Workgroup: HTTP
Internet-Draft:
draft-egorbaty-httpbis-secondary-server-
certs-01
Published: 12 October 2023
Intended Status: Standards Track
Expires: 14 April 2024
Authors: E. Gorbaty  M. Bishop
Apple  Akamai

# Secondary Certificate Authentication of HTTP Servers

## Abstract

This document defines a way for HTTP/2 and HTTP/3 servers to send additional certificate-based credentials after a TLS connection is established, based on TLS Exported Authenticators.

## About This Document

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

The latest revision of this draft can be found at https://egorbaty.github.io/draft-httpbis-secondary-server-certs/draft-egorbaty-httpbis-secondary-server-certs.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-egorbaty-httpbis-secondary-server-certs/.

Discussion of this document takes place on the HTTP Working Group mailing list (mailto:ietf-http-wg@w3.org), which is archived at https://lists.w3.org/Archives/Public/ietf-http-wg/.

Source for this draft and an issue tracker can be found at https://github.com/egorbaty/draft-httpbis-secondary-server-certs.

## 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 https://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 14 April 2024.



**Table of Contents**

1. Introduction
  1.1. Server Certificate Authentication
  1.2. TLS Exported Authenticators
  1.3. HTTP-Layer Certificate Authentication
2. Conventions and Definitions
3. Discovering Additional Certificates at the HTTP Layer
  3.1. Indicating Support for HTTP-Layer Certificate Authentication
  3.2. Making Certificates Available
4. SETTINGS_HTTP_SERVER_CERT_AUTH
  4.1. The SETTINGS_HTTP_SERVER_CERT_AUTH HTTP/2 SETTINGS Parameter{#http2-setting}
  4.2. The SETTINGS_HTTP_SERVER_CERT_AUTH HTTP/3 SETTINGS Parameter{#http3-setting}
5. CERTIFICATE frame
  5.1. HTTP/2 CERTIFICATE frame
  5.2. HTTP/3 CERTIFICATE frame
  5.3. Exported Authenticator Characteristics
6. Indicating Failures During HTTP-Layer Certificate Authentication
  6.1. Misbehavior
  6.2. Invalid Certificates
7. Security Considerations
  7.1. Impersonation
  7.2. Fingerprinting
  7.3. Persistence of Service
  7.4. Confusion About State
8. IANA Considerations
  8.1. Frame Types
  8.2. Settings Parameters
9. References
  9.1. Normative References

9.2. Informative References
Acknowledgments
Authors' Addresses

## 1. Introduction

HTTP [HTTP] clients need to know that the content they receive on a connection comes from the origin from which they intended to retrieve it. The traditional form of server authentication in HTTP has been in the form of a single X.509 certificate provided during the TLS [TLS13] handshake.

TLS supports one server and one client certificate on a connection. These certificates may contain multiple identities, but only one certificate may be provided.

Many HTTP servers host content from several origins. HTTP/2 [H2] and HTTP/3 [H3] permit clients to reuse an existing HTTP connection to a server provided that the secondary origin is also in the certificate provided during the TLS handshake. In many cases, servers choose to maintain separate certificates for different origins but still desire the benefits of a shared HTTP connection. This document defines a capability for servers to use and to authenticate with those seperate certificates over a shared connection.

The ability to maintain seperate certificates for different origins can also allow proxies that cache content from secondary origins to communicate to clients that they can service some of those origins directly, allowing the proxy to behave as a TLS-terminating reverse proxy for those origins instead of establishing a TLS encrypted tunnel through the proxy.

### 1.1. Server Certificate Authentication

Section 9.1.1 of [H2] and Section 3.3 of [H3] describe how connections may be used to make requests from multiple origins as long as the server is authoritative for both. A server is considered authoritative for an origin if DNS resolves the origin to the IP address of the server and (for TLS) if the certificate presented by the server contains the origin in the Subject Alternative Names field.

[ALTSVC] enables a step of abstraction from the DNS resolution. If both hosts have provided an Alternative Service at hostnames which resolve to the IP address of the server, they are considered authoritative just as if DNS resolved the origin itself to that address. However, the server's one TLS certificate is still required to contain the name of each origin in question.

[ORIGIN] relaxes the requirement to perform the DNS lookup if already connected to a server with an appropriate certificate which claims support for a particular origin.

