Internet Draft   Ronald Tschalar (Trustpoint)
PKIX Working Group   Amit Kapoor (Trustpoint)
Expires in 6 months   Carlisle Adams (Entrust)

Sep. 1999

Using TCP as a Transport Protocol for CMP
<draft-ietf-pkix-cmp-tcp-00.txt>

Abstract

This document describes how to layer Certificate Management Protocols [CMP] over [TCP]. A method for doing so is described in section 5.2 of [CMP], but that method does not solve problems encountered by implementors. This document specifies an enhanced method which extends the protocol.

The key words "MUST", "MUST NOT", "REQUIRED", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document (in uppercase, as shown) are to be interpreted as described in [RFC2119].

1. Motivation

Section 5.2 of the CMP spec specifies sending the DER-encoded CMP message directly over TCP. However, implementors, during various interoperability workshops, found the protocol lacking in the following respects:

1. No clear definition on when the connection is to be closed and by whom.
2. No version number specified to allow for extensions.
3. Error messages cannot be processed by applications.

Realizing that this could not be achieved in a backward compatible way, the decision was made to enhance the protocol now to avoid interoperability conflicts later. This enhancement tries to keep as much of the older protocol as possible, while ensuring that implementations using the old protocol will not mistake a TCP-Message (defined in Section 2.1) for a valid message in the RFC-2510 format.

2. TCP-Based Management Protocol

The following simple TCP-based protocol is to be used for transport of PKI messages. This protocol is suitable for cases where an end entity (or an RA) initiates a transaction and can poll to pick up the results.

The client sends a TCP-message to the server, and the server responds with another TCP-message.

The protocol basically assumes a listener process on an RA or CA which can accept TCP-messages on a well-defined port (default port number is 829). Typically a client initiates connection to the server and submits a PKI message. The server replies with a PKI message or with a reference number to be used later when polling for the actual PKI message response.

If a polling-reference was supplied then the client will send a polling request using this polling-reference after waiting for at least the specified time. The server may again reply with a polling-reference or with the actual PKI message response.

When the final PKI response message has been picked up by the client then no new polling reference is supplied.

If a transaction is initiated by a PKI entity (RA or CA) then an end entity must either supply a listener process or be supplied with a polling reference (see below) in order to allow it to pick up the PKI message from the PKI management component.

2.1 General Form

A TCP-message consists of:

```
      length (32-bits)
      version (8-bits)
      flags (variable length)
      message-type (8-bits),
      value (defined below)
```

The length field contains the number of octets of the remainder of the TCP-message (i.e., number of octets of <value> plus <flags-length> plus 2). All bit values in this protocol are specified to be in network byte order.

The version field indicates the version of the TCP-message. It MUST be incremented for each specification which changes the flags field in a way that is not fully backwards compatible with the previous version (e.g. when the length of the flags field is changed), or which introduces a new message-type.

The flags field is for transporting TCP-message specific data. The length of this field is version dependent and is fixed for a given version.

The message-type field is used to indicate the type of TCP-message.

The value field contains message-type dependent data.

2.2 Version Negotiation

If a client knows the protocol version(s) supported by the server (e.g. from a previous TCP-message exchange or via some out-of-band means) then it SHOULD send a TCP-message with the highest version supported both by it and the server. If a client does not know what version(s) the server supports then it SHOULD send a TCP-message using the highest version it supports.

If a server receives a TCP-message version that it supports, then it MUST reply with a TCP-message of the same version. If the version received is higher than what the server supports, it MUST send back a VersionNotSupported errorMsgRep (defined below) containing the highest version it supports.

2.3 TCP-message Version 10

The TCP-message version will be 10 for this document. The number has deliberately been chosen to prevent RFC-2510 compliant applications from treating it as a valid message type. Applications receiving a version less than 10 SHOULD interpret the message as being an RFC-2510 style message.

The length of the flags field for this version is 1 octet. The LSB is used to indicate a connection close; all other bits in the flags octet MUST be ignored by servers, and MUST be set to zero by senders.

