Network Working Group
Internet-Draft
Expires: October 31, 2012

A. Langley
Google Inc

May 2012

Transport Layer Security (TLS) Next Protocol Negotiation Extension
draft-agl-tls-nextprotoneg-04

Abstract

This document describes a Transport Layer Security (TLS) extension for application layer protocol negotiation. This allows the application layer to negotiate which protocol should 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 October 31, 2012.

Table of Contents

1. Introduction

The Next Protocol Negotiation extension (NPN) is currently used to negotiate the use of SPDY [spdy] as an application level protocol on port 443, and to perform SPDY version negotiation. However, it is not SPDY specific in any way.

Designers of new application level protocols are faced with a problem: there are no good options for establishing a clean transport for a new protocol and negotiating its use. Negotiations on port 80 will run afoul of intercepting proxies. Ports other than 80 and 443 are likely to be firewalled without any fast method of detection, and are also unlikely to traverse HTTP proxies with CONNECT. Negotiating on port 443 is possible, but may run afoul of MITM proxies and also uses a round trip for negotiation on top of the round trips for establishing the TLS connection. Negotiation at that level is also dependent on the application level protocol, i.e. the real world tolerance of servers to HTTP Upgrade requests.

Next Protocol Negotiation allows application level protocols to be negotiated without additional round trips and with clean fallback in the case of an unsupportive MITM proxy.

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 Negotiation Extension

A new extension type ("next_protocol_negotiation(TBD)") is defined and 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 echo the extension in its "ServerHello".

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

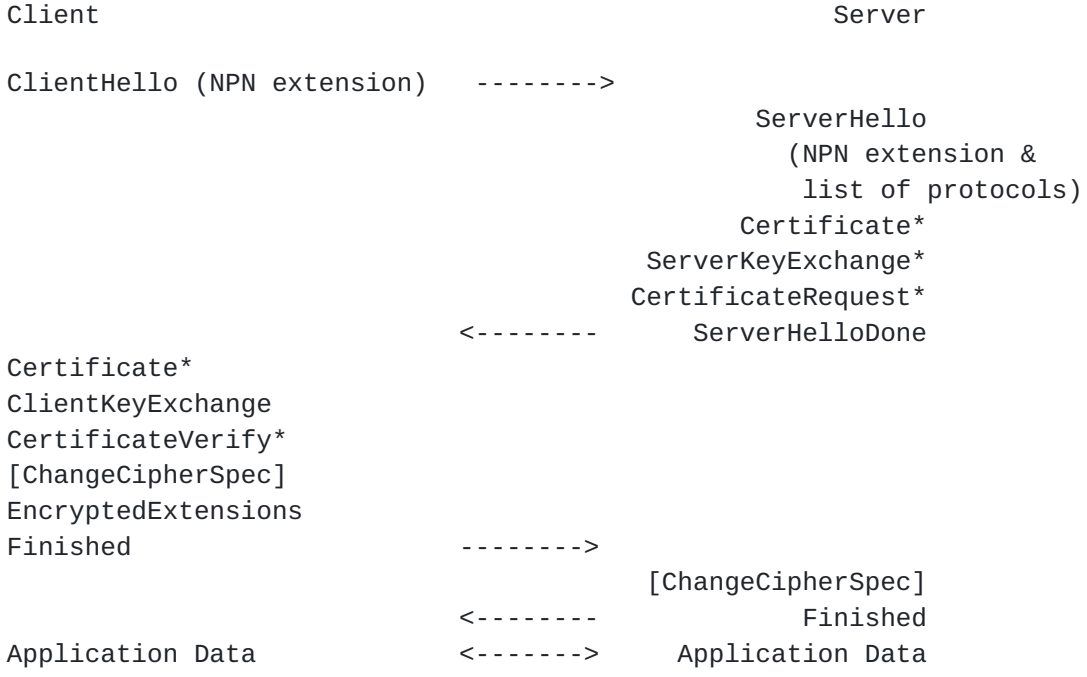
The "extension_data" field of a "next_protocol_negotiation" extension in a "ClientHello" MUST be empty.

The "extension_data" field of a "next_protocol_negotiation" extension in a "ServerHello" contains an optional list of protocols advertised by the server. Protocols are named by opaque, non-empty byte strings and the list of protocols is serialized as a concatenation of 8-bit, length prefixed byte strings. Implementations MUST ensure that the empty string is not included and that no byte strings are truncated.

