HTTPBis Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 18, 2014

S. Loreto
J. Mattsson
R. Skog
H. Spaak
Ericsson
G. Gus
D. Druta
M. Hafeez
AT&T

February 14, 2014

Explicit Trusted Proxy in HTTP/2.0
draft-loreto-httpbis-trusted-proxy20-01

Abstract

The purpose of this Internet Draft is to continue the discussion on explicit and trusted proxy as intermediary of HTTP2 traffic.

The httpbis wg has agreed on the HTTP2 usage with HTTP URIs, with or without TLS, without any constraints from the standard (see: issue 314).

To distinguish between an HTTP2 connection meant to transport "https" URIs resources and an HTTP2 connection meant to transport "http" URIs resource, the draft proposes to

register a new value in the Application Layer Protocol negotiation (ALPN) Protocol IDs registry specific to signal the usage of HTTP2 to transport "http" URIs resources: h2clr.

This document describes two alternative methods for an user-agent to automatically discover and for an user to provide consent for a Trusted Proxy to be securely involved when he or she is requesting an HTTP URI resource over HTTP2 with TLS. The consent is supposed to be per network access. The draft also describes the role of the Trusted Proxy in helping the user to fetch HTTP URIs resource when the user has provided consent to the Trusted Proxy to be involved.

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 18, 2014.

Table of Contents

1. Introduction

The current (i.e. HTTP/1.1) and previous HTTP protocol versions have provided room and made it possible to create a Web ecosystem of various kind of proxies and intermediaries: cache servers, security gateways, web accelerators, content filters, and many others. In some cases their presence is explicit (i.e. configured proxies), and in other they are completely transparent to the end user (i.e. transparent proxy, reverse proxy).

The success and the presence of those intermediaries is also a problem for the evolution of the HTTP protocol as the intermediary behaviour on protocol extension and especially on alternative wire format of the protocol is not predictable. This not predictable behaviour can lead to difficulties to deploy a new version of the protocol before the intermediate are themselves update to support it (see the difficult in deploying the WebSocket Protocol [RFC6455] in clear). Relying on establishing an HTTPS tunnel has then become the popular way to bypass the intermediate proxies as it provides reliable deployment model for web protocols: the encrypted tunnel obfuscates the data from all intermediaries.

HTTPS tunnels, while speeding up the deployment, makes it difficult for a forward proxy and other intermediaries to be used to allow caching, to provide anonymity to a User-Agent, or to provide security by using an application-layer firewall to inspect the HTTP traffic on behalf of the User-Agent (e.g. to protect it against cross-site scripting attacks). HTTPS tunnels also remove the possibility to enhance delivery performance based on the knowledge of the network status, and this become an important limitation especially with HTTP2 when multiple streams are multiplexed on top of the same TCP connection.

In the presence of HTTPS tunnels the only possibility to have a trusted intermediary (and still providing confidentiality towards not trusted elements in the network) is then to have separate TLS sessions between the User-Agent and the proxy, on one side, and the proxy and the server on the other side (see Figure 1).

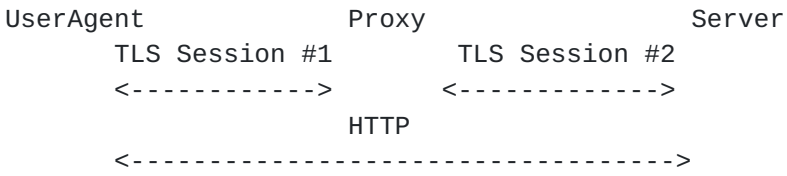

Figure 1: A proxied HTTPS session, with two independent TLS sessions

Several drafts analysing the role and the requirements for proxy have

been submitted:

1. [I-D.nottingham-http-proxy-problem] discusses the use and configuration of proxies in HTTP, pointing out problems in the currently deployed Web infrastructure along the way

2. [I-D.vidya-httpbis-explicit-proxy-ps] describes the issues with HTTP proxies for TLS protected traffic and motivates the need for explicit proxying capability in HTTP. It also presents the goals that such a solution would need to satisfy and some example solution directions.

3. [I-D.rpeon-httpbis-exproxy] describes a method for connecting to a proxy via a secure channel, allowing, disallowing, and detecting any transforms that the proxy may perform, and allowing the proxy to connect via secure channel to another site on the user's behalf..

Use cases in form of stories for proxies are also listed in the wiki Proxy-User-Stories [1] and analysed in a matrix form in Trusted Proxy Use Case Analysis and Alternatives [2].

1.1. Explicit and Trusted Proxy

The httpbis wg has agreed on the HTTP2 usage with HTTP URIs, with or without TLS, without any constraints from the standard (see issue 314 [3]), however the role of explicit and trusted proxy as intermediary in HTTP2 is still to be specified in the standard.

