Network Working Group   S. Friedl
Internet-Draft   Cisco Systems, Inc.
Intended status: Standards Track   A. Popov
Expires: April 18, 2014   Microsoft Corp.
                                                          A. Langley
                                                         Google Inc.
                                                          E. Stephan
                                                              Orange
                                                    October 15, 2013

       Transport Layer Security (TLS) Application Layer Protocol Negotiation
                                Extension
                   draft-ietf-tls-applayerprotoneg-03

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

   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 April 18, 2014.

Table of Contents

1. Introduction

Increasingly, application layer protocols are encapsulated in the TLS
security protocol [RFC5246]. This encapsulation enables applications
to use the existing, secure communications links already present on
port 443 across virtually the entire global IP infrastructure.

When multiple application protocols are supported on a single server-
side port number, such as port 443, the client and the server need to
negotiate an application protocol for use with each connection. It
is desirable to accomplish this negotiation without adding network
round-trips between the client and the server, as each round-trip
will degrade an end-user's experience. Further, it would be
advantageous to allow certificate selection based on the negotiated
application protocol.

This document specifies a TLS extension which permits the application
layer to negotiate protocol selection within the TLS handshake. This
work was requested by the HTTPbis WG to address the negotiation of
HTTP version ([RFC2616], [I-D.ietf-httpbis-http2]) over TLS, however
ALPN facilitates negotiation of arbitrary application layer
protocols.

   With ALPN, the client sends the list of supported application
   protocols as part of the TLS ClientHello message. The server chooses
   a protocol and sends the selected protocol as part of the TLS
   ServerHello message. The application protocol negotiation can thus
   be accomplished within the TLS handshake, without adding network
   round-trips, and allows the server to associate a different
   certificate with each application protocol, if desired.

2. 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 [RFC2119].

3. Application Layer Protocol Negotiation

3.1. The Application Layer Protocol Negotiation Extension

   A new extension type ("application_layer_protocol_negotiation(16)")
   is defined and MAY be included by the client in its "ClientHello"
   message.

```
   enum {
       application_layer_protocol_negotiation(16), (65535)
   } ExtensionType;
```

   The "extension_data" field of the
   ("application_layer_protocol_negotiation(16)") extension SHALL
   contain a "ProtocolNameList" value.

```
   opaque ProtocolName<1..2^8-1>;

   struct {
       ProtocolName protocol_name_list<2..2^16-1>
   } ProtocolNameList;
```

   "ProtocolNameList" contains the list of protocols advertised by the
   client, in descending order of preference. Protocols are named by
   IANA registered, opaque, non-empty byte strings, as described further
   in Section 6 "IANA Considerations" of this document. Implementations
   MUST ensure that an empty string is not included and that no byte
   strings are truncated.

   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 protocol name that it does not recognize. A new
ServerHello extension type
("application_layer_protocol_negotiation(16)") MAY be returned to the
client within the extended ServerHello message. The "extension_data"
field of the ("application_layer_protocol_negotiation(16)") extension
SHALL be structured the same as described above for the client
"extension_data", except that the "ProtocolNameList" MUST contain
exactly one "ProtocolName".

Therefore, a full handshake with the
"application_layer_protocol_negotiation" extension in the ClientHello
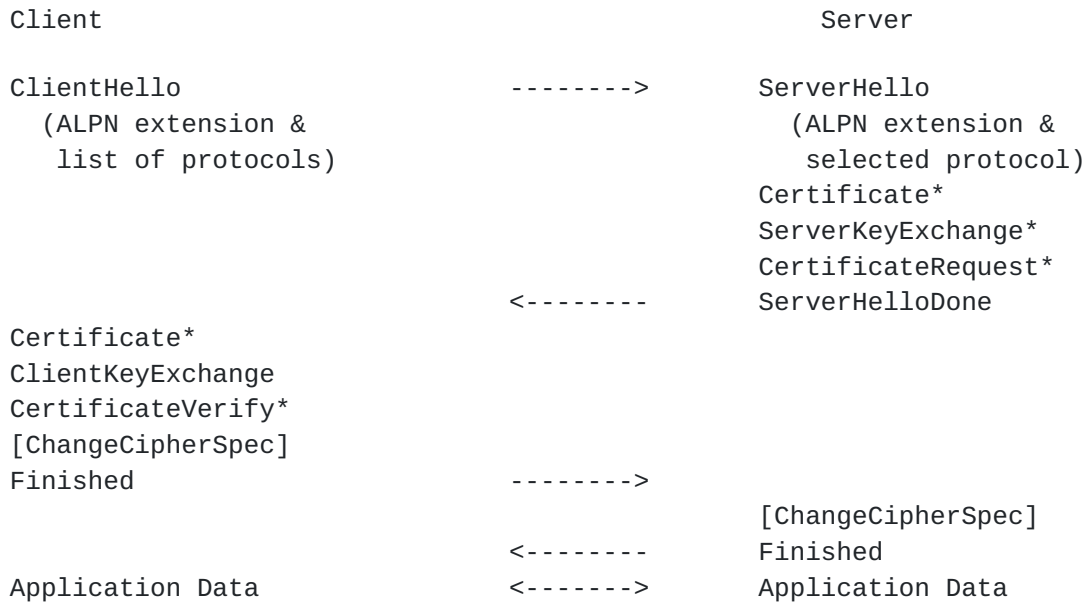and ServerHello messages has the following flow (contrast with
section 7.3 of [RFC5246]):

```
Client                                          Server

ClientHello                      -------->       ServerHello
  (ALPN extension &                                (ALPN extension &
   list of protocols)                               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" extension has the following
flow:

