Network Working Group M. Nottingham
Internet-Draft October 1, 2013
Intended status: Standards Track
Expires: April 4, 2014

Encryption for HTTP URIs Using Alternate Services
draft-nottingham-http2-encryption-00

Abstract

This document proposes a way to optimistically encrypt HTTP/2.0 using TLS for HTTP URIs.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

In discussion at IETF87, it was proposed that the current means of bootstrapping encryption in HTTP [I-D.ietf-httpbis-p1-messaging] - using the "HTTPS" URI scheme - unintentionally gives the server disproportionate power in determining whether encryption is used.

Furthermore, HTTP's current use of TLS [RFC5246] for "https://" URIs is inflexible; it is difficult to introduce new trust roots, for example.

This document proposes changes to HTTP that decouple the URI scheme from the use and configuration of underlying encryption, as well as other aspects of the protocol.

In particular, it defines the concept of an "alternate service" that allows an origin to advertise when its resources are available at a separate location, using a different configuration of protocols.

This allows a "http://" URI to be upgraded to use TLS optimistically.

Because deploying TLS requires acquiring and configuring a valid certificate, some deployments may find supporting it difficult. Therefore, this document also specifies a "relaxed" profile of HTTP/2.0 over TLS that does not require strong server authentication, specifically for use with "http://" URIs.

Note: This is a preliminary draft that attempts to capture the state of relevant discussion to this point. It has not be reviewed for security, deployability, or effectiveness, and is only intended to serve as the basis of further discussion in the HTTPbis Working Group.

1.1. Goals and Non-Goals

This proposal attempts to de-couple a HTTP URI's scheme from the specific wire protocol in use, as well as that protocol's layering onto the network.

The immediate goal is to make HTTP URIs more robust in the face of passive monitoring.

Such passive attacks are often opportunistic; they rely on sensitive information being available in the clear. Furthermore, they are often broad, where all available data is collected en masse, being analyzed separately for relevant information.

It is not a goal of this document to address active or targeted

attacks, although future solutions may be complementary.

Other goals include ease of implementation and deployment, with minimal impact upon performance (in keeping with the goals of HTTP/2.0).

Furthermore, since this proposal is designed as an alternate negotiation mechanism for HTTP/2.0, it is expected that it is useful for that use case as well.

1.2. Notational Conventions

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].

2. Alternate Services

On the Web, a resource is accessed through a scheme (e.g., "https" or "http") on a nominated host / port combination.

These three pieces of information collectively can be used to establish the authority for ownership of the resource (its "origin"; see [RFC6454]), as well as providing enough information to bootstrap access to it.

This document introduces the notion of an "Alternate Service"; when an origin's resources are accessible through a different protocol /

host / port combination, it is said to have an alternate service.

For example, an origin:

("http", "www.example.com", "80")

Might declare that its resources are also accessible at the alternate service:

("http2-tls", "new.example.com", "443")

Alternate services do not replace or change the origin for any given resource; in general, they are not visible to the software above the access mechanism.

Furthermore, it is important to note that the first member of an alternate service tuple is different from the "scheme" component of an origin; it is more specific, identifying not only the major version of the protocol being used, but potentially communication

options for that protocol.

Practically speaking, clients using an alternate service will change the host, port and protocol that they are using to fetch resources, but these changes MUST NOT be propagated to the application that is using HTTP; from that standpoint, the URI being accessed and all information derived from it (scheme, host, port) are the same as before.

Importantly, this includes the security context of the connection; by default, the alternate server will need to present a certificate for the origin's host name, not that of the alternate. Likewise, the Host header is still derived from the origin, not the alternate service.

The changes SHOULD, however, be made visible in debugging tools, consoles, etc.

Clients MUST NOT use alternate services on a host other than the origin's without strong server authentication; one way to achieve this is for the alternate to use TLS with a certificate that is valid for that origin.

For example, if the origin's host is "www.example.com" and an alternate is offered on "other.example.com" with the "http2-tls" protocol, and the certificate offered is valid for "www.example.com", the client can use the alternate. However, if "other.example.com" is offered with the "http2" protocol, the client cannot use it, because there is no mechanism in that protocol to establish strong server authentication.