By default connections are kept open after the receipt of a response. Either party (client or server) MAY set the connection close bit at any time. If the connection close bit is set on a request, then the server SHOULD set the bit in the response and close the connection after sending the response. If the bit is set on a response from the server, the client MUST NOT send any further requests on that connection. Applications MAY decide to close an idle connection (one on which no response is outstanding) after some time-out. Because of the problem where a client sends a request and the server closes the connection while the request is still in flight, clients SHOULD automatically retry a request for which no part of the response could be read due to a connection close or reset.

If the connection is kept open, it MUST only be used for subsequent request/response transactions started by the client - the server MUST NOT use it to send requests to the client.

2.4 Detecting and Interoperating with RFC-2510 Conformant Implementations

Servers wishing to interoperate with clients conforming to RFC-2510 can do so by treating any received message with a version less than 10 as an RFC-2510 message and responding in that format. Servers not wishing to support RFC-2510 messages MUST respond with a RFC-2510 errorMsgRep.

Clients wishing to interoperate with RFC-2510 compliant servers SHOULD treat a response with a version less than 10 as an RFC-2510 style message. If this message is an errorMsgRep (message-type 06) then the client MAY automatically retry the request using the RFC-2510 format; if the message is not an errorMsgRep or the implementation does not wish to support RFC-2510 then it MUST abort the corresponding CMP transaction.

2.5 Message Types

message-types 0-127 are reserved and will be issued under IANA auspices. message-types 128-255 are reserved for application use.

The message-type's currently defined are:

Message name	Message-type
pkiReq	'00'H
pollRep	'01'H
pollReq	'02'H

finRep	'03'H
pkiRep	'05'H
errorMsgRep	'06'H

The different TCP-messages are discussed in the following sections:

2.5.1 pkiReq

The pkiReq is to be used to carry a PKIMessage from the client to the server. The <value> portion of this TCP-message will contain:

        DER-encoded PKIMessage.

The type of PKIMessages that can be carried by this TCP-message are:

        Initialization Request
        Certification Request
        PKCS-10 Request
        POP Response
        Key Update Request
        Key Recovery Request
        Revocation Request
        Cross-Certification Request
        CA Key Update Announcement
        CRL Announcement
        Certificate Announcement
        Confirmation
        Nested Message
        General Message
        Error Message

2.5.2 pkiRep

This TCP-message is to be used to send back the response to the request. The <value> portion of the finalMsgRep will contain:

        DER encoded PKI message

The type of PKIMessages that can be carried by this TCP-message are:

        Initialization Response
        Certification Response
        POP Request
        Key Update Response
        Key Recovery Response
        Revocation Response
        General Response
        Error Message

2.5.3 pollReq

The pollReq will be the used by the client to check the status of a pending PKI message. The <value> portion of the pollReq will contain:

```
polling-reference (32 bits)
```

The <polling-reference> MUST be the one returned via the pollRep TCP-message.

2.5.4 pollRep

The pollRep will be the response sent by the server to the client when there are no PKI message response ready. The <value> portion of the pollRep will contain:

```
polling-reference (32 bits)
time-to-check-back (32 bits)
```

The <polling-reference> is a unique 32-bit number sent by the server. The <time-to-check-back> is the time in seconds indicating the minimum interval after which the client SHOULD check the status again.

The duration for which the server keeps the <polling-reference> unique is left to the implementation.

2.5.5 finRep

finRep will be the response from the server indicating end of transaction, i.e., there are no further messages to be delivered from the server. The <value> portion of the finRep will contain:

```
'00'H (8 bits)
```

2.5.6 errorMsgRep

This TCP-message is sent when a TCP-message level protocol error is detected. Please note that PKIError messages MUST NOT be sent using this. Examples of TCP-message level errors are:

1. Invalid protocol version
2. Invalid TCP message-type
3. Invalid polling reference number

The <value> field of the TCP-message SHALL contain:

```
error-type (16-bits)
data-length (16-bits)
data (<data-length> octets)
UTF8 String (SHOULD include a RFC 1766 language tag)
```

The <error-type> is of the form MMNN where M and N are hex digits [0-F] and MM represents the major category and NN the minor. The major categories defined by this specification are:

'01'H	TCP-message version negotiation
'02'H	client errors
'03'H	server errors

The <data-length> and <data> are additional information about the error to be used by programs for further processing and recovery. <data-length> contains the length of the <data> field in number of octets. Error messages not needing additional information to be conveyed should set the <data-length> to 0.

