HTTPBIS
Internet-Draft
Intended status: Standards Track
Expires: April 30, 2015

W. Chow
Mobolize
S. Mishra
Verizon Communications
J. McEachern, Ed.
ATIS

October 27, 2014

Web Proxy Description (WPD) Proxy Discovery
draft-chow-httpbis-proxy-discovery-00

Abstract

This document proposes mechanisms for applications to discover web proxy description files across different network configurations that are complementary to but not reliant upon an external network service or function, such as DHCP or DNS.

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 30, 2015.

Copyright Notice

Table of Contents

1. Terminology

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. Introduction

2.1. Scope of the document

[I-D.nottingham-web-proxy-desc] proposes a web proxy description ("WPD") format and use of well-known URI to download it, but it does not specify a mechanism to identify the authority for the URL of the WPD.

This document proposes a process to bootstrap the WPD process, and outlines mechanisms to locate the authority for the WPD file, and how to validate that authority.

The mechanisms specified in this document leverage HTTP to enable a wide variety of user agents to discover a WPD without reliance on extensions being implemented in another protocol or external network service, such as DNS or DHCP. This approach also allows the WPD to be automatically discovered for a zero configuration setup.

2.2. Use cases

The mechanisms in this document were designed to enable the following use cases:

1. Support secure and private access to a proxy on a public WiFi hotspot.

2. Support a mobile device accessing multiple networks with different proxies. The mobile device must be able to access the correct network proxy, and to change proxies as it switches between WiFi and cellular, or between operator networks.

3. Support proxies operating in different locations, possibly in a cooperative manner. The proxies could be in a private network, in a service provider network, or in the cloud, and could be limited to a specific service domain or a specific access technology.

4. Support proxies deployed on a corporate or home network.

5. Allow an app or content provider to have its own proxy.

The mechanisms proposed in this document are designed to meet the following goals:

1. Enable wide scale support across many network types, including, but not limited to, wireless, satellite, enterprise and private networks.

2. Enable applications to discover a proxy without relying on support from lower software layers or external network services, such as DNS or DHCP.

3. Simplify the process for a user (or user agent) to use a proxy, while still providing robust security.

4. Enable both automatic and manual configuration, depending on the user and network requirements.

5. Enable dynamic WPDs (i.e., different WPDs for different networks).

3. Proxy locations

The location of a proxy will influence the characteristics of a proxy and how it is used. This document considers the following proxy locations:

Internet/Cloud: Proxies that are based in the cloud can be owned and managed by 3rd parties, independent of the service provider or end user. Although these proxies are owned by 3rd parties, they can be invoked by the user, the network operator, or by application providers. Over time, the party accessing the proxy can change, depending on the configuration. This makes cloud based proxies useful for 3rd party optimization services that can be used for a range of applications, or for application specific optimization. Examples include browser-specific compression services, and Content Distribution Networks (CDN).

Service provider: Proxies can also be owned and operated by the network service provider, and can be used to manage and/or optimize traffic carried over the specific network. The proxy can be physically located in the service provider core network, the access network, or even on the customer premise. These proxies can be used to provide services such as privacy, content filtering or security services such as malware detection. Service provider proxies are also used to enhance capacity through network optimization or to increase effective network speed and improve page load times. These proxies can also be used to apply network policy and track billing, including zero rating for selected applications. Examples of service provider proxies include those deployed in the MNO packet core, ISP central office, public hotspots, or inflight WiFi.

Private: Proxies can be owned or controlled by the device owner, and deployed in a controlled network environment. These proxies can be used to provide content filtering and security services. They can also provide various services to enhance the effective speed or decrease bandwidth usage. Examples of private proxies include corporate proxies and firewalls, MiFi devices, home routers and localhost.

4. WPD Authorities

A proxy’s location determines the authority that is used to discover the WPD that describes it. The WPD authority may be distinct from the proxies described in the WPD, so the WPD Server is considered a logically distinct entity from the proxy. This document describes several possible WPD authorities, and provides a mechanism to identify and authenticate each one in the following sections.

4.1. Pre-defined authority

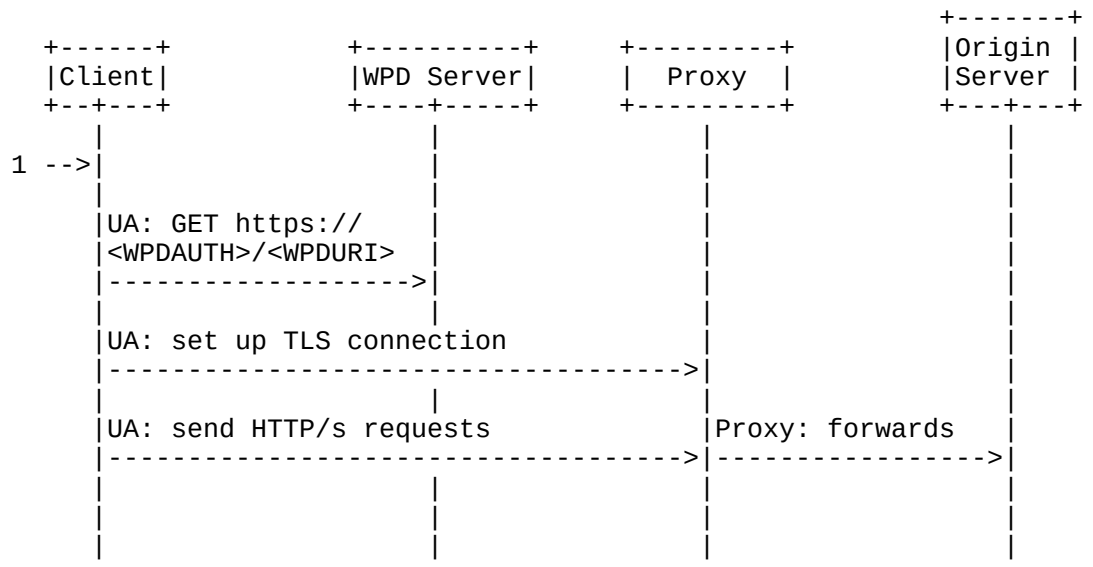
In this case, the authority is pre-configured into the device or application, and is used by the UA to probe for the appropriate proxy. This mechanism would be limited to instances where the proxy

provider is explicitly authorized to control the configuration of the device or application. Examples would include "locked" mobile phones, PCs owned by the enterprise/school, or a browser associated with a cloud-based compression service.

Mechanism: The pre-configured authority is seeded into the device or an app beforehand through an out-of-band (OOB) mechanism, so that it is available at the time that the WPD is requested. The OOB mechanism can often be static, such as being hard-coded into the UA, but this approach is complementary to dynamic mechanisms, such as those leveraging an external network service, such as a PDP context for mobile devices or LDAP for PCs.

Proxy scope: The scope of the UA determines the scope of traffic affected by the WPD. If the UA has device wide administrative privileges, it can apply the WPD configuration to all traffic on the device. If the UA is an application without administrative privileges, it can only apply the WPD configuration to the traffic for that application.

Sequence diagram:



1: OOB-defined WPD authority

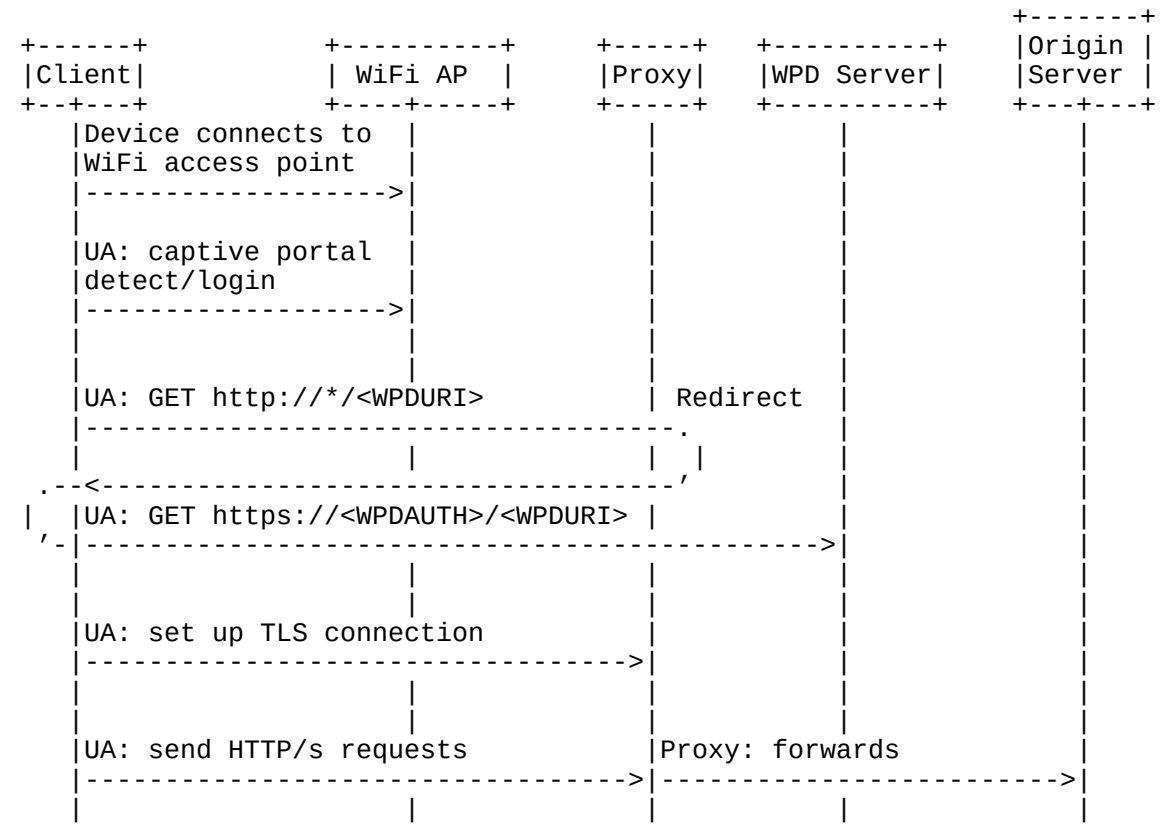
4.2. Network authority

In this case, the UA probes the network for the WPD, allowing the current network to advertise its web proxies, such as a privacy proxy on a public WiFi hotspot, optimization proxy on a MiFi device, or a content filtering proxy on a corporate network. This approach allows the proxy configuration to be customized for the current network, and is dynamically updated when the user device attaches to a different network.

Mechanism: The network authority for the WPD request is identified by the UA sending a request to the WPD URI with the "http" scheme to any authority, so that the network can respond with a redirect to the authority where its WPD can be found. The redirect MUST specify a URL with the "https" scheme, to ensure that the network authority can be securely authenticated by the UA.

Proxy scope: The scope of the WPD is limited to the current network. Proxy information can be cached, but a change in the network MUST clear the proxies it configured. The scope of the UA determines the scope of traffic affected by the WPD. If the UA has device wide administrative privileges, it can apply the WPD configuration to all traffic on the device. If the UA is an application without administrative privileges, it can only apply the WPD configuration to the traffic for that application.

Sequence diagram:



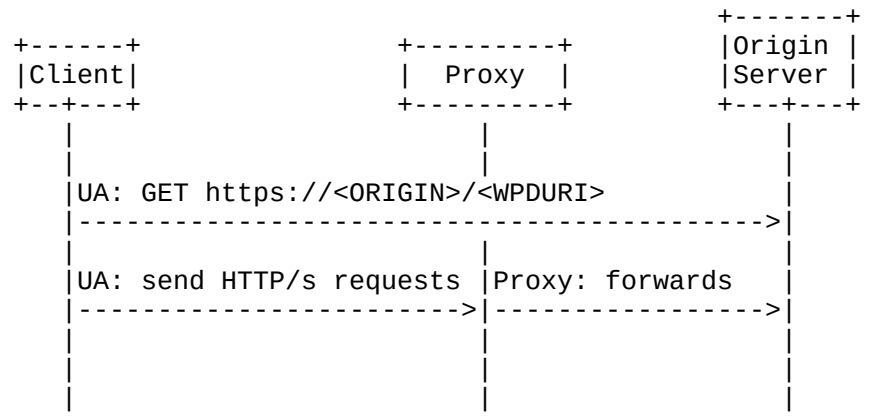
4.3. Origin authority

In this case, the UA probes for the WPD using the origin server as the authority for the WPD URI. This allows the content provider to specify a proxy for its content. The scope of the WPD from the origin server is inherently limited to the traffic to/from that origin and the UA MUST enforce this.

Mechanism: The WPD from the origin server MUST be requested using the "https" scheme, to ensure that it is coming from that origin authority. This ensures it bypasses all intermediaries and avoids overlapping with the other network authority mechanism.

Proxy scope: The scope of the WPD from the origin server MUST be limited to the traffic to that origin server.

Sequence diagram:



5. IANA Considerations

This document does not contain any considerations for IANA.

6. Security Considerations

This document specifies mechanisms to locate the authority for the WPD across different applications and network types, so it is important that these mechanisms allow the user agent to securely authenticate the WPD. Therefore, this document proposes that WPD requests SHOULD use the "https" scheme whenever possible to ensure strong authentication for the WPD server for each of the mechanisms. However, certain network scenarios may provide strong authentication of the WPD authority through other means, such as through operational security on a private network, so this document does not preclude those alternatives.

The proxy can observe unencryped traffic that traverses it, but it MUST NOT alter the end-to-end encryption for "https" requests. Also, the scope of the WPD and the traffic it affects is enforced by the user agent. In the case where the WPD is advertised by the content provider, the user agent MUST ensure that the scope of the network traffic is limited to the authority specified when requesting that WPD. This ensures that an origin server can only direct requests to proxies for its own traffic and it cannot reroute traffic for other origins.

7. Acknowledgments

The editor and authors thank Dan Druta, Vijay K. Gurbani, David Lerner, Peter Lepska, John Border and Chi-Jiun Su for feedback and suggestions.

8. Normative References

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

[I-D.nottingham-web-proxy-desc] Nottingham, M., "The Web Proxy Description Format", draft-nottingham-web-proxy-desc-01 (work in progress), October 2014.

[I-D.ietf-httpbis-http2] Belshe, M., Peon, R., and M. Thomson, "Hypertext Transfer Protocol version 2", draft-ietf-httpbis-http2-14 (work in progress), July 2014.

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

[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.

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

[RFC5785] Nottingham, M. and E. Hammer-Lahav, "Defining Well-Known Uniform Resource Identifiers (URIs)", RFC 5785, April 2010.

[RFC7301] Friedl, S., Popov, A., Langley, A., and E. Stephan, "Transport Layer Security (TLS) Application-Layer Protocol Negotiation Extension", RFC 7301, July 2014.

Authors' Addresses

William Chow
Mobolize

Email: wchow@mobolize.com

Sanjay Mishra
Verizon Communications

Email: sanjay.mishra@verizon.com

James McEachern (editor)
ATIS

Email: jmceachern@atis.org

HTTP Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 9, 2016

M. Nottingham
Akamai
P. McManus
Mozilla
J. Reschke
greenbytes

March 8, 2016

HTTP Alternative Services
draft-ietf-httpbis-alt-svc-14

Abstract

This document specifies "Alternative Services" for HTTP, which allow an origin’s resources to be authoritatively available at a separate network location, possibly accessed with a different protocol configuration.

Editorial Note (To be removed by RFC Editor)

Discussion of this draft takes place on the HTTPBIS working group mailing list (ietf-http-wg@w3.org), which is archived at <https://lists.w3.org/Archives/Public/ietf-http-wg/>.

Working Group information can be found at <http://httpwg.github.io/>; source code and issues list for this draft can be found at <https://github.com/httpwg/http-extensions>.

The changes in this draft are summarized in Appendix A.

Status of This Memo

Table of Contents

1. Introduction

HTTP [RFC7230] conflates the identification of resources with their location. In other words, "http://" and "https://" URIs are used to both name and find things to interact with.

In some cases, it is desirable to separate identification and location in HTTP; keeping the same identifier for a resource, but interacting with it at a different location on the network.

For example:

- An origin server might wish to redirect a client to a different server when it is under load, or it has found a server in a location that is more local to the client.

- An origin server might wish to offer access to its resources using a new protocol, such as HTTP/2 [RFC7540], or one using improved security, such as Transport Layer Security (TLS) [RFC5246].

- An origin server might wish to segment its clients into groups of capabilities, such as those supporting Server Name Indication (SNI) (Section 3 of [RFC6066]), for operational purposes.

This specification defines a new concept in HTTP, "Alternative Services", that allows an origin server to nominate additional means of interacting with it on the network. It defines a general framework for this in Section 2, along with specific mechanisms for advertising their existence using HTTP header fields (Section 3) or HTTP/2 frames (Section 4), plus a way to indicate that an alternative service was used (Section 5).

It also endorses the status code 421 (Misdirected Request) (Section 6) that origin servers or their nominated alternatives can use to indicate that they are not authoritative for a given origin, in cases where the wrong location is used.

1.1. 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].

This document uses the Augmented BNF defined in [RFC5234] and updated by [RFC7405] along with the "#rule" extension defined in Section 7 of [RFC7230]. The rules below are defined in [RFC5234], [RFC7230], and [RFC7234]:

```
OWS           = <OWS, see [RFC7230], Section 3.2.3>
delta-seconds = <delta-seconds; see [RFC7234], Section 1.2.1>
port          = <port, see [RFC7230], Section 2.7>
quoted-string = <quoted-string, see [RFC7230], Section 3.2.6>
token         = <token, see [RFC7230], Section 3.2.6>
uri-host      = <uri-host, see [RFC7230], Section 2.7>
```

2. Alternative Services Concepts

This specification defines a new concept in HTTP, the "Alternative Service". When an origin [RFC6454] has resources that are accessible through a different protocol / host / port combination, it is said to have an alternative service available.

An alternative service can be used to interact with the resources on an origin server at a separate location on the network, possibly using a different protocol configuration. Alternative services are considered authoritative for an origin’s resources, in the sense of [RFC7230], Section 9.1.

For example, an origin:

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

might declare that its resources are also accessible at the alternative service:

("h2", "new.example.com", "81")

By their nature, alternative services are explicitly at the granularity of an origin; they cannot be selectively applied to resources within an origin.

Alternative 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. The alternative service is essentially alternative routing information that can also be used to reach the origin in the same way that DNS CNAME or SRV records define routing information at the name resolution level. Each origin maps to a set of these routes -- the default route is derived from the origin itself and the other routes are introduced based on alternative-service information.

Furthermore, it is important to note that the first member of an alternative 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.