The main aim for this document is to analyse and define the role of an Explicit and Trusted proxy for HTTP URI resources transported over HTTP2 with TLS.

To distinguish between an HTTP2 connection meant to transport "https" URIs resources and an HTTP2 connection meant to transport "http" URIs resource, the draft proposes to

> register a new value in the Application Layer Protocol negotiation (ALPN) Protocol IDs registry specific to signal the usage of HTTP2 to transport "http" URIs resources: h2clr.

This document describes two alternative methods for an user-agent to automatically discover and then for an user to provide consent for a Trusted Proxy to be securely involved when an HTTP URI resource is requested.

Section 3.1 proposes a solution based on sending a proxy certificate in the TLS handshake.

Section 3.2 proposes a solution based on the presence of a Captive Proxy.

The consent is supposed to be per network access.

Section 4 describes the role of the Trusted Proxy in helping the user to fetch HTTP URIs resource when the user has provided consent to the Trusted Proxy to be involved.

2. 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].

This document defines the following terms:

Explicit proxy: an intermediary that explicit signals its presence to the User-Agent and eventually also to the Origin Server.

Trusted proxy: an intermediary that in order to authenticate itself provides a proxy certificate issued by a trusted certification authority, where the root certificate is assumed to be preinstalled in the User-Agent.

3. How a Forward Proxy establishes the trust

An explicit and trusted proxy is preferable to a transparent MITM proxy as an intermediary of HTTP2 traffic. However how a proxy is interposed into a network flow often has great affect on perceptions of its operation by end users and origin servers.

The following sections describe two mechanisms by which a proxy can establish trust.

3.1. Proxy certificate

To help HTTP User-Agents identify and distinguish proxies from other servers (e.g. web servers), a certification authority can issue special public key certificates to trusted proxies. More specifically, the certification authority can use the Extended Key Usage extension as specified in [RFC5280] to indicate a key purpose "proxyAuthentication" (a new object identifier needs to be assigned by IANA for this key purpose). The certification authority also marks this Extended Key Usage extension as critical. As the user needs to have high trust in the Proxy, the validation procedure for

proxy certificates should be more rigorous than for ordinary SSL certificates. All proxy certificates should therefore be Extended Validation (EV) SSL Certificates.

3.1.1. TLS Handshake with Proxy certificate

When a (TLS and HTTP) User-Agent receives a Server Certificate message, it checks whether the certificate contains an Extended Key Usage extension and if so whether the "proxyAuthentication" key purpose id is included. If it is included, the User-Agent concludes that the certificate belongs to a proxy. It then verifies that the proxy certificate is a valid EV certificate. The user-agent then SHOULD secure user consent.

When the user has given consent to the use of a proxy, the User-Agent SHOULD store this consent so that the user does not have to give consent for each new TLS connection involving the proxy. The consent SHOULD be limited to the specific access and MAY be limited to a single connection to that access or limited in time. How the consent information is stored is implementation specific, but as a network may have several proxies (for network resilience) it is RECOMMENDED that the consent is only tied to the Subject field of the proxy certificate so that the consent applies to all proxy certificates with the same name.

If the user has previously given consent to use the specific proxy and the user-agent has stored that, the user-agent may conclude that the user has given consent without asking the user again.

If the user provides consent, the User-Agent continues the TLS handshake with the proxy.

Figure 2: TLS Handshake with Proxy certificate

3.1.2. Opt Out

If the user does not give consent, or decides to opt out from the proxy for a specific connection, the user-agent will negotiate HTTP2 connection using "h2" value in the Application Layer Protocol Negotiation (ALPN) extension field. The proxy will then notice that the TLS connection is to be used for a https resource or for a http resource for which the user wants to opt out from the proxy. The proxy will then forward the ClientHello message to the Server and the TLS connection will be end-to-end between the user-agent and the Server.

