Network Working Group   S. Friedl
Internet-Draft   Cisco Systems, Inc.
Intended status: Standards Track   October 16, 2012
Expires: April 19, 2013

Transport Layer Security (TLS) Application Layer Protocol Negotiation Extension
draft-friedl-tls-applayerprotoneg-00

Abstract

This document describes a Transport Layer Security (TLS) extension for application layer protocol negotiation within the TLS handshake. For instances in which the TLS connection is established over a well known TCP/IP port not associated with the desired application layer protocol, this extension allows the application layer to negotiate which protocol will be used within the TLS session.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

Currently, the Next Protocol Negotiation extension (NPN) is used to establish a SPDY [spdy] protocol session within a TLS RFC 5246 [RFC5246] session on port 443. NPN is not specific to SPDY and can be used to negotiate sessions for a wide variety of protocols within the TLS handshake.

NPN seeks to provide a reliable mechanism for application developers to establish secure sessions for arbitrary protocols without interference from firewalls, HTTP proxies and MITM proxies. It addresses this goal by introducing a protocol negotiation process into the TLS handshake under the constraints that no additional roundtrips be added to the handshake and that the final protocol selection be opaque to the network carrying the TLS session. To do this, NPN introduces a non-symmetric and slightly idiosyncratic extension to the TLS handshake. Within the NPN extension, it is the server that first generates and transmits an offer of supported protocols to the client. The offer is sent as part of the TLS ServerHello message before the [ChangeCipherSpec] subprotocol has been started, therefore the list of protocols supported by the server is transmitted in plaintext. The client chooses a protocol which may or may not appear in the offer from the server and then responds with the definitive protocol selection answer. The client response is sent after the [ChangeCipherSpec] subprotocol has been initiated, so the protocol selected is encrypted in the client response.

In many other application layer protocol negotiation processes, it is the client that first sends an offer of protocols it supports to the server. The server then selects the protocol to be used in the session and includes this answer in the response. RFC 3264 [RFC3264] describes a SDP based offer/answer model which is not proscriptive in terms of which party generates the offer, however in practice it is typically the client generating the offer and the server replying with the answer. This permits the server to act as the definitive entity for selection of the application layer protocol.

History demonstrates that there exist a multiplicity of compelling needs for TCP/IP networks to provide differentiated network treatment based on protocol. This differentiated treatment may include QOS and/or firewalling to permit enterprises and carriers to manage the flows within their networks. QOS requirements may be driven by the needs of real-time protocols or service provider SLAs or service tiers. Firewalling is required to meet a variety of regulatory, compliance, appropriate use and security mandates.

This draft proposes an alternative formulation of the NPN protocol which 1) brings the offer/answer negotiation into alignment with the

majority of other application layer protocol negotiation standards, 2) improves the symmetry of the offer/answer exchange and 3) makes the definitive protocol selection answer from the server visible to the network.

1.1. Requirements Language

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].

1.2. Application Layer Protocol Negotiation Extension

A new extension type ("application_layer_protocol_negotiation(TBD)") is defined and MAY be included by the client in its "ClientHello" message.

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

The "extension_data" field of the ("application_layer_protocol_negotiation(TBD)") extension SHALL contain "ProtocolIdentifierList" where:

```
struct {
    IdentifierType id_type;
    select (id_type) {
        case IANA_application_id: ApplicationID;
    } id;
} ProtocolIdentifier;

enum {
    IANA_application_id(0), (255)
} IdentifierType;

opaque ApplicationID<2^32-1>;

struct {
    ProtocolIdentifier protocol_id_list<1..2^16-1>
} ProtocolIdentifierList;
```

"ApplicationID" contains the base 10 IANA registered application number associated with the requested protocol. This new IANA controlled value is 32 bits in length and each application layer protocol negotiated within the ALPN handshake must have an entry in this IANA registry. "ApplicationID" will be serialized as a fixed

length 10 byte string using UTF-8 encoding [UTF8]. Leading zeros SHALL be used to pad the identifiers to 10 digits. Spaces SHALL be used to separate port numbers in a "ProtocolIdentifierList".

