflect mandatory RequestHash and ResponseSignature. Added explicit words about compromise of the SCVP server. Removed the first paragraph because it was confusing and will be fixed in later versions of the draft.

A: Added reference to OCSP.

D: Updated.

C.2 Difference between -01 and -02

Abstract: Updated to include design goals.

Throughout: Changed TBSRequest to PSRequest. Changed UsageID to PolicyID. Changed Greenwich Mean Time to UTC.

1.2: Changed wording to match RFC 2119.

1.3: Removed first open issue (cert hashes) because we removed cert hashes. Removed third open issue (optional response signing) because the draft now clarifies which responses must be signed and which ones don't. Added new open issue (making signatures on responses optional).

3.1: Removed the RequestHash from the request.

3.2: Removed the RequestHash from the request. Added explanation of PSRequest name. Added SignerName here.

3.4: Added note about other types of queries being added in the future.

3.5: Removed CertHash.

3.7: Removed the CertHash item. Filled in the hole that would have been created with SignerName from below.

3.10: Minor edit to last line.

3.12: Removed most of the second paragraph because it was confusing.

3.14: Removed the arc stuff.

3.15: Made the PolicyID be a URL instead of an OID.

3.17: Removed the arc stuff. Also added last sentence after the list.

3.18: Removed the arc stuff.

3.19: Removed the surperfluous NextUpdate from the last sentence. Detailed what no ValidityTime request means. Changed what should happen if the client requests information for a time that the server does not have.

3.21: Changed last sentence to indicate that the RequestHash is only returned in the response, not sent in the request.

3.22: Removed the last sentence because the RequestHash is only returned in the response, not sent in the request. Moved the second paragraph up to 3.2 to make it clearer why someone might or might not sign their request. Got rid of the optional KeyID. Removed the SignerName.

3.23: Moved SignerName up in the document to 3.7. Renumbered the rest of this section.

3.26: Got rid of KeyID item.

4.2: Added SignerName here.

4.4: Got rid of 11 and 12 and made the description of 10 more sensible. Changed 25 to "encoding not understood".

4.5: Removed the last sentence because it was confusing.

4.9: Got rid of "temporarily unknown".

4.12: Made the response signature optional in the first sentence of the second paragraph. Got rid of KeyID. Removed the SignerName.

5: Removed RequestHash from FullRequest. Removed CertItem and made CertBundle a SEQUENCE OF Cert. Changed type of policyID to UTF8string to hold the URL. Got rid of KeyID. Moved signerName out of Signature and into PSRequest and TBSResponse, and made it optional.

6: Added the XML syntax and example.

7: Removed the second paragraph because it dealt with RequestHash in the request.

C.3 Difference between -02 and -03

1. Changed TBSResponse and TBSRequest to PSResponse and PSRequest. Made signatures optional in both requests and responses.

2. Added a tag to the optional signatures in both requests and responses.

3. Changed RevocationInfos to RevocationInfo.

4. Removed CertHash completely.

5. Simplified section 3.5, since FullCert has gone away

6. Replaced section 3.6 to talk about Cert, rather than FullCert

7. Replaced ExtensionParameter with ExtensionValue in Section 3.11.

8. Made sure that all SEQUENCE OF are SEQUENCE SIZE (1..MAX) OF

9. Import Extension and used the same definition for Extension as in RFC2459

10. Replace "trusted root" with "trusted certificate", because a server or client might decide to put its trust in a certificate that might not be self-signed. Replaced trustedRoot with trustedCert.

11. Fixed once occurance of definition of requestNonce

12. Removed scvp, scvpReq and scvpResp tags in the XML.

13. Removed the last 2 sentences of the second paragraph Section 3.4

14. Changed last sentence of section 3.13, since you have have multiple cert chains for a certificate even if there is no cross certification.

15. Changed last sentence of section 3.17.

16. Moved section 3.21 to the response section - 4.4a. We need to renumber all sections when we are close to being done.

17. Added a default value for the attribute value of ReplyStatus in the XML.

18. Added IMPORTS to the ASN.1 module.

19. Gave the extensions in different places different names.

20. Changed the way criticality is specified for Extension in XML

21. Added the mime type registration requests

22. Added appendix E and moved Author Information to appendix F

23. Moved signerName from the PSRequest and PSResponse to the signature part.

24. Removed the second paragraph in section 3.13.

25. Changed a line in section 3.14, first para (about where a client may have obtained an OCSP response to send to the SCVP server).