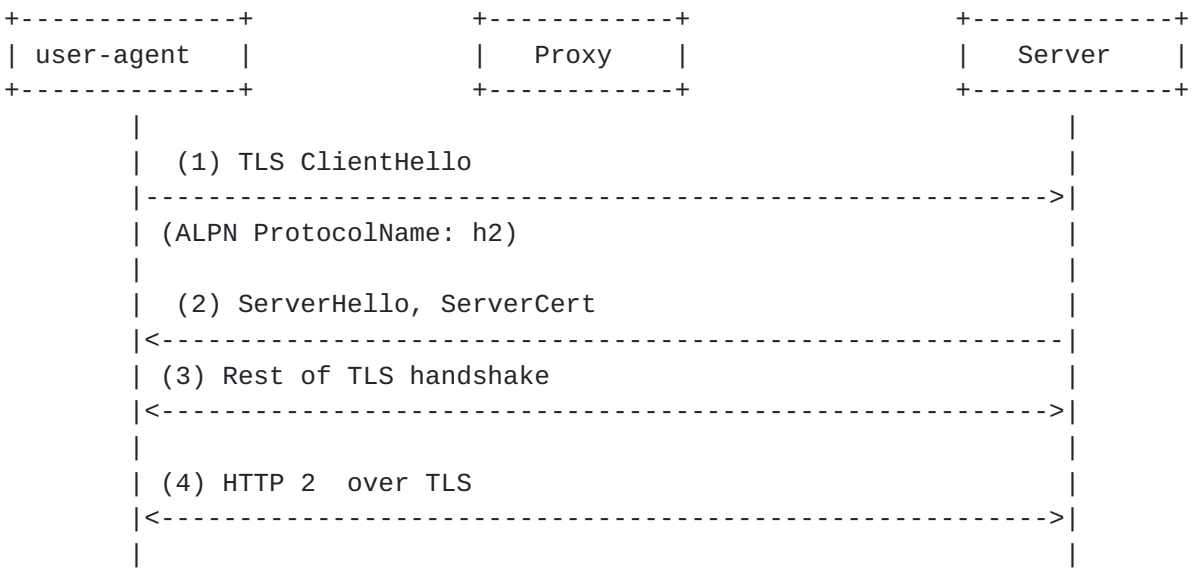

Figure 3: Opt Out

3.2. Captive Proxy

The Captive Proxy mechanism suggests that a Secure Proxy may indicate its presence and identity by intercepting a TLS ClientHello message, analysing the application layer protocol negotiation extension field and if it contains "h2clr" value forcing the User-Agent to redirect to a secure page on a Portal where the user is request to consent to the presence of the Secure Proxy for all the request towards "http" URI resources while it stays in that network (see Figure 4).

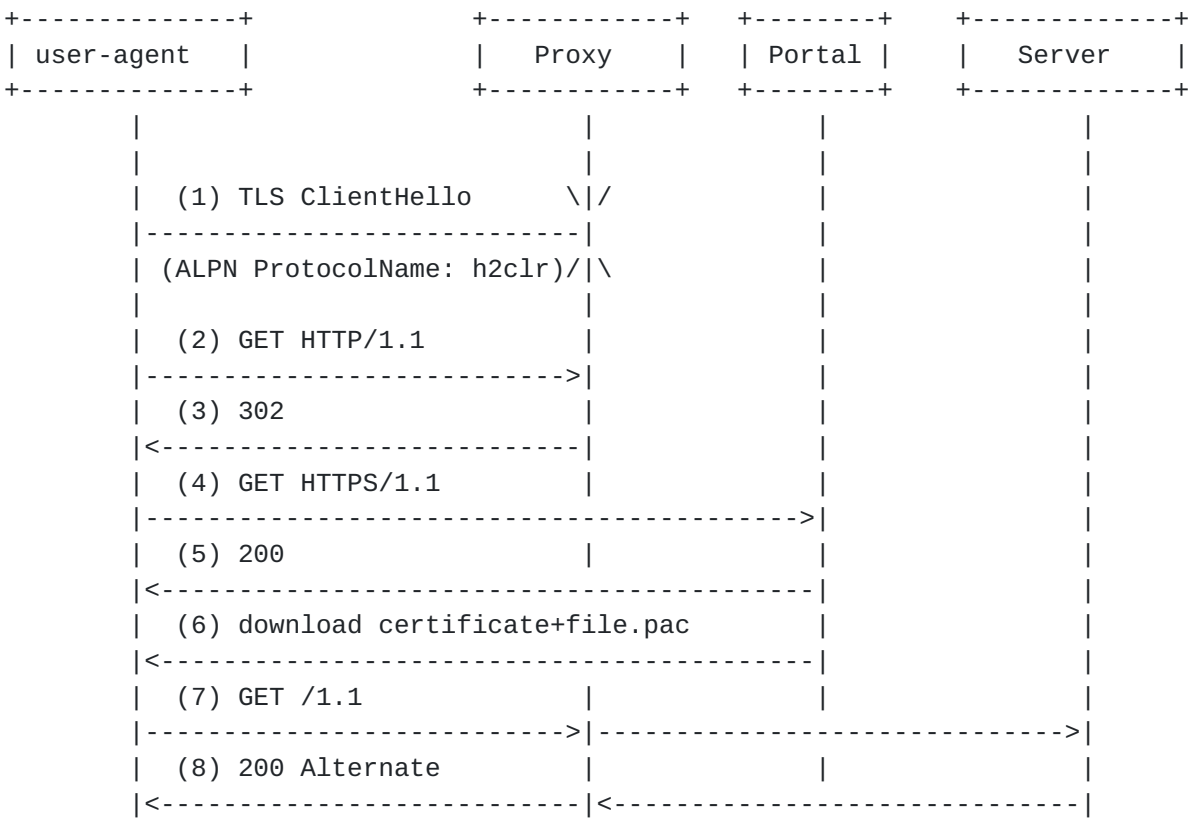

Figure 4: Captive Proxy flow

Figure 4 steps are explained below

(1) A User-Agent that makes a request to an "http" URI without prior knowledge about support for HTTP2 uses TLS, with the application level protocol negotiation extension inserting the h2clr tag, to start the HTTP2 connection. The Proxy intercepts the TLS ClientHello analyses the application layer protocol negotiation extension field and if it contains "h2clr" value it blocks the TLS ClientHello.

