        Extensible Provisioning Protocol (EPP) Transport Over TCP

Status of this Memo

Copyright Notice

Abstract

   This document describes how an Extensible Provisioning Protocol (EPP)
   session is mapped onto a single Transmission Control Protocol (TCP)
   connection. This mapping requires use of the Transport Layer
   Security (TLS) protocol to protect information exchanged between an
   EPP client and an EPP server.

Table of Contents

1. Introduction

   This document describes how the Extensible Provisioning Protocol
   (EPP) is mapped onto a single client-server TCP connection. Security
   services beyond those defined in EPP are provided by the Transport
   Layer Security (TLS) Protocol [RFC2246]. EPP is described in
   [RFC3730]. TCP is described in [RFC793].

1.1. Conventions Used In This Document

   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 [RFC2119].

2. Session Management

   Mapping EPP session management facilities onto the TCP service is
   straight forward. An EPP session first requires creation of a TCP
   connection between two peers, one that initiates the connection
   request and one that responds to the connection request. The
   initiating peer is called the "client", and the responding peer is
   called the "server". An EPP server MUST listen for TCP connection
   requests on a standard TCP port assigned by IANA.

   The client MUST issue an active OPEN call, specifying the TCP port
   number on which the server is listening for EPP connection attempts.
   The server MUST respond with a passive OPEN call, which the client
   MUST acknowledge to establish the connection. The EPP server MUST
   return an EPP <greeting> to the client after the TCP session has been
   established.

   An EPP session is normally ended by the client issuing an EPP
   <logout> command. A server receiving an EPP <logout> command MUST
   end the EPP session and close the TCP connection through an active
   CLOSE call. The client MUST respond with a passive CLOSE call.

   A client MAY end an EPP session by issuing an active CLOSE call. A
   server SHOULD respond with a passive CLOSE call.

   A server MAY limit the life span of an established TCP connection.
   EPP sessions that are inactive for more than a server-defined period
   MAY be ended by a server issuing an active CLOSE call. A server MAY
   also close TCP connections that have been open and active for longer
   than a server-defined period.

   Peers SHOULD respond to an active CLOSE call with a passive CLOSE
   call. The closing peer MAY issue an ABORT call if the responding
   peer does not respond to the active CLOSE call.

3. Message Exchange

   With the exception of the EPP server greeting, EPP messages are
   initiated by the EPP client in the form of EPP commands. An EPP
   server MUST return an EPP response to an EPP command on the same TCP
   connection that carried the command. If the TCP connection is closed
   after a server receives and successfully processes a command but
   before the response can be returned to the client, the server MAY
   attempt to undo the effects of the command to ensure a consistent
   state between the client and the server. EPP commands are
   idempotent, so processing a command more than once produces the same
   net effect on the repository as successfully processing the command
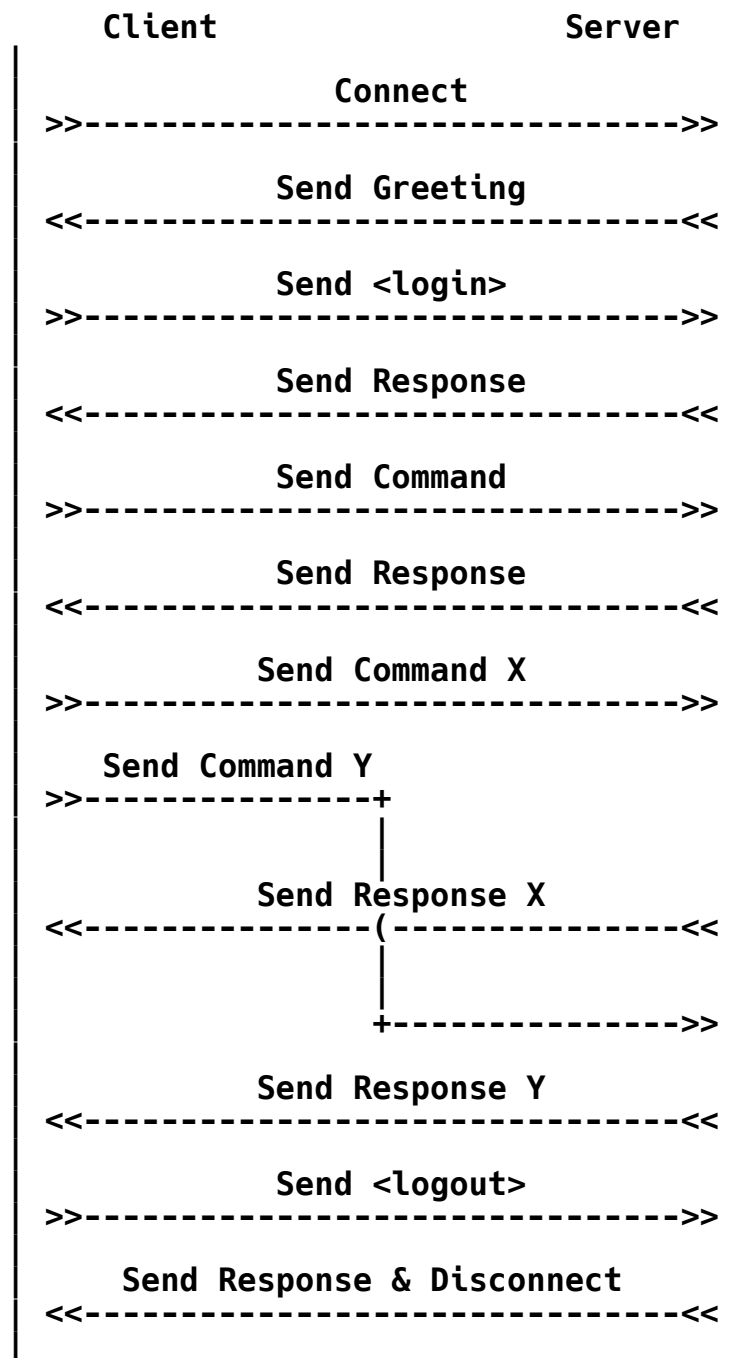   once.

