Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: November 15, 2011

A. Langley
Google Inc
P. Hoffman
VPNC

May 14, 2011

Transport Layer Security (TLS) Next Protocol Extension
draft-agl-tls-nextproto-00

Abstract

This document describes a Transport Layer Security (TLS) extension for application layer protocol probing and announcement. This allows the application client to specify which protocol will be performed over the secure connection.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on November 15, 2011.

Copyright Notice

Table of Contents

1. Introduction

As the Internet has evolved, it has become commonplace for hosts to initiate connections based on untrusted and possibly hostile data. HTTP [RFC2616] clients are currently the most widespread example of this as they will fetch URLs based on the contents of untrusted webpages.

Any time that a connection is initiated based on untrusted data there is the possibility of a cross-protocol attack. If the attacker can control the contents of the connection in any way (for example, the requested URL in an HTTP connection) they may be able to encode a valid message in another protocol. The connecting host believes that it is speaking one protocol but the server understands it to be another. The application of Postel's Law exacerbates the issue as many servers will permit gross violations of the expected protocol in order to achieve maximum compatibility with clients.

The WebSockets [websockets] protocol seeks to allow low-latency, full-duplex communication between browsers and HTTP servers. However, it also permits an unprecedented amount of attacker control over the contents of the connection. In order to prevent cross-protocol attacks, a mechanism to assure that both client and server are speaking the same protocol is required. To this end, the Next Protocol extension described in this document extends the TLS [RFC5246] handshake to allow the client to tell the server the intended application protocol.

1.1. Design of the Protocol

The basic design of the extension is that the client expresses that it knows how to specify which application protocol it will use after the TLS session is established. The server responds with some or all the types of application protocols that it knows. The client responds with the protocol that it intends to use; this might even be one that was not listed by the server.

Note that this protocol is not a negotiation in the classic sense of "client says what it wants and server picks one choice". Instead, it allows the client to not reveal in the clear which application protocol it intends to use after TLS is established. Intermediaries who might prevent TLS from being established for a particular application cannot determine which protocol will be used; this, in turn, leads to secure connections for more protocols.

2. Requirements Notation

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 RFC 2119 [RFC2119].

3. Next Protocol Extension

This document defines a new extension type, "next_protocol(TBD)".

```
enum {
  next_protocol(TBD), (65535)
} ExtensionType;
```

The next_protocol extension MAY be included by the client in its "ClientHello" message. If, and only if, the server sees this extension in the "ClientHello", it MAY choose to include the extension in its "ServerHello".

The "extension_data" field of a "next_protocol" in a "ClientHello" MUST be empty. The "extension_data" field of a "next_protocol" in a "ServerHello" contains the list of application-layer protocols that the server wishes to advertise that it supports. That list is a set of two-octet (uint16) values, in network byte order, taken from the ports numbers assigned by IANA; see <http://www.iana.org/assignments/port-numbers>.

This document also defines a new handshake message type, "next_protocol_ann(TBD)").

```
struct {
  uint16 announced_protocol;
} NextProtocolAnnounce;
```

If, and only if, the server included a "next_protocol" extension in its ServerHello message, the client MUST send a "NextProtocolAnnounce" message after its "ChangeCipherSpec" and before its "Finished" message. The NextProtocolAnnounce message contains the single application-layer protocol that the client will use in this connection after the TLS handshake completes; that value does not need to match any of those given by the server in the

next_protocol extension.

3.1. Handshake Summary

```
   -> ClientHello (contains next_protocol extension
          with empty extension_data )
   <- ServerHello (contains next_protocol extension
          with list of protocols)
   <- ...
   <- ServerHelloDone
   -> ClientKeyExchange
   -> ...
   -> ChangeCipherSpec
   -> NextProtocolAnnounce (contains announced_protocol)
   -> Finished
   <- ChangeCipherSpec
   <- Finished
```

3.2. Session Resumption and Renegotiation

Unlike many other TLS extensions, this extension does not establish properties of the session, only of the connection. When session resumption or session tickets [RFC5077] are used, the previous contents of this extension are irrelevant and only the values in the new handshake messages are considered.

For the same reasons, after a handshake has been performed for a given connection, renegotiations on the same connection MUST NOT include the "next_protocol" extension.

4. Security Considerations

This extension sends the server's list of supported protocols in the clear. This may be undesirable for certain protocols (such as Tor [tor]) where one could imagine that hostile networks would terminate any TLS connection with a server that advertised such a capability. Thus, if a client knows through out-of-band methods that a server supports a particular protocol, it can specify that protocol in the NextProtocolAnnounce message and use that protocol after TLS is set up.

5. IANA Considerations

This document requires IANA to update its registry of TLS extensions to assign an entry, referred herein as "next_protocol".

This document also requires IANA to update its registry of TLS handshake types to assign an entry, referred herein as "next_protocol_ann".

6. Acknowledgements

This document benefitted specifically from discussions with Wan-Teh Chang and Nagendra Modadugu.

7. References

7.1. Normative References

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

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[RFC5077] Salowey, J., Zhou, H., Eronen, P., and H. Tschofenig, "Transport Layer Security (TLS) Session Resumption without Server-Side State", RFC 5077, January 2008.

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

7.2. Informative References

[websockets] Fette, I., "The Web Socket protocol", draft-ietf-hybi-thewebsocketprotocol (work in progress), 2011.

[tor] "Tor Onion Router", www.tor.org , 2011.

Authors' Addresses

Adam Langley
Google Inc

Email: agl@google.com

Paul Hoffman
VPNC

Email: paul.hoffman@vpnc.org