26. Got rid of the multiple places where we say what is signed by the RequestSignature or ResponseSignature (e.g. section 3.1 and 3.2). Also simplified the definition of the RequestSignature and

ResponseSignature in sections 3 and 4. The should be defined in detail in the encoding sections.

C.4 Difference between -03 and -04

1. Added format information in the http header in Appendix E.1.1

2. Changed the numbers in the want-arc to start with 0 in section 4.10

3. Added error states to indicate that the request contained unsupported items in section 4.4.

4. Added acknowledgement to Frank Balluffi and Ameya Talwalkar in Appendix B.

5. Made nextUpdate optional (renumbered tags in CertReply).

6. Specified the criticality bit in ReplyTypesOfCheck and ReplyWantBack (sections 4.9 and 4.10)

7. Specified the encoding for the replyTypesOfCheck field

8. Renumbered tag fields for PSResponse.

9. Added a TODO to section 3.4 about Cert URLs.

10. Corrected the section on the ConfigurationIdentifier.

11. Modified TypesOfCheck to allow client to request a non-validated path.

12. Removed an old (unneeded) line in the security section.

D. MIME Registrations

D.1 application/scvp-request

   To: ietf-types@iana.org
   Subject: Registration of MIME media type application/scvp-request

   MIME media type name: application

   MIME subtype name: scvp-request

   Required parameters: format

   Optional parameters: None

   Encoding considerations: binary or XML

   Security considerations: Carries a request for information. This

request may optionally be cryptographically signed.

Interoperability considerations: None

Published specification: IETF PKIX Working Group Draft on Simple Certificate Validation Protocol - SCVP

Applications which use this media type: SCVP clients

Additional information:

Magic number(s): None
File extension(s): .SCQ
Macintosh File Type Code(s): none

Person & email address to contact for further information:
Ambarish Malpani <ambarish@valicert.com>

Intended usage: COMMON

Author/Change controller:
Ambarish Malpani <ambarish@valicert.com>

D.2 application/scvp-response

To: ietf-types@iana.org
Subject: Registration of MIME media type application/scvp-response

MIME media type name: application

MIME subtype name: scvp-response

Required parameters: format

Optional parameters: None

Encoding considerations: binary or XML

Security considerations: Carries a cryptographically signed response

Interoperability considerations: None

Published specification: IETF PKIX Working Group Draft on Simple Certificate Validation Protocol - SCVP

Applications which use this media type: SCVP servers

Additional information:

Magic number(s): None
File extension(s): .SCS

Macintosh File Type Code(s): none

Person & email address to contact for further information:
Ambarish Malpani <ambarish@valicert.com>

Intended usage: COMMON

Author/Change controller:
Ambarish Malpani <ambarish@valicert.com>

E. SCVP data format

E.1 SCVP over HTTP

This section describes the formatting that will be done to the request and response to support HTTP.

E.1.1 Request

HTTP based SCVP requests can use the POST method to submit their requests. Where privacy is a requirement, SCVP transactions exchanged using HTTP MAY be protected using either TLS/SSL or some other lower layer protocol.

An SCVP request using the POST method is constructed as follows: The Content-Type header MUST have the value "application/scvp-request". In addition, the format of the message must be specified as either "format=xml" or "format=asn1". The Content-Length header MUST be present and have the exact length of the request. The body of the message is the binary value of the DER encoding of the FullRequest, or XML encoding of FullRequest. Other HTTP headers MAY be present and MAY be ignored if not understood by the requestor.

Sample Content-Type headers are:

```
Content-Type: application/scvp-request;format=xml
Content-Type: application/scvp-request;format=asn1
```

E.1.2 Response

An HTTP-based SCVP response is composed of the appropriate HTTP headers, followed by the binary value of the DER encoding of the FullResponse or XML encoding of FullResponse. The Content-Type header MUST have the value "application/scvp-response". In addition, the format of the message must be specified as either "format=xml" or "format=asn1". The Content-Length header MUST be present and specify the length of the response. Other HTTP headers MAY be present and MAY be ignored if not understood by the requestor.

F. Author Contact Information

Ambarish Malpani
ValiCert, Inc.
339 N. Bernardo Ave.
Mountain View, CA 94043
ambarish@valicert.com

Paul Hoffman
VPN Consortium
127 Segre Place
Santa Cruz, CA 95060 USA
paul.hoffman@vpnc.org

Russell Housley
SPYRUS
381 Elden Street
Suite 1120
Herndon, VA 20170 USA
housley@spyrus.com