   An EPP client streams EPP commands to an EPP server on an established
   TCP connection. A client MAY but SHOULD NOT establish multiple TCP
   connections to create multiple command exchange channels. A server
   SHOULD limit a client to a maximum number of TCP connections based on
   server capabilities and operational load.

   EPP describes client-server interaction as a command-response
   exchange where the client sends one command to the server and the
   server returns one response to the client. A client might be able to
   realize a slight performance gain by pipelining (sending more than
   one command before a response for the first command is received)
   commands with TCP transport, but this feature does not change the
   basic single command, single response operating mode of the core
   protocol. The amount of data that can be outstanding is limited to
   the current TCP window size.

   Each EPP data unit MUST contain a single EPP message. Commands MUST
   be processed independently and in the same order as sent from the
   client.

   A server SHOULD impose a limit on the amount of time required for a
   client to issue a well-formed EPP command. A server SHOULD end an
   EPP session and close an open TCP connection if a well-formed command
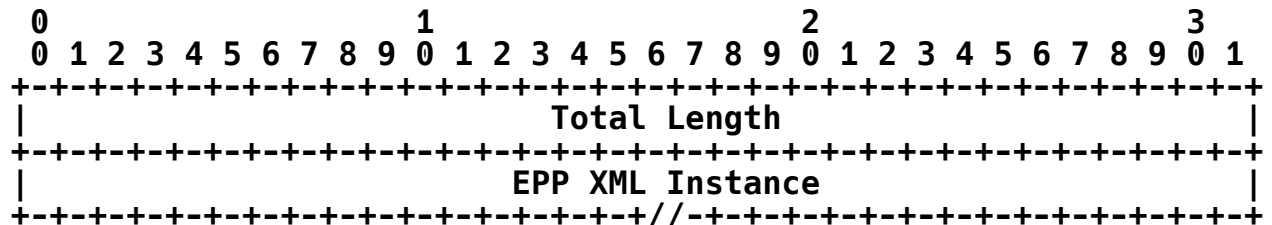   is not received within the time limit.

   A general state machine for an EPP server is described in section 2
   of [RFC3730]. General client-server message exchange using TCP
   transport is illustrated in Figure 1.

```
                    Client                    Server
                    |                              |
                         Connect
                    >>----------------------------->>
                    |                              |
                       Send Greeting
                    <<-----------------------------<<
                    |                              |
                       Send <login>
                    >>----------------------------->>
                    |                              |
                       Send Response
                    <<-----------------------------<<
                    |                              |
                       Send Command
                    >>----------------------------->>
                    |                              |
                       Send Response
                    <<-----------------------------<<
                    |                              |
                       Send Command X
                    >>----------------------------->>
                    |                              |
                  Send Command Y
                    >>--------------+
                    |               |             |
                       Send Response X
                    <<-------------(-------------<<
                    |               |             |
                    |               +-------------->>
                    |                              |
                       Send Response Y
                    <<-----------------------------<<
                    |                              |
                       Send <logout>
                    >>----------------------------->>
                    |                              |
                    Send Response & Disconnect
                    <<-----------------------------<<
                    |                              |
```

Figure 1: TCP Client-Server Message Exchange

4. Data Unit Format

   The data field of the TCP header MUST contain an EPP data unit. The
   EPP data unit contains two fields: a 32-bit header that describes the
   total length of the data unit, and the EPP XML instance.

   EPP Data Unit Format (one tick mark represents one bit position):

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                          Total Length                         |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                        EPP XML Instance                       |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+//-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

   Total Length (32 bits): The total length of the EPP data unit
   measured in octets in network (big endian) byte order. The octets
   contained in this field MUST be included in the total length
   calculation.