The UTF8 text string is for user readable error messages.

2.5.6.1 VersionNotSupported errorMsgRep

The VersionNotSupported errorMsgRep is defined as follows:

```
error-type:                         '0101'H
data-length:                              1
data:                             <version>
UTF8-text String:   implementation defined
```

where <version> is the highest version the server supports.

2.5.6.2 GeneralClientError errorMsgRep

The GeneralClientError errorMsgRep is defined as follows:

```
error-type:                         '0200'H
data-length:                              0
data:                               <empty>
UTF8-text String:   implementation defined
```

2.5.6.3 MessageTypeUnknown errorMsgRep

The MessageTypeUnknown errorMsgRep is defined as follows:

```
error-type:                         '0201'H
data-length:                              1
data:                        <message-type>
UTF8-text String:   implementation defined
```

where <message-type> is the message-type received by the server.

2.5.6.4 InvalidPollID errorMsgRep

The InvalidPollID errorMsgRep is defined as follows:

```
    error-type:                        '0202'H
    data-length:                             4
    data:                  <polling-reference>
    UTF8-text String:   implementation defined
```

where <polling-reference> is the polling-reference received by the server.

3. CMP over TCP

The following sections describe how the above protocol is to be used to carry the PKI messages. Similar behaviour has been put in the same section.

3.1 Initialization/Certificate/Key Update/Key Recovery/Revocation

These requests MUST be sent using the pkiReq message-type. In response, the server SHOULD send back one of the following:

1. errorMsgRep
2. pollRep
3. pkiRep

Any other TCP-message is to be treated as an error.

On receiving the pollRep, the client SHOULD use the delta time specified in the pollRep to check the status, using pollReq.

The final PKI message in the transaction MUST be sent in pkiRep from the server to the client. The client MUST send the PKIConfirm in pkiReq, and on successful processing SHOULD get back a finRep indicating an end of the transaction.

3.2 Cross Certification/General Message

These requests MUST be sent using the pkiReq message-type. In response, the server SHOULD send back one of the following:

1. errorMsgRep
2. pollRep
3. pkiRep

Any other TCP-message is to be treated as an error.

On receiving the pollRep, the client SHOULD use the delta time specified in the pollRep to check the status later, using pollReq. It is RECOMMENDED, however, that the server process the PKI message immediately and return the PKI response (i.e. instead of sending a pollRep).

The server MUST send back the Cross Certification response/ General response in a pkiRep.

3.3 CA Key Update/Certificate Ann/ Revocation Ann/CRL Ann

These requests MUST be sent using the pkiReq message-type. In response, the server SHOULD send back one of the following:

1. errorMsgRep
2. pollRep
3. pkiRep

Any other TCP-message is to be treated as an error.

The server MUST send the pkiRep only if it contains a PKIError message. Successful processing of the request MUST result in a finRep.

4. Security Considerations

No new security considerations with respect to [CMP] are introduced.

5. Acknowledgments

The following individuals also deserve credit for their review and input:

Robert Moskowitz, ICSA;
Keith Brady, Baltimore;
Ron Chittaro, Entrust;
Mike Shanzer, Iris Associates;
John Wray, Iris Associates;

6. References

[CMP] Adams, C., Farrell, S., "Internet X.509 Public Key Infrastructure, Certificate Management Protocols", RFC 2510, March 1999.

[HTTP] Fielding, R.T., et. al, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, March 1997.

[RFC821] Postel, J., "Simple Mail Transfer Protocol", RFC 821, August 1982.

Authors' Addresses

Amit Kapoor
Trustpoint

429 Castro Street, Suite B
Mountain View, CA 94041
US

E-Mail: amit@trustpoint.com

Ronald Tschal r
Trustpoint
429 Castro Street, Suite B
Mountain View, CA 94041
US

E-Mail: ronald@trustpoint.com

Carlisle Adams
Entrust Technologies
750 Heron Road, Suite E08,
Ottawa, Ontario
Canada K1V 1A7

EMail: cadams@entrust.com

Full Copyright Statement