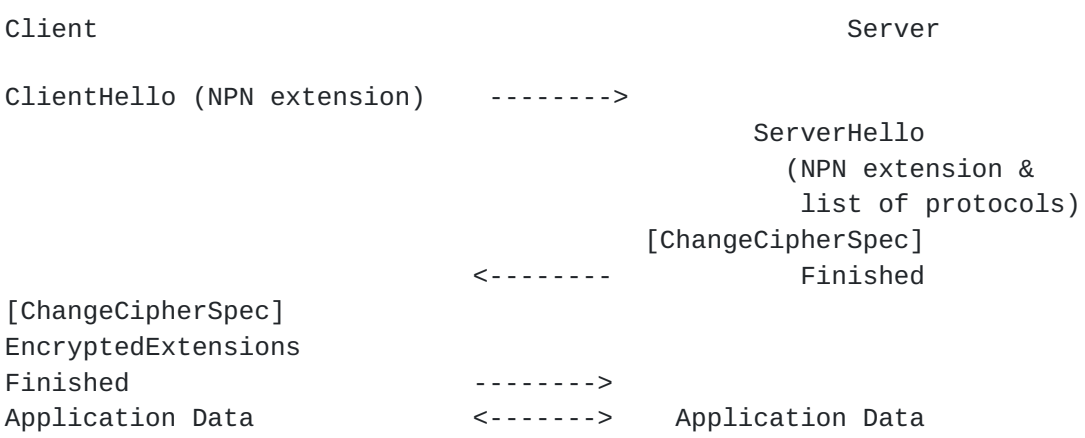
A new handshake message type ("encrypted_extensions(TBD)") is defined. If the server included a "next_protocol_negotiation" extension in its "ServerHello" message, the client MUST send a "EncryptedExtensions" message after its "ChangeCipherSpec" and before its "Finished" message.

```
enum {
  encrypted_extensions(TBD), (65535)
} HandshakeType;
```

Therefore a full handshake with "EncryptedExtensions" has the following flow (contrast with section 7.3 of RFC 5246 [RFC5246]):



An abbreviated handshake with "EncryptedExtensions" has the following flow:



The "EncryptedExtensions" message contains a series of "Extension" structures (see section 7.4.1.4 of RFC 5246 [RFC5246]

If the server included a "next_protocol_negotiation" extension in its "ServerHello" message, the client MUST include an "Extension" with "extension_type" equal to "next_protocol_negotiation(TBD)". The "extension_data" of which has the following format:

```
struct {
  opaque selected_protocol<0..255>;
  opaque padding<0..255>;
} NextProtocolNegotiationEncryptedExtension;
```

The contents of "selected_protocol" are an opaque protocol string, but need not have been advertised by the server. The length of "padding" SHOULD be 32 - ((len(selected_protocol) + 2) % 32). Note that len(selected_protocol) does not include its length prefix.

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_negotiation" extension.

4. Protocol selection

It's expected that a client will have a list of protocols that it supports, in preference order, and will only select a protocol if the server supports it. In that case, the client SHOULD select the first protocol advertised by the server that it also supports. In the event that the client doesn't support any of server's protocols, or the server doesn't advertise any, it SHOULD select the first protocol

that it supports.

There may be cases where the client knows, via other means, that a server supports an unadvertised protocol. In these cases the client can simply select that protocol.

5. Design discussion

NPN is an outlier from TLS in several respects: firstly that it introduces a handshake message between the "ChangeCipherSpec" and "Finished" message, that the handshake message is padded, and that the negotiation isn't done purely with the hello messages. All these aspects of the protocol are intended to prevent middle-ware discrimination based on the negotiated protocol and follow the general principle that anything that can be encrypted, should be encrypted. The server's list of advertised protocols is in the clear as a compromise between performance and robustness.

6. Security considerations

The server's list of supported protocols is still advertised in the clear with this extension. 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. In this case, clients may wish to opportunistically select a protocol that wasn't advertised by the server. However, the workings of such a scheme are outside the scope of this document.

7. IANA Considerations

This document requires IANA to update its registry of TLS extensions to assign an entry referred to here as "next_protocol_negotiation".

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

This document also requires IANA to create a registry of TLS Next Protocol Negotiation protocol strings on a first come, first served basis, initially containing the following entries:

- "http/1.1": HTTP/1.1 [RFC2616]
- "spdy/1": (obsolete) SPDY version 1
- "spdy/2": SPDY version 2
- "spdy/3": SPDY version 3

8. Acknowledgments

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

9. References

9.1. Normative References

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

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

9.2. Informative References

[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.

[tor] Dingledine, R., Matthewson, N. and P. Syverson, "Tor: The Second-Generation Onion Router", August 2004.

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

Author's Address

Adam Langley
Google Inc

Email: agl@google.com