(2), (3), (4) The User-Agent receives an TLS Alert (e.g. unsupported_extension) indicating that that the resource does not support HTTP2 and it will fall back to HTTP/1.1 and will issue a GET /1.1. The proxy intercepts the GET and redirects the User-Agent to a portal secure page.

Note that the portal page is an https URI resource, so that the user can be sure of the identity of the portal.

(5) When the User-Agent arrives to the portal page it becomes aware of the existence of a Proxy in the access network and receives a consent request for the proxy to stay in the path for HTTP URI resources. The user-agent then SHOULD secure user consent.

When the user has given consent to the use of a proxy, both the User-Agent and the Proxy SHOULD store this consent so that the user does not have to give consent for each new TLS connection involving the proxy.

(6) If the user provides consent, then it will start to download a certificate identifying the proxy. The proxy certificate should be issued by a trusted certification authority, where the root certificate is assumed to be preinstalled in the User-Agent. The proxy certificate SHOULD be stored in a proxy certificate repository distinct from the repository for the server certificates.

The portal page may also offer the possibility to download a signed (with the proxy certificate) pac file that contains all the information to automatically configure the proxy in the User-Agent.

(7) (8) The User-Agent is then forwarded to the origin initially requested and it receives the requested content, however the Proxy will insert an Alternate header that will tell the User-Agent to asynchronously upgrade to HTTP2.

The presence of the pac file or of the proxy certificate for that access network will automatically configure the User-Agent in the Proxy mode.

3.2.1. Opt Out

This section specifies how an user can opt out (i.e. refuse) the presence of a Proxy for all the subsequent requests toward "http" URI resources while it stays in that network (see Figure 5).

If the user decides to opt out from the proxy, the User-Agent will start (step 6 in the figure below) to negotiate HTTP2 connection only using "h2" vale in the Application Layer Protocol Negotiation (ALPN) extension field, while staying in that access network.