This means that clients using an alternative service can 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 its security context; in particular, when TLS [RFC5246] is used to authenticate, the alternative service will need to present a certificate for the origin's host name, not that of the alternative. Likewise, the Host header field ([RFC7230], Section 5.4) is still derived from the origin, not the alternative service (just as it would if a CNAME were being used).

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

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

- An Application Layer Protocol Negotiation (ALPN) protocol name, as per [RFC7301]
- A host, as per [RFC3986], Section 3.2.2
- A port, as per [RFC3986], Section 3.2.3

The ALPN protocol name is used to identify the application protocol or suite of protocols used by the alternative service. Note that for the purpose of this specification, an ALPN protocol name implicitly includes TLS in the suite of protocols it identifies, unless specified otherwise in its definition. In particular, the ALPN name "http/1.1", registered by Section 6 of [RFC7301], identifies HTTP/1.1 over TLS.

Additionally, each alternative service MUST have:

- A freshness lifetime, expressed in seconds; see Section 2.2

There are many ways that a client could discover the alternative service(s) associated with an origin. This document describes two such mechanisms: the "Alt-Svc" HTTP header field (Section 3) and the "ALTSVC" HTTP/2 frame type (Section 4).

The remainder of this section describes requirements that are common to alternative services, regardless of how they are discovered.

2.1. Host Authentication

Clients MUST have reasonable assurances that the alternative service is under control of and valid for the whole origin. This mitigates the attack described in Section 9.2.

For the purposes of this document, "reasonable assurances" can be established through use of a TLS-based protocol with the certificate checks defined in [RFC2818]. Clients MAY impose additional criteria for establishing reasonable assurances.

For example, if the origin's host is "www.example.com" and an alternative is offered on "other.example.com" with the "h2" protocol, and the certificate offered is valid for "www.example.com", the client can use the alternative. However, if either is offered with the "h2c" protocol, the client cannot use it, because there is no mechanism (at the time of the publication of this specification) in that protocol to establish the relationship between the origin and the alternative.

2.2. Alternative Service Caching

Mechanisms for discovering alternative services also associate a freshness lifetime with them; for example, the Alt-Svc header field uses the "ma" parameter.

Clients can choose to use an alternative service instead of the origin at any time when it is considered fresh; see Section 2.4 for specific recommendations.

Clients with existing connections to an alternative service do not need to stop using it when its freshness lifetime ends; the caching mechanism is intended for limiting how long an alternative service can be used for establishing new connections, not limiting the use of existing ones.

Alternative services are fully authoritative for the origin in question, including the ability to clear or update cached alternative service entries, extend freshness lifetimes, and any other authority the origin server would have.

When alternative services are used to send a client to the most optimal server, a change in network configuration can result in cached values becoming suboptimal. Therefore, clients SHOULD remove from cache all alternative services that lack the "persist" flag with the value "1" when they detect such a change, when information about network state is available.

2.3. Requiring Server Name Indication

A client MUST NOT use a TLS-based alternative service unless the client supports TLS Server Name Indication (SNI). This supports the conservation of IP addresses on the alternative service host.

Note that the SNI information provided in TLS by the client will be that of the origin, not the alternative (as will the Host HTTP header field value).

2.4. Using Alternative Services

By their nature, alternative services are OPTIONAL: clients do not need to use them. However, it is advantageous for clients to behave in a predictable way when alternative services are used by servers, to aid purposes like load balancing.

Therefore, if a client supporting this specification becomes aware of an alternative service, the client SHOULD use that alternative service for all requests to the associated origin as soon as it is available, provided the alternative service information is fresh (Section 2.2) and the security properties of the alternative service protocol are desirable, as compared to the existing connection. A viable alternative service is then treated in every way as the origin; this includes the ability to advertise alternative services.

If a client becomes aware of multiple alternative services, it chooses the most suitable according to its own criteria, keeping security properties in mind. For example, an origin might advertise multiple alternative services to notify clients of support for multiple versions of HTTP.

A client configured to use a proxy for a given request SHOULD NOT directly connect to an alternative service for this request, but instead route it through that proxy.

When a client uses an alternative service for a request, it can indicate this to the server using the Alt-Used header field (Section 5).

The client does not need to block requests on any existing connection; it can be used until the alternative connection is established. However, if the security properties of the existing connection are weak (for example, cleartext HTTP/1.1) then it might make sense to block until the new connection is fully available in order to avoid information leakage.

Furthermore, if the connection to the alternative service fails or is

unresponsive, the client MAY fall back to using the origin or another alternative service. Note, however, that this could be the basis of a downgrade attack, thus losing any enhanced security properties of the alternative service. If the connection to the alternative service does not negotiate the expected protocol (for example, ALPN fails to negotiate h2, or an Upgrade request to h2c is not accepted), the connection to the alternative service MUST be considered to have failed.

3. The Alt-Svc HTTP Header Field

An HTTP(S) origin server can advertise the availability of alternative services to clients by adding an Alt-Svc header field to responses.

```
Alt-Svc       = clear / 1#alt-value
clear         = %s"clear"; "clear", case-sensitive
alt-value     = alternative *( OWS ";" OWS parameter )
alternative   = protocol-id "=" alt-authority
protocol-id   = token ; percent-encoded ALPN protocol name
alt-authority = quoted-string ; containing [ uri-host ] ":" port
parameter     = token "=" ( token / quoted-string )
```

The field value consists either of a list of values, each of which indicates one alternative service, or the keyword "clear".

A field value containing the special value "clear" indicates that the origin requests all alternatives for that origin to be invalidated (including those specified in the same response, in case of an invalid reply containing both "clear" and alternative services).

ALPN protocol names are octet sequences with no additional constraints on format. Octets not allowed in tokens ([RFC7230], Section 3.2.6) MUST be percent-encoded as per Section 2.1 of [RFC3986]. Consequently, the octet representing the percent character "%" (hex 25) MUST be percent-encoded as well.

In order to have precisely one way to represent any ALPN protocol name, the following additional constraints apply:

1. Octets in the ALPN protocol name MUST NOT be percent-encoded if they are valid token characters except "%", and

2. When using percent-encoding, uppercase hex digits MUST be used.

With these constraints, recipients can apply simple string comparison to match protocol identifiers.

The "alt-authority" component consists of an OPTIONAL uri-host ("host" in Section 3.2.2 of [RFC3986]), a colon (":"), and a port number.

For example:

Alt-Svc: h2=":8000"

This indicates the "h2" protocol ([RFC7540]) on the same host using the indicated port 8000.

An example involving a change of host:

Alt-Svc: h2="new.example.org:80"

This indicates the "h2" protocol on the host "new.example.org", running on port 80. Note that the "quoted-string" syntax needs to be used because ":" is not an allowed character in "token".

Examples for protocol name escaping:

ALPN protocol name	protocol-id	Note
h2	h2	No escaping needed
w=x:y#z	w%3Dx%3Ay#z	"=" and ":" escaped
x%y	x%25y	"%" needs escaping

Alt-Svc MAY occur in any HTTP response message, regardless of the status code. Note that recipients of Alt-Svc can ignore the header field (and are required to in some situations; see Sections 2.1 and 6).

The Alt-Svc field value can have multiple values:

Alt-Svc: h2="alt.example.com:8000", h2=":443"

When multiple values are present, the order of the values reflects the server’s preference (with the first value being the most preferred alternative).

The value(s) advertised by Alt-Svc can be used by clients to open a new connection to an alternative service. Subsequent requests can start using this new connection immediately, or can continue using the existing connection while the new connection is created.

When using HTTP/2 ([RFC7540]), servers SHOULD instead send an ALTSVC frame (Section 4). A single ALTSVC frame can be sent for a connection; a new frame is not needed for every request. Note that, despite this recommendation, Alt-Svc header fields remain valid in responses delivered over HTTP/2.

Each "alt-value" is followed by an OPTIONAL semicolon-separated list of additional parameters, each such "parameter" comprising a name and a value.

This specification defines two parameters: "ma" and "persist", defined in Section 3.1. Unknown parameters MUST be ignored. That is, the values (alt-value) they appear in MUST be processed as if the unknown parameter was not present.

New parameters can be defined in extension specifications (see Section 7.3 for registration details).

Note that all field elements that allow "quoted-string" syntax MUST be processed as per Section 3.2.6 of [RFC7230].

3.1. Caching Alt-Svc Header Field Values

When an alternative service is advertised using Alt-Svc, it is considered fresh for 24 hours from generation of the message. This can be modified with the ’ma’ (max-age) parameter.

Syntax:

ma = delta-seconds; see [RFC7234], Section 1.2.1

The delta-seconds value indicates the number of seconds since the response was generated the alternative service is considered fresh for.

Alt-Svc: h2=":443"; ma=3600

See Section 4.2.3 of [RFC7234] for details of determining response age.

For example, a response:

```
HTTP/1.1 200 OK
Content-Type: text/html
Cache-Control: max-age=600
Age: 30
Alt-Svc: h2=":8000"; ma=60
```

indicates that an alternative service is available and usable for the next 60 seconds. However, the response has already been cached for 30 seconds (as per the Age header field value), so therefore the alternative service is only fresh for the 30 seconds from when this response was received, minus estimated transit time.

Note that the freshness lifetime for HTTP caching (here, 600 seconds) does not affect caching of Alt-Svc values.

When an Alt-Svc response header field is received from an origin, its value invalidates and replaces all cached alternative services for that origin.

By default, cached alternative services will be cleared when the client detects a network change. Alternative services that are intended to be longer-lived (such as those that are not specific to the client access network) can carry the "persist" parameter with a value "1" as a hint that the service is potentially useful beyond a network configuration change.

Syntax:

```
persist = "1"
```

For example:

```
Alt-Svc: h2=":443"; ma=2592000; persist=1
```

This specification only defines a single value for "persist". Clients MUST ignore "persist" parameters with values other than "1".

See Section 2.2 for general requirements on caching alternative services.

4. The ALTSVC HTTP/2 Frame

The ALTSVC HTTP/2 frame ([RFC7540], Section 4) advertises the availability of an alternative service to an HTTP/2 client.

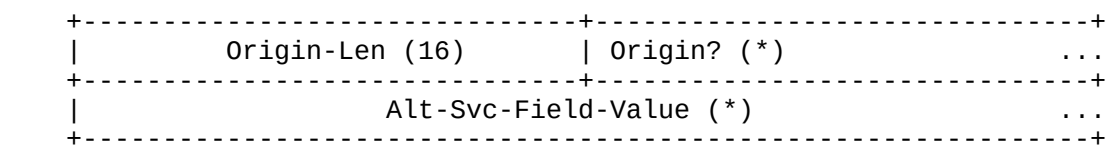
The ALTSVC frame is a non-critical extension to HTTP/2. Endpoints

that do not support this frame will ignore it (as per the extensibility rules defined in Section 4.1 of [RFC7540]).

An ALTSVC frame from a server to a client on a stream other than stream 0 indicates that the conveyed alternative service is associated with the origin of that stream.

An ALTSVC frame from a server to a client on stream 0 indicates that the conveyed alternative service is associated with the origin contained in the Origin field of the frame. An association with an origin that the client does not consider authoritative for the current connection MUST be ignored.

The ALTSVC frame type is 0xa (decimal 10).



ALTSVC Frame Payload

The ALTSVC frame contains the following fields:

Origin-Len: An unsigned, 16-bit integer indicating the length, in octets, of the Origin field.

Origin: An OPTIONAL sequence of characters containing the ASCII serialization of an origin ([RFC6454], Section 6.2) that the alternative service is applicable to.

Alt-Svc-Field-Value: A sequence of octets (length determined by subtracting the length of all preceding fields from the frame length) containing a value identical to the Alt-Svc field value defined in Section 3 (ABNF production "Alt-Svc").

The ALTSVC frame does not define any flags.

The ALTSVC frame is intended for receipt by clients. A device acting as a server MUST ignore it.

An ALTSVC frame on stream 0 with empty (length 0) "Origin" information is invalid and MUST be ignored. An ALTSVC frame on a stream other than stream 0 containing non-empty "Origin" information is invalid and MUST be ignored.

The ALTSVC frame is processed hop-by-hop. An intermediary MUST NOT

forward ALTSVC frames, though it can use the information contained in ALTSVC frames in forming new ALTSVC frames to send to its own clients.

Receiving an ALTSVC frame is semantically equivalent to receiving an Alt-Svc header field. As a result, the ALTSVC frame causes alternative services for the corresponding origin to be replaced. Note that it would be unwise to mix the use of Alt-Svc header fields with the use of ALTSVC frames, as the sequence of receipt might be hard to predict.

5. The Alt-Used HTTP Header Field

The Alt-Used header field is used in requests to indicate the identity of the alternative service in use, just as the Host header field (Section 5.4 of [RFC7230]) identifies the host and port of the origin.

```
Alt-Used     = uri-host [ ":" port ]
```

Alt-Used is intended to allow alternative services to detect loops, differentiate traffic for purposes of load balancing, and generally to ensure that it is possible to identify the intended destination of traffic, since introducing this information after a protocol is in use has proven to be problematic.

When using an alternative service, clients SHOULD include an Alt-Used header field in all requests.

For example:

```
GET /thing HTTP/1.1
Host: origin.example.com
Alt-Used: alternate.example.net
```

6. The 421 Misdirected Request HTTP Status Code

The 421 (Misdirected Request) status code is defined in Section 9.1.2 of [RFC7540] to indicate that the current server instance is not authoritative for the requested resource. This can be used to indicate that an alternative service is not authoritative; see Section 2).

Clients receiving 421 (Misdirected Request) from an alternative service MUST remove the corresponding entry from its alternative service cache (see Section 2.2) for that origin. Regardless of the idempotency of the request method, they MAY retry the request, either at another alternative server, or at the origin.

An Alt-Svc header field in a 421 (Misdirected Request) response MUST be ignored.

7. IANA Considerations

7.1. Header Field Registrations

HTTP header fields are registered within the "Message Headers" registry maintained at <https://www.iana.org/assignments/message-headers/>.

This document defines the following HTTP header fields, so their associated registry entries shall be added according to the permanent registrations below (see [BCP90]):

Header Field Name	Protocol	Status	Reference
Alt-Svc	http	standard	Section 3
Alt-Used	http	standard	Section 5

The change controller is: "IETF (iesg@ietf.org) - Internet Engineering Task Force".

7.2. The ALTSVC HTTP/2 Frame Type

This document registers the ALTSVC frame type in the HTTP/2 Frame Types registry ([RFC7540], Section 11.2).

Frame Type: ALTSVC

Code: 0xa

Specification: Section 4 of this document

7.3. Alt-Svc Parameter Registry

The HTTP Alt-Svc Parameter Registry defines the name space for parameters. It will be created and maintained at (the suggested URI) <http://www.iana.org/assignments/http-alt-svc-parameters>.

7.3.1. Procedure

A registration MUST include the following fields:

- Parameter Name

o Pointer to specification text

Values to be added to this name space require Expert Review (see [RFC5226], Section 4.1).

7.3.2. Registrations

The HTTP Alt-Svc Parameter Registry is to be populated with the registrations below:

Alt-Svc Parameter	Reference
ma	Section 3.1
persist	Section 3.1

8. Internationalization Considerations

An internationalized domain name that appears in either the header field (Section 3) or the HTTP/2 frame (Section 4) MUST be expressed using A-labels ([RFC5890], Section 2.3.2.1).

9. Security Considerations

9.1. Changing Ports

Using an alternative service implies accessing an origin's resources on an alternative port, at a minimum. An attacker that can inject alternative services and listen at the advertised port is therefore able to hijack an origin. On certain servers, it is normal for users to be able to control some personal pages available on a shared port, and also to accept to requests on less-privileged ports.

For example, an attacker that can add HTTP response header fields to some pages can redirect traffic for an entire origin to a different port on the same host using the Alt-Svc header field; if that port is under the attacker's control, they can thus masquerade as the HTTP server.

This risk is mitigated by the requirements in Section 2.1.

On servers, this risk can also be reduced by restricting the ability to advertise alternative services, and restricting who can open a port for listening on that host.

9.2. Changing Hosts

When the host is changed due to the use of an alternative service, it presents an opportunity for attackers to hijack communication to an origin.

For example, if an attacker can convince a user agent to send all traffic for "innocent.example.org" to "evil.example.com" by successfully associating it as an alternative service, they can masquerade as that origin. This can be done locally (see mitigations in Section 9.1) or remotely (e.g., by an intermediary as a man-in-the-middle attack).

This is the reason for the requirement in Section 2.1 that clients have reasonable assurances that the alternative service is under control of and valid for the whole origin; for example, presenting a certificate for the origin proves that the alternative service is authorized to serve traffic for the origin.

Note that this assurance is only as strong as the method used to authenticate the alternative service. In particular, when TLS authentication is used to do so, there are well-known exploits to make an attacker’s certificate appear as legitimate.

Alternative services could be used to persist such an attack. For example, an intermediary could man-in-the-middle TLS-protected communication to a target, and then direct all traffic to an alternative service with a large freshness lifetime, so that the user agent still directs traffic to the attacker even when not using the intermediary.

Implementations MUST perform any certificate-pinning validation (such as [RFC7469]) on alternative services just as they would on direct connections to the origin. Implementations might also choose to add other requirements around which certificates are acceptable for alternative services.

9.3. Changing Protocols

When the ALPN protocol is changed due to the use of an alternative service, the security properties of the new connection to the origin can be different from that of the "normal" connection to the origin, because the protocol identifier itself implies this.

For example, if an "https://" URI has a protocol advertised that does not use some form of end-to-end encryption (most likely, TLS), it violates the expectations for security that the URI scheme implies. Therefore, clients cannot blindly use alternative services, but

instead evaluate the option(s) presented to assure that security requirements and expectations of specifications, implementations and end users are met.

9.4. Tracking Clients Using Alternative Services

Choosing an alternative service implies connecting to a new, server-supplied host name. By using unique names, servers could conceivably track client requests. Such tracking could follow users across multiple networks, when the "persist" flag is used.

Clients that wish to prevent requests from being correlated can decide not to use alternative services for multiple requests that would not otherwise be allowed to be correlated.

In a user agent, any alternative service information MUST be removed when origin-specific data is cleared (typically, when cookies [RFC6265] are cleared).

9.5. Confusion Regarding Request Scheme

Some server-side HTTP applications make assumptions about security based upon connection context; for example, equating being served upon port 443 with the use of an "https://" URI and the various security properties that implies.

This affects not only the security properties of the connection itself, but also the state of the client at the other end of it; for example, a Web browser treats "https://" URIs differently than "http://" URIs in many ways, not just for purposes of protocol handling.

