Workgroup: Network Working Group
Internet-Draft: draft-thomson-tls-snip-01
Published: 4 January 2021
Intended Status: Informational
Expires: 8 July 2021
Authors: M. Thomson
Mozilla

Secure Negotiation of Incompatible Protocols in TLS

Abstract

An extension is defined for TLS that allows a client and server to detect an attempt to force the use of less-preferred application protocol even where protocol options are incompatible. This supplements application-layer protocol negotiation, which allows choices between compatible protocols to be authenticated.

Discussion Venues

This note is to be removed before publishing as an RFC.

Discussion of this document takes place on the TLS Working Group mailing list (tls@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/tls/.

Source for this draft and an issue tracker can be found at https://github.com/martinthomson/snip.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

With increased diversity in protocol choice, some applications are able to use one of several semantically-equivalent protocols to achieve their goals. This is particularly notable in HTTP where there are currently three distinct protocols: HTTP/1.1 [HTTP11], HTTP/2 [HTTP2], and HTTP/3 [HTTP3]. This is also true of protocols that support variants based on both TLS [TLS] and DTLS [DTLS].

For protocols that are mutually compatible, Application-Layer Protocol Negotiation (ALPN; [ALPN]) provides a secure way to negotiate protocol selection.

In ALPN, the client offers a list of options in a TLS ClientHello and the server chooses the option that it most prefers. A downgrade attack occurs where both client and server support a protocol that the server prefers more than than the selected protocol. ALPN protects against this attack by ensuring that the server is aware of all options the client supports and including those options and the server choice under the integrity protection provided by the TLS handshake.

This downgrade protection functions because protocol negotiation is part of the TLS handshake. The introduction of semantically-equivalent protocols that use incompatible handshakes introduces new opportunities for downgrade attack. For instance, it is not possible to negotiate the use of HTTP/2 based on an attempt to connect using HTTP/3. The former relies on TCP, whereas the latter uses UDP. These protocols are therefore mutually incompatible.

This document defines an extension to TLS that allows clients to discover when servers support alternative protocols that are incompatible with the currently-selected TLS version. This might be used to avoid downgrade attack caused by interference in protocol discovery mechanisms.

This extension is motivated by the addition of new mechanisms, such as [SVCB]. SVCB enables the discovery of servers that support multiple different protocols, some of which are incompatible. The extension can also be used to authenticate protocol choices that are discovered by other means.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

Two protocols are consider "compatible" if it is possible to negotiate either using the same connection attempt. In comparison, protocols are "incompatible" if they require separate attempts to establish a connection.

3. Incompatible Protocols and SVCB

The SVCB record [SVCB] allows a client to learn about services associated with a domain name. This includes how to locate a server, along with supplementary information about the server, including protocols that the server supports. This allows a client to start using a protocol of their choice without added latency, as the lookup can be performed concurrently with other name resolution. The added cost of the additional DNS queries is minimal.

However, SVCB provides no protection against a downgrade attack between incompatible protocols. An attacker could remove DNS records for client-preferred protocols, leaving the client to believe that only less-prefered, mutually-incompatible options are available. The client only offers compatible options to a server in its TLS handshake. Even if a client were to inform the server that it supports a more preferred protocol, the server would not be able to act upon it.

Authenticating all of the information presented in SVCB records might provide clients with complete information about server support, but this is impractical for several reasons:

* it is not possible to ensure that all server instances in a deployment have the same protocol configuration, as deployments for a single name routinely include multiple providers that cannot coordinate closely;

* the ability to provide a subset of valid DNS records is integral to many strategies for managing servers; and

* it is difficult to ensure that cached DNS records are synchronized with server state.

Overall, an authenticated TLS handshake is a better source of authoritative information about the protocols that are supported.

4. Authenticating Incompatible Protocols

The incompatible_protocols(TBD) TLS extension provides clients with information about the incompatible protocols that are supported by servers.