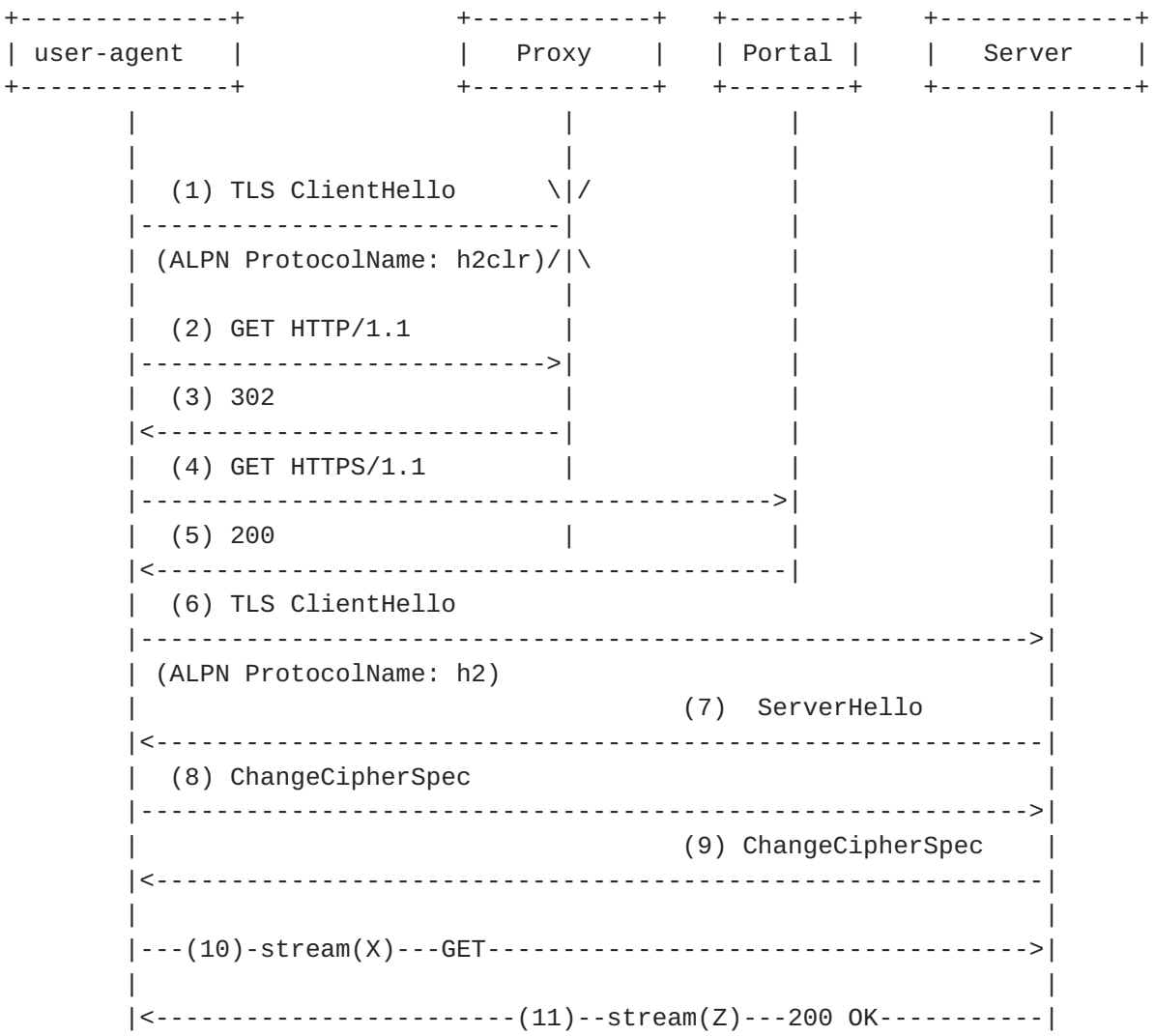

Figure 5: Proxy Opt Out

4. Explicit Proxy behaviour

This section describes the role of the Trusted Proxy in helping the user to fetch HTTP URI resources when the user has provided consent to the Trusted Proxy to be involved.