Since one of the uses of Alternative Services is to allow a connection to be migrated to a different protocol and port, these applications can become confused about the security properties of a given connection, sending information (for example, cookies and content) that is intended for a secure context (such as an "https://" URI) to a client that is not treating it as one.

This risk can be mitigated in servers by using the URI scheme explicitly carried by the protocol (such as ":scheme" in HTTP/2 or the "absolute form" of the request target in HTTP/1.1) as an indication of security context, instead of other connection properties ([RFC7540], Section 8.1.2.3 and [RFC7230], Section 5.3.2).

When the protocol does not explicitly carry the scheme (as is usually the case for HTTP/1.1 over TLS), servers can mitigate this risk by either assuming that all requests have an insecure context, or by

refraining from advertising alternative services for insecure schemes (for example, HTTP).

10. References

10.1. Normative References

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

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, DOI 10.17487/RFC2818, May 2000, <http://www.rfc-editor.org/info/rfc2818>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <http://www.rfc-editor.org/info/rfc3986>.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, DOI 10.17487/RFC5226, May 2008, <http://www.rfc-editor.org/info/rfc5226>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <http://www.rfc-editor.org/info/rfc5234>.

[RFC5890] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, DOI 10.17487/RFC5890, August 2010, <http://www.rfc-editor.org/info/rfc5890>.

[RFC6066] Eastlake, D., "Transport Layer Security (TLS) Extensions: Extension Definitions", RFC 6066, DOI 10.17487/RFC6066, January 2011, <http://www.rfc-editor.org/info/rfc6066>.

[RFC6454] Barth, A., "The Web Origin Concept", RFC 6454, DOI 10.17487/RFC6454, December 2011, <http://www.rfc-editor.org/info/rfc6454>.

[RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <http://www.rfc-editor.org/info/rfc7230>.

[RFC7234] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Caching", RFC 7234, DOI 10.17487/RFC7234, June 2014, <http://www.rfc-editor.org/info/rfc7234>.

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

[RFC7405] Kyzivat, P., "Case-Sensitive String Support in ABNF", RFC 7405, DOI 10.17487/RFC7405, December 2014, <http://www.rfc-editor.org/info/rfc7405>.

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

10.2. Informative References

[BCP90] Klyne, G., Nottingham, M., and J. Mogul, "Registration Procedures for Message Header Fields", BCP 90, RFC 3864, September 2004, <http://www.rfc-editor.org/info/bcp90>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <http://www.rfc-editor.org/info/rfc5246>.

[RFC6265] Barth, A., "HTTP State Management Mechanism", RFC 6265, DOI 10.17487/RFC6265, April 2011, <http://www.rfc-editor.org/info/rfc6265>.

[RFC7469] Evans, C., Palmer, C., and R. Sleevi, "Public Key Pinning Extension for HTTP", RFC 7469, DOI 10.17487/RFC7469, April 2015, <http://www.rfc-editor.org/info/rfc7469>.

Appendix A. Change Log (to be removed by RFC Editor before publication)

A.1. Since draft-nottingham-httpbis-alt-svc-05

This is the first version after adoption of draft-nottingham-httpbis-alt-svc-05 as Working Group work item. It only contains editorial changes.

A.2. Since draft-ietf-httpbis-alt-svc-00

Selected 421 as proposed status code for "Not Authoritative".

Changed header field syntax to use percent-encoding of ALPN protocol names (<https://github.com/http2/http2-spec/issues/446>).

A.3. Since draft-ietf-httpbis-alt-svc-01

Updated HTTP/1.1 references.

Renamed "Service" to "Alt-Svc-Used" and reduced information to a flag to address fingerprinting concerns (<https://github.com/http2/http2-spec/issues/502>).

Note that ALTSVC frame is preferred to Alt-Svc header field (<https://github.com/http2/http2-spec/pull/503>).

Incorporate ALTSRV frame (<https://github.com/http2/http2-spec/pull/507>).

Moved definition of status code 421 to HTTP/2.

Partly resolved <https://github.com/httpwg/http-extensions/issues/5>.

A.4. Since draft-ietf-httpbis-alt-svc-02

Updated ALPN reference.

Resolved <https://github.com/httpwg/http-extensions/issues/2>.

A.5. Since draft-ietf-httpbis-alt-svc-03

Renamed "Alt-Svc-Used" to "Alt-Used" (<https://github.com/httpwg/http-extensions/issues/17>).

Clarify ALTSVC Origin information requirements (<https://github.com/httpwg/http-extensions/issues/19>).

Remove/tune language with respect to tracking risks (see <https://github.com/httpwg/http-extensions/issues/34>).

A.6. Since draft-ietf-httpbis-alt-svc-04

Mention tracking by alt-svc host name in Security Considerations (<https://github.com/httpwg/http-extensions/issues/36>).

"421 (Not Authoritative)" -> "421 (Misdirected Request)".

Allow the frame to carry multiple indicator and use the same payload formats for both (<https://github.com/httpwg/http-extensions/issues/37>).

A.7. Since draft-ietf-httpbis-alt-svc-05

Go back to specifying the origin in Alt-Used, but make it a "SHOULD" (<https://github.com/httpwg/http-extensions/issues/34>).

Restore Origin field in ALT-SVC frame (<https://github.com/httpwg/http-extensions/issues/38>).

A.8. Since draft-ietf-httpbis-alt-svc-06

Disallow use of alternative services when the protocol might not carry the scheme (<https://github.com/httpwg/http-extensions/issues/12>).

Align opp-sec and alt-svc (<https://github.com/httpwg/http-extensions/issues/33>).

alt svc frame on pushed (even and non-0) frame (<https://github.com/httpwg/http-extensions/issues/44>).

"browser" -> "user agent" (<https://github.com/httpwg/http-extensions/pull/61>).

ABNF for "parameter" (<https://github.com/httpwg/http-extensions/issues/65>).

Updated HTTP/2 reference.

A.9. Since draft-ietf-httpbis-alt-svc-07

Alt-Svc alternative cache invalidation (<https://github.com/httpwg/http-extensions/issues/16>).

Unexpected Alt-Svc frames (<https://github.com/httpwg/http-extensions/issues/18>).

Associating Alt-Svc header with an origin (<https://github.com/httpwg/http-extensions/issues/21>).

ALPN identifiers in Alt-Svc (<https://github.com/httpwg/http-extensions/issues/43>).

Number of alternate services used (<https://github.com/httpwg/http-extensions/issues/58>).

Proxy and .pac interaction
(<https://github.com/httpwg/http-extensions/issues/62>).

Need to define extensibility for alt-svc parameters
(<https://github.com/httpwg/http-extensions/issues/69>).

Persistence of alternates across network changes
(<https://github.com/httpwg/http-extensions/issues/71>).

Alt-Svc header with 421 status
(<https://github.com/httpwg/http-extensions/issues/75>).

Incorporate several editorial improvements suggested by Mike Bishop
(<https://github.com/httpwg/http-extensions/pull/77>,
<https://github.com/httpwg/http-extensions/pull/78>).

Alt-Svc response header field in HTTP/2 frame
(<https://github.com/httpwg/http-extensions/issues/87>).

A.10. Since draft-ietf-httpbis-alt-svc-08

Remove left over text about ext-params, applying to an earlier
version of Alt-Used (see
<https://github.com/httpwg/http-extensions/issues/34>).

Conflicts between Alt-Svc and ALPN
(<https://github.com/httpwg/http-extensions/issues/72>).

Elevation of privilege
(<https://github.com/httpwg/http-extensions/issues/73>).

Alternates of alternates
(<https://github.com/httpwg/http-extensions/issues/74>).

Alt-Svc and Cert Pinning
(<https://github.com/httpwg/http-extensions/issues/76>).

Using alt-svc on localhost (no change to spec, see
<https://github.com/httpwg/http-extensions/issues/89>).

IANA procedure for alt-svc parameters
(<https://github.com/httpwg/http-extensions/issues/96>).

Alt-svc from https (1.1) to https (1.1)
(<https://github.com/httpwg/http-extensions/issues/91>).

Alt-svc vs the ability to convey the scheme inside the protocol
(<https://github.com/httpwg/http-extensions/issues/92>).

Reconciling MAY/can vs. SHOULD
(<https://github.com/httpwg/http-extensions/issues/101>).

Typo in alt-svc caching example
(<https://github.com/httpwg/http-extensions/issues/117>).

A.11. Since draft-ietf-httpbis-alt-svc-09

Editorial improvements
(<https://github.com/httpwg/http-extensions/issues/118>,
<https://github.com/httpwg/http-extensions/issues/119>,
<https://github.com/httpwg/http-extensions/issues/120>,
<https://github.com/httpwg/http-extensions/issues/121>,
<https://github.com/httpwg/http-extensions/issues/122>,
<https://github.com/httpwg/http-extensions/issues/123>,
<https://github.com/httpwg/http-extensions/issues/125>,
<https://github.com/httpwg/http-extensions/issues/126>).

A.12. Since draft-ietf-httpbis-alt-svc-10

Editorial improvements
(<https://github.com/httpwg/http-extensions/issues/130>).

Use RFC 7405 ABNF extension
(<https://github.com/httpwg/http-extensions/issues/131>).

A.13. Since draft-ietf-httpbis-alt-svc-11

Security considerations wrt system ports
(<https://github.com/httpwg/http-extensions/issues/139>).

A.14. Since draft-ietf-httpbis-alt-svc-12

Editorial changes triggered by <https://lists.w3.org/Archives/Public/ietf-http-wg/2016JanMar/0243.html>.

Reasonable Assurances and H2C
(<https://github.com/httpwg/http-extensions/issues/148>).

Appendix B. Acknowledgements

Thanks to Adam Langley, Bence Beky, Chris Lonvick, Eliot Lear, Erik Nygren, Guy Podjarny, Herve Ruellan, Lucas Pardue, Martin Thomson, Matthew Kerwin, Mike Bishop, Paul Hoffman, Richard Barnes, Richard Bradbury, Stephen Farrell, Stephen Ludin, and Will Chan for their feedback and suggestions.

The Alt-Svc header field was influenced by the design of the

Alternate-Protocol header field in SPDY.

Authors' Addresses

Mark Nottingham
Akamai

EMail: mnot@mnot.net
URI: https://www.mnot.net/

Patrick McManus
Mozilla

EMail: mcmanus@ducksong.com
URI: https://mozillians.org/u/pmcmanus/

Julian F. Reschke
greenbytes GmbH

EMail: julian.reschke@greenbytes.de
URI: https://greenbytes.de/tech/webdav/

HTTP Working Group
Internet-Draft
Intended status: Experimental
Expires: September 18, 2017

M. Nottingham

M. Thomson
Mozilla

March 17, 2017

Opportunistic Security for HTTP/2
draft-ietf-httpbis-http2-encryption-11

Abstract

This document describes how "http" URIs can be accessed using Transport Layer Security (TLS) and HTTP/2 to mitigate pervasive monitoring attacks. This mechanism not a replacement for "https" URIs; it is vulnerable to active attacks.

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 September 18, 2017.

Table of Contents

1. Introduction

This document describes a use of HTTP Alternative Services [RFC7838] to decouple the URI scheme from the use and configuration of underlying encryption. It allows an "http" URI to be accessed using HTTP/2 [RFC7230] and Transport Layer Security (TLS) [RFC5246] with Opportunistic Security [RFC7435].

This document describes a usage model whereby sites can serve "http" URIs over TLS, thereby avoiding the problem of serving Mixed Content (described in [W3C.CR-mixed-content-20160802]) while still providing protection against passive attacks.

Opportunistic Security does not provide the same guarantees as using TLS with "https" URIs, because it is vulnerable to active attacks, and does not change the security context of the connection. Normally, users will not be able to tell that it is in use (i.e., there will be no "lock icon").

1.1. Goals and Non-Goals

The immediate goal is to make the use of HTTP more robust in the face of pervasive passive monitoring [RFC7258].

A secondary (but significant) goal is to provide for ease of implementation, deployment and operation. This mechanism is expected

to have a minimal impact upon performance, and require a trivial administrative effort to configure.

Preventing active attacks (such as a Man-in-the-Middle) is a non-goal for this specification. Furthermore, this specification is not intended to replace or offer an alternative to "https", since "https" both prevents active attacks and invokes a more stringent security model in most clients.

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. Using HTTP URIs over TLS

An origin server that supports the resolution of "http" URIs can indicate support for this specification by providing an alternative service advertisement [RFC7838] for a protocol identifier that uses TLS, such as "h2" [RFC7540]. Such a protocol MUST include an explicit indication of the scheme of the resource. This excludes HTTP/1.1; HTTP/1.1 clients are forbidden from including the absolute form of a URI in requests to origin servers (see Section 5.3.1 of [RFC7230]).

A client that receives such an advertisement MAY make future requests intended for the associated origin [RFC6454] to the identified service (as specified by [RFC7838]), provided that the alternative service opts in as described in Section 2.1.

A client that places the importance of protection against passive attacks over performance might choose to withhold requests until an encrypted connection is available. However, if such a connection cannot be successfully established, the client can resume its use of the cleartext connection.

A client can also explicitly probe for an alternative service advertisement by sending a request that bears little or no sensitive information, such as one with the OPTIONS method. Likewise, clients with existing alternative services information could make such a request before they expire, in order minimize the delays that might be incurred.

Client certificates are not meaningful for URLs with the "http" scheme, and therefore clients creating new TLS connections to alternative services for the purposes of this specification MUST NOT present them. A server that also provides "https" resources on the

same port can request a certificate during the TLS handshake, but it MUST NOT abort the handshake if the client does not provide one.

2.1. Alternative Server Opt-In

It is possible that the server might become confused about whether requests' URLs have a "http" or "https" scheme, for various reasons; see Section 4.4. To ensure that the alternative service has opted into serving "http" URLs over TLS, clients are required to perform additional checks before directing "http" requests to it.

Clients MUST NOT send "http" requests over a secured connection, unless the chosen alternative service presents a certificate that is valid for the origin as defined in [RFC2818]. Using an authenticated alternative service establishes "reasonable assurances" for the purposes of [RFC7838]. In addition to authenticating the server, the client MUST have obtained a valid http-opportunistic response for an origin (as per Section 2.3) using the authenticated connection. An exception to the latter restriction is made for requests for the "http-opportunistic" well-known URI.

For example, assuming the following request is made over a TLS connection that is successfully authenticated for those origins, the following request/response pair would allow requests for the origins "http://www.example.com" or "http://example.com" to be sent using a secured connection:

```
HEADERS
  + END_STREAM
  + END_HEADERS
    :method = GET
    :scheme = http
    :authority = example.com
    :path = /.well-known/http-opportunistic

HEADERS
    :status = 200
    content-type = application/json
DATA
  + END_STREAM
[ "http://www.example.com", "http://example.com" ]
```

Though this document describes multiple origins, this is only for operational convenience. Only a request made to an origin (over an authenticated connection) can be used to acquire this resource for that origin. Thus in the example, the request to "http://example.com" cannot be assumed to also provide an http-opportunistic response for "http://www.example.com".

2.2. Interaction with "https" URIs

Clients MUST NOT send "http" requests and "https" requests on the same connection. Similarly, clients MUST NOT send "http" requests for multiple origins on the same connection.

2.3. The "http-opportunistic" well-known URI

This specification defines the "http-opportunistic" well-known URI [RFC5785]. A client is said to have a valid http-opportunistic response for a given origin when:

- The client has requested the well-known URI from the origin over an authenticated connection and a 200 (OK) response was provided, and
- That response is fresh [RFC7234] (potentially through revalidation [RFC7232]), and
- That response has the media type "application/json", and
- That response's payload, when parsed as JSON [RFC7159], contains an array as the root, and
- The array contains a string that is a case-insensitive character-for-character match for the origin in question, serialised into Unicode as per Section 6.1 of [RFC6454].

A client MAY treat an "http-opportunistic" resource as invalid if values it contains are not strings.

This document does not define semantics for "http-opportunistic" resources on an "https" origin, nor does it define semantics if the resource includes "https" origins.

Allowing clients to cache the http-opportunistic resource means that all alternative services need to be able to respond to requests for "http" resources. A client is permitted to use an alternative service without acquiring the http-opportunistic resource from that service.

A client MUST NOT use any cached copies of an http-opportunistic resource that was acquired (or revalidated) over an unauthenticated connection. To avoid potential errors, a client can request or revalidate the http-opportunistic resource before using any connection to an alternative service.

Clients that use cached http-opportunistic responses MUST ensure that their cache is cleared of any responses that were acquired over an unauthenticated connection. Revalidating an unauthenticated response using an authenticated connection does not ensure the integrity of the response.

3. IANA Considerations

This specification registers a Well-Known URI [RFC5785]:

- URI Suffix: http-opportunistic
- Change Controller: IETF
- Specification Document(s): Section 2.3 of [this specification]
- Related Information:

4. Security Considerations

4.1. Security Indicators

User Agents MUST NOT provide any special security indicators when an "http" resource is acquired using TLS. In particular, indicators that might suggest the same level of security as "https" MUST NOT be used (e.g., a "lock device").

4.2. Downgrade Attacks

A downgrade attack against the negotiation for TLS is possible.

For example, because the "Alt-Svc" header field [RFC7838] likely appears in an unauthenticated and unencrypted channel, it is subject to downgrade by network attackers. In its simplest form, an attacker that wants the connection to remain in the clear need only strip the "Alt-Svc" header field from responses.

4.3. Privacy Considerations

Cached alternative services can be used to track clients over time; e.g., using a user-specific hostname. Clearing the cache reduces the ability of servers to track clients; therefore clients MUST clear cached alternative service information when clearing other origin-based state (i.e., cookies).

4.4. Confusion Regarding Request Scheme

HTTP implementations and applications sometimes use ambient signals to determine if a request is for an "https" resource; for example, they might look for TLS on the stack, or a server port number of 443.

This might be due to expected limitations in the protocol (the most common HTTP/1.1 request form does not carry an explicit indication of the URI scheme and the resource might have been developed assuming HTTP/1.1), or it may be because how the server and application are implemented (often, they are two separate entities, with a variety of possible interfaces between them).

Any security decisions based upon this information could be misled by the deployment of this specification, because it violates the assumption that the use of TLS (or port 443) means that the client is accessing a HTTPS URI, and operating in the security context implied by HTTPS.

Therefore, server implementers and administrators need to carefully examine the use of such signals before deploying this specification.