Servers which host many origins often would prefer to have separate certificates for some sets of origins. This may be for ease of certificate management (the ability to separately revoke or renew them), due to different sources of certificates (a CDN acting on behalf of multiple origins), or other factors which might drive this administrative decision. Clients connecting to such origins cannot currently reuse connections, even if both client and server would prefer to do so.

Because the TLS SNI extension is exchanged in the clear, clients might also prefer to retrieve certificates inside the encrypted context. When this information is sensitive, it might be advantageous to request a general-purpose certificate or anonymous ciphersuite at the TLS layer, while acquiring the "real" certificate in HTTP after the connection is established.

### 1.2. TLS Exported Authenticators

TLS Exported Authenticators [EXPORTED-AUTH] are structured messages that can be exported by either party of a TLS connection and validated by the other party. Given an established TLS connection, an authenticator message can be constructed proving possession of a certificate and a corresponding private key. The mechanisms that this draft defines are primarily focused on the server's ability to generate TLS Exported Authenticators.

Each Authenticator is computed using a Handshake Context and Finished MAC Key derived from the TLS session. The Handshake Context is identical for both parties of the TLS connection, while the Finished MAC Key is dependent on whether the Authenticator is created by the client or the server.

Successfully verified Authenticators result in certificate chains, with verified possession of the corresponding private key, which can be supplied into a collection of available certificates. Likewise, descriptions of desired certificates can also be supplied into these collections.

### 1.3. HTTP-Layer Certificate Authentication

This draft defines HTTP/2 and HTTP/3 CERTIFICATE frames (Section 5) to carry the relevant certificate messages, enabling certificate-based authentication of servers independent of TLS version. This mechanism can be implemented at the HTTP layer without breaking the existing interface between HTTP and applications above it.

TLS Exported Authenticators [EXPORTED-AUTH] allow the opportunity for an HTTP/2 and HTTP/3 servers to send certificate frames which can be used to prove the servers authenticity for multiple origins.

This draft additionally defines SETTINGS parameters for HTTP/2 and HTTP/3 (Section 4) that allow the client and server to indicate support for HTTP-Layer certificate authentication.

## 2. Conventions and Definitions

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

## 3. Discovering Additional Certificates at the HTTP Layer

A certificate chain with proof of possession of the private key corresponding to the end-entity certificate is sent as a sequence of CERTIFICATE frames (see Section 5.1, Section 5.2) to the client. Once the holder of a certificate has sent the chain and proof, this certificate chain is cached by the recipient and available for future use.

### 3.1. Indicating Support for HTTP-Layer Certificate Authentication

The SETTINGS_HTTP_SERVER_CERT_AUTH parameters for HTTP/2 and HTTP/3 are defined in Section 4 so that clients and servers can indicate support for secondary certificate authentication of servers.

HTTP/2 and HTTP/3 endpoints who wish to indicate support for HTTP-Layer certificate authentication **MUST** send a SETTINGS_HTTP_SERVER_CERT_AUTH parameter set to "1" in their SETTINGS frame. Endpoints **MUST NOT** use any of the authentication functionality described in this draft unless the parameter has been negotiated by both sides.

Endpoints **MUST NOT** send a SETTINGS_HTTP_SERVER_CERT_AUTH parameter with a value of 0 after previously sending a value of 1.

SETTINGS_HTTP_SERVER_CERT_AUTH indicates that servers are able to offer additional certificates to demonstrate control over other origin hostnames, and that clients are able to make requests for hostnames received in a TLS Exported Authenticator that the server sends.

### 3.2. Making Certificates Available

When both peers have advertised support for HTTP-layer certificates in a given direction as in Section 3.1, the indicated endpoint can supply additional certificates into the connection at any time. That is, if both endpoints have sent SETTINGS_HTTP_SERVER_CERT_AUTH and validated the value received from the peer, the server may send certificates spontaneously, at any time, as described by the Spontaneous Server Authentication message sequence in Section 3 of [EXPORTED-AUTH].

This does mean that if a server knows it supports secondary certificate authentication, and it receives SETTINGS_HTTP_SERVER_CERT_AUTH from the client, that it can enqueue certificates immediately following the received SETTINGS frame.