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

A client that supports the extension advertises an empty extension. In response, a server that supports this extension includes a list of application protocol identifiers. The "extension_data" field of the value server extension uses the ProtocolName type defined in [ALPN], which is repeated here. This syntax is shown in Figure 1.

```
enum {
  default(0), svcb(1), quic(2), (255)
} ProtocolAuthenticationScope;

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

struct {
  ProtocolAuthenticationScope scope;
  ProtocolName protocol;
} IncompatibleProtocol;

struct {
  select (Handshake.msg_type) {
    case client_hello:
      Empty;
    case encrypted_extensions:
      IncompatibleProtocol incompatible_protocols<3..2^16-1>;
  };
} IncompatibleProtocols;
```

Figure 1: TLS Syntax for incompatible_protocols Extension

This extension only applies to the ClientHello and EncryptedExtensions messages. An implementation that receives this extension in any other handshake message MUST send a fatal illegal_parameter alert.

A server deployment that supports multiple incompatible protocols MAY advertise all protocols that are supported. Each protocol is paired with an identifier for the Protocol Authentication Scope, which defines how endpoints for that protocol might be discovered; see Section 6.

A server needs to ensure that protocols advertised in this fashion are available to the client within the same protocol authentication scope.

A server MUST omit any compatible protocols from this extension. That is, any protocol that the server might be able to select, had the client offered the protocol in the application_layer_protocol_negotiation extension. Clients are expected to include all compatible protocols in the application_layer_protocol_negotiation extension.

A server MAY limit the incompatible protocols that it advertises to those that have similar semantics to protocols that the client lists in its application_layer_protocol_negotiation extension.

The definition of what a server includes is intentionally flexible. It is better that a server offer more information than less as the needs of a client are not necessarily well reflected in its ALPN extension. However, it might not be feasible for a server to advertise all potential protocols; see Section 8 for more discussion on this point.

5. Incompatible Protocol Selection

This document expands the definition of protocol negotiation to include both compatible and incompatible protocols and provide protection against downgrade for both types of selection. ALPN [ALPN] only considers compatible protocols: the client presents a set of compatible options and the server chooses its most preferred.

With an selection of protocols that includes incompatible options, the client makes a selection between incompatible options before making a connection attempt. Therefore, this design does not enable negotiation, it instead provides the client with information about other incompatible protocols that the server might support.

Detecting a potential downgrade between incompatible protocols does not automatically imply that a client abandon a connection attempt. It only provides the client with authenticated information about its options. What a client does with this information is left to client policy.

In brief:

* For compatible protocols, the client offers all acceptable options and the server selects its most preferred

* For incompatible protocols, information the server offers is authenticated and the client is able to act on that

For a protocol like HTTP/3, this might not result in the client choosing to use HTTP/3, even if HTTP/3 is preferred and the server indicates that a service endpoint supporting HTTP/3 is available. Blocking of UDP or QUIC is known to be widespread. As a result, clients might adopt a policy of tolerating a downgrade to a TCP-based protocol, even if HTTP/3 were preferred. However, as blocking of UDP is highly correlated by access network, clients that are able to establish HTTP/3 connections to some servers might choose to apply a stricter policy when a server that indicates HTTP/3 support is unreachable.

6. Protocol Authentication Scope

A protocol authentication scope includes a set of service endpoints that are provided downgrade protection by this mechanism. There are multiple types of protocol authentication scope, each identified by a different type. The type of protocol authentication scope is encoded in the ProtocolAuthenticationScope enum.

The type of protocol authentication scope describes how a client might learn of all of the service endpoints that a server offers in that scope. If a client has attempted to discover service endpoints using the methods defined by the protocol authentication scope, receiving an incompatible_protocols extension from a server is a strong indication of a potential downgrade attack.

A client considers that a downgrade attack might have occurred if all of the following occur:

1. A server advertises that there are endpoints that support a protocol that the client prefers over the protocol that is currently in use.

2. The protocol authentication scope associated with that protocol is understood by the client and the client attempted to discover services in that scope.

In response to detecting a potential downgrade attack, a client might abandon the current connection attempt and report an error. A client that supports discovery of incompatible protocols, but chooses not to make a discovery attempt under normal conditions might instead not fail, but it could use what it learns as cause to initiate discovery.