4.5. Server Controls

This specification requires that a server send both an Alternative Service advertisement and host content in a well-known location to send HTTP requests over TLS. Servers SHOULD take suitable measures to ensure that the content of the well-known resource remains under their control. Likewise, because the Alt-Svc header field is used to describe policies across an entire origin, servers SHOULD NOT permit user content to set or modify the value of this header.

5. References

5.1. Normative References

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

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, DOI 10.17487/RFC2818, May 2000, <http://www.rfc-editor.org/info/rfc2818>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <http://www.rfc-editor.org/info/rfc5246>.

[RFC5785] Nottingham, M. and E. Hammer-Lahav, "Defining Well-Known Uniform Resource Identifiers (URIs)", RFC 5785, DOI 10.17487/RFC5785, April 2010, <http://www.rfc-editor.org/info/rfc5785>.

[RFC6454] Barth, A., "The Web Origin Concept", RFC 6454, DOI 10.17487/RFC6454, December 2011, <http://www.rfc-editor.org/info/rfc6454>.

[RFC7159] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, DOI 10.17487/RFC7159, March 2014, <http://www.rfc-editor.org/info/rfc7159>.

[RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <http://www.rfc-editor.org/info/rfc7230>.

[RFC7232] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Conditional Requests", RFC 7232, DOI 10.17487/RFC7232, June 2014, <http://www.rfc-editor.org/info/rfc7232>.

[RFC7234] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Caching", RFC 7234, DOI 10.17487/RFC7234, June 2014, <http://www.rfc-editor.org/info/rfc7234>.

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

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

5.2. Informative References

[RFC7258] Farrell, S. and H. Tschofenig, "Pervasive Monitoring Is an Attack", BCP 188, RFC 7258, DOI 10.17487/RFC7258, May 2014, <http://www.rfc-editor.org/info/rfc7258>.

[RFC7435] Dukhovni, V., "Opportunistic Security: Some Protection Most of the Time", RFC 7435, DOI 10.17487/RFC7435, December 2014, <http://www.rfc-editor.org/info/rfc7435>.

[RFC7469] Evans, C., Palmer, C., and R. Sleevi, "Public Key Pinning Extension for HTTP", RFC 7469, DOI 10.17487/RFC7469, April 2015, <http://www.rfc-editor.org/info/rfc7469>.

[W3C.CR-mixed-content-20160802] West, M., "Mixed Content", World Wide Web Consortium CR CR-mixed-content-20160802, August 2016, <https://www.w3.org/TR/2016/CR-mixed-content-20160802>.

Appendix A. Acknowledgements

Mike Bishop contributed significant text to this document.

Thanks to Patrick McManus, Stefan Eissing, Eliot Lear, Stephen Farrell, Guy Podjarny, Stephen Ludin, Erik Nygren, Paul Hoffman, Adam Langley, Eric Rescorla, Julian Reschke, Kari Hurtta, and Richard Barnes for their feedback and suggestions.

Authors' Addresses

Mark Nottingham

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

Martin Thomson
Mozilla

Email: martin.thomson@gmail.com

HTTPbis Working Group
Internet-Draft
Obsoletes: 7238 (if approved)
Intended status: Standards Track
Expires: August 9, 2015

J. Reschke
greenbytes

February 5, 2015

The Hypertext Transfer Protocol Status Code 308 (Permanent Redirect) draft-ietf-httpbis-rfc7238bis-03

Abstract

This document specifies the additional Hypertext Transfer Protocol (HTTP) status code 308 (Permanent Redirect).

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 August 9, 2015.

Copyright Notice

Table of Contents

1. Introduction

HTTP defines a set of status codes for the purpose of redirecting a request to a different URI ([RFC3986]). The history of these status codes is summarized in Section 6.4 of [RFC7231], which also classifies the existing status codes into four categories.

The first of these categories contains the status codes 301 (Moved Permanently), 302 (Found), and 307 (Temporary Redirect), which can be classified as below:

	Permanent	Temporary
Allows changing the request method from POST to GET	301	302
Does not allow changing the request method from POST to GET	-	307

Section 6.4.7 of [RFC7231] states that it does not define a permanent variant of status code 307; this specification adds the status code 308, defining this missing variant (Section 3).

This specification contains no technical changes from the experimental RFC 7238, which it obsoletes.

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].

3. 308 Permanent Redirect

The 308 (Permanent Redirect) status code indicates that the target resource has been assigned a new permanent URI and any future references to this resource ought to use one of the enclosed URIs. Clients with link editing capabilities ought to automatically re-link references to the effective request URI (Section 5.5 of [RFC7230]) to one or more of the new references sent by the server, where possible.

The server SHOULD generate a Location header field ([RFC7231], Section 7.1.2) in the response containing a preferred URI reference for the new permanent URI. The user agent MAY use the Location field value for automatic redirection. The server’s response payload usually contains a short hypertext note with a hyperlink to the new URI(s).

A 308 response is cacheable by default; i.e., unless otherwise indicated by the method definition or explicit cache controls (see [RFC7234], Section 4.2.2).

Note: This status code is similar to 301 (Moved Permanently) ([RFC7231], Section 6.4.2), except that it does not allow changing the request method from POST to GET.

4. Deployment Considerations

Section 6 of [RFC7231] requires recipients to treat unknown 3xx status codes the same way as status code 300 Multiple Choices ([RFC7231], Section 6.4.1). Thus, servers will not be able to rely on automatic redirection happening similar to status codes 301, 302, or 307.

Therefore, the use of status code 308 is restricted to cases where the server has sufficient confidence in the client's understanding the new code or when a fallback to the semantics of status code 300 is not problematic. Server implementers are advised not to vary the status code based on characteristics of the request, such as the User-Agent header field ("User-Agent Sniffing") -- doing so usually results in code that is both hard to maintain and hard to debug and would also require special attention to caching (i.e., setting a "Vary" response header field, as defined in Section 7.1.4 of [RFC7231]).

Note that many existing HTML-based user agents will emulate a refresh when encountering an HTML <meta> refresh directive ([HTML], Section 4.2.5.3). This can be used as another fallback. For example:

Client request:

```
GET / HTTP/1.1
Host: example.com
```

Server response:

```
HTTP/1.1 308 Permanent Redirect
Content-Type: text/html; charset=UTF-8
Location: http://example.com/new
Content-Length: 356

<!DOCTYPE HTML>
<html>
   <head>
      <title>Permanent Redirect</title>
      <meta http-equiv="refresh"
            content="0; url=http://example.com/new">
   </head>
   <body>
      <p>
         The document has been moved to
         <a href="http://example.com/new"
         >http://example.com/new</a>.
      </p>
   </body>
</html>
```

5. Security Considerations

All security considerations that apply to HTTP redirects apply to the 308 status code as well (see Section 9 of [RFC7231]).

Unsecured communication over the Internet is subject to man in the middle modification of messages, including changing status codes or redirect targets. Use of TLS is one way to mitigate those attacks. See Section 9 of [RFC7230] for related attacks on authority and message integrity.

6. IANA Considerations

The "Hypertext Transfer Protocol (HTTP) Status Code Registry" (defined in Section 8.2 of [RFC7231] and located at <http://www.iana.org/assignments/http-status-codes>) needs to be updated with the registration below:

Value	Description	Reference
308	Permanent Redirect	Section 3 of this specification

7. Acknowledgements

The definition for the new status code 308 reuses text from the HTTP/1.1 definitions of status codes 301 and 307.

Furthermore, thanks to Ben Campbell, Cyrus Daboo, Adrian Farrell, Eran Hammer-Lahav, Bjoern Hoehrmann, Barry Leiba, Subramanian Moonesamy, Kathleen Moriarty, Peter Saint-Andre, Robert Sparks, and Roy Fielding for feedback on this document.

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.

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

[RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, June 2014.

[RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, June 2014.

[RFC7234] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Caching", RFC 7234, June 2014.

8.2. Informative References

[HTML] Hickson, I., Berjon, R., Faulkner, S., Leithead, T., Doyle Navara, E., O'Connor, E., and S. Pfeiffer, "HTML5", W3C Recommendation REC-html5-20141028, October 2014, <http://www.w3.org/TR/2014/REC-html5-20141028/>.

Latest version available at <http://www.w3.org/TR/html5/>.

Author's Address

Julian F. Reschke
greenbytes GmbH
Hafenweg 16
Muenster, NW 48155
Germany

EMail: julian.reschke@greenbytes.de
URI: http://greenbytes.de/tech/webdav/

HTTP Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 13, 2015

A. Hutton
Unify
J. Uberti
Google
M. Thomson
Mozilla

June 11, 2015

The ALPN HTTP Header Field
draft-ietf-httpbis-tunnel-protocol-05

Abstract

This specification allows HTTP CONNECT requests to indicate what protocol is intended to be used within the tunnel once established, using the ALPN header field.

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 December 13, 2015.

Table of Contents

1. Introduction

The HTTP CONNECT method (Section 4.3.6 of [RFC7231]) requests that the recipient establish a tunnel to the identified origin server and thereafter forward packets, in both directions, until the tunnel is closed. Such tunnels are commonly used to create end-to-end virtual connections, through one or more proxies.

The HTTP ALPN header field identifies the protocol or protocols that the client intends to use within a tunnel that is established using CONNECT. This uses the Application Layer Protocol Negotiation identifier (ALPN, [RFC7301]).

For a tunnel that is then secured using TLS [RFC5246], the header field carries the same application protocol label as will be carried within the TLS handshake [RFC7301]. If there are multiple possible application protocols, all of those application protocols are indicated.

The ALPN header field carries an indication of client intent only. An ALPN identifier is used here only to identify the application protocol or suite of protocols that the client intends to use in the tunnel. No negotiation takes place using this header field. In TLS, the final choice of application protocol is made by the server from the set of choices presented by the client. Other substrates could negotiate the application protocol differently.

Proxies do not implement the tunneled protocol, though they might choose to make policy decisions based on the value of the header

field. For example, a proxy could use the application protocol to select appropriate traffic prioritization.

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].

2. The ALPN HTTP Header Field

Clients include the ALPN header field in an HTTP CONNECT request to indicate the application layer protocol that a client intends to use within the tunnel, or a set of protocols that might be used within the tunnel.

2.1. Header Field Values

Valid values for the protocol field are taken from the "Application-Layer Protocol Negotiation (ALPN) Protocol ID" registry ([1]) established by [RFC7301].

2.2. Syntax

The ABNF (Augmented Backus-Naur Form) syntax for the ALPN header field value is given below. It uses the syntax defined in Section 1.2 of [RFC7230].

```
ALPN = 1#protocol-id
protocol-id     = token ; percent-encoded ALPN protocol identifier
```

ALPN protocol names are octet sequences with no additional constraints on format. Octets not allowed in tokens ([RFC7230], Section 3.2.6) MUST be percent-encoded as per Section 2.1 of [RFC3986]. Consequently, the octet representing the percent character "%" (hex 25) MUST be percent-encoded as well.

In order to have precisely one way to represent any ALPN protocol name, the following additional constraints apply:

- Octets in the ALPN protocol MUST NOT be percent-encoded if they are valid token characters except "%", and
- When using percent-encoding, uppercase hex digits MUST be used.

With these constraints, recipients can apply simple string comparison to match protocol identifiers.

For example:

```
CONNECT www.example.com HTTP/1.1
Host: www.example.com
ALPN: h2, http%2F1.1
```

2.3. Usage

When used in the ALPN header field, an ALPN identifier is used to identify an entire application protocol stack, not a single protocol layer or component.

For a CONNECT tunnel that conveys a protocol secured with TLS, the value of the ALPN header field contains the same list of ALPN identifiers that will be sent in the TLS ClientHello message [RFC7301].

Where no protocol negotiation is expected to occur, such as in protocols that do not use TLS, the ALPN header field contains a single ALPN Protocol Identifier corresponding to the application protocol that is intended to be used. If an alternative form of protocol negotiation is possible, the ALPN header field contains the set of protocols that might be negotiated.

A proxy can use the value of the ALPN header field to more cleanly and efficiently reject requests for a CONNECT tunnel. Exposing protocol information at the HTTP layer allows a proxy to deny requests earlier, with better error reporting (such as a 403 status code). The ALPN header field can be falsified and is therefore not sufficient basis for authorizing a request.

A proxy could attempt to inspect packets to determine the protocol in use. This requires that the proxy understand each ALPN identifier. Protocols like TLS could hide negotiated protocols, or protocol negotiation details could change over time. Proxies SHOULD NOT break a CONNECT tunnel solely on the basis of a failure to recognize the protocol.

A proxy can use the ALPN header field value to change how it manages or prioritizes connections.

3. IANA Considerations

HTTP header fields are registered within the "Permanent Message Header Field Names" registry maintained at [2]. This document defines and registers the ALPN header field, according to [RFC3864] as follows:

Header Field Name: ALPN

Protocol: http

Status: Standard

Reference: Section 2

Change Controller: IETF (iesg@ietf.org) - Internet Engineering Task Force

4. Security Considerations

In case of using HTTP CONNECT to a TURN server ("Traversal Using Relays around NAT", [RFC5766]) the security considerations of Section 4.3.6 of [RFC7231] apply. It states that there "are significant risks in establishing a tunnel to arbitrary servers, particularly when the destination is a well-known or reserved TCP port that is not intended for Web traffic. Proxies that support CONNECT SHOULD restrict its use to a limited set of known ports or a configurable whitelist of safe request targets."

The ALPN header field described in this document is OPTIONAL. Clients and HTTP proxies could choose to not support it and therefore either fail to provide it, or ignore it when present. If the header field is not available or ignored, a proxy cannot identify the purpose of the tunnel and use this as input to any authorization decision regarding the tunnel. This is indistinguishable from the case where either client or proxy does not support the ALPN header field.

There is no confidentiality protection for the ALPN header field. ALPN identifiers that might expose confidential or sensitive information SHOULD NOT be sent, as described in Section 5 of [RFC7301].

The value of the ALPN header field could be falsified by a client. If the data being sent through the tunnel is encrypted (for example, with TLS [RFC5246]), then the proxy might not be able to directly inspect the data to verify that the claimed protocol is the one which is actually being used, though a proxy might be able to perform traffic analysis [TRAFFIC]. A proxy therefore cannot rely on the value of the ALPN header field as a policy input in all cases.

5. References

5.1. Normative References

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

[RFC3864] Klyne, G., Nottingham, M., and J. Mogul, "Registration Procedures for Message Header Fields", BCP 90, RFC 3864, DOI 10.17487/RFC3864, September 2004, <http://www.rfc-editor.org/info/rfc3864>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <http://www.rfc-editor.org/info/rfc3986>.

[RFC7230] Fielding, R. and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <http://www.rfc-editor.org/info/rfc7230>.

[RFC7231] Fielding, R. and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, <http://www.rfc-editor.org/info/rfc7231>.

[RFC7301] 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, <http://www.rfc-editor.org/info/rfc7301>.

5.2. Informative References

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <http://www.rfc-editor.org/info/rfc5246>.

[RFC5766] Mahy, R., Matthews, P., and J. Rosenberg, "Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)", RFC 5766, DOI 10.17487/RFC5766, April 2010, <http://www.rfc-editor.org/info/rfc5766>.

[TRAFFIC] Pironti, A., Strub, P-Y., and K. Bhargavan, "Website Users by TLS Traffic Analysis: New Attacks and Effective Countermeasures, Revision 1", 2012, <https://alfredo.pironti.eu/research/publications/full/identifying-website-users-tls-traffic-analysis-new-attacks-and-effective-counterme>.

5.3. URIs

[1] http://www.iana.org/assignments/tls-extensiontype-values/#alpn-protocol-ids

[2] https://www.iana.org/assignments/message-headers

Authors' Addresses

Andrew Hutton
Unify
Technology Drive
Nottingham NG9 1LA
UK

EMail: andrew.hutton@unify.com

Justin Uberti
Google
747 6th Ave S
Kirkland, WA 98033
US

EMail: justin@uberti.name

Martin Thomson
Mozilla
331 E Evelyn Street
Mountain View, CA 94041
US

EMail: martin.thomson@gmail.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: April 30, 2015

D. Druta
G. Bourg
AT&T
S. Loreto
Ericsson

October 27, 2014

Using WPD to configure network proxies
draft-loreto-wpd-usage-00

Abstract

This specification provides feedbacks and improvements suggestions to "The Web Proxy Description Format" draft. It is an initial set of considerations collected while reviewing the WPD draft to understand how it could be used in a Telecom Operator network; specifically Cellular and Operated WiFi Networks.

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 30, 2015.

Table of Contents

1. Introduction

Draft [I-D.nottingham-web-proxy-desc] presents a simple and reliable way to describe a Web Proxy Configuration (WPD), and proposes the usage of a well-known URI [RFC5785] to designate a well-know location where the WPD can be found.

The WPD allows the description of multiple proxies in one WPD file as long as they are under the same authority; clients may choose any proxy they wish, and may use more than one proxy at a time. However the proxy description should be extended to provide better user experience and smoother interaction with the network.

2. WPD Proxies

[I-D.nottingham-web-proxy-desc] defines the "WPD Proxy" concept and place additional requirements upon this proxy, as well as upon clients using it. The following sections proposes improvements and changes to those requirements.

2.1. Traffic Flow label

3GPP Mobile Networks support Traffic Flow Templates that allow specific flows to traverse different data bearers; a flow is identified by the 5-tuple {dest addr, source addr, protocol, dest port, source port}. These data bearers can have different characteristics based on how they are configured. Specifically, in scheduled networks there is an inherent balancing act between absolute data rate, latency, and jitter. Today 3GPP mobile networks

balance the needs of different types of traffic (e.g. the high bit rate download that is unaffected by jitter vs real time communications that are lower bit rate but require minimal jitter or latency impact) by detecting traffic and assigning flows to bearers that are best configured for the type of traffic. Traffic classification is not used to discriminate between clients or between servers. Instead, it is necessary to optimally utilize the radio resources resulting in improved user experience.

WPD proxy mandates the usage of HTTP2 [I-D.ietf-httpbis-http2] over TLS for connections from clients.

The usage of a single TCP connection to an addressable and explicit proxy provides several benefits, however at same time it prevents a 3GPP mobile network from working as previously described, because it is not possible to de-multiplex the traffic within the single TCP socket into multiple data bearers. For a 3GPP mobile network, this limits the ability to tune the network efficiently based on traffic classification.

As draft [I-D.nottingham-web-proxy-desc] already allows a client to choose multiple proxies to be used at a time, it should be possible for a client to use multiple proxies for different purposes as long as it is clear to the client the different type of advantages offered by each proxy listed in the retrieved WPD file. In this way a different proxy instance could be used for flows of each traffic type.

