Network Working Group
Request for Comments: 5734
STD: 69
Obsoletes: 4934
Category: Standards Track

S. Hollenbeck
VeriSign, Inc.
August 2009

Extensible Provisioning Protocol (EPP) Transport over TCP

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. This document obsoletes RFC 4934.

Status of This Memo

Copyright Notice

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 [RFC5730]. TCP is described in [RFC0793]. This document obsoletes RFC 4934 [RFC4934].

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 straightforward. 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 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 with a CLOSE call. A client MAY end an EPP session by issuing a 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 a CLOSE call. A server MAY also close TCP connections that have been open and active for longer than a server-defined period.

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 MUST NOT distribute commands from a single EPP session over multiple TCP connections. A client MAY establish multiple TCP connections to support multiple EPP sessions with each session mapped to a single connection. 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.

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 [RFC5730]. 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 EPP data unit contains two fields: a 32-bit header that describes the total length of the data unit, and the EPP XML instance. The length of the EPP XML instance is determined by subtracting four octets from the total length of the data unit. A receiver must successfully read that many octets to retrieve the complete EPP XML instance before processing the EPP message.

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 [RFC5730] describes considerations to be addressed by protocol transport mappings. This document 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 [RFC0793] 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 document describes how EPP sessions are mapped to TCP connections.

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

- Section 3 of this document 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 document describes features to frame data units by explicitly specifying the number of octets used to represent a data unit.

6. Internationalization Considerations

This document 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.

When layered over TCP, the Transport Layer Security (TLS) Protocol version 1.0 [RFC2246] or its successors (such as TLS 1.2 [RFC5246]), using the latest version supported by both parties, MUST be used to provide integrity, confidentiality, and mutual strong client-server authentication. Implementations of TLS often contain a weak 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.

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. Acceptable certificate identities MUST be

negotiated between client operators and server operators using an out-of-band mechanism. Presented certificate identities MUST match negotiated identities before EPP service is granted.

There is a risk of login credential compromise if a client does not properly identify a server before attempting to establish an EPP session. Before sending login credentials to the server, a client needs to confirm that the server certificate received in the TLS handshake is an expected certificate for the server. A client also needs to confirm that the greeting received from the server contains expected identification information. After establishing a TLS session and receiving an EPP greeting on a protected TCP connection, clients MUST compare the certificate subject and/or subjectAltName to expected server identification information and abort processing if a mismatch is detected. If certificate validation is successful, the client then needs to ensure that the information contained in the received certificate and greeting is consistent and appropriate. As described above, both checks typically require an out-of-band exchange of information between client and server to identify expected values before in-band connections are attempted.

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. TLS Usage Profile

The client should initiate a connection to the server and then send the TLS Client Hello to begin the TLS handshake. When the TLS handshake has finished, the client can then send the first EPP message.

TLS implementations are REQUIRED to support the mandatory cipher suite specified in the implemented version:

- o TLS 1.0 [RFC2246]: TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA

- o TLS 1.1 [RFC4346]: TLS_RSA_WITH_3DES_EDE_CBC_SHA

- o TLS 1.2 [RFC5246]: TLS_RSA_WITH_AES_128_CBC_SHA

This document is assumed to apply to future versions of TLS, in which case the mandatory cipher suite for the implemented version MUST be supported.

Mutual client and server authentication using the TLS Handshake Protocol is REQUIRED. Signatures on the complete certification path 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. A complete description of the issues associated with certification path validation can be found in RFC 5280 [RFC5280]. 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.

If the client has external information as to the expected identity of the server, the server name check MAY be omitted. For instance, a client may be connecting to a machine whose address and server name are dynamic, but the client knows the certificate that the server will present. In such cases, it is important to narrow the scope of acceptable certificates as much as possible in order to prevent man-in-the-middle attacks. In special cases, it might be appropriate for the client to simply ignore the server's identity, but it needs to be understood that this leaves the connection open to active attack.

During the TLS negotiation, the EPP client MUST check its understanding of the server name / IP address against the server's identity as presented in the server Certificate message in order to prevent man-in-the-middle attacks. In this section, the client's understanding of the server's identity is called the "reference identity". Checking is performed according to the following rules in the specified order:

- o If the reference identity is a server name:

  - * If a subjectAltName extension of the dNSName [CCITT.X509.1988] type is present in the server's certificate, then it SHOULD be used as the source of the server's identity. Matching is performed as described in Section 7.2 of [RFC5280], with the exception that wildcard matching (see below) is allowed for dNSName type. If the certificate contains multiple names (e.g., more than one dNSName field), then a match with any one of the fields is considered acceptable.