4.1. Secure Forward Proxy towards HTTP2 origin server

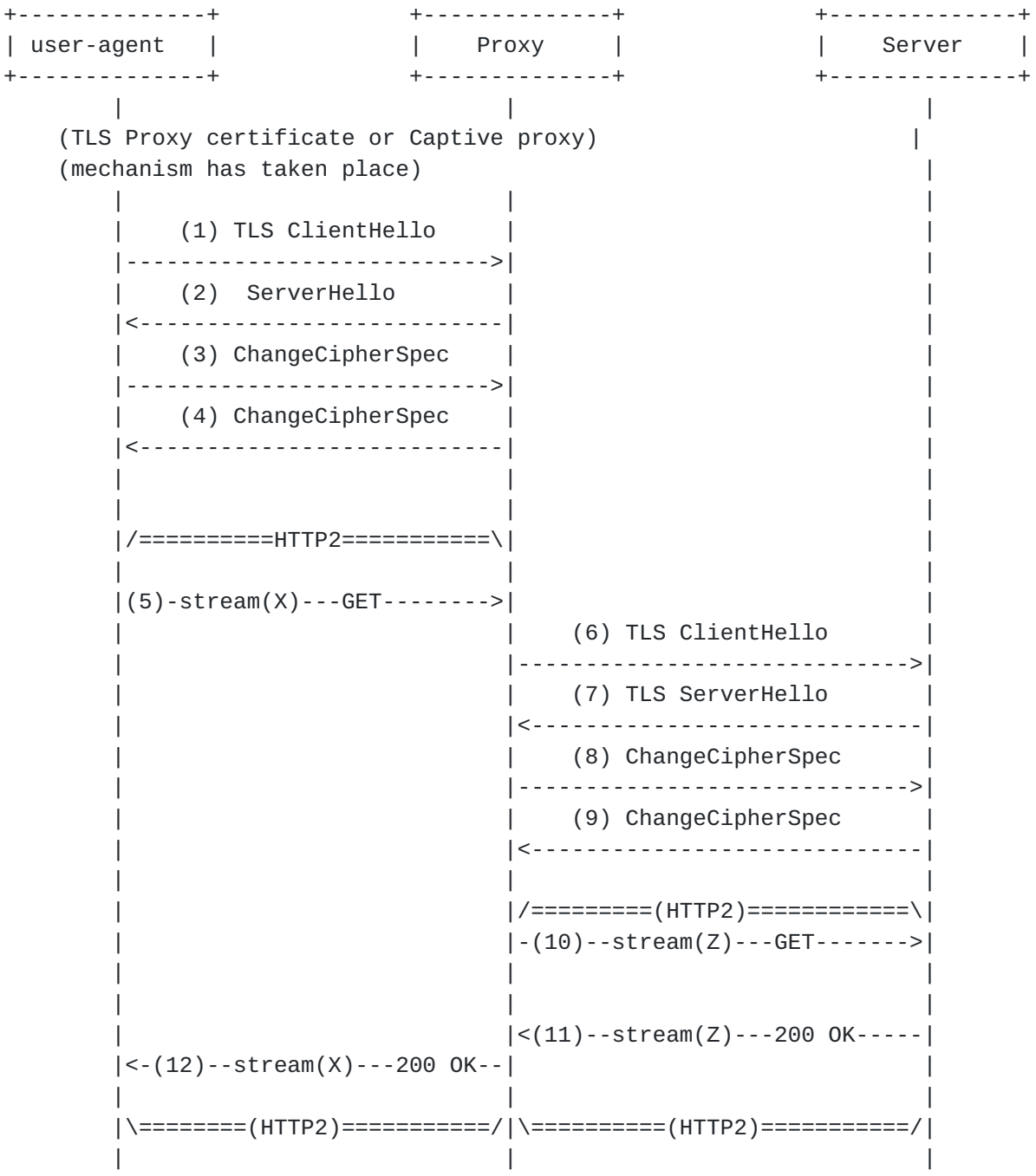

Figure 6: requesting an HTTP resource

(0) The TLS Proxy Announcement (Section 3.1) or Captive Proxy (Section 3.2) mechanism has already taken place, so the User-Agent is now configure in the proxy mode.

(1). . .(4) For each "http" URI resource towards a not yet contacted Server Origin, then the User-Agent negotiates a new TLS session, using the ALPN extension containing the "h2clr" tag, to establish an HTTP2 connection.

(5) The User-Agent will then use the streams in the HTTP2 connection to request any resources hosted on that Origin Server.

(6),(7),(8),(9) In the case the Proxy receives a request for an URI resource towards a not yet contacted Server Origin, then the trusted Proxy negotiates a new TLS session, using the ALPN extension containing the "h2clr" tag, to establish an HTTP2 connection.

(10) Once the Proxy has established the HTTP2 connection toward the origin, then it picks one stream to forward the request

(11) (12) The Proxy forward the answer it receives from the Origin Server to the User-Agent.

4.2. Secure Forward Proxy towards HTTP/1.1 Origin Server

In the case the "http" URI resources requested by the user-agent will be only available over HTTP/1.1 or the proxy does not have a previous knowledge about it, the proxy will then attempt to contact the resource based on its knowledge. If the proxy does not know if the resource is a HTTP2 or not it will contact it using HTTP1.1 (see Figure 7).