Certificates supplied by servers can be considered by clients without further action by the server. A server **SHOULD NOT** send certificates which do not cover origins which it is prepared to service on the current connection, and **SHOULD NOT** send them if the client has not indicated support with SETTINGS_HTTP_SERVER_CERT_AUTH.

A client **MUST NOT** send certificates to the server. The server **SHOULD** close the connection upon receipt of a CERTIFICATE frame from a client.

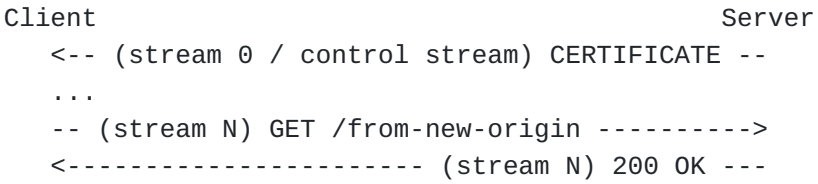


Figure 1: Simple unprompted server authentication

A server **MAY** send a CERTIFICATE immediately after sending its SETTINGS. However, it **MAY** also send certificates at any time later. For example, a proxy might discover that a client is interested in an origin that it can reverse proxy at the time that a client sends a CONNECT request. It can then send certificates for those origins to allow for TLS-terminated reverse proxying to those origins for the remainder of the connection lifetime. Figure 2 illustrates this behavior.

```
Client                                          Server
   -- (stream N) CONNECT /to-new-origin -------->
   <-- (stream 0 / control stream) CERTIFICATE --
   <-- (stream 0 / control stream) 200 OK -------
   ...
   -- (stream M) GET /to-new-origin ------------>
   <--- (stream M, direct from server) 200 OK ---
```

Figure 2: Reverse proxy server authentication

## 4. SETTINGS_HTTP_SERVER_CERT_AUTH

SETTINGS parameters for HTTP/2 and HTTP/3 seperately are defined below.

### 4.1. The SETTINGS_HTTP_SERVER_CERT_AUTH HTTP/2 SETTINGS Parameter{#http2-setting}

This document adds a new HTTP/2 SETTINGS(0xTBD) parameter to those defined by Section 6.5.2 of [H2].

The new parameter name is SETTINGS_HTTP_SERVER_CERT_AUTH. The value of the parameter **MUST** be 0 or 1.

The usage of this parameter is described in Section 3.1.

### 4.2. The SETTINGS_HTTP_SERVER_CERT_AUTH HTTP/3 SETTINGS Parameter{#http3-setting}

This document adds a new HTTP/3 SETTINGS(0xTBD) parameter to those defined by Section 7.2.4.1 of [H3].

The new parameter name is SETTINGS_HTTP_SERVER_CERT_AUTH. The value of the parameter **MUST** be 0 or 1.

The usage of this parameter is described in Section 3.1.

## 5. CERTIFICATE frame

The CERTIFICATE frame contains an exported authenticator message from the TLS layer that provides a chain of certificates and associated extensions, proving possession of the private key corresponding to the end-entity certificate.

A server sends a CERTIFICATE frame on stream 0 for HTTP/2 and on the control stream for HTTP/3. The client is permitted to make subsequent requests for resources upon receipt of a CERTIFICATE frame without further action from the server.

Upon receiving a complete series of CERTIFICATE frames, the receiver may validate the Exported Authenticator value by using the exported authenticator API. This returns either an error indicating that the message was invalid or the certificate chain and extensions used to create the message.

### 5.1. HTTP/2 CERTIFICATE frame

A CERTIFICATE frame in HTTP/2 (type=0xTBD) carrries a TLS Exported authenticator that clients can use to authenticate secondary origins from a sending server.

The CERTIFICATE frame **MUST** be sent on stream 0. A CERTIFICATE frame received on any other stream **MUST** not be used for server authentication.

```
CERTIFICATE Frame {
  Length (24),
  Type (8) = 0xTBD,

  Unused Flags (8),

  Reserved (1),
  Stream Identifier (31) = 0,

  Authenticator (..),
}
```

Figure 3: HTTP/2 CERTIFICATE Frame

The Length, Type, Unused Flag(s), Reserved, and Stream Identifier fields are described in Section 4 of [H2].

The CERTIFICATE frame does not define any flags.

The authenticator field is a portion of the opaque data returned from the TLS connection exported authenticator authenticate API. See Section 5.3 for more details on the input to this API.