This document defines a new WPD’s "proxy object" member ("label"), describing the purpose and the specific services offered by a proxy. Labels can be used by the client to identify the right proxy that should be used for a specific traffic type or them can be used to identify proxies offering specific optimisation functions.

A client can use the "label" member to identify the right proxy to be used for a specific class of traffic (e.g. "dft" for a traffic balance between latency and jitter, "rtc" for low bandwidth but jitter sensitive real time communications, or "hdr" for High Data Rate traffic that is unaffected by jitter). The client is, in many situations, able to determine from the context of an HTTP transaction what type of "label" to apply to an http resource. The client may then decide to map the label’s resource with the proxy with the same label listed in the retrieved WPD or to use the default proxy. Additionally, it should be possible for the content provider to explicitly label resources to suggest they traverse a specific proxy path.

2.2. Web Protocol

> "Clients MUST use HTTP/2 over TLS to connect to a WPD proxy; if one cannot be established, the client MUST consider that proxy "failed"."

Using TLS to provide security, confidentiality and integrity is an important requirement to have. However restricting the possibility to establishing connection from client only to "HTTP/2 over TLS" closes the door to innovation. It would much better to define a list of criteria that protocol has to fulfil in order to be used between a Client and a WPD Proxy. In order to maximize interoperability, HTTP/2 MUST be mandatory to implement, however other protocols that meet the minimum requirements may also be used.

A possible but not complete list of requirements can be: "The protocol session between the client and server must use a multiplexed and flow controlled protocol. This is necessary to effectively scale the proxy server and provide differentiated service priority between requests while protecting resource constraints on both ends. The reduction in connection count is particularly meaningful to an intermediary, and the protocol support for prioritization is needed to mitigate the change in connection granularity."

3. The Web Proxy Description (WPD) Format

The proxy information currently defined in the WPD proposal should be extended in order to both provide a better user experience and a smoother interaction with the network. For example:

```
{
    "name": "ExampleCorp Web Proxy",
    "desc": "ExampleCorp's Proxy Gateway for Web access. Note that
             all traffic through this proxy is logged, and may be
             filtered for content.",
    "moreInfo": "https://inside.example.com/proxy/",
    "proxies": [
        {
            "host": "proxy.example.com",
            "port": 8080,
            "clientNetworks": ["192.0.2.0/24"]
            "label": "dft"
            "WebProtocol": "h2"

        },
        {
            "host": "proxy1.example.com",
            "port": 8080,
            "clientNetworks": ["192.0.2.0/24"]
            "label": "rtc"
            "WebProtocol": "h2"
        }
    ],
    "alwaysDirect": ["example.com", "192.0.2.0/24"],
    "failDirect": False
}
```

The reminder of this section defines the proposed extensions of the WPD object members.

3.1. proxy objects members

A proxy object as defined in [I-D.nottingham-web-proxy-desc] represents a HTTP proxy endpoint within the WPD JSON representation. At moment the only Proxy objects' "label" value defined are the following "host", "port", "clientNetworks".

The following sections define new members necessary to completely describe a WPD Proxy.

3.1.1. Label

A string containing the tag specifying the class of traffic. This specification defines the following three three labels:

dft: a traffic class with balance between latency and jitter;

rtc: a traffic class for low bandwidth but jitter sensitive real time communications;

hdr: a traffic class for High Data Rate traffic that is unaffected by jitter.

It should be possible to use implementation defined labels for which the Client MUST use the label on the http resource URL to send the request to the corresponding proxy entry in the WPD. If the label does not have a matching entry, the Client MUST use the default proxy (dft).

[TBD] create a new IANA register for the label.

3.1.2. WebProtocol

A string containing the ALPN tag [RFC7301] describing the protocol supported by the proxy to establish a session between the client and proxy itself. This member MUST be present.

4. IANA Considerations

This specification creates a new IANA registry, in accordance with the principles set out in [RFC5226], for the "label" Proxy objects' parameter namespace describing the specific class of traffic or optimisation feature provided by a proxy.

5. Security Considerations

See [I-D.nottingham-web-proxy-desc] for general security considerations on the WPD usage.

It should be noted that the usage of multiple proxies, proposed in this document, discloses the type of traffic requested and used by the client (i.e. the end user).

6. Acknowledgments

The authors wish to thank Diego Lopez, Kevin Smith, Martin Nilsson, Sanjay Mishra and Mathilde Cayla for their comments and useful feedback.

7. Normative References

[I-D.ietf-httpbis-http2]
Belshe, M., Peon, R., and M. Thomson, "Hypertext Transfer Protocol version 2", draft-ietf-httpbis-http2-14 (work in progress), July 2014.

[I-D.nottingham-web-proxy-desc]
Nottingham, M., "The Web Proxy Description Format", draft-nottingham-web-proxy-desc-01 (work in progress), October 2014.

[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.

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

[RFC5785] Nottingham, M. and E. Hammer-Lahav, "Defining Well-Known Uniform Resource Identifiers (URIs)", RFC 5785, April 2010.

[RFC7301] Friedl, S., Popov, A., Langley, A., and E. Stephan, "Transport Layer Security (TLS) Application-Layer Protocol Negotiation Extension", RFC 7301, July 2014.

Authors' Addresses

Dan Druta
AT&T

Email: dd5826@att.com

Gus Bourg
AT&T

Email: gb3635@att.com

Salvatore Loreto
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

Email: salvatore.loreto@ericsson.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 17, 2015

M. Nottingham

October 14, 2014

The Web Proxy Description Format
draft-nottingham-web-proxy-desc-01

Abstract

This specification defines a simple means of configuring Web proxies, and places additional requirements upon them in order to promote improved interoperability, security, and error handling.

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 17, 2015.

Copyright Notice

Table of Contents

1. Introduction

Web proxies can be configured in a variety of ways, but existing approaches suffer from security, usability and interoperability issues.

This specification defines:

- A simple format for describing a Web proxy ("WPD"; see Section 3) to facilitate configuration, and to allow proxies to be represented to users in a consistent way, and

- A way to discover the proxy description using a well-known URL (Section 4), so that direct configuration of a proxy is as simple as entering a hostname, and

- A set of additional requirements for proxies described in this fashion, as well as requirements for User Agents connecting to

them, designed to improve security, usability and interoperability. See Section 2.

It can be used in a variety of ways, but is designed to meet the following goals:

- Users should always be aware of a configured proxy and be able to confidently identify it, and
- Configuring a proxy should be a deliberate act, but simple to do for non-technical users, and
- Proxies should always respect the wishes of the end user and Web site, and
- Proxies should never reduce or compromise the security of connections, and improve it where possible, and
- Proxies should be able to reliably communicate with their end users regarding their policies and problems that are encountered.

Furthermore, it is designed to be useful in the following cases:

- An end user wants to use a proxy network that provides improved performance, by re-compressing responses to http:// resources.
- An end user wants to use a proxy network that provides improved privacy, by routing requests through any number of intermediaries.
- An end user is required to use a proxy to access Internet resources by their network (e.g., a school, workplace or prison).
- A network wants to offer enhanced access to selected Web sites, through interposition of a proxy.

Importantly, this specification does not address the automatic discovery of proxy configuration for a given network, because proxy configuration is a security-sensitive action, and ought never be performed without explicit user or administrator action.

It is expected that the mechanisms described could be implemented by a single program (e.g., a Web browser), or through an Operating System facility.

1.1. 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. WPD Proxies

This specification defines a particular kind of HTTP proxy (as per [RFC7230] Section 2.3) known as a "WPD proxy" that has additional requirements placed upon it, as well as upon those using it.

WPD Proxies MUST support HTTP/2 [I-D.ietf-httpbis-http2] over TLS for connections from clients. Clients MUST use HTTP/2 over TLS to connect to a WPD proxy; if one cannot be established, the client MUST consider that proxy "failed."

WPD Proxies MUST support forwarding requests with the "http" scheme [RFC7230], and SHOULD support the CONNECT method, as specified in [I-D.ietf-httpbis-http2] Section 8.3.

[RFC7230] Section 5.7.2 requires proxies to honour the semantic of the "no-transform" cache-control directive, and append the 214 (Transformation Applied) warn-code to other messages that have been transformed; WPD proxies MUST honour these requirements.

When connecting to a WPD proxy, clients MUST validate the proxy hostname as per [RFC2818] Section 3.1. If the proxy presents an invalid certificate, that proxy MUST be considered "failed" and not used (until a valid certificate is presented).

User agents MUST use a CONNECT tunnel when retrieving URLs with the "https" scheme through WPD proxies.

When user agents encounter 5xx responses to a CONNECT request from a WPD proxy, they MUST present the response to the end user, but MUST NOT present or process it as a response to the eventual request to be made through the tunnel (i.e., it has an unidentified payload, as per [RFC7231] Section 3.1.4.1).

NOTE: Many user agents refuse to show an error response to a CONNECT to the user, in order to deal with the issues brought to light by [bad-proxy]. While effective in dealing with those attacks, doing so effectively disallows communication between the proxy and the end user; this requirement is designed to re-open that channel.

If a WPD proxy becomes unresponsive, clients SHOULD consider it failed and attempt to use another proxy (if available) or inform the

end user (if not available). Clients SHOULD regularly attempt to re-establish contact with failed WPD proxies (e.g., every minute).

Requests for the "localhost" [RFC6761] and "local" [RFC6762] top-level domains MUST NOT be routed through a WPD proxy.

Likewise, requests to the Loopback address blocks (127.0.0.0/8 and ::1/128) and the Link Local block (169.254.0.0/16 and fe80::/10) MUST NOT be routed through a WPD proxy; see [RFC6890]. Note that clients are not required to perform a reverse lookup on hostnames to conform to this requirement.

3. The Web Proxy Description (WPD) Format

WPD is a JSON [RFC7159] format that describes a Web proxy to potential clients. Its root is an object containing WPD-specific object members. For example:

```
{
    "name": "ExampleCorp Web Proxy",
    "desc": "ExampleCorp's Proxy Gateway for Web access. Note that
             all traffic through this proxy is logged, and may be
             filtered for content.",
    "moreInfo": "https://inside.example.com/proxy/",
    "proxies": [
        {
            "host": "proxy.example.com",
            "port": 8080,
            "clientNetworks": ["192.0.2.0/24"]
        },
        {
            "host": "proxy1.example.com",
            "port": 8080,
            "clientNetworks": ["192.0.2.0/24"]
        }
    ],
    "alwaysDirect": ["example.com", "192.0.2.0/24"],
    "failDirect": False
}
```

When configuring a proxy through WPD, a user agent SHOULD present the relevant information contained within (i.e., the 'name', 'desc' and 'moreInfo' members, the latter as a link) to the end user. User agents SHOULD also make this information available to the end user whenever the WPD is in use.

The remainder of this section defines the content of the WPD object members. Unrecognized members SHOULD be ignored.

3.1. name

A string containing a short, memorable name for the proxy; typically 64 characters or less. This member MUST be present for the WPD to be considered valid.

3.2. desc

A string containing a textual description of the proxy's function(s); typically 256 characters or less. This member MUST be present for the WPD to be considered valid.

3.3. moreInfo

A string containing a URL [RFC3986] that leads to more information about the proxy, its operation, who operates it, etc. The URL MUST have a scheme of "https" [RFC7230], and MUST be able to respond with an HTML [W3C.CR-html5-20140731] representation. This member MUST be present for the WPD to be considered valid.

3.4. proxies

An array containing one or more proxy objects; each proxy object represents a HTTP proxy endpoint that can be used when this WPD is configured. See Section 2 for requirements specific to these proxies, as well as those clients connecting to them.

Proxy objects' members are defined by the following subsections; unrecognized members SHOULD be ignored.

The ordering of proxy objects within the proxies array is not significant; clients MAY choose any proxy they wish (as long as the specific requirement so the proxy object are met), and MAY use more than one at a time.

NOTE: the array of proxy objects is functionally similar to, but not as expressive as, the commonly-used "proxy.pac" format. While it would be expedient for WPD to just reference a proxy.pac, feedback so far is that proxy.pac has a number of deficiencies, and interoperability is poor. Therefore, this document specifies the proxy object instead, in order to gather feedback on an alternative approach.

3.4.1. host

A string containing the host (as per [RFC3986], section 3.2.2) of the proxy. This member MUST be present.

3.4.2. port

A number representing the port that the proxy is listening on. This member MUST be present, and MUST be an integer.

3.4.3. clientNetworks

An array containing strings; each string contains a classless prefix (see [RFC4632]) which the proxy can be used within. Clients MUST NOT attempt to use the proxy if their IP address is not within one of the stated ranges.

This member is optional.

For example, if the value of clientNetworks is

["192.168.1.0/32", "192.168.2.0/24"]

then the only clients that could use the proxy would have IP addresses in the ranges 192.168.1.0 to 192.168.1.3 and 192.168.2.0 to 192.168.2.255.

Note that by their nature private networks (as specified in [RFC1918]) are not unique, and therefore there may be false positives. As such, clients SHOULD NOT automatically configure a WPD based upon clientNetworks when the IP address is in these ranges, although they MAY notify the user of a WPD's possible applicability, and SHOULD use additional information to correlate a WPD to its proper network. For example, the MAC address of the network's gateway (as discovered by ARP [RFC0826]) can be used to disambiguate multiple instances of the same network.

3.5. forReferers

An array containing strings; each string is a host (as per [RFC3986] Section 3.2.2).

When forReferers is present, Clients MUST use the WPD's proxies to access these hosts, hostnames that have the host as a root, and for traffic generated by that content. They MUST NOT be used for other traffic.

This member is optional.

For example, if the value of forReferers is

["friendface.example.com"]

then requests to "friendface.example.com", "www.friendface.example.com", "app.friendface.example.com" etc. would use the associated proxies; likewise, if processing a response from one of these hosts generated further requests to "images.example.net" and "scripts.example.org", they would also use the proxies.

Note that alwaysDirect takes precedence over forReferers.

TODO: tighten up what "processing" means here; the intent is to omit a href

3.6. alwaysDirect

An array containing strings; each string is one of:

- a host (as per [RFC3986] Section 3.2.2),
- a classless prefix [RFC4632].
- the string "CONNECT".

Clients MUST NOT use the WPD's proxies to access nominated hosts and hostnames that have the a nominated host as its root. Likewise, clients MUST NOT use the WPD's proxies to access bare IP addresses that fall within the classless prefix.

If the string "CONNECT" (case-sensitive) appears in alwaysDirect, it indicates that requests that require establishment of a tunnel (e.g., for "https" URLs) MUST NOT use the WPD's proxies, but instead ought to be made directly to the origin (i.e., without a tunnel).

Note that when a "bare" IP address or classless prefix is used in alwaysDirect, clients are not required to perform a reverse lookup on hostnames; these forms are only intended for accessing URLs that use the IP-literal or IPv4address forms.

This member is optional.

For example, if the value of alwaysDirect is:

["example.com", "192.168.5/24"]

then requests to "example.com", "www.example.com", "foo.example.com" etc would not use any proxy. Likewise, requests whose URL authority were bare IP addresses in the range 192.168.5.0 to 192.168.5.255 would not use any proxy.

3.7. failDirect

A boolean indicating whether the client should attempt to directly access the origin server if all applicable proxies are unavailable.

When False, clients MUST NOT attempt to directly access the origin server when no proxy is available, but instead SHOULD inform the user that the proxy is unavailable.

When True, clients MAY do so. If failDirect is not present, clients MAY default to this behavior.

3.8. exclusive

A boolean indicating whether the client is required to route all network traffic through the proxy.

When True, clients MUST NOT initiate network traffic to any host except a valid WPD (once its identity and location are established), and MUST NOT allow network traffic from any host except valid WPDs. This includes all traffic from and to the client, no matter how it is generated or handled (e.g., browser "plug-ins").

This directive is designed to accommodate privacy-enhancing proxies; therefore, clients that cannot reasonably assure conformance to the requirements in this section MUST NOT allow a WPD with this flag set to be configured.

3.9. privateMode

A boolean indicating whether the client should be configured in "private mode" when this WPD is active.

When True, clients SHOULD configure "private mode" browsing.

4. Discovering WPD Files

To facilitate easy configuration of WPD proxies, this specification defines a well-known URI [RFC5785]. Doing so allows a proxy’s description to be found with a simple hostname; e.g., "proxy.example.net" or even just "example.net".

Clients MUST NOT use the DHCP "WPAD" mechanism to discover WPDs.

4.1. The web-proxy-desc well-known URI

The "web-proxy-desc" well-known URI allows discovery of a Web Proxy Description (Section 3).

This well-known URI is only valid when used with the "https" URI Scheme [RFC7230]; it MUST NOT be used with "http" URIs. In other words, WPD discovery is always protected by TLS [RFC5246].

The description found at this location is considered valid for its freshness lifetime, as defined in [RFC7234] Section 4.2. Once stale, clients SHOULD refresh it and apply any changes.

If the WPD is not retrievable (e.g., a 404 response status), invalid (as per JSON [RFC7159] or the requirements in Section 3), or its certificate is not valid for the host (as per [RFC2818] Section 3.1), the client MUST NOT use the WPD, and if a user agent, SHOULD inform the end user.

The well-known URI MAY use proactive content negotiation ([RFC7231] Section 3.4.1) to select an appropriate language for the response representation. Therefore, clients SHOULD send an Accept-Language request header field ([RFC7231] Section 5.3.5) when they wish to advertise their configured language.

The registration template is:

- URI suffix: web-proxy-desc
- Change controller: IETF
- Specification document(s): [this document]
- Related information: only to be used with ’https’ scheme

5. IANA Considerations

This specification registers a new well-known URI, as per [RFC5785]. See Section 4.1 for the template.

6. Security Considerations

If a user can be convinced to configure a WPD hostname as their proxy, that host can observe all unencrypted traffic by the client. As such, WPD configuration interfaces ought only allow configuration of proxies once their identity is validated (as required), and the user ought to be given access to all relevant information about the WPD proxy (i.e., ’name’, ’desc’ and ’moreInfo’, the latter as a

link). Furthermore, WPD proxies ought only be configured as the result of an intentional act, not as a side effect of normal Web browsing.

7. Acknowledgements

Thanks to Patrick McManus for his feedback and suggestions.

8. References

8.1. Normative References

[I-D.ietf-httpbis-http2] Belshe, M., Peon, R., and M. Thomson, "Hypertext Transfer Protocol version 2", draft-ietf-httpbis-http2-14 (work in progress), July 2014.

[RFC1918] Rekhter, Y., Moskowitz, R., Karrenberg, D., Groot, G., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, February 1996.

[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.