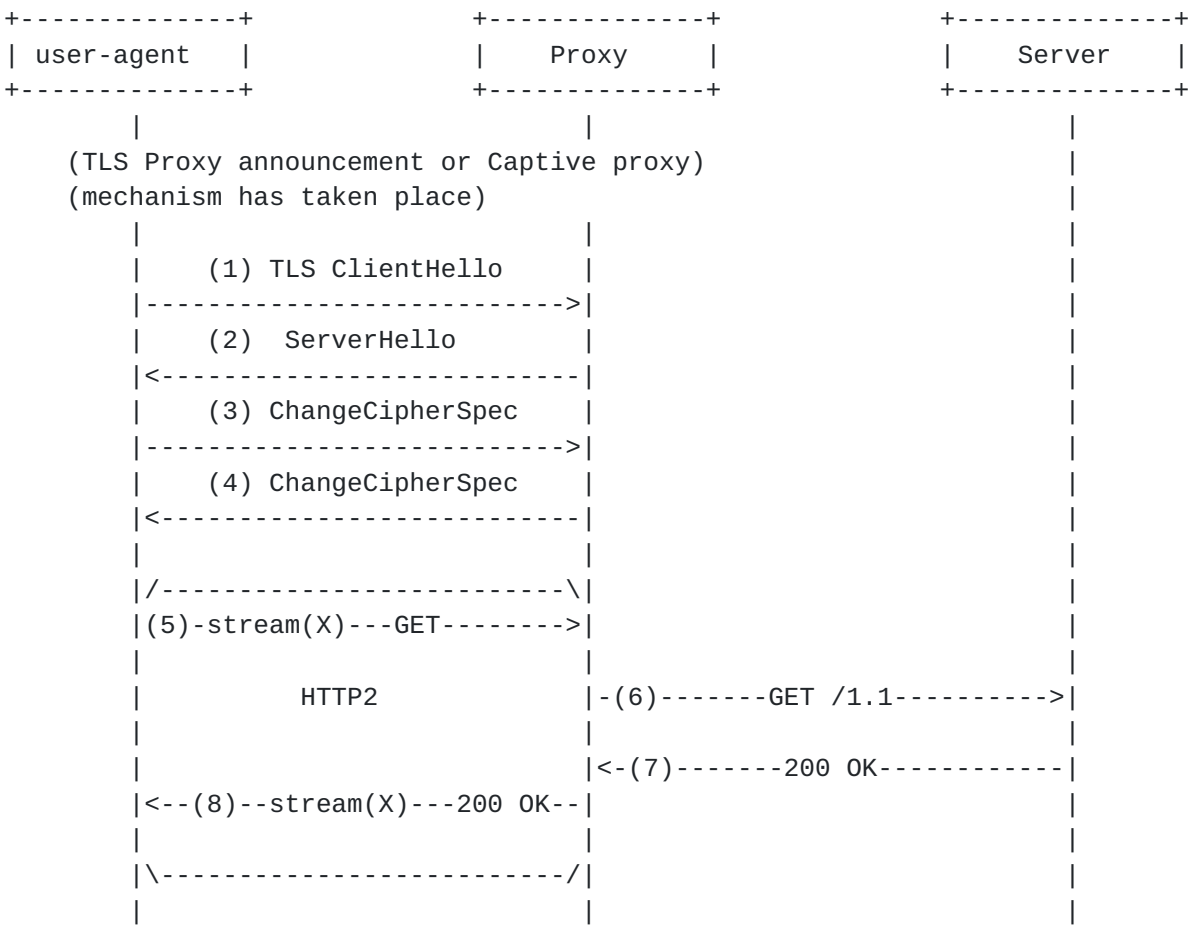

Figure 7: Origin server with only HTTP/1.1 support

4.3. Secure Forward Proxy and https URIs

The Proxy intercepts the TLS ClientHello analyses the application layer protocol negotiation extension field and if it contains "h2" value it does not do anything and let the TLS handshake continue and the TLS session be established between the User-Agent and the Server (see Figure 8).

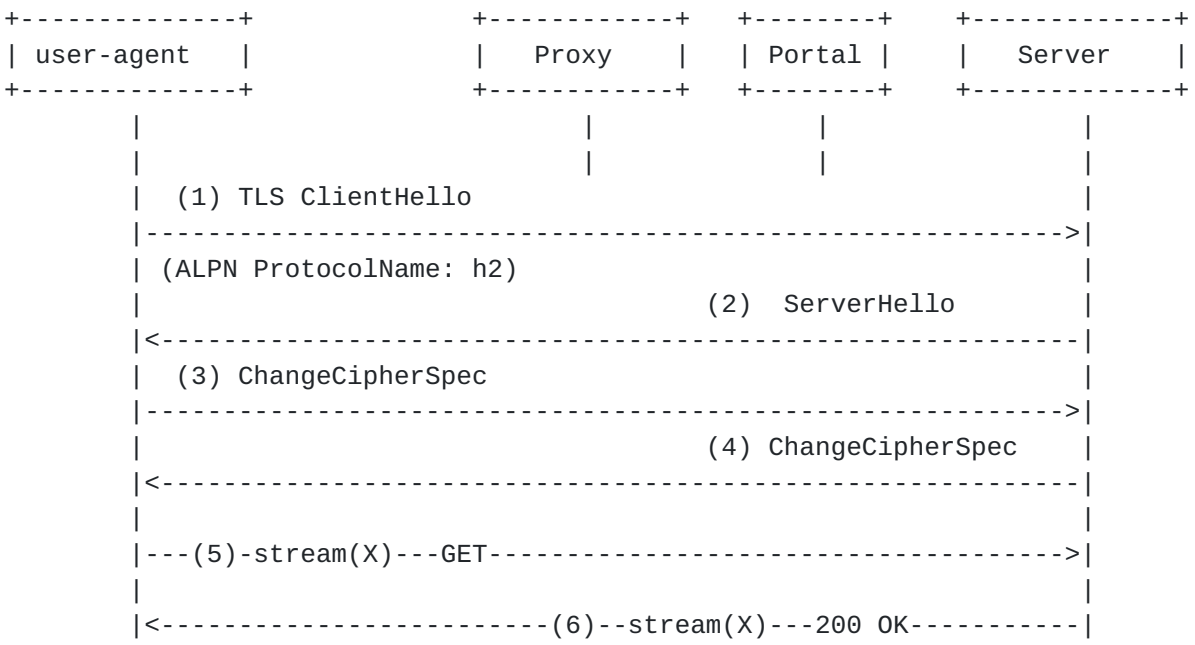

Figure 8: Trusted Proxy and HTTPS URI resources

5. Signalling the presence of a Proxy in between

The Via general-header field MUST be used by the user-agent to indicate the presence of the secure proxy between the User-Agent and the server on requests, and between the origin server and the User-Agent on responses.