Formally, an alternate service is identified by the combination of:

- An ALPN protocol, as per [I-D.ietf-tls-applayerprotoneg]
- A host, as per [RFC3986]
- A port, as per [RFC3986]

Potentially, there are many ways that a client could discover the alternate service(s) associated with an origin; this document currently defines one, the Alt-Svc HTTP Header Field.

2.1. The Alt-Svc HTTP Header Field

A HTTP server can advertise the availability of alternate services to HTTP/1.1 and HTTP/2.0 clients by adding an Alt-Svc header field to responses. For example:

```
Alt-Svc: http2-tls-relaxed=:443
```

This indicates that the "http2tls-relaxed" protocol on the same host using the indicated port (in this case, 443).

Alt-Svc can also contain a host:

```
Alt-Svc: http2-tls=other.example.com:443
```

This indicates that all resources on the origin are available using the "http2-tls" profile on other.example.com port 443.

It can also have multiple values:

```
Alt-Svc: http2-tls=:443, http2-tls=other.example.com:443
```

The value(s) advertised by Alt-Svc can be used by clients to open a new connection to one or more alternate services immediately, or simultaneously with subsequent requests on the same connection.

When an alternate service is advertised using Alt-Svc, it is considered to be valid for all resources associated with the origin, and by default is valid for 24 hours from generation of the message. This can be modified with the 'ma' (max-age') parameter;

```
Alt-Svc: http2-tls=:443;ma=3600
```

which indicates the number of seconds since the response was generated the policy is considered fresh for. See [I-D.ietf-httpbis-p6-cache] Section 4.2.3 for details of determining response age.

[[TODO: ABNF]]

2.2. HTTP-related Protocol Identifiers

To accommodate this approach, HTTP/2.0 will need to nominate several protocol negotiation strings (a.k.a. "profiles") to allow negotiation for the desired properties in alternate services.

This might include:

- http1 - http/1.x over TCP
- http1-tls - http/1.x over TLS over TCP (as per [RFC2818])
- http2 - http/2.x over TCP
- http2-tls - http/2.x over TLS over TCP (as per [RFC2818])
- http2-tls-relaxed - http/2.x over TLS over TCP (see below)

Most of these are already latently defined by HTTP/2.0, with the exception being http2-tls-relaxed, defined below.

These profiles are expected to be used not only in the Alt-Svc header field, but also in other HTTP/2.0 negotiation mechanisms; e.g., ALPN, the "Upgrade dance" and so forth.

Note that, as discussed in Security Considerations, there may be situations (e.g,. ALPN) where advertising some of these profiles are inapplicable or inadvisable.

For example, in an ALPN negotiation for a "https://" URI, it is only sensible to offer http1-tls and http2-tls.

2.2.1. The "http2-tls-relaxed" Protocol

Servers that support the "http2-tls-relaxed" protocol indicate that they support TLS for access to URIs with the "http" URI scheme using HTTP/2.0 or greater.

Servers MAY advertise the "http2-tls-relaxed" profile for resources with a "http" origin scheme; they MUST NOT advertise it for resources with a "https" origin.

When a client connects to an "http2-tls-relaxed" alternate service, it MUST use TLS1.1 or greater, and MUST use HTTP/2.x. HTTP/2.0 SHOULD be used as soon as TLS negotiation is completed; i.e., the "Upgrade dance" SHOULD NOT be performed.

When connecting to an "http2-tls-relaxed" service, the algorithm for authenticating the server described in [RFC2818] Section 3.1 changes; the client does not necessarily validate its certificate for expiry, hostname match or relationship to a known certificate authority (as it would with "normal" HTTPS).

However, the client MAY perform additional checks on the certificate and make a decision as to its validity before using the server. Definition of such additional checks are out of scope for this specification.

Upon initial adoption of this proposal, it is expected that no such additional checks will be performed. Therefore, the client MUST NOT use the "http2-tls-relaxed" profile to connect to alternate services whose host does not match that of the origin, unless additional checks are performed.

This requirement bounds the risk of a service being hijacked and redirected to another host; see Security Considerations for details.