The CERTIFICATE frame applies to the connection, not a specific stream. An endpoint **MUST** treat a CERTIFICATE frame with a stream identifier other than 0x00 as a connection error.

### 5.2. HTTP/3 CERTIFICATE frame

A CERTIFICATE frame in HTTP/3 (type=0xTBD) carrries a TLS Exported authenticator that clients can use to authenticate secondary origins from a sending server.

The CERTIFICATE frame **MUST** be sent on the control stream. A CERTIFICATE frame received on any other stream **MUST** not be used for server authentication.

```
CERTIFICATE Frame {
  Type (i) = 0xTBD,
  Length (i),
  Authenticator (...),
}
```

Figure 4: HTTP/3 CERTIFICATE Frame

The Type and Length fields are described in Section 7.1 of [H3].

The authenticator field is a portion of the opaque data returned from the TLS connection exported authenticator authenticate API. See Section 5.3 for more details on the input to this API.

The CERTIFICATE frame applies to the connection, not a specific stream. An endpoint **MUST** treat a CERTIFICATE frame received on any stream other than the control stream as a connection error.

### 5.3. Exported Authenticator Characteristics

The Exported Authenticator API defined in [EXPORTED-AUTH] takes as input a request, a set of certificates, and supporting information about the certificate (OCSP, SCT, etc.). The result is an opaque token which is used when generating the CERTIFICATE frame.

Upon receipt of a CERTIFICATE frame, an endpoint which has negotiated support for secondary certfiicates **MUST** perform the following steps to validate the token it contains:

- Using the get context API, retrieve the certificate_request_context used to generate the authenticator, if any. Because the certificate_request_context for spontaneous server certificates is chosen by the server, the usage of the certificate_request_context is implementation-dependent. For details, see Section 5 of [EXPORTED-AUTH].

- Use the validate API to confirm the validity of the authenticator with regard to the generated request, if any.

If the authenticator cannot be validated, this **SHOULD** be treated as a connection error.

Once the authenticator is accepted, the endpoint can perform any other checks for the acceptability of the certificate itself.

## 6. Indicating Failures During HTTP-Layer Certificate Authentication

Because this draft permits certificates to be exchanged at the HTTP framing layer instead of the TLS layer, several certificate-related errors which are defined at the TLS layer might now occur at the HTTP framing layer.

There are two classes of errors which might be encountered, and they are handled differently.

### 6.1. Misbehavior

This category of errors could indicate a peer failing to follow requirements in this document or might indicate that the connection is not fully secure. These errors are fatal to stream or connection, as appropriate.

**CERTIFICATE_UNREADABLE (0xERROR-TBD):** An exported authenticator could not be validated.

### 6.2. Invalid Certificates

Unacceptable certificates (expired, revoked, or insufficient to satisfy the request) are not treated as stream or connection errors. This is typically not an indication of a protocol failure. Clients **SHOULD** establish a new connection in an attempt to reach an authoritative server if they deem a certificate from the server unacceptable.

## 7. Security Considerations

This mechanism defines an alternate way to obtain server and client certificates other than in the initial TLS handshake. While the signature of exported authenticator values is expected to be equally secure, it is important to recognize that a vulnerability in this code path is at least equal to a vulnerability in the TLS handshake.

### 7.1. Impersonation

This mechanism could increase the impact of a key compromise. Rather than needing to subvert DNS or IP routing in order to use a compromised certificate, a malicious server now only needs a client to connect to *some* HTTPS site under its control in order to present the compromised certificate. Clients **SHOULD** consult DNS for hostnames presented in secondary certificates if they would have done so for the same hostname if it were present in the primary certificate.

As recommended in [ORIGIN], clients opting not to consult DNS ought to employ some alternative means to increase confidence that the certificate is legitimate, such as an ORIGIN frame.

As noted in the Security Considerations of [EXPORTED-AUTH], it is difficult to formally prove that an endpoint is jointly authoritative over multiple certificates, rather than individually authoritative on each certificate. As a result, clients **MUST NOT** assume that because one origin was previously colocated with another, those origins will be reachable via the same endpoints in the future. Clients **MUST NOT** consider previous secondary certificates to be validated after TLS session resumption. Servers **MAY** re-present certificates if a TLS Session is resumed.