6. Security Considerations

This document addresses proxies that act as intermediary for HTTP2 traffic and therefore the security and privacy implications of having those proxies in the path need to be considered. MITM [4], [I-D.nottingham-http-proxy-problem] and [I-D.vidya-httpbis-explicit-proxy-ps] discuss various security and privacy issues associated with the use of proxies. Users should be made aware that, different than end-to-end HTTPS, the achievable security level is now also dependent on the security features/ capabilities of the proxy as to what cipher suites it supports, which root CA certificates it trusts, how it checks certificate revocation status, etc. Users should also be made aware that the proxy has visibility to the actual content they exchange with Web servers, including personal and sensitive information.

By requiring proxy certificates to be extended validation

certificates the barrier to attaining a proxy certificate is significantly increased. To further increase the security, the validation by the CA could also include technical details and processes relevant for the security. The owner could for example be obliged to apply security patches in a timely fashion.

This document proposes two alternative approaches to making the presence of proxy explicit to users and letting them decide whether they accept that. A user can opt out and choose to bypass the proxy. This ensures that a proxy never acts as intermediary for HTTP2 traffic unless authorized by the user.

When the user has given consent to the presence of the proxy, the User-Agent switches to a Proxy mode in which it does not check the hostname of the origin server against the server's identity as presented in the Server Certificate message. However if any of the following checks fails the User-Agent should immediately exit this Proxy mode:

1. the server's certificate is issued by a trusted CA and the certificate is valid;

2. the Extended Key Usage extension is present in the certificate and indicates the owner of this certificate is a proxy;

3. the server possesses the private key corresponding to the certificate.

7. Privacy Considerations

8. IANA Considerations

This document creates a registration for the identification of HTTP2 used to transport "http" URIs resources in the "Application Layer Protocol Negotiation (ALPN) Protocol IDs" registry established in [TLSALPN].

Protocol: HTTP2 used to transport "http" URIs resources

Identification Sequence: 0x68 0x32 0x63 0x6C 0x72 ("h2clr")

9. Acknowledgments

The authors wish to thank Yi Cheng, Goran Eriksson, Stefan Hakansson, Nicolas Mailhot and Salman Taj for their ideas, technical suggestions

and comments.

10. References

10.1. Normative References

[I-D.ietf-httpbis-http2]
Belshe, M., Peon, R., Thomson, M., and A. Melnikov, "Hypertext Transfer Protocol version 2.0", draft-ietf-httpbis-http2-08 (work in progress), November 2013.

[I-D.ietf-httpbis-p1-messaging]
Fielding, R. and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", draft-ietf-httpbis-p1-messaging-24 (work in progress), September 2013.

[I-D.nottingham-http-proxy-problem]
Nottingham, M., "Problems with Proxies in HTTP", draft-nottingham-http-proxy-problem-00 (work in progress), October 2013.

[I-D.rpeon-httpbis-exproxy]
Peon, R., "Explicit Proxies for HTTP/2.0", draft-rpeon-httpbis-exproxy-00 (work in progress), June 2012.

[I-D.vidya-httpbis-explicit-proxy-ps]
Narayanan, V., "Explicit Proxying in HTTP - Problem Statement And Goals", draft-vidya-httpbis-explicit-proxy-ps-00 (work in progress), October 2013.

[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.

[RFC2817] Khare, R. and S. Lawrence, "Upgrading to TLS Within HTTP/1.1", RFC 2817, May 2000.

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

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, May 2008.

10.2. Informative References

[RFC3040] Cooper, I., Melve, I., and G. Tomlinson, "Internet Web Replication and Caching Taxonomy", RFC 3040, January 2001.

[RFC4732] Handley, M., Rescorla, E., and IAB, "Internet Denial-of-Service Considerations", RFC 4732, December 2006.

[RFC6455] Fette, I. and A. Melnikov, "The WebSocket Protocol", RFC 6455, December 2011.

URIs

[1] <https://github.com/http2/http2-spec/wiki/Proxy-User-Stories>

[2] <https://github.com/bizzbyster/TrustedProxy/wiki/Trusted-Proxy-Use-Case-Analysis-and-Alternatives>

[3] <https://github.com/http2/http2-spec/issues/314>

[4] <Jarmoc, J., SSL/TLS Interception Proxies and Transitive Trust, 2012 https://www.grc.com/miscfiles/HTTPS_Interception_Proxies.pdf>

Authors' Addresses

Salvatore Loreto
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

Email: salvatore.loreto@ericsson.com

John Mattsson
Ericsson
Kista
Sweden

Email: john.mattsson@ericsson.com

Robert Skog
Ericsson
Kista
Sweden

Email: robert.skog@ericsson.com

Hans Spaak
Ericsson
Kista
Sweeden

Email: hans.spaak@ericsson.com

Gus Bourg
AT&T

Phone:
Email:

Dan Druta
AT&T

Phone:
Email: dd5826@att.com

Mohammad Hafeez
AT&T

Phone:
Email: mh2897@att.com