   EPP XML Instance (variable length): The EPP XML instance carried in
   the data unit.

5. Transport Considerations

   Section 2.1 of the EPP core protocol specification [RFC3730]
   describes considerations to be addressed by protocol transport
   mappings. This mapping addresses each of the considerations using a
   combination of features described in this document and features
   provided by TCP as follows:

   - TCP includes features to provide reliability, flow control,
      ordered delivery, and congestion control. Section 1.5 of RFC 793
      [RFC793] describes these features in detail; congestion control
      principles are described further in RFC 2581 [RFC2581] and RFC
      2914 [RFC2914]. TCP is a connection-oriented protocol, and
      Section 2 of this mapping describes how EPP sessions are mapped to
      TCP connections.

   - Sections 2 and 3 of this mapping describe how the stateful nature
      of EPP is preserved through managed sessions and controlled
      message exchanges.

   - Section 3 of this mapping notes that command pipelining is
      possible with TCP, though batch-oriented processing (combining
      multiple EPP commands in a single data unit) is not permitted.

-   Section 4 of this mapping describes features to frame data units
    by explicitly specifying the number of octets used to represent a
    data unit.

6. Internationalization Considerations

    This mapping does not introduce or present any internationalization
    or localization issues.

7. IANA Considerations

    System port number 700 has been assigned by the IANA for mapping EPP
    onto TCP.

    User port number 3121 (which was used for development and test
    purposes) has been reclaimed by the IANA.

8. Security Considerations

    EPP as-is provides only simple client authentication services using
    identifiers and plain text passwords. A passive attack is sufficient
    to recover client identifiers and passwords, allowing trivial command
    forgery. Protection against most other common attacks MUST be
    provided by other layered protocols.

    EPP provides protection against replay attacks through command
    idempotency. A replayed or repeated command will not change the
    state of any object in any way, though denial of service through
    consumption of connection resources is a possibility.

    When layered over TCP, the Transport Layer Security (TLS) Protocol
    described in [RFC2246] MUST be used to prevent eavesdropping,
    tampering, and command forgery attacks. Implementations of TLS often
    contain a US-exportable cryptographic mode that SHOULD NOT be used to
    protect EPP. Clients and servers desiring high security SHOULD
    instead use TLS with cryptographic algorithms that are less
    susceptible to compromise.

    Mutual client and server authentication using the TLS Handshake
    Protocol is REQUIRED. Signatures on the complete certificate chain
    for both client machine and server machine MUST be validated as part
    of the TLS handshake. Information included in the client and server
    certificates, such as validity periods and machine names, MUST also
    be validated. EPP service MUST NOT be granted until successful

completion of a TLS handshake and certificate validation, ensuring
that both the client machine and the server machine have been
authenticated and cryptographic protections are in place.

Authentication using the TLS Handshake Protocol confirms the identity
of the client and server machines. EPP uses an additional client
identifier and password to identify and authenticate the client's
user identity to the server, supplementing the machine authentication
provided by TLS. The identity described in the client certificate
and the identity described in the EPP client identifier can differ,
as a server can assign multiple user identities for use from any
particular client machine.

EPP TCP servers are vulnerable to common TCP denial of service
attacks including TCP SYN flooding. Servers SHOULD take steps to
minimize the impact of a denial of service attack using combinations
of easily implemented solutions, such as deployment of firewall
technology and border router filters to restrict inbound server
access to known, trusted clients.

9. Acknowledgements

This document was originally written as an individual submission
Internet-Draft. The provreg working group later adopted it as a
working group document and provided many invaluable comments and
suggested improvements. The author wishes to acknowledge the efforts
of WG chairs Edward Lewis and Jaap Akkerhuis for their process and
editorial contributions.

Specific suggestions that have been incorporated into this document
were provided by Chris Bason, Randy Bush, Patrik Faltstrom, Ned
Freed, James Gould, Dan Manley, and John Immordino.

10. References

10.1. Normative References

[RFC793]   Postel, J., "Transmission Control Protocol", STD 7, RFC
            793, September 1981.

[RFC2119]   Bradner, S., "Key Words for Use in RFCs to Indicate
            Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2246]   Dierks, T. and C. Allen, "The TLS Protocol Version 1.0",
            RFC 2246, January 1999.

[RFC2581]   Allman, M., Paxson, V. and W. Stevens, "TCP Congestion
            Control", RFC 2581, April 1999.

   [RFC2914] Floyd, S., "Congestion Control Principles", BCP 41, RFC
              2914, September 2000.

   [RFC3730] Hollenbeck, S., "Extensible Provisioning Protocol (EPP)",
              RFC 3730, March 2004.

10.2. Informative References

   None

11. Author's Address

   Scott Hollenbeck
   VeriSign Global Registry Services
   21345 Ridgetop Circle
   Dulles, VA 20166-6503
   USA

   EMail: shollenbeck@verisign.com

12. Full Copyright Statement

Intellectual Property

Acknowledgement