  - * The '*' (ASCII 42) wildcard character is allowed in subjectAltName values of type dNSName, and then only as the left-most (least significant) DNS label in that value. This wildcard matches any left-most DNS label in the server name. That is, the subject *.example.com matches the server names a.example.com and b.example.com, but does not match example.com or a.b.example.com.

  * The server's identity MAY also be verified by comparing the reference identity to the Common Name (CN) [RFC4519] value in the leaf Relative Distinguished Name (RDN) of the subjectName field of the server's certificate. This comparison is performed using the rules for comparison of DNS names in bullet 1 above (including wildcard matching). Although the use of the Common Name value is existing practice, it is deprecated, and Certification Authorities are encouraged to provide subjectAltName values instead. Note that the TLS implementation may represent DNs in certificates according to X.500 or other conventions. For example, some X.500 implementations order the RDNs in a DN using a left-to-right (most significant to least significant) convention instead of LDAP's right-to-left convention.

- If the reference identity is an IP address:

  * The iPAddress subjectAltName SHOULD be used by the client for comparison. In such a case, the reference identity MUST be converted to the "network byte order" octet string representation. For IP Version 4 (as specified in RFC 791 [RFC0791]), the octet string will contain exactly four octets. For IP Version 6 (as specified in RFC 2460 [RFC2460]), the octet string will contain exactly sixteen octets. This octet string is then compared against subjectAltName values of type iPAddress. A match occurs if the reference identity octet string and value octet strings are identical.

If the server identity check fails, user-oriented clients SHOULD either notify the user (clients MAY give the user the opportunity to continue with the EPP session in this case) or close the transport connection and indicate that the server's identity is suspect. Automated clients SHOULD return or log an error indicating that the server's identity is suspect and/or SHOULD close the transport connection. Automated clients MAY provide a configuration setting that disables this check, but MUST provide a setting which enables it.

During the TLS negotiation, the EPP server MUST verify that the client certificate matches the reference identity previously negotiated out of band, as specified in Section 8. The server should match the entire subject name or the subjectAltName as described in RFC 5280. The server MAY enforce other restrictions on the subjectAltName, for example if it knows that a particular client is always connecting from a particular hostname / IP address.

All EPP messages MUST be sent as TLS "application data". It is possible that multiple EPP messages are contained in one TLS record, or that an EPP message is transferred in multiple TLS records.

When no data is received from a connection for a long time (where the application decides what "long" means), a server MAY close the connection. The server MUST attempt to initiate an exchange of close_notify alerts with the client before closing the connection. Servers that are unprepared to receive any more data MAY close the connection after sending the close_notify alert, thus generating an incomplete close on the client side.

10. Acknowledgements

RFC 3734 is a product of the PROVREG working group, which suggested improvements and provided many invaluable comments. The author wishes to acknowledge the efforts of WG chairs Edward Lewis and Jaap Akkerhuis for their process and editorial contributions. RFC 4934 and this document are individual submissions, based on the work done in RFC 3734.

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.

11. References

11.1. Normative References

[CCITT.X509.1988]
International Telephone and Telegraph Consultative Committee, "Information Technology - Open Systems Interconnection - The Directory: Authentication Framework", CCITT Recommendation X.509, November 1988.

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.

[RFC0793] 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.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[RFC4519] Sciberras, A., "Lightweight Directory Access Protocol (LDAP): Schema for User Applications", RFC 4519, June 2006.

[RFC5730] Hollenbeck, S., "Extensible Provisioning Protocol (EPP)", STD 69, RFC 5730, August 2009.

11.2. Informative References

[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.

[RFC4346] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.1", RFC 4346, April 2006.

[RFC4934] Hollenbeck, S., "Extensible Provisioning Protocol (EPP) Transport Over TCP", RFC 4934, May 2007.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[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, May 2008.

Appendix A. Changes from RFC 4934

1. Changed "This document obsoletes RFC 3734" to "This document obsoletes RFC 4934".

2. Replaced references to RFC 3280 with references to 5280.

3. Replaced references to RFC 3734 with references to 4934.

4. Updated references to RFC 4346 and TLS 1.1 with references to 5246 and TLS 1.2.

5. Replaced references to RFC 4930 with references to 5730.

6. Added clarifying TLS Usage Profile section and included references.

7. Moved the paragraph that begins with "Mutual client and server authentication" from the Security Considerations section to the TLS Usage Profile section.

Author’s Address

Scott Hollenbeck
VeriSign, Inc.
21345 Ridgetop Circle
Dulles, VA 20166-6503
US

EMail: shollenbeck@verisign.com