### 7.2. Fingerprinting

This draft defines a mechanism which could be used to probe servers for origins they support, but it opens no new attack that was not already possible by making repeat TLS connections with different SNI values.

### 7.3. Persistence of Service

CNAME records in the DNS are frequently used to delegate authority for an origin to a third-party provider. This delegation can be changed without notice, even to the third-party provider, simply by modifying the CNAME record in question.

After the owner of the domain has redirected traffic elsewhere by changing the CNAME, new connections will not arrive for that origin, but connections which are properly directed to this provider for other origins would continue to claim control of this origin (via Secondary Certificates). This is proper behavior based on the third-party provider's configuration, but would likely not be what is intended by the owner of the origin.

This is not an issue which can be mitigated by the protocol, but something about which third-party providers **SHOULD** educate their customers before using the features described in this document.

### 7.4. Confusion About State

Implementations need to be aware of the potential for confusion about the state of a connection. The presence or absence of a validated certificate can change during the processing of a request, potentially multiple times, as CERTIFICATE frames are received. A client that uses certificate authentication needs to be prepared to reevaluate the authorization state of a request as the set of certificates changes.

Behavior for TLS-Terminated reverse proxies is also worth considering. If a server which situationally reverse-proxies wishes for the client to view a request made prior to receipt of certificates as TLS-Terminated, or wishes for the client to start a new tunnel alternatively, this draft does not currently define formal mechanisms to facilitate that intention.

## 8. IANA Considerations

This document registers the CERTIFICATE frame type and SETTINGS_HTTP_SERVER_CERT_AUTH setting for both [H2] and [H3].

### 8.1. Frame Types

This specification registers the following entry in the "HTTP/2 Frame Type" registry defined in [H2]:

Code: : TBD

Frame Type: : CERTIFICATE

Reference: : This document

This specification registers the following entry in the "HTTP/3 Frame Types" registry established by [H3]:

Value: : TBD

Frame Type: : CERTIFICATE

Status: : permanent

Reference: : This document

Change Controller: : IETF

Contact: : ietf-http-wg@w3.org

### 8.2. Settings Parameters

This specification registers the following entry in the "HTTP/2 Settings" registry defined in [H2]:

Code: : TBD

Name: : SETTINGS_HTTP_SERVER_CERT_AUTH

Initial Value: : 0

Reference: : This document

This specification registers the following entry in the "HTTP/3 Settings" registry defined in [H3]:

Code: : TBD

Name: : SETTINGS_HTTP_SERVER_CERT_AUTH

Default: : 0

Reference: : This document

Change Controller: : IETF

Contact: : ietf-http-wg@w3.org

# 9. References

## 9.1. Normative References


**[EXPORTED-AUTH]** Sullivan, N., "Exported Authenticators in TLS", RFC 9261, DOI 10.17487/RFC9261, July 2022, <https://www.rfc-editor.org/rfc/rfc9261>.

**[H2]** Thomson, M., Ed. and C. Benfield, Ed., "HTTP/2", RFC 9113, DOI 10.17487/RFC9113, June 2022, <https://www.rfc-editor.org/rfc/rfc9113>.

**[H3]** Bishop, M., Ed., "HTTP/3", RFC 9114, DOI 10.17487/RFC9114, June 2022, <https://www.rfc-editor.org/rfc/rfc9114>.

**[HTTP]** Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Semantics", STD 97, RFC 9110, DOI 10.17487/RFC9110, June 2022, <https://www.rfc-editor.org/rfc/rfc9110>.

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

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

**[TLS13]** Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/rfc/rfc8446>.


## 9.2. Informative References

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

**[ORIGIN]** Nottingham, M. and E. Nygren, "The ORIGIN HTTP/2 Frame", RFC 8336, DOI 10.17487/RFC8336, March 2018, <https://www.rfc-editor.org/rfc/rfc8336>.

## Acknowledgments

Thanks to Mike Bishop, Nick Sullivan, Martin Thomson and other contributors for their work on the draft that this is based on.

And thanks to Eric Kinnear and Tommy Pauly for their guidance and editorial contributions to this draft.

TODO: Other acknowledgements

## Authors' Addresses

Eric Gorbaty
Apple

Email: e_gorbaty@apple.com

Mike Bishop
Akamai

Email: mbishop@evequefou.be