Servers that receive a client hello containing the "application_layer_protocol_negotiation" extension, MAY return a suitable protocol selection response to the client. The server will ignore any "ProtocolIdentifier" that it does not recognize. A new ServerHello extension type ("connection_protocol(TBD)") MAY be returned to the client within the extended ServerHello message. The "extension_data" field of the ("connection_protocol(TBD)") extension SHALL contain a single protocol identifier serialized as an "ApplicationID" as described for the client "extension_data" protocol list.

The additional content associated with this extension MUST be included in the hash calculations associated with the "Finished" messages.

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

struct {
    IdentifierType id_type;
    select (id_type) {
        case IANA_application_id: ApplicationID;
    } id;
} ProtocolIdentifier;

opaque ApplicationID<2^32-1>;
```

Therefore, a full handshake with the "application_layer_protocol_negotiation" and "connection_protocol" extensions has the following flow (contrast with section 7.3 of RFC 5246 [RFC5246]):

```
Client                                               Server

ClientHello                     -------->       ServerHello
  (ALPN extension &                               (connection protocol
   list of protocols)                              extension &
                                                   selected protocol)
                                                Certificate*
                                                ServerKeyExchange*
                                                CertificateRequest*
                                <--------       ServerHelloDone
Certificate*
ClientKeyExchange
CertificateVerify*
[ChangeCipherSpec]
Finished                        -------->
                                                [ChangeCipherSpec]
                                <--------       Finished
Application Data                <------->       Application Data
```

Figure 1

An abbreviated handshake with the "application_layer_protocol_negotiation" and "connection_protocol" extensions the following flow:

```
Client                                               Server

ClientHello                     -------->       ServerHello
  (ALPN extension &                               (connection protocol
   list of protocols)                              extension &
                                                   selected protocol)
                                                [ChangeCipherSpec]
                                <--------       Finished
[ChangeCipherSpec]
Finished                        -------->
Application Data                <------->       Application Data
```

Figure 2

Unlike many other TLS extension, this extension does not establish properties of the session, only of the connection. When session resumption or session tickets RFC 5077 [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 "application_layer_protocol_negotiation" or

"connection_protocol" extensions.

1.3. Protocol Selection

It is expected that a server will have a list of protocols that it supports, in preference order, and will only select a protocol if the client supports it. In that case, the server SHOULD select the most highly preferred protocol it supports which is also advertised by the client. In the event that the server supports no protocols that the client advertises, then the server SHOULD select the first protocol in its ordered list.

The protocol identified in the "connection_protocol" extension type in the ServerHello SHALL be definitive for the connection. The server SHALL NOT respond with a selected protocol and subsequently use a different protocol for application data exchange.

1.4. Design Considerations

The ALPN extension is intended to follow the typical design of TLS protocol extensions. Specifically, the negotiation is performed entirely within the hello messages and the ClientHello and ServerHello extensions conform to the same general pattern used by other TLS extensions. The "connection_protocol" extension is intended to be definitive for the connection and is sent in plaintext to permit network elements to provide differentiated service for the connection when the TCP/IP port number is not definitive for the application layer protocol to be used in the connection.

1.5. Security Considerations

The ALPN extension does not impact the security of TLS session establishment or application data exchange. ALPN serves to provide an externally visible marker for the application layer protocol associated with the TLS connection. Historically, the application layer protocol associated with a connection could be ascertained from the TCP/IP port number in use.

1.6. IANA Considerations

This document requires the IANA to update its registry of TLS extensions to assign two entries referred to here as "application_layer_protocol_negotiation" for extended ClientHello messages and "connection_protocol" for extended ServerHello messages.

This document requires the IANA to create a new registry of application identifiers to serve as protocol identifiers for ALPN. It is suggested this identifier be 32 bit numeric value with the

bottom 16 bits associated with current IANA port numbers when the upper 16 bits are all set to zero.

2. Acknowledgements

This document benefitted specifically from the NPN extension draft authored by Adam Langley of Google and from discussions with Tom Wesselman and Cullen Jennings both of Cisco.

3. References

3.1. Normative References

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

[RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.

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

[RFC6066] Eastlake, D., "Transport Layer Security (TLS) Extensions: Extension Definitions", RFC 6066, January 2011.

3.2. Informative References

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

[spdy] Belshe, M. and R. Peon, "SPDY Protocol (Internet Draft)", 2012.

Author's Address

Stephan Friedl
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134
USA

Phone: (720)562-6785
Email: sfriedl@cisco.com