Internet Engineering Task Force S. Hollenbeck
Internet-Draft VeriSign, Inc.
January 24, 2002 Expires: July 24, 2002

Extensible Provisioning Protocol Transport Over TCP
<draft-ietf-provreg-epp-tcp-04.txt>

Status of this Memo

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.

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].

Table of Contents

1. Introduction 3
2. Session Management 4
3. Message Exchange 5
4. Header Format 6
5. Internationalization Considerations 6
6. IANA Considerations 7
7. Security Considerations 7
8. Acknowledgements 8
9. References 8
10. Author's Address 8
A. Revisions From Previous Version 9
B. Full Copyright Statement 10

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 [EPP]. TCP is described in [RFC793].

This document is being discussed on the "ietf-provreg" mailing list. To join the list, send a message to <majordomo@cafax.se> with the words "subscribe ietf-provreg" in the body of the message. There is a web site for the list archives at http://www.cafax.se/ietf-provreg.

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 nominally 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 establish multiple TCP connections to create multiple command exchange channels. A server MAY limit a client to a maximum number of TCP connections based on server capabilities and operational load.

An EPP command MUST be a well-formed XML instance. An EPP command begins with a RECOMMENDED XML declaration, followed by an <epp> element, EPP child elements, and ending with an </epp> element. A server MUST receive data from a client until an </epp> element is received, signaling the end of a potentially well-formed XML instance. XML parsing and command processing begins after the server has received a complete XML instance.

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.

EPP clients can initiate asynchronous command-response exchanges. In the course of a single read operation, a server might receive data that includes multiple client commands or command fragments. A server MUST scan the incoming client data, extract and execute properly formed commands as described above, and carry over any remaining data as a prefix to the data received in the next read operation.

4. Datagram Format

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

EPP Datagram 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 datagram measured in octets. The octets contained in this field MUST be included in the total length calculation.

EPP XML Instance (variable length): The EPP XML instance being carried in the datagram.

5. Internationalization Considerations

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

6. IANA Considerations

Mapping EPP onto TCP requires a TCP port assignment from IANA for public operation. TCP port 3121 (a port number in the user port range) has been assigned by IANA for development and test purposes. A system port will need to be assigned, and this user port assignment will need to be reclaimed, if this document advances to RFC status.

System Port number XXX - TBA by IANA.

7. 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. EPP service MUST NOT be granted until successful completion of a TLS handshake, ensuring that both client and server have been authenticated and cryptographic protections are in place.

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.

8. 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, James Gould, Dan Manley, and John Immordino.

9. References

Normative References:

[EPP] S. Hollenbeck: "Extensible Provisioning Protocol", work in progress.

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

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

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

Informative References:

None

10. Author's Address

Scott Hollenbeck
VeriSign Global Registry Services
21345 Ridgetop Circle
Dulles, VA 20166-6503
USA
shollenbeck@verisign.com

A. Revisions From Previous Version

(Note to RFC editor: please remove this section completely before publication as an RFC.)

-03 to -04 (WG last call updates):

Added datagram format section.

Changed some lower-case "must"s, "may"s, etc. to avoid confusion with RFC 2119 directives.

Separated references into normative and informative subsections.

B. Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.