[RFC4632] Fuller, V. and T. Li, "Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan", BCP 122, RFC 4632, August 2006.

[RFC6761] Cheshire, S. and M. Krochmal, "Special-Use Domain Names", RFC 6761, February 2013.

[RFC6762] Cheshire, S. and M. Krochmal, "Multicast DNS", RFC 6762, February 2013.

[RFC6890] Cotton, M., Vegoda, L., Bonica, R., and B. Haberman, "Special-Purpose IP Address Registries", BCP 153, RFC 6890, April 2013.

[RFC7159] Bray, T., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, March 2014.

[RFC7230] Fielding, R. and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, June 2014.

[RFC7234] Fielding, R., Nottingham, M., and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Caching", RFC 7234, June 2014.

[W3C.CR-html5-20140731] Berjon, R., Faulkner, S., Leithead, T., Navara, E., O'Connor, E., and S. Pfeiffer, "HTML5", World Wide Web Consortium CR CR-html5-20140731, July 2014, <http://www.w3.org/TR/2014/CR-html5-20140731>.

8.2. Informative References

[RFC0826] Plummer, D., "Ethernet Address Resolution Protocol: Or converting network protocol addresses to 48.bit Ethernet address for transmission on Ethernet hardware", STD 37, RFC 826, November 1982.

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

[RFC5785] Nottingham, M. and E. Hammer-Lahav, "Defining Well-Known Uniform Resource Identifiers (URIs)", RFC 5785, April 2010.

[RFC7231] Fielding, R. and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, June 2014.

[bad-proxy] Chen, S., Mao, Z., Wang, Y., and M. Zhang, "Pretty-Bad-Proxy: An Overlooked Adversary in Browsers' HTTPS Deployments", January 2009, <research.microsoft.com/jump/79323>.

Appendix A. User Experience for WPDs

There are a variety of ways to present proxy configuration to users and administrators, so this specification does not constrain how this is done. That said, guidance for the common case (visual Web browsers) can be helpful in assuring consistent user experience.

One of the core principles of this specification is that WPDs need to be explicitly configured, either by the end user or an administrator on their behalf. This is because using a proxy is a security-sensitive operation; if an attacker can automatically configure a

proxy, or convince a user to do so as part of accessing a site, they can gain access to the user's traffic, even when the user leaves the attacking network.

Therefore, a user agent might allow configuration by entering a hostname (e.g., "example.net"), whereupon it retrieves the WPD, validates its certificate and contents, and present its information to the end user for confirmation.

Once a WPD is confirmed, a user agent might "remember" it for future use; e.g., by allowing quick configuration through a drop-down menu. When a WPD nominates clientNetworks and the client does not have a suitable IP address, the drop-down might make that option unavailable.

It is envisioned that only a single WPD ought be configured at a time; combining WPDs leads to ambiguity regarding precedence and therefore user confusion.

When a WPD is active, its ought be visible to the end user, to remind them of its presence, and to offer more information about the configured proxy.

Author's Address

Mark Nottingham

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

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 30, 2015

D. Druta
AT&T
T. Fossati
Alcatel-Lucent
M. Ihlar
Ericsson
G. Klas
Vodafone
D. Lopez
Telefonica I+D
J. Reschke, Ed.
greenbytes

October 27, 2014

A Rationale for Fine-grained Intermediary-aware End-to-End Protocols
draft-reschke-objsec-01

Abstract

A tremendous growth in different uses of the Internet has let to a growing need to protect data sent over public networks, including data sent via http. Use of end-to-end TLS for the majority of traffic looks at first a most feasible response. However, the web architecture has become more sophisticated and as it has now gone beyond the simple client-server model, the end-to-end used of TLS is increasingly showing its downside. The end-to-end use of TLS excludes the use of beneficial intermediaries such as use of caches or proxies that provide instrumental services. Then need for greater privacy seems to collide with the equally growing desire for better end-to-end performance and user experience. As an example, the use of HTTP/TLS often appears to maximise the benefit for the combination of both.

This document describes the above dichotomy and lays out a number of objectives of what can ideally be achieved, namely catering for sufficient security and privacy whilst providing users with the opportunity to make use of intermediaries' services where considered beneficial. This document introduces a number of potential solutions towards use of suitable protocol mechanisms and data formats. End-to-end protocols which are aware of intermediaries should enable users and/or content providers to exercise fine-grained control over what intermediaries should be able to do and what exposure to data or metadata they shall be permitted to get. The document then highlights anticipated benefits to key stakeholders such as users, content providers and intermediaries. As elements such as object security can play a useful role, this document encourages the analysis of related work to discern their applicability, limitations,

and coverage of use cases. Such an effort may us espouse innovation to frame an overall architecture and motivate more detailed work on protocols and mechanisms in the future.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

In the last decade, the generalization of network access, the cloud and the ubiquity of the Web as an application platform has translated into an unprecedented increase in the use of the Internet, facilitated by the development of new web standards and the innovations in mobile technologies. With this growth in use, there has been an increased amount of personal data being sent over multi hop public links.

In order to protect the integrity and confidentiality of the online transactions, HTTP traffic can be secured with transport layer security using https. While https is great for e-commerce and banking and while there is a sense of understanding in the user community around the secure nature of https, using TLS and https for the majority of the traffic has performance and functional drawbacks, mainly because the HTTP session is encrypted as a whole.

Looking from the privacy and security perspective, it is clear that users must be aware if and when an intermediary node is intercepting their traffic and they have the right to continue or demand higher levels of privacy by encrypting what they deem to be sensitive information. The privacy threshold depends on user's tolerance and the trust they put in the intermediary's reputation, as well as whether the user is the ultimate consent authority for a given connection: for example a parent or employer may take that role for a connection used by a child or employee.

Modern web architecture involves sophisticated caching schemes that involve fetching various objects (images, libraries, etc.) from various locations in the path to avoid latency and improve the overall user experience while reducing bandwidth use. This is an important aspect especially in the developing countries, remote locations and in general areas that lack fast network infrastructure.

Issues thus arise from the clash of two trends: One is towards enhanced privacy calling for integrity, confidentiality and

anonymity. The other one is towards improved performance and lower latency, calling for caching, compression, and adaptability.

This is also reflected in the views of important stakeholders. Users want to make informed decisions in regards to whom they trust with their data. They also want to have control over what data they share with whom.

Web site owners and content providers on the other hand are keen to get the most cost efficient and reliable way to deliver information and services to their users and customers, while preserving their confidentiality, protecting their privacy and the integrity of their data.

As different entities seek to meet the requirements of their stakeholders, they sometimes apply solutions which generate conflicts. Clients act on behalf of end users and solutions may include local caches and browser proxies. Servers act on behalf of content providers and solutions may include the use of CDNs and reverse proxies. Intermediaries operated by communication service providers and network operators act on behalf of users and/or content providers and solutions include means for access network optimization and content filtering.

In above context, the use of TLS and https looks like a "black and white approach", or an "all or nothing" approach which doesn’t lend itself to resolving above-mentioned conflicts. The question arises: Can "color be added"?

TLS is a client server protocol and it serves its purpose perfectly in many client-server scenarios and use cases. But then the web has evolved to become a mesh. Average traffic flows now involve various intermediaries between clients and servers. They add value by performing different functions including multiple levels of optimization.

The application of TLS forces point-to-point flows which cuts out intermediaries and can lead to significant drawbacks. It reduces e.g. the optimization options which translates into increasing traffic volumes in access networks, higher latency and overall decreasing quality of experience for users.

It can be observed that ignoring the role of intermediaries on the Internet does not necessarily make the Internet more secure. In fact, in some cases it forces various parties to break the TLS promise of e2e integrity and confidentiality in order to meet their legal obligations (enterprises).

We suggest the solution to the challenge lies in "adding color". An example of this are fine-grained intermediary-aware end-to-end protocols.

Assuming the existence of such a fine-grained protocol, the following benefits could be imagined which leads to satisfying the justified needs of multiple stakeholders:

The ability to atomically encrypt objects or even HTTP frames should support this sophisticated caching mechanisms while allowing content providers to avoid distributing their server key material across the network nodes and prevent the risk of compromising their security.

2. Objectives

Given the short description of the problem above, the following objectives can be derived.

1. To improve security and user-controlled privacy.

2. To minimize passive interception and man in the middle attacks.

3. To allow the client (user) and the server (content provider) to negotiate what and whom they want to give (or not) visibility into their traffic flows.

4. To enable multiple levels of optimization that don’t conflict with each other and either meet all parties expectations or maximise the benefit to as many involved parties as possible.

In a world of TLS and https only, it is difficult to achieve in particular objectives 3. and 4.

The challenge therefore is in finding mechanisms or protocols which meet objectives 1. and 2. (e.g. in the way TLS is delivering against those objectives) AND simultaneously provide the added flexibility to leverage the services of 3rd party intermediaries located between client and origin server.

3. Characteristics of Solutions

From above, one can draw some conclusions about the characteristics possible solutions or new protocols may have to show. Below is a non-exhaustive list. A new fine-grained intermediary-aware end-to-end protocol may need to feature:

1. a mechanism to enable users to choose their preferred level of privacy, adequate for a particular context and use case. The

context may be determined by the presence or absence of particular intermediaries or proxies which offer particular services (e.g. caching, data compression etc.).

2. mechanisms that enable certain communication data to be exchanged securely, whereby the end user shall be able to select the set of security services deemed adequate for a particular communication context (e.g. confidentiality, data integrity, entity authentication etc.).

3. mechanisms that enable the end user to select which communication data shall be subject to particular security services (like confidentiality, data integrity etc.). Note that this might be all application level data transferred between server and client, or it might be a subset of application level data. This refers to the notion of "fine-grained" control.

4. mechanisms that protect against passive interception and man-in-the-middle attacks.

5. mechanisms that allow the two ultimate communication end points, namely client and origin server, to negotiate whether and if so which intermediaries shall be permitted to play a role in delivering application data from origin server to client given particular end user expectations, requirements and preferences, and information about the status of the network between client and server. This refers to the notion of "intermediary-aware end-to-end protocol".

6. mechanisms that allow end users or origin server or both to determine in real-time which traffic optimizations are available at the time of communication setup.

7. mechanisms that allow end users or origin servers or both to eventually select zero or more optimizations to be applied to traffic flows between origin server and client.

8. mechanisms that allow the simultaneous or sequential application of optimizations such that those optimizations on traffic and traffic flow don’t conflict with each other.

As said, above list is not exhaustive and additional characteristics may be either required or useful.

The intent of this draft is not to introduce a solution yet. However, it may help to consider possible elements which might play a role in any solution. One element is "object security".

4. Benefits for the content provider, for the users, for the intermediaries

An object security approach will allow the different parties to establis end-to-end and hop-by-hop security mechanisms to different data and metadata elements, and therefore address what can be seen as conflicting requirements in terms of optimization and security capabilitites.

The following benefits will arise for the different stakeholders:

Content providers:

- Can select the type of security service that is optimal and sufficient for particular types of content: e.g. confidentiality, integrity protection, entity authentication etc.
- Can select which parts of their content shall be secured or not and how content shall be securely retrievable.
- Can increase their confidence in secure temporary content storage during delivery through encrypting/signing sensitive content objects.
- Can leverage the business services of 3rd parties (intermediaries) through enabling the intermediaries to perform value-added services. Content providers may outsource particular tasks (like caching, or offering higher security level to users) to intermediaries.
- When using the services of content delivery networks, can benefit from faster, optimised delivery over the "last mile" (as seen from the perspective of a content delivery network). Content delivery networks can optimise delivery on behalf of content providers over the first and middle mile, however they often rely on other ISPs and mobile network operators to deliver content over the last mile. Intermediaries in the last mile can optimise traffic engineering.

Users:

- Are able to enjoy sufficient privacy and security as dictated by different use cases and equally their personal preferences (e.g. protection from traffic analysis, integrity of content).
- Can benefit from value-added services delivered by intermediaries on behalf of content providers.

- Can have access to services offered by intermediaries which enhance end user quality of experience (e.g. malware detection, parental controls).

- Can access web resources and services with lower latency and better response times (e.g. through intermediaries performing video pacing or traffic engineering to avoid congestion on particular networks).

Intermediaries:

- Can play their specific roles in content delivery and communication on behalf of content providers, like
  - Caching of content on behalf of content providers
  - Optimisation of content for optimal delivery on behalf of content providers
  - Video pacing on behalf of content providers.
- Can provide value-added services on behalf of users like parental control, malware detection etc.
- Can optimise content delivery and data communication within a network they are associated with or control e.g. through traffic engineering and traffic management by taking into account the inherent needs of content types and the explicit real- and non-real-time requirements of content providers and content delivery networks. Thereby, intermediaries contribute to an improved "end-to-end" user experience in the interest of both users and content providers.
  - Intermediaries are enabled to perform congestion management and can therefore reduce latency and response times.
- Can meet regulatory requirements as they may prevail in particular jurisdictions through an approach which is more open and transparent to both users and content providers, and which may be in the national interest.

5. Analysis of Related Work

The concept of object security is not something new, several approaches targeted at different application areas exist today, and we can even root them at the original S/MIME proposal ([RFC5751]).

As one of our first tasks, we intend to perform a detailed analysis of this related work, producing a list of the gaps of each technology solution in the scenarios we foresee. In particular, we have already identified at least a couple of such related work:

- JOSE, which stands for "JSON Object Signing and Encryption". It is a series of standards produced by the IETF under the JOSE charter ([1]) offering encryption, digital signatures, and Message Authentication Codes (MACs).

- Subresource Integrity ([SRI]), a W3C specification defining mechanisms by which user agents may verify that a fetched resource has been delivered without unexpected manipulation.

6. Architectural Considerations

The purpose of an object security architecture is to be able to provide more flexible security services than strict end-to-end encryption. A content owner should be able to express what security levels different objects should be associated with.

Such an architecture needs to define two types of logical channels between end-points. One channel is strictly end-to-end encrypted where sensitive data is transferred between end points without the risk of third-party access. The second channel is more relaxed in allowing third-party nodes be part of the flow (i.e hop-by-hop encrypted channel). The amount of information exposed in the second channel is determined by the content provider alone or in agreement with the end-user.

There are several ways to design an architecture that fulfills these requirements. An important question to analyze is whether an object security architecture should be designed at the application layer or further down the stack as an alternative to TLS.

7. Analysis of the Impacts on HTTP/2

[[CREF1: TBD]]

8. Analysis of the Impacts on TLS

[[CREF2: TBD]]

9. Impacts on the current browser architecture

[[CREF3: TBD]]

10. Impacts on the existing deployment / how to make this proposal coexist with the current

[[CREF4: TBD]]

11. Privacy Impact

[[CREF5: TBD]]

12. Security Considerations

[[CREF6: TBD]]

13. Contributors

The following people are not listed as authors, but contributed significantly to the discussions leading to this document: Liliana Dinale, Vijay Gurbani, Mike Jones, Eliot Lear, Salvatore Loreto, John Mattsson, Sanjay Mishram, Robert Moskowitz, Kevin Smith, Dan Wing.

14. References

14.1. Informative References

[RFC5751] Ramsdell, B. and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification", RFC 5751, January 2010.

[SRI] Braun, F., Akhawe, D., Weinberger, J., and M. West, "Subresource Integrity", W3C Working Draft WD-SRI-20140318, March 2014, <http://www.w3.org/TR/2014/WD-SRI-20140318/>.

Latest version available at [2].

14.2. URIs

[1] https://datatracker.ietf.org/wg/jose/charter/

Authors' Addresses

Dan Druta
AT&T

Email: dd5826@att.com

Thomas Fossati
Alcatel-Lucent

Email: thomas.fossati@alcatel-lucent.com

Marcus Ihlar
Ericsson

Email: marcus.ihlar@ericsson.com

Guenter Klas
Vodafone

Email: Guenter.Klas@vodafone.com

Diego R. Lopez
Telefonica I+D

Email: diego.r.lopez@telefonica.com

Julian F. Reschke (editor)
greenbytes GmbH

Email: julian.reschke@greenbytes.de

HTTPBIS
Internet-Draft
Intended status: Standards Track
Expires: January 5, 2015

M. Thomson
Mozilla

July 4, 2014

Client Authentication over New TLS Connection
draft-thomson-httpbis-cant-01

Abstract

This document defines an HTTP authentication scheme that can be added to an error response to indicate to a client that a successful response will only be provided over a new TLS connection, and only if the client has provided a certificate on that 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 January 5, 2015.

Copyright Notice

Table of Contents

1. Introduction

Client authentication in HTTP sometimes relies on certificate-based authentication of clients in Transport Layer Security (TLS) ([RFC5246], [I-D.ietf-tls-tls13]). Some existing uses of client authentication rely on TLS renegotiation, triggering renegotiation and a request for a client certificate in response to a request for a particular resource.

HTTP/2 [I-D.ietf-httpbis-http2] forbids the use of renegotiation, except for at the very beginning of a connection. TLS 1.3 [I-D.ietf-tls-tls13] does not support renegotiation at all. Both of these restrictions result in a server being unable to use renegotiation to conditionally request certificate-based authentication for clients in those protocol versions.

This document defines a new authentication scheme, "ClientCertificate", for use in HTTP authentication challenges [I-D.ietf-httpbis-p7-auth]. In combination with the 401 (Unauthorized) status code, this indicates that client authentication at the TLS layer is required in order to access the resource.

1.1. Conventions and Terminology

At times, this document falls back on shorthands for establishing interoperability requirements on implementations: the capitalized words "MUST", "SHOULD" and "MAY". These terms are defined in [RFC2119].

2. Client Certificate Challenge

A new authentication scheme ([I-D.ietf-httpbis-p7-auth]) for the "WWW-Authenticate" and "Proxy-Authenticate" header fields is defined with the name "ClientCertificate".

This challenge cannot be satisfied by constructing an Authorization header field [I-D.ietf-httpbis-p7-auth], it can only be satisfied by making the request on a TLS connection where an appropriate certificate has been provided by the client.

This authentication scheme cannot be used for "http" URIs.

To effectively use this authentication scheme, a new connection is needed for every protection space used by a given origin server. A client can use the "ClientCertificate" challenge as a trigger to open a new connection and to use client authentication on that connection.

For the new connection, a client can use the mechanism in [I-D.thomson-tls-care] to prompt the server to request a client certificate, to avoid the problem where the server doesn't know to make a CertificateRequest.

[[TBD: In TLS 1.3 a client can unilaterally provide authentication information without a request from the server.]]

[[TBD For versions of TLS prior to 1.3,]] a client can immediately request renegotiation immediately after the initial handshake. A server that supports conditional client authentication MUST request a client certificate if it receives a renegotiation request prior to receiving any application data.