6.1. The Default Scope

The default protocol authentication scope reserves an identifier of 0. A client cannot act on information about incompatible protocols advertised with this scope. A server MUST NOT advertise incompatible protocols with this scope; however, a client MUST ignore advertisements it receives.

The default protocol authentication scope is reserved for discovery methods that have no explicit scope; see Section 7 for more on this subject.

6.2. SVCB Scope

The SVCB protocol authentication scope uses an identifier of 1. A server that lists incompatible protocols with this scope indicates that SVCB records ServiceForm records with the same SvcDomainName exist that refer to services that support the indicated protocol.

The SVCB protocol authentication scope also applies to records that use the SVCB form, like HTTPS.

This ensures that the final choice a client makes between ServiceForm SVCB records is protected by this extension. If the client does not receive a SVCB record for a protocol that the server includes in its incompatible_protocols extension, then it can assume that this omission was caused by an error or attack.

A choice between AliasForm records (or CNAME or DNAME records) is not authenticated, but choices between ServiceForm records is. This allows for server deployments for the same name to have different administrative control and protocol configurations.

6.3. QUIC Version Negotiation Scope

The QUIC version negotiation protocol authentication scope uses an identifier of 2. A server that lists incompatible protocols with this scope indicates that QUIC version negotiation at the same server IP and port could be used to learn of incompatible QUIC versions that support the indicated protocol.

Using this protocol authentication scope depends on application protocols that are dependent on a specific QUIC version.

7. Other Discovery Methods

For other discovery methods, a definition for protocol authentication scope is needed before a client can act on what is learned using the incompatible_protocols extension. That definition needs to define how to discover server instances that support all incompatible protocols in the scope.

In particular, a server that is discovered using forms of DNS-based name resolution other than SVCB uses the default protocol authentication scope; see Section 6.1. Discovering services in this way does not provide enough information to locate other incompatible protocols.

For instance, an HTTPS server that is discovered using purely A or AAAA records (and CNAME or DNAME records) might advertise support for incompatible protocols, but as there is no way to determine where those protocols are supported, a client cannot act on the information. Note that Alternative Services do not change the protocol authentication scope.

Deployments of discovery methods that define a protocol authentication scope larger than the default need to ensure that every server provides information that is consistent with every protocol authentication scope that includes that server. A server that fails to indicate support for a protocol that is within a protocol authentication scope does not offer any protection against attack; a server that advertises a protocol that the client cannot discover risks this misconfiguration being identified as an attack by clients.

7.1. Alternative Services

It is possible to negotiate protocols based on an established connection without exposure to downgrade. The Alternative Services [ALTSVC] bootstrapping in HTTP/3 [HTTP3] does just that. Assuming that HTTP/2 or HTTP/1.1 are not vulnerable to attacks that would compromise integrity, a server can advertise the presence of an endpoint that supports HTTP/3.

Under these assumptions Alternative Services is secure, but it has performance trade-offs. A client could attempt the protocol it prefers most, but that comes at a risk that this protocol is not supported by a server. A client could implement a fallback, which might even be performed concurrently (see [HAPPY-EYEBALLS]), but this costs time and resources. A client avoids these costs by attempting the protocol it believes to be most widely supported, though this comes with a performance penalty in cases where the most-preferred protocol is supported.

A server that is discovered using Alternative Services uses the default protocol authentication scope. As use of Alternative Services is discretionary for both client and server, a client cannot expect to receive information about incompatible protocols. To avoid downgrade, a client only has to limit its use of Alternative Services to those that it prefers more than the active protocol.

8. Operational Considerations

By listing incompatible protocols, a server does not indicate how to find endpoints that support those protocols, only that they exist. This ensures that server configuration is minimized, as servers do not require tight coordination. Providing even this much information could present operational difficulties as it requires that incompatible protocols are only listed when those protocols are deployed.

Server deployments can choose not to provide information about incompatible protocols, which denies clients information about downgrade attacks but might avoid the operational complexity of providing accurate information.