[[TODO: define "match"]]

Servers SHOULD use the same certificate consistently over time, to aid future extensions for building trust and adding other services.

[[TODO: define "same"; likely not the same actual certificate.]]

When the http2-tls-relaxed protocol is in use, User Agents MUST NOT indicate the connection has the same level of security as https:// (e.g. using a "lock device").

3. Security Considerations

3.1. Alt-Svc Header Field Downgrade Attacks

Because the Alt-Svc header field appears in the clear (for "http://" URIs), it is subject to downgrade by attackers that are able to Man-in-the-Middle the network connection; in its simplest form, an attacker that wants the connection to remain in the clear need only strip the Alt-Svc header from responses.

This proposal does not offer a remedy for this risk. However, it's important to note that it is no worse than current use of unencrypted HTTP in the face of such active attacks.

3.2. Poor Client Profile Choices

Furthermore, because different protocols can have different security properties, clients ought not blindly use alternate services, but instead they option(s) presented for conformance to implementation policy, user preferences, and general security.

For example, in theory the header field could be used to advertise a downgrade to plain TCP from a TLS-protected connection, or to relax certificate checks for a HTTPS URI; opting to use such an alternate would seldom be desirable.

3.3. Alt-Svc Header Field Hijacking

An attacker local to the Web server who can inject response header fields can redirect HTTP traffic to a different port on the same host using the "http2-tls-relaxed" protocol; if it is under their control, the can masquerade as the HTTP server.

An attacker local to the Web server who can inject response header fields can redirect HTTP traffic to an arbitrary host and port using the "http2-tls" protocol; if they can present a certificate which validates for the host under attack, they can masquerade as that server.

Both of these risks can be mitigated by appropriate controls to setting response header fields, as well as control over who can open a port for listening (in the former case) and good certificate hygiene (in the latter case).

An attacker who can Man-in-the-Middle the network connection and inject response header fields directly can redirect HTTP traffic to a different port and (presumably) masquerade as that server.

As with HTTP today, it is not possible to mitigate this latter risk without cryptographic solutions.

3.4. Alt-Svc Header Field "Gap"

When the Alt-Svc header field is used in "http://" URIs, the client needs to send an unencrypted HTTP request to the server to discover the alternates. In doing so, it potentially exposes sensitive information (e.g., cookies [RFC6265]) to surveillance.

This risk can be mitigated if the client is willing to send a separate request (e.g., OPTIONS *) to the origin to discover policy before making requests containing potentially sensitive information. However, doing so adds a round-trip of latency to such requests.

Likewise, if the Alt-Svc is cacheable for a long period (using a large ma parameter), it reduces the window for such attacks (but does not eliminate it).

Alternatively, this risk can be mitigated by using an out-of-band discovery mechanism (e.g., DNS).

4. References

4.1. Normative 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.

[I-D.ietf-httpbis-p1-messaging]
Fielding, R. and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", draft-ietf-httpbis-p1-messaging-24 (work in progress),

September 2013.

[I-D.ietf-httpbis-p6-cache]
Fielding, R., Nottingham, M., and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Caching", draft-ietf-httpbis-p6-cache-24 (work in progress), September 2013.

[I-D.ietf-tls-applayerprotoneg]
Friedl, S., Popov, A., Langley, A., and S. Emile, "Transport Layer Security (TLS) Application Layer Protocol Negotiation Extension", draft-ietf-tls-applayerprotoneg-02 (work in progress), September 2013.

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

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

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

[RFC6454] Barth, A., "The Web Origin Concept", RFC 6454, December 2011.

4.2. Informative References

[I-D.mbelshe-httpbis-spdy]
Belshe, M. and R. Peon, "SPDY Protocol", draft-mbelshe-httpbis-spdy-00 (work in progress), February 2012.

[RFC2804] IAB and IESG, "IETF Policy on Wiretapping", RFC 2804, May 2000.

[RFC3365] Schiller, J., "Strong Security Requirements for Internet Engineering Task Force Standard Protocols", BCP 61, RFC 3365, August 2002.