3. Client Certificate Challenge Parameters

In addition to the "realm" parameter, a challenge with this authentication scheme MAY include parameters that provide a client with information that assists with selecting an appropriate certificate to offer. This can be necessary, since the necessary context that the server relies on is derived from the request and this is not available at the server when establishing a new connection.

3.1. Distinguished Name Parameter ("dn")

The "dn" attribute includes a distinguished name for the certificate authority. This matches the "certificate_authorities" value sent in a TLS CertificateRequest handshake message. The "dn" parameter is repeated for every distinguished name that is permitted.

A distinguished name is defined in Abstract Syntax Notation number 1 (ASN.1) [X680] and encoded using Distinguished Encoding Rules (DER) [X690] when used in TLS. The value of the "dn" parameter is a DER-encoded distinguished name that is further encoded using base 64 [RFC4648] with the URL and filename safe alphabet. Multiple alternative distinguished names are carried by repeating the "dn" parameter.

```
dn-parameter = base64url
base64url    = ALPHA / DIGIT / "_" / "~"
```

[[Issue: We could use the string encoding defined in RFC 4514, which could be friendlier for production and consumption. That means that this would need to use quoted-string, RFC 4514 escaping would need to be escaped twice, and non-ASCII characters in the DN would need to be escaped. Since the fingerprint parameters are strictly better anyway, and it’s also highly likely that selection criteria are unnecessary due to clients rarely having more than one certificate for any given site, I’m not inclined to support that level of complication.]]

3.2. Fingerprint Parameters ("sha-256", ...)

A server can instead include the fingerprint of a certificate that is on the certificate chain of an acceptable client certificate. For instance, this might include the fingerprint of an acceptable end-entity certificate, though what is more likely is that this includes the fingerprint of a certificate issuer.

The name of fingerprint parameters is taken from the hash function textual names registry defined in [RFC4572]. Clients MUST support a "sha-256" parameter, which indicates that the fingerprint is calculated using SHA-256 [RFC6234]. The value of a fingerprint parameter is encoded using base 64 [RFC4648] with the URL and filename safe alphabet.

```
sha-256-parameter = base64url
```

Like the "dn" parameter, fingerprint parameters can be repeated to provide clients with alternative values.

For example, a server could indicate that a set of permissible certificates based on a SHA-256 fingerprint, as follows:

```
WWW-Authenticate: ClientCertificate realm="home",
                    sha-256=NjUwZjA0Mzcy..., sha-256=MzJiMTQ3ODF...
```

These fingerprint values could be matched against an end-entity certificate or any issuer in the certificate chain. Note that line breaks are added to this example for formatting reasons only.

Where fingerprints are provided with multiple hash function names, a client can use any of the provided algorithms to determine which certificate to provide.

Note that strong collision-resistance is not important for the hash function that is used for certificate fingerprints, since clients only use this value to select between available certificates. The only consequence of a collision is that it becomes more difficult for the client to select the correct certificate.

4. Security Considerations

Clients that support this authentication scheme will create a new connection each time that they see a challenge. This could be exploited in order to generate additional load from connections on both server and client. This authentication scheme MUST only be used for "https" URIs.

Using new connections for client authentication has additional processing costs to the client in proving access to the private keys associated with the client certificate; and to the server in proving access to the private keys associated with their certificate twice in the case that the client opts for confidentiality protection on the client certificate (though only for TLS versions prior to 1.3, see Section 5).

HTTP/2 [I-D.ietf-httpbis-http2] allows clients to use the same connection for multiple domains. Certificate-based client authentication as defined by this specification is bound to a single canonical root URI (see [I-D.ietf-httpbis-p7-auth]). This could create issues where the security properties of a connection become unclear. Clients MUST ensure that a client-authenticated connection is only used for the canonical root URI for which it was created.

5. Privacy Considerations

TLS version 1.2 and prior do not provide confidentiality protection for client certificates in the initial handshake. Confidentiality protection for handshake messages, including the client certificate, is provided only for renegotiation handshakes.

Clients can initiate renegotiation immediately after the TLS connection is established to ensure that passive observers aren't able to view the selected certificate.

Revealing that a certificate is in use could alert a passive observer to the fact that a client has requested particular resources, thereby aiding traffic analysis. While renegotiation hides the contents of a client certificate, the presence of a new connection could reveal that some form of client authentication is being used. This is an especially strong signal in HTTP/2, where new connections are discouraged and are therefore exceptional.

Clients MUST avoid offering their client certificate if it will lack confidentiality protection, unless they are explicitly configured to send credentials in the clear.

6. IANA Considerations

IANA is requested to create an entry in the HTTP Authentication Scheme Registry with the following information:

Authentication Scheme Name: ClientCertificate

Pointer to specification text: This document

Notes This scheme does not rely on the Authorization header field.

7. Acknowledgements

Eric Rescorla helped identify the problem and formulate this mechanism. Julian Reschke and Michael Koeller provided excellent feedback. Andrei Popov observed that the information in the TLS CertificateRequest message is needed so that clients can select an appropriate certificate.

8. References

8.1. Normative References

[I-D.ietf-httpbis-p7-auth]
Fielding, R. and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Authentication", draft-ietf-httpbis-p7-auth-26 (work in progress), February 2014.

[I-D.ietf-tls-tls13]
Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3", draft-ietf-tls-tls13-01 (work in progress), April 2014.

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

[RFC4572] Lennox, J., "Connection-Oriented Media Transport over the Transport Layer Security (TLS) Protocol in the Session Description Protocol (SDP)", RFC 4572, July 2006.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, October 2006.

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

[RFC6234] Eastlake, D. and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, May 2011.

[X680] ITU-T, "Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation", ISO/IEC 8824-1:2002, 2002.

[X690] ITU-T, "Information technology - ASN.1 encoding Rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)", ISO/IEC 8825-1:2002, 2002.

8.2. Informational References

[I-D.ietf-httpbis-http2]
Belshe, M., Peon, R., and M. Thomson, "Hypertext Transfer Protocol version 2", draft-ietf-httpbis-http2-13 (work in progress), June 2014.

[I-D.thomson-tls-care]
Thomson, M., "Client Authentication Request Extension for (D)TLS", draft-thomson-tls-care-00 (work in progress), March 2014.

Author's Address

Martin Thomson
Mozilla
331 E Evelyn Street
Mountain View, CA 94041
US

Email: martin.thomson@gmail.com

HTTPbis
Internet-Draft
Updates: 6265 (if approved)
Intended status: Standards Track
Expires: October 8, 2016

M. West
Google, Inc
M. Goodwin
Mozilla

April 6, 2016

Same-site Cookies
draft-west-first-party-cookies-07

Abstract

This document updates RFC6265 by defining a "SameSite" attribute which allows servers to assert that a cookie ought not to be sent along with cross-site requests. This assertion allows user agents to mitigate the risk of cross-origin information leakage, and provides some protection against cross-site request forgery attacks.

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 8, 2016.

Table of Contents

1. Introduction

Section 8.2 of [RFC6265] eloquently notes that cookies are a form of ambient authority, attached by default to requests the user agent sends on a user's behalf. Even when an attacker doesn't know the contents of a user's cookies, she can still execute commands on the user's behalf (and with the user's authority) by asking the user agent to send HTTP requests to unwary servers.

Here, we update [RFC6265] with a simple mitigation strategy that allows servers to declare certain cookies as "same-site", meaning they should not be attached to "cross-site" requests (as defined in section 2.1).

Note that the mechanism outlined here is backwards compatible with the existing cookie syntax. Servers may serve these cookies to all user agents; those that do not support the "SameSite" attribute will simply store a cookie which is attached to all relevant requests, just as they do today.

1.1. Goals

These cookies are intended to provide a solid layer of defense-in-depth against attacks which require embedding an authenticated request into an attacker-controlled context:

1. Timing attacks which yield cross-origin information leakage (such as those detailed in [pixel-perfect]) can be substantially mitigated by setting the "SameSite" attribute on authentication cookies. The attacker will only be able to embed unauthenticated resources, as embedding mechanisms such as "<iframe>" will yield cross-site requests.

2. Cross-site script inclusion (XSSI) attacks are likewise mitigated by setting the "SameSite" attribute on authentication cookies. The attacker will not be able to include authenticated resources via "<script>" or "<link>", as these embedding mechanisms will likewise yield cross-site requests.

3. Cross-site request forgery (CSRF) attacks which rely on top-level navigation (HTML "<form>" POSTs, for instance) can also be mitigated by treating these navigational requests as "cross-site".

4. Same-site cookies have some marginal value for policy or regulatory purposes, as cookies which are not delivered with cross-site requests cannot be directly used for tracking purposes. It may be valuable for an origin to assert that its cookies should not be sent along with cross-site requests in order to limit its exposure to non-technical risk.

1.2. Examples

Same-site cookies are set via the "SameSite" attribute in the "Set-Cookie" header field. That is, given a server’s response to a user agent which contains the following header field:

```
Set-Cookie: SID=31d4d96e407aad42; SameSite=Strict
```

Subsequent requests from that user agent can be expected to contain the following header field if and only if both the requested resource and the resource in the top-level browsing context match the cookie.

2. Terminology and 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 [RFC2119].

This specification uses the Augmented Backus-Naur Form (ABNF) notation of [RFC5234].

Two sequences of octets are said to case-insensitively match each other if and only if they are equivalent under the "i;ascii-casemap" collation defined in [RFC4790].

The terms "active document", "ancestor browsing context", "browsing context", "document", "WorkerGlobalScope", "sandboxed origin browsing context flag", "parent browsing context", "the worker's Documents", "nested browsing context", and "top-level browsing context" are defined in [HTML].

"Service Workers" are defined in the Service Workers specification [SERVICE-WORKERS].

The term "origin", the mechanism of deriving an origin from a URI, and the "the same" matching algorithm for origins are defined in [RFC6454].

"Safe" HTTP methods include "GET", "HEAD", "OPTIONS", and "TRACE", as defined in Section 4.2.1 of [RFC7231].

The term "public suffix" is defined in a note in Section 5.3 of [RFC6265] as "a domain that is controlled by a public registry". For example, "example.com"'s public suffix is "com". User agents SHOULD use an up-to-date public suffix list, such as the one maintained by Mozilla at [PSL].

An origin's "registrable domain" is the origin's host's public suffix plus the label to its left. That is, "https://www.example.com"'s registrable domain is "example.com". This concept is defined more rigorously in [PSL].

The term "request", as well as a request's "client", "current url", "method", and "target browsing context", are defined in [FETCH].

2.1. "Same-site" and "cross-site" Requests

A request is "same-site" if its target's URI's origin's registrable domain is an exact match for the request's initiator's "site for cookies", and "cross-site" otherwise. To be more precise, for a

given request ("request"), the following algorithm returns "same-site" or "cross-site":

1. If "request"'s client is "null", return "same-site".

2. Let "site" be "request"'s client's "site for cookies" (as defined in the following sections).

3. Let "target" be the registrable domain of "request"'s current url.

4. If "site" is an exact match for "target", return "same-site".

5. Return "cross-site".

2.1.1. Document-based requests

The URI displayed in a user agent's address bar is the only security context directly exposed to users, and therefore the only signal users can reasonably rely upon to determine whether or not they trust a particular website. The registrable domain of that URI's origin represents the context in which a user most likely believes themselves to be interacting. We'll label this domain the "top-level site".

For a document displayed in a top-level browsing context, we can stop here: the document's "site for cookies" is the top-level site.

For documents which are displayed in nested browsing contexts, we need to audit the origins of each of a document's ancestor browsing contexts' active documents in order to account for the "multiple-nested scenarios" described in Section 4 of [RFC7034]. These document's "site for cookies" is the top-level site if and only if the document and each of its ancestor documents' origins have the same registrable domain as the top-level site. Otherwise its "site for cookies" is the empty string.

Given a Document ("document"), the following algorithm returns its "site for cookies" (either a registrable domain, or the empty string):

1. Let "top-document" be the active document in "document"'s browsing context's top-level browsing context.

2. Let "top-origin" be the origin of "top-document"'s URI if "top-document"'s sandboxed origin browsing context flag is set, and "top-document"'s origin otherwise.

3. Let "documents" be a list containing "document" and each of "document"'s ancestor browsing contexts' active documents.

4. For each "item" in "documents":

    1. Let "origin" be the origin of "item"'s URI if "item"'s sandboxed origin browsing context flag is set, and "item"'s origin otherwise.

    2. If "origin"'s host's registrable domain is not an exact match for "top-origin"'s host's registrable domain, return the empty string.

5. Return "top-site".

2.1.2. Worker-based requests

Worker-driven requests aren't as clear-cut as document-driven requests, as there isn't a clear link between a top-level browsing context and a worker. This is especially true for Service Workers [SERVICE-WORKERS], which may execute code in the background, without any document visible at all.

Note: The descriptions below assume that workers must be same-origin with the documents that instantiate them. If this invariant changes, we'll need to take the worker's script's URI into account when determining their status.

2.1.2.1. Dedicated and Shared Workers

Dedicated workers are simple, as each dedicated worker is bound to one and only one document. Requests generated from a dedicated worker (via "importScripts", "XMLHttpRequest", "fetch()", etc) define their "site for cookies" as that document's "site for cookies".

Shared workers may be bound to multiple documents at once. As it is quite possible for those documents to have distinct "site for cookie" values, the worker's "site for cookies" will be the empty string in cases where the values diverge, and the shared value in cases where the values agree.

Given a WorkerGlobalScope ("worker"), the following algorithm returns its "site for cookies" (either a registrable domain, or the empty string):

1. Let "site" be "worker"'s origin's host's registrable domain.

2. For each "document" in "worker"'s Documents:

1. Let "document-site" be "document"'s "site for cookies" (as defined in Section 2.1.1).

2. If "document-site" is not an exact match for "site", return the empty string.

3. Return "site".

2.1.2.2. Service Workers

Service Workers are more complicated, as they act as a completely separate execution context with only tangential relationship to the Document which registered them.

Requests which simply pass through a service worker will be handled as described above: the request's client will be the Document or Worker which initiated the request, and its "site for cookies" will be those defined in Section 2.1.1 and Section 2.1.2.1

Requests which are initiated by the Service Worker itself (via a direct call to "fetch()", for instance), on the other hand, will have a client which is a ServiceWorkerGlobalScope. Its "site for cookies" will be the registrable domain of the Service Worker's URI.

Given a ServiceWorkerGlobalScope ("worker"), the following algorithm returns its "site for cookies" (either a registrable domain, or the empty string):

1. Return "worker"'s origin's host's registrable domain.

3. Server Requirements

This section describes extensions to [RFC6265] necessary to implement the server-side requirements of the "SameSite" attribute.

3.1. Grammar

Add "SameSite" to the list of accepted attributes in the "Set-Cookie" header field's value by replacing the "cookie-av" token definition in Section 4.1.1 of [RFC6265] with the following ABNF grammar:

```
cookie-av      = expires-av / max-age-av / domain-av /
                 path-av / secure-av / httponly-av /
                 samesite-av / extension-av
samesite-av    = "SameSite" / "SameSite=" samesite-value
samesite-value = "Strict" / "Lax"
```

3.2. Semantics of the "SameSite" Attribute (Non-Normative)

The "SameSite" attribute limits the scope of the cookie such that it will only be attached to requests if those requests are "same-site", as defined by the algorithm in Section 2.1. For example, requests for "https://example.com/sekrit-image" will attach same-site cookies if and only if initiated from a context whose "site for cookies" is "example.com".

If the "SameSite" attribute's value is "Strict", or if the value is invalid, the cookie will only be sent along with "same-site" requests. If the value is "Lax", the cookie will be sent with "same-site" requests, and with "cross-site" top-level navigations, as described in Section 4.1.1.

The changes to the "Cookie" header field suggested in Section 4.3 provide additional detail.

4. User Agent Requirements

This section describes extensions to [RFC6265] necessary in order to implement the client-side requirements of the "SameSite" attribute.

4.1. The "SameSite" attribute

The following attribute definition should be considered part of the the "Set-Cookie" algorithm as described in Section 5.2 of [RFC6265]:

If the "attribute-name" case-insensitively matches the string "SameSite", the user agent MUST process the "cookie-av" as follows:

1. If "cookie-av"'s "attribute-value" is not a case-sensitive match for "Strict" or "Lax", ignore the "cookie-av".

2. Let "enforcement" be "Lax" if "cookie-av"'s "attribute-value" is a case-insensitive match for "Lax", and "Strict" otherwise.

3. Append an attribute to the "cookie-attribute-list" with an "attribute-name" of "SameSite" and an "attribute-value" of "enforcement".

4.1.1. "Strict" and "Lax" enforcement

By default, same-site cookies will not be sent along with top-level navigations. As discussed in Section 5.2, this might or might not be compatible with existing session management systems. In the interests of providing a drop-in mechanism that mitigates the risk of CSRF attacks, developers may set the "SameSite" attribute in a "Lax"

enforcement mode that carves out an exception which sends same-site cookies along with cross-site requests if and only if they are top-level navigations which use a "safe" (in the [RFC7231] sense) HTTP method.

Lax enforcement provides reasonable defense in depth against CSRF attacks that rely on unsafe HTTP methods (like "POST"), but do not offer a robust defense against CSRF as a general category of attack:

1. Attackers can still pop up new windows or trigger top-level navigations in order to create a "same-site" request (as described in section 2.1), which is only a speedbump along the road to exploitation.

2. Features like "<link rel='prerender'>" [prerendering] can be exploited to create "same-site" requests without the risk of user detection.

When possible, developers should use a session management mechanism such as that described in Section 5.2 to mitigate the risk of CSRF more completely.

4.2. Monkey-patching the Storage Model

Note: There's got to be a better way to specify this. Until I figure out what that is, monkey-patching!

Alter Section 5.3 of [RFC6265] as follows:

1. Add "samesite-flag" to the list of fields stored for each cookie. This field's value is one of "None", "Strict", or "Lax".

2. Before step 11 of the current algorithm, add the following:

    1. If the "cookie-attribute-list" contains an attribute with an "attribute-name" of "SameSite", set the cookie's "samesite-flag" to "attribute-value" ("Strict" or "Lax"). Otherwise, set the cookie's "samesite-flag" to "None".

    2. If the cookie's "samesite-flag" is not "None", and the request which generated the cookie's client's "site for cookies" is not an exact match for "request-uri"'s host's registrable domain, then abort these steps and ignore the newly created cookie entirely.