During rollout of a new, incompatible protocol, until the deployment is stable and not at risk of being disabled, servers SHOULD NOT advertise the existence of the new protocol. Protocol deployments that are disabled, first need to be removed from the incompatible_protocols extension or there could be some loss of service. Though the incompatible_protocols extension only applies at the time of the TLS handshake, clients might take some time to act on the information. If an incompatible protocol is removed from deployment between when the client completes a handshake and when it acts, this could be treated as an error by the client.

If a server does not list available, incompatible protocols, clients cannot learn about other services and so cannot detect downgrade attacks against those protocols.

9. Security Considerations

This design depends on the integrity of the TLS handshake across all forms, including TLS [RFC8446], DTLS [DTLS], and QUIC [QUIC-TLS]. An attacker that can modify a TLS handshake in any one of these protocols can cause a client to believe that other options do not exist.

A server deployment that uses AliasForm SVCB records and does not uniformly support a client-preferred protocol is vulnerable to downgrade attacks that steer clients toward instances that lack support for that protocol. This attack is ineffective for protocols that are consistently supported by all server instances.

10. IANA Considerations

TODO: register the extension

TODO: create a registry of scopes

11. References

11.1. Normative References

[ALPN] Friedl, S., Popov, A., Langley, A., and E. Stephan, "Transport Layer Security (TLS) Application-Layer Protocol Negotiation Extension", RFC 7301, DOI 10.17487/RFC7301, July 2014, <https://www.rfc-editor.org/info/rfc7301>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

11.2. Informative References

[ALTSVC]
Nottingham, M., McManus, P., and J. Reschke, "HTTP Alternative Services", RFC 7838, DOI 10.17487/RFC7838, April 2016, <https://www.rfc-editor.org/info/rfc7838>.

[DTLS] Rescorla, E., Tschofenig, H., and N. Modadugu, "The Datagram Transport Layer Security (DTLS) Protocol Version 1.3", Work in Progress, Internet-Draft, draft-ietf-tls-dtls13-39, 2 November 2020, <http://www.ietf.org/internet-drafts/draft-ietf-tls-dtls13-39.txt>.

[HAPPY-EYEBALLS] Wing, D. and A. Yourtchenko, "Happy Eyeballs: Success with Dual-Stack Hosts", RFC 6555, DOI 10.17487/RFC6555, April 2012, <https://www.rfc-editor.org/info/rfc6555>.

[HTTP11] Fielding, R., Nottingham, M., and J. Reschke, "HTTP/1.1", Work in Progress, Internet-Draft, draft-ietf-httpbis-messaging-13, 14 December 2020, <http://www.ietf.org/internet-drafts/draft-ietf-httpbis-messaging-13.txt>.

[HTTP2] Belshe, M., Peon, R., and M. Thomson, Ed., "Hypertext Transfer Protocol Version 2 (HTTP/2)", RFC 7540, DOI 10.17487/RFC7540, May 2015, <https://www.rfc-editor.org/info/rfc7540>.

[HTTP3] Bishop, M., "Hypertext Transfer Protocol Version 3 (HTTP/3)", Work in Progress, Internet-Draft, draft-ietf-quic-http-33, 15 December 2020, <http://www.ietf.org/internet-drafts/draft-ietf-quic-http-33.txt>.

[QUIC-TLS] Thomson, M. and S. Turner, "Using TLS to Secure QUIC", Work in Progress, Internet-Draft, draft-ietf-quic-tls-33, 13 December 2020, <http://www.ietf.org/internet-drafts/draft-ietf-quic-tls-33.txt>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[SVCB] Schwartz, B., Bishop, M., and E. Nygren, "Service binding and parameter specification via the DNS (DNS SVCB and HTTPSSVC)", Work in Progress, Internet-Draft, draft-ietf-dnsop-svcb-httpssvc-03, 11 June 2020, <http://www.ietf.org/internet-drafts/draft-ietf-dnsop-svcb-httpssvc-03.txt>.

[TLS] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

Appendix A. Acknowledgments

Benjamin Schwartz provided significant input into the design of the mechanism and helped clarify many points.

Author's Address

Martin Thomson
Mozilla

Email: mt@lowentropy.net