[RFC6265] Barth, A., "HTTP State Management Mechanism", RFC 6265, April 2011.

[RFC6555] Wing, D. and A. Yourtchenko, "Happy Eyeballs: Success with Dual-Stack Hosts", RFC 6555, April 2012.

[RFC6962] Laurie, B., Langley, A., and E. Kasper, "Certificate

Transparency", RFC 6962, June 2013.

[RFC6973] Cooper, A., Tschofenig, H., Aboba, B., Peterson, J., Morris, J., Hansen, M., and R. Smith, "Privacy Considerations for Internet Protocols", RFC 6973, July 2013.

[firesheep] Butler, E., "Firesheep", 2010, <http://codebutler.com/firesheep/>.

[streetview] Kravets, D., "The Anatomy of Google's Wi-Fi Sniffing Debacle", 2012, <http://www.wired.com/threatlevel/2012/05/google-wifi-fcc-investigation/>.

[xkeyscore] Greenwald, G., "NSA tool collects 'nearly everything a user does on the internet'", 2013, <http://www.theguardian.com/world/2013/jul/31/nsa-top-secret-program-online-data>.

Appendix A. Acknowledgements

Thanks to Patrick McManus, Eliot Lear, Stephen Farrell, Guy Podjarny, Stephen Ludin, Erik Nygren, Paul Hoffman and Adam Langley for their feedback and suggestions.

Appendix B. Recent History and Background

One of the design goals for SPDY [I-D.mbelshe-httpbis-spdy] was increasing the use of encryption on the Web, achieved by only supporting the protocol over a connection protected by TLS [RFC5246].

This was done, in part, because sensitive information - including not only login credentials, but also personally identifying information (PII) and even patterns of access - are increasingly prevalent on the Web, being evident in potentially every HTTP request made.

Attacks such as FireSheep [firesheep] showed how easy it is to gather such information when it is sent in the clear, and incidents such as Google's collection of unencrypted data by its StreetView Cars [streetview] further illustrated the risks.

In adopting SPDY as the basis of HTTP/2 [I-D.ietf-httpbis-http2], the HTTPbis Working Group agreed not to make TLS mandatory to implement

(MtI) or mandatory to use (MtU) in our charter, despite an IETF policy to prefer the "best security available" [RFC3365].

There were a variety of reasons for this, but most significantly, HTTP is used for much more than the traditional browsing case, and encryption is not needed for all of these uses. Making encryption MtU or MtI was seen as unlikely to succeed because of the wide deployment of HTTP URIs.

However, since making that decision, there have been developments that have caused the Working Group to discuss these issues again:

1. Active contributors to some browser implementations have stated that their products will not use HTTP/2 over unencrypted connections. If this eventuates, it will prevent wide deployment of the new protocol (i.e., it couldn't be used with those products for HTTP URIs; only HTTPS URIs).
2. It has been reported that surveillance of HTTP traffic takes place on a broad scale [xkeyscore]. While the IETF does not take a formal, moral position on wiretapping, we do have a strongly held belief "that both commercial development of the Internet and adequate privacy for its users against illegal intrusion requires the wide availability of strong cryptographic technology" [RFC2804]. This requirement for privacy is further reinforced by [RFC6973].

As a result, we decided to revisit the issue of how encryption is used in HTTP/2.0 at IETF87.

Appendix C. Next Steps

There are three separable aspects to this proposal:

- The concept of alternate services
- The Alt-Svc header field
- The http2-tls-relaxed protocol

In evaluating it, they should be considered separately.

Depending on what aspects we decide to adopt, there are also a number of related issues that should be discussed:

- DNS: Alternate services are also amenable to DNS-based discovery. If there is sufficient interest, a future revision may include a proposal for that.