```
Client                                          Server

ClientHello                      -------->       ServerHello
  (ALPN extension &                                (ALPN extension &
   list of protocols)                               selected protocol)
                                                 [ChangeCipherSpec]
                                 <--------       Finished
[ChangeCipherSpec]
Finished                         -------->
Application Data                 <------->       Application Data
```
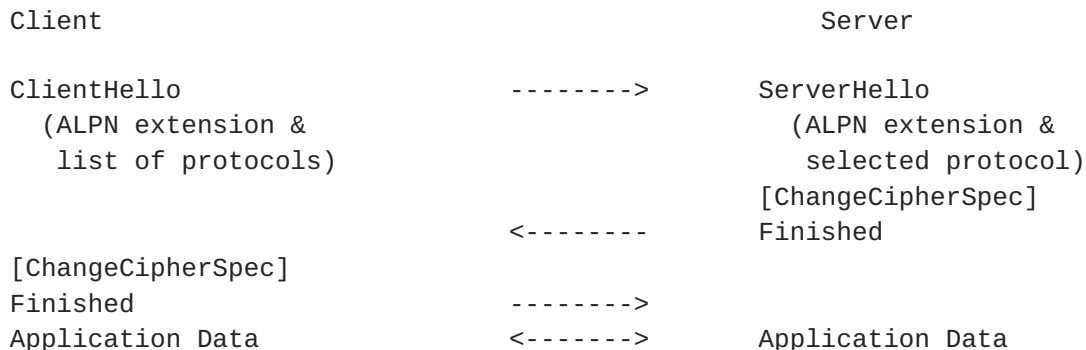
Figure 2

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.

3.2. 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 SHALL respond with a fatal "no_application_protocol" alert.

```
enum {
    no_application_protocol(120),
    (255)
} AlertDescription;
```

The "no_application_protocol" fatal alert is only defined for the "application_layer_protocol_negotiation" extension and MUST NOT be sent unless the server has received a ClientHello message containing this extension.

The protocol identified in the "application_layer_protocol_negotiation" 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.

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 client/server hello exchange in accordance with
established TLS architecture. The
"application_layer_protocol_negotiation" ServerHello 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. By placing
ownership of protocol selection on the server, ALPN facilitates
scenarios in which certificate selection or connection rerouting may
be based on the negotiated protocol.

Finally, by managing protocol selection in the clear as part of the
handshake, ALPN avoids introducing false confidence with respect to
the ability to hide the negotiated protocol in advance of
establishing the connection. If hiding the protocol is required,
then renegotiation after connection establishment, which would
provide true TLS security guarantees, would be a preferred
methodology.

[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.

[6](). IANA Considerations

The IANA has updated its Registry of TLS ExtensionType Values to
include the following entry:

    16 application_layer_protocol_negotiation

This document establishes a registry for protocol identifiers
entitled "Application Layer Protocol Negotiation (ALPN) Protocol IDs"
under the existing "Transport Layer Security (TLS)" heading.

Entries in this registry require the following fields:

o Protocol: The name of the protocol.

o Identification Sequence: The precise set of octet values that
   identifies the protocol. This could be the UTF-8 encoding
   [RFC3629] of the protocol name.

   o Specification: A reference to a specification that defines the
      protocol.

   This registry operates under the "Expert Review" policy as defined in
   [RFC5226]. The designated expert is advised to encourage the
   inclusion of a reference to a permanent and readily available
   specification that enables the creation of interoperable
   implementations of the identified protocol.

   An initial set of registrations for this registry follows:

      Protocol: HTTP/1.1

      Identification Sequence: 0x68 0x74 0x74 0x70 0x2f 0x31 0x2e 0x31
      ("http/1.1")

      Specification: http://tools.ietf.org/html/rfc2616

      Protocol: SPDY/1

      Identification Sequence: 0x73 0x70 0x64 0x79 0x2f 0x31 ("spdy/1")

      Specification: http://dev.chromium.org/spdy/spdy-protocol/spdy-
      protocol-draft1

      Protocol: SPDY/2

      Identification Sequence: 0x73 0x70 0x64 0x79 0x2f 0x32 ("spdy/2")

      Specification: http://dev.chromium.org/spdy/spdy-protocol/spdy-
      protocol-draft2

      Protocol: SPDY/3

      Identification Sequence: 0x73 0x70 0x64 0x79 0x2f 0x33 ("spdy/3")

      Specification: http://dev.chromium.org/spdy/spdy-protocol/spdy-
      protocol-draft3

7. Acknowledgements

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

8. References

8.1. 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.

   [RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO
              10646", STD 63, RFC 3629, November 2003.

   [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an
              IANA Considerations Section in RFCs", BCP 26, RFC 5226,
              May 2008.

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

8.2. Informative References

   [I-D.ietf-httpbis-http2]
              Belshe, M., Peon, R., Thomson, M., and A. Melnikov,
              "Hypertext Transfer Protocol version 2.0", draft-ietf-
              httpbis-http2-06 (work in progress), August 2013.

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

Authors' Addresses

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

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

Andrei Popov
Microsoft Corp.
One Microsoft Way
Redmond, WA 98052
USA

Email: andreipo@microsoft.com

Adam Langley
Google Inc.
USA

Email: agl@google.com

Emile Stephan
Orange
2 avenue Pierre Marzin
Lannion F-22307
France

Email: emile.stephan@orange.com