4.3. Monkey-patching the "Cookie" header

Note: There's got to be a better way to specify this. Until I figure out what that is, monkey-patching!

Alter Section 5.4 of [RFC6265] as follows:

1. Add the following requirement to the list in step 1:

    * If the cookie's "samesite-flag" is not "None", and the HTTP request is cross-site (as defined in Section 2.1 then exclude the cookie unless all of the following statements hold:

        1. "samesite-flag" is "Lax"

        2. The HTTP request's method is "safe".

        3. The HTTP request's target browsing context is a top-level browsing context.

Note that the modifications suggested here concern themselves only with the "site for cookies" of the request's client, and the registrable domain of the resource being requested. The cookie's "domain", "path", and "secure" attributes do not come into play for these comparisons.

5. Authoring Considerations

5.1. Defense in depth

"SameSite" cookies offer a robust defense against CSRF attack when deployed in strict mode, and when supported by the client. It is, however, prudent to ensure that this designation is not the extent of a site's defense against CSRF, as same-site navigations and submissions can certainly be executed in conjunction with other attack vectors such as cross-site scripting.

Developers are strongly encouraged to deploy the usual server-side defenses (CSRF tokens, ensuring that "safe" HTTP methods are idempotent, etc) to mitigate the risk more fully.

Additionally, client-side techniques such as those described in [app-isolation] may also prove effective against CSRF, and are certainly worth exploring in combination with "SameSite" cookies.

5.2. Top-level Navigations

Setting the "SameSite" attribute in "strict" mode provides robust defense in depth against CSRF attacks, but has the potential to confuse users unless sites' developers carefully ensure that their session management systems deal reasonably well with top-level navigations.

Consider the scenario in which a user reads their email at MegaCorp Inc's webmail provider "https://example.com/". They might expect that clicking on an emailed link to "https://projects.com/secret/project" would show them the secret project that they're authorized to see, but if "projects.com" has marked their session cookies as "SameSite", then this cross-site navigation won't send them along with the request. "projects.com" will render a 404 error to avoid leaking secret information, and the user will be quite confused.

Developers can avoid this confusion by adopting a session management system that relies on not one, but two cookies: one conceptualy granting "read" access, another granting "write" access. The latter could be marked as "SameSite", and its absence would provide a reauthentication step before executing any non-idempotent action. The former could drop the "SameSite" attribute entirely, or choose the "Lax" version of enforcement, in order to allow users access to data via top-level navigation.

5.3. Mashups and Widgets

The "SameSite" attribute is inappropriate for some important use-cases. In particular, note that content intended for embedding in a cross-site contexts (social networking widgets or commenting services, for instance) will not have access to such cookies. Cross-site cookies may be required in order to provide seamless functionality that relies on a user's state.

Likewise, some forms of Single-Sign-On might require authentication in a cross-site context; these mechanisms will not function as intended with same-site cookies.

6. Privacy Considerations

6.1. Server-controlled

Same-site cookies in and of themselves don't do anything to address the general privacy concerns outlined in Section 7.1 of [RFC6265]. The attribute is set by the server, and serves to mitigate the risk of certain kinds of attacks that the server is worried about. The user is not involved in this decision. Moreover, a number of side-

channels exist which could allow a server to link distinct requests even in the absence of cookies. Connection and/or socket pooling, Token Binding, and Channel ID all offer explicit methods of identification that servers could take advantage of.

6.2. Pervasive Monitoring

As outlined in [RFC7258], pervasive monitoring is an attack. Cookies play a large part in enabling such monitoring, as they are responsible for maintaining state in HTTP connections. We considered restricting same-site cookies to secure contexts [secure-contexts] as a mitigation but decided against doing so, as this feature should result in a strict reduction in the number of cookies floating around in cross-site contexts. That is, even if "http://not-example.com" embeds a resource from "http://example.com/", that resource will not be "same-site", and "http://example.com"'s cookies simply cannot be used to correlate user behavior across distinct origins.

7. References

7.1. Normative References

[FETCH] van Kesteren, A., "Fetch", n.d., <https://fetch.spec.whatwg.org/>.

[HTML] Hickson, I., Pieters, S., van Kesteren, A., Jaegenstedt, P., and D. Denicola, "HTML", n.d., <https://html.spec.whatwg.org/>.

[PSL] "Public Suffix List", n.d., <https://publicsuffix.org/list/>.

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

[RFC4790] Newman, C., Duerst, M., and A. Gulbrandsen, "Internet Application Protocol Collation Registry", RFC 4790, DOI 10.17487/RFC4790, March 2007, <http://www.rfc-editor.org/info/rfc4790>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <http://www.rfc-editor.org/info/rfc5234>.

[RFC6265] Barth, A., "HTTP State Management Mechanism", RFC 6265, DOI 10.17487/RFC6265, April 2011, <http://www.rfc-editor.org/info/rfc6265>.

[RFC6454] Barth, A., "The Web Origin Concept", RFC 6454, DOI 10.17487/RFC6454, December 2011, <http://www.rfc-editor.org/info/rfc6454>.

[RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, <http://www.rfc-editor.org/info/rfc7231>.

[RFC7258] Farrell, S. and H. Tschofenig, "Pervasive Monitoring Is an Attack", BCP 188, RFC 7258, DOI 10.17487/RFC7258, May 2014, <http://www.rfc-editor.org/info/rfc7258>.

[SERVICE-WORKERS] Russell, A., Song, J., and J. Archibald, "Service Workers", n.d., <http://www.w3.org/TR/service-workers/>.

7.2. Informative References

[app-isolation] Chen, E., Bau, J., Reis, C., Barth, A., and C. Jackson, "App Isolation - Get the Security of Multiple Browsers with Just One", n.d., <http://www.collinjackson.com/research/papers/appisolation.pdf>.

[pixel-perfect] Stone, P., "Pixel Perfect Timing Attacks with HTML5", n.d., <http://www.contextis.com/documents/2/Browser_Timing_Attacks.pdf>.

[prerendering] Bentzel, C., "Chrome Prerendering", n.d., <https://www.chromium.org/developers/design-documents/prerender>.

[RFC7034] Ross, D. and T. Gondrom, "HTTP Header Field X-Frame-Options", RFC 7034, DOI 10.17487/RFC7034, October 2013, <http://www.rfc-editor.org/info/rfc7034>.

[samedomain-cookies] Goodwin, M. and J. Walker, "SameDomain Cookie Flag", 2011, <http://people.mozilla.org/~mgoodwin/SameDomain/samedomain-latest.txt>.

[secure-contexts]
West, M., "Secure Contexts", n.d., <https://w3c.github.io/webappsec-secure-contexts/>.

Appendix A. Acknowledgements

The same-site cookie concept documented here is indebited to Mark Goodwin's and Joe Walker's [samedomain-cookies]. Michal Zalewski, Artur Janc, Ryan Sleevi, and Adam Barth provided particularly valuable feedback on this document.

Authors' Addresses

Mike West
Google, Inc

Email: mkwst@google.com
URI: https://mikewest.org/

Mark Goodwin
Mozilla

Email: mgoodwin@mozilla.com
URI: https://www.computerist.org/

HTTPbis
Internet-Draft
Updates: 6265 (if approved)
Intended status: Standards Track
Expires: October 22, 2015

M. West
Google, Inc

April 20, 2015

Origin Cookies
draft-west-origin-cookies-01

Abstract

This document updates RFC6265, defining the "origin" attribute for cookies and the "Origin-Cookie" header field, which together allow servers to choose to harmonize the security policy of their cookies with the same-origin policy which governs other available client-side storage mechanisms.

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 22, 2015.

Copyright Notice

Table of Contents

1. Introduction

Cookies, as defined by [RFC6265], diverge from the web's general security policy in a number of ways which may be surprising to implementers and authors who haven't carefully read that document's discussion of "domain matching", and "path matching", or who ignored the admonitions regarding "Weak Confidentiality" and "Weak Integrity".

This document updates [RFC6265], describing a mechanism by which servers can opt-in to harmonizing cookies' security policy with the same-origin policy, as described in [RFC6454]. User agents that support these "origin cookies" will ignore a "Set-Cookie" header's value's "Path", "Domain", and "Secure" attributes if an "Origin" attribute is present, instead tying the cookie to the origin that set it. These "origin cookies" will be returned in a new "Origin-Cookie" header field (see Section 4.4 for detail), separating them from non-origin cookies in a way a server can easily distinguish.

Harmonizing with the same-origin policy mitigates the confidentiality and integrity risks noted above by ensuring that origin cookies are not influenced by malicious code running on a server's subdomain or a non-standard port or scheme.

Note that the mechanism outlined here is backwards compatible with the existing cookie syntax. Servers may serve origin cookies to all user agents; those that do not support the "Origin" attribute will simply store a non-origin cookie, just as they do today.

1.1. Examples

Origin cookies are set via the "Origin" attribute in the "Set-Cookie" header field. That is, given a server's response to a user agent which contains the following header field:

```
Set-Cookie: SID=31d4d96e407aad42; Secure; HttpOnly; Origin
```

Subsequent requests from that user agent can be expected to contain the following header field:

```
Origin-Cookie: SID=31d4d96e407aad42
```

Non-origin cookies are returned in the "Cookie" header field as usual. If both non-origin and origin cookies are present for an origin, then both a "Cookie" and "Origin-Cookie" header field will be present. That is, given a server's response to a user agent which contains the following header fields:

```
Set-Cookie: SID=31d4d96e407aad42; Origin
Set-Cookie: lang=en-US;
```

Subsequent requests from that user agent can be expected to contain the following header fields:

```
Cookie: lang=en-US
Origin-Cookie: SID=31d4d96e407aad42
```

User agents that support origin cookies are required to advertise their support for such by sending an "Origin-Cookie" header whenever a "Cookie" header is sent. That is, given the following server response:

```
Set-Cookie: lang=en-US; Secure; HttpOnly
```

Subsequent requests from a user agent that supports origin cookies can be expected to contain the following header fields:

```
Cookie: lang=en-US
Origin-Cookie:
```

Note that the "Origin-Cookie" field is empty.

2. Terminology and 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 [RFC2119].

This specification uses the Augmented Backus-Naur Form (ABNF) notation of [RFC5234].

Two sequences of octets are said to case-insensitively match each other if and only if they are equivalent under the "i;ascii-casemap" collation defined in [RFC4790].

3. Server Requirements

This section describes extensions to [RFC6265] necessary to implement the server-side requirements of the "Origin" attribute.

3.1. Grammar

Add "Origin" to the list of accepted attributes in the "Set-Cookie" header field’s value by replacing the "cookie-av" token definition in Section 4.1.1 of [RFC6265] with the following ABNF grammar:

```
cookie-av = expires-av / max-age-av / domain-av /
            path-av / secure-av / httponly-av /
            origin-av / extension-av
origin-av = "origin"
```

3.2. Semantics of the "Origin" Attribute (Non-Normative)

The "Origin" attribute limits the scope of the cookie such that it will only be attached to requests if those request match the origin which set the cookie. For example, requests for "https://example.com/" will attach origin cookies if and only if those cookies were set by "https://example.com/".

The changes to the "Cookie" header field suggested in Section 4.3 provide additional detail.

3.3. The <spanx style="verb">Origin-Cookie</spanx> header

3.3.1. Syntax

The user agent sends stored origin cookies to the origin server in the "Origin-Cookie" header. If the server conforms to the requirements in Section 3 (and the user agent conforms to the requirements in Section 4), the user agent will send an "Origin-Cookie" header which conforms to the following grammar:

origin-cookie-header = "Origin-Cookie:" OWS [cookie-string] OWS

3.3.2. Semantics

The semantics of the "cookie-string" are the same as those of the same token in the "Cookie" header.

Note, however, that the "Origin-Cookie" header MAY be empty, and MUST be sent with every request, even if no origin cookies are present in the cookie store. This allows conformant servers to detect a user agent's support for origin cookies, and therefore to make a secure decision about whether or not to fallback to searching through the "Cookie" header for specific cookies. See Section 5.2 for details.

4. User Agent Requirements

This section describes extensions to [RFC6265] necessary in order to implement the client-side requirements of the "Origin" attribute and "Origin-Cookie" header field.

4.1. The "Origin" attribute

The following attribute definition should be considered part of the the "Set-Cookie" algorithm as described in Section 5.2 of [RFC6265]:

If the attribute-name case-insensitively matches the string "Origin", the user agent MUST append an attribute to the cookie-attribute-list with an attribute-name of "Origin" and an empty attribute-value.

4.2. Monkey-patching the Storage Model

Note: There's got to be a better way to specify this. Until I figure out what that is, monkey-patching!

Alter Section 5.3 of [RFC6265] as follows:

1. Add "origin" and "origin-flag" to the list of fields stored about each cookie.

2. Before step 11 of the current algorithm, add the following:

    1. If the "cookie-attribute-list" contains an attribute with an "attribute-name" of "Origin":

        1. Set the cookie’s "domain" attribute to the empty string.

        2. Set the cookie’s "host-only-flag" to true.

        3. Set the cookie’s "origin" to the origin of "request-uri", as defined by Section 4 of [RFC6454].

        4. Set the cookie’s "origin-flag" to true.

        5. Set the cookie’s "path" attribute to the empty string.

        6. Set the cookie’s "secure-only-flag" to false.

        Otherwise: set the cookie’s "origin-flag" to false, and its "origin" to "null".

    2. If the newly created cookie’s "origin-flag" is set to true, and the cookie store contains a cookie with the same"name", "origin", and "origin-flag" as the newly created cookie:

        1. Let "old-cookie" be the existing cookie with the same "name", "origin", and "origin-flag" as the newly created cookie.

        2. Update the "creation-time" of the newly created cookie to match the "creation-time" of "old-cookie".

        3. Remove "old-cookie" from the cookie store.

3. Change the priority order for excess cookie removal to the following:

    1. Expired cookies.

    2. Cookies whose "origin-flag" is false that share a "domain" field with more than a predetermined number of other cookies.

    3. Cookies whose "origin-flag" is true that share a "domain" field with more than a predetermined number of other cookies.

    4. Cookies whose "origin-flag" is false.

    5. All cookies.

4.3. Monkey-patching the "Cookie" header

Note: There's got to be a better way to specify this. Until I figure out what that is, monkey-patching!

Alter Section 5.4 of [RFC6265] as follows:

1. Add the following requirement to the list in step 1:

    * The cookie's "origin-flag" is false.

4.4. The "Origin-Cookie" header field

The user agent includes stored cookies whose "origin-flag" is set in the "Origin-Cookie" request header. When the user agent generates an HTTP request, it MUST NOT attach more than one "Origin-Cookie" header field.

A user agent MAY omit the "Origin-Cookie" header in its entirety. For example, the user agent may wish to block sending cookies during "third-party" requests. If, however, a "Cookie" header is sent, a user agent MUST send an "Origin-Cookie" header.

If the user agent does attach an "Origin-Cookie" header field to an HTTP request, the user agent MUST send the "cookie-string" as defined below as the value of the header field.

The user agent MUST use an algorithm equivalent to the following algorithm to compute the "cookie-string" from a cookie store and a "request-uri":

1. Let "cookie-list" be the set of cookies from the cookie store that meets all the following requirements:

    * The cookie's "origin-flag" is true.

    * The cookie's "origin" matches the origin of "request-uri". [RFC6454]

2. The user agent SHOULD sort the "cookie-list" in the following order:

    * Cookies with earlier "creation-time"s are listed before cookies with later "creation-time"s.

3. Update the "last-access-time" of each cookie in the "cookie-list" to the current date and time.

4. Serialize the "cookie-list" into a "cookie-string" by processing each cookie in the "cookie-list" in order:

    1. Output the cookie's "name", the %x3D ("=") character, and the cookie's "value".

    2. If there is an unprocessed cookie in the "cookie-list", output the characters %x3B and %x20 ("; ").

5. Security Considerations

The security considerations listed in Section 8 of [RFC6265] apply equally to origin cookies, with the exceptions of Sections 8.6 ("Weak Confidentiality") and Sections 8.7 ("Weak Isolation"), both of which are substantially improved if the "Origin" attribute is set. Further:

5.1. Paths are ignored

Origin cookies will break the (flawed) "Path"-based isolation strategy which some servers may be attempting to implement. If a server has used the "Path" attribute to limit cookies to specific areas of a site (say "/admin"), then they may be surprised by origin cookies' pathless behavior.

That said, paths offer little to no protection against malicious code. The origin is the only security boundry enforced rigorously by user agents; it is therefore the only security boundry that server operators ought to rely on for isolation.

5.2. Downgrade attacks

If a server chooses to scan both the "Origin-Cookie" and "Cookie" headers in order to provide backwards compatibility with user agents that don't support origin cookies, it ought to be done carefully. Careless fallback strategies can provide a window of opportunity for an attacker to inject cookies with the same name as origin cookies from a subdomain, bypassing origin cookies' main advantage.

- If the "Origin-Cookie" header is present, servers SHOULD NOT check the "Cookie" header for cookies which it set as origin cookies.

- If a user agent is known to support origin cookies, servers SHOULD check only the "Origin-Cookie" header for origin cookies, and SHOULD NOT fallback to the "Cookie" header if the "Origin-Cookie" header is not present, or if a particular cookie is not found there.

6. IANA Considerations

The permanent message header field registry (see [RFC3864]) shall be updated with the following registration:

6.1. Origin-Cookie

- Header field name: Origin-Cookie
- Applicable protocol: http
- Status: standard
- Author/Change controller: IETF
- Specification document: This specification (see Section 4.4)

7. References

7.1. Normative References

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

[RFC4790] Newman, C., Duerst, M., and A. Gulbrandsen, "Internet Application Protocol Collation Registry", RFC 4790, March 2007.

[RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

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

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

7.2. Informative References

[RFC3864] Klyne, G., Nottingham, M., and J. Mogul, "Registration Procedures for Message Header Fields", BCP 90, RFC 3864, September 2004.

[draft-abarth-cake-01] Barth, A., "Origin Cookies", September 2011, <https://tools.ietf.org/html/draft-abarth-cake-01>.

[origin-cookies-w2sp]
Bortz,, A., Barth, A., and A. Czeskis, "Origin Cookies: Session Integrity for Web Applications", 2011, <http://w2spconf.com/2011/papers/session-integrity.pdf>.

Appendix A. Acknowledgements

The origin cookie concept documented here is heavily indebted to and based upon Adam Barth's [draft-abarth-cake-01] document, as well as Andrew Bortz, Adam Barth, and Alexei Czeskis' paper [origin-cookies-w2sp].

Author's Address

Mike West
Google, Inc

Email: mkwst@google.com
URI: https://mikewest.org/