- Upgrade: For some flows, it may be advantageous to do an "upgrade dance" to the tls-relaxed protocol, a la STARTTLS. If there is sufficient interest, a future revision may also include a proposal for that.
- http1-tls-relaxed: If there is sufficient interest, it may also be worthwhile defining a HTTP/1-based tls-relaxed protocol.
- Priority and Weight: It may be advantageous to include measures of priority and weight in the Alternate Services model (similar to SRV).
- Indicating Chosen Service: It's likely necessary for the server to know which protocol the client has chosen, and perhaps even the hostname (for load balancing). This could be conveyed as part of the "magic", or as a request header. There are also security implications here; for example, without this information, the server doesn't know if the client has checked the certificate, leading to a situation where an intermediary can downgrade a HTTPS connection to relaxed HTTP.
- Client Behavior: Currently, this mechanism is completely

declarative, and clients have free reign as to how they use the alternate services. It may be desirable to specify this more closely.

- IPV6: The intersection between Alternate Services and IPV6 / Happy Eyeballs [RFC6555] should be investigated.

Appendix D. Frequently Asked Questions

D.1. Will this make encryption mandatory in HTTP/2.0?

Not in the sense that this proposal would have it required (with a MUST) in the specification.

What might happen, however, is that some browser implementers will take the flexibility that this approach grants and decide to not negotiate for HTTP/2.0 without one of the encryption profiles. That means that servers would need to implement one of the encryption-enabling profiles to interoperate using HTTP/2.0 for HTTP URIs.

D.2. No certificate checks? Really?

http2-tls-relaxed has the effect of relaxing certificate checks on "http://" - but not "https://" - URIs when TLS is in use. Since TLS isn't in use for any "http://" URIs today, there is no net loss of security, and we gain some privacy from passive attacks.

In the future, if the certificate trust system can be improved such that it's both more reliable and has a lower barrier to entry (e.g., see [RFC6962]), it may be possible to modify or even drop the http2-

tls-relaxed profile (even before HTTP/2 ships, depending on progress there).

D.3. Why do this if a downgrade attack is so easy?

There are many attack scenarios (e.g., third parties in coffee shops) where active attacks are not feasible, or much more difficult.

Furthermore, active attacks can be more easily detected. Future infrastructure (again, along similar lines to [RFC6962]) might be able to detect them and mitigate the risk.

D.4. What about using DNS?

Using DNS for discovery of alternate services has attractive performance characteristics, and also avoids the "gap" vulnerability. However, it is significantly more difficult to deploy, compared to a HTTP header.

If there is implementer interest, a future revision might include a DNS approach.

D.5. Doesn't Alt-Svc make it easy to hijack a Web server?

In introducing Alt-Svc, we are taking a bounded risk, in that anyone who has access to write a response header for an origin can effectively take over the Web site.

To mitigate this, we require the alternate server to either a) be a port on the same hostname (as the Alternate-Protocol header from SPDY did), or if it's on another host b) present a certificate that's valid for the origin server.

D.6. What about using Upgrade?

While it's possible that the HTTP Upgrade header could be used in a STARTTLS-like connection upgrade, that's more difficult to deploy with existing infrastructure, and is constrained to upgrading the same connection, leading to possible latency issues. Alt-Svc offers a more flexible and less intrusive approach.

That said, if there is sufficient interest, we'll look at defining an Upgrade-based mechanism.

D.7. Why not 305 Use Proxy?

While it's possible to use a HTTP response code to redirect the client to an alternate service, this would unavoidably introduce a round trip (at least) before the new connection is established, which violates the performance focus of HTTP/2.0.

D.8. Will this make negotiation too "chatty"?

Putting more information into the protocol string implies that more protocols will be created, to cover the possible space of identifiers. In turn, this brings the risk that the negotiation phase could become bloated by a mass of identifiers that can impact performance, much as HTTP content negotiation has become in some cases.

There are a few factors that should mitigate this. First, as discussed above, it's not necessary to advertise every protocol you support; only those that are applicable to the current context need to be sent.

Moreover, we expect that the protocol mechanism will be used to negotiate coarse-grained, backwards-incompatible changes to the protocol; this is one of the reasons the "http2-tls-relaxed" protocol is so loosely defined, so that future mechanisms can be easily layered upon it.

Nevertheless, the appropriate role of an ALPN protocol needs to be scrutinized to make sure we have agreement upon what's in and out of scope for its function.

Author's Address

Mark Nottingham

Email: mnot@mnot.net
URI: http://www.mnot.net/