HTTP
Internet-Draft
Intended status: Informational
Expires: 26 November 2022

B. Campbell
Ping Identity
M. Bishop, Ed.
Akamai

25 May 2022

Client-Cert HTTP Header Field
draft-ietf-httpbis-client-cert-field-02

Abstract

This document defines HTTP extension header fields that allow a TLS terminating reverse proxy to convey the client certificate information of a mutually-authenticated TLS connection to the origin server in a common and predictable manner.

About This Document

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

Status information for this document may be found at https://datatracker.ietf.org/doc/draft-ietf-httpbis-client-cert-field/.

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/. Working Group information can be found at https://httpwg.org/.

Source for this draft and an issue tracker can be found at https://github.com/httpwg/http-extensions/labels/client-cert-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 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 26 November 2022.

Copyright Notice

Table of Contents

1. Introduction

A fairly common deployment pattern for HTTPS applications is to have the origin HTTP application servers sit behind a reverse proxy that terminates TLS connections from clients. The proxy is accessible to the internet and dispatches client requests to the appropriate origin server within a private or protected network. The origin servers are not directly accessible by clients and are only reachable through the reverse proxy. The backend details of this type of deployment are typically opaque to clients who make requests to the proxy server and see responses as though they originated from the proxy server itself. Although HTTPS is also usually employed between the proxy and the origin server, the TLS connection that the client establishes for HTTPS is only between itself and the reverse proxy server.

The deployment pattern is found in a number of varieties such as n-tier architectures, content delivery networks, application load balancing services, and ingress controllers.

Although not exceedingly prevalent, TLS client certificate authentication is sometimes employed and in such cases the origin server often requires information about the client certificate for its application logic. Such logic might include access control decisions, audit logging, and binding issued tokens or cookies to a certificate, and the respective validation of such bindings. The specific details from the certificate needed also vary with the application requirements. In order for these types of application deployments to work in practice, the reverse proxy needs to convey information about the client certificate to the origin application server. A common way this information is conveyed in practice today is by using non-standard fields to carry the certificate (in some encoding) or individual parts thereof in the HTTP request that is dispatched to the origin server. This solution works but interoperability between independently developed components can be cumbersome or even impossible depending on the implementation choices respectively made (like what field names are used or are

configurable, which parts of the certificate are exposed, or how the certificate is encoded). A well-known predictable approach to this commonly occurring functionality could improve and simplify interoperability between independent implementations.

This document aspires to standardize two HTTP header fields, Client-Cert and Client-Cert-Chain, which a TLS terminating reverse proxy (TTRP) adds to requests sent to the backend origin servers. The Client-Cert field value contains the end-entity client certificate from the mutually-authenticated TLS connection between the originating client and the TTRP. Optionally, the Client-Cert-Chain field value contains the certificate chain used for validation of the

end-entity certificate. This enables the backend origin server to utilize the client certificate information in its application logic. While there may be additional proxies or hops between the TTRP and the origin server (potentially even with mutually-authenticated TLS connections between them), the scope of the Client-Cert header field is intentionally limited to exposing to the origin server the certificate that was presented by the originating client in its connection to the TTRP.

1.1. Requirements Notation and Conventions

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.

1.2. Terminology and Applicability

Phrases like TLS client certificate authentication or mutually-authenticated TLS are used throughout this document to refer to the process whereby, in addition to the normal TLS server authentication with a certificate, a client presents its X.509 certificate [RFC5280] and proves possession of the corresponding private key to a server when negotiating a TLS connection or the resumption of such a connection. In contemporary versions of TLS [RFC8446] [RFC5246] this requires that the client send the Certificate and CertificateVerify messages during the handshake and for the server to verify the CertificateVerify and Finished messages.

HTTP/2 restricts TLS 1.2 renegotiation (Section 9.2.1 of [RFC7540]) and prohibits TLS 1.3 post-handshake authentication [RFC8740]. However, they are sometimes used to implement reactive client certificate authentication in HTTP/1.1 [RFC7230] where the server decides whether to request a client certificate based on the HTTP request. HTTP application data sent on such a connection after receipt and verification of the client certificate is also mutually-authenticated and thus suitable for the mechanisms described in this document. With post-handshake authentication there is also the possibility, though unlikely in practice, of multiple certificates and certificate chains from the client on a connection, in which case only the certificate and chain of the last post-handshake authentication are to be utilized for the header fields described herein.

2. HTTP Header Fields and Processing Rules

This document designates the following headers, defined further in Section 2.2 and Section 2.3 respectively, to carry the client certificate information of a mutually-authenticated TLS connection from a reverse proxy to origin server.

Client-Cert: Conveys the end-entity certificate used by the client in the TLS handshake with the reverse proxy from the reverse proxy to the origin server.

Client-Cert-Chain: Conveys the certificate chain used for validation of the end-entity certificate used by the client in the TLS handshake from the reverse proxy to the origin server.

2.1. Encoding

The headers in this document encode certificates as Structured Field Byte Sequences (Section 3.3.5 of [RFC8941]) where the value of the binary data is a DER encoded [ITU.X690.1994] X.509 certificate [RFC5280]. In effect, this means that the binary DER certificate is encoded using base64 (without line breaks, spaces, or other

characters outside the base64 alphabet) and delimited with colons on either side.

Note that certificates are often stored encoded in a textual format, such as the one described in Section 5.1 of [RFC7468], which is already nearly compatible with a Structured Field Byte Sequence; if so, it will be sufficient to replace ---(BEGIN|END) CERTIFICATE--- with : and remove line breaks in order to generate an appropriate item.

2.2. Client-Cert HTTP Header Field

In the context of a TLS terminating reverse proxy deployment, the proxy makes the TLS client certificate available to the backend application with the Client-Cert HTTP header field. This field contains the end-entity certificate used by the client in the TLS handshake.

Client-Cert is an Item Structured Header [RFC8941]. Its value MUST be a Byte Sequence (Section 3.3.5 of [RFC8941]). Its ABNF is:

```
Client-Cert = sf-binary
```

The value of the header is encoded as described in Section 2.1.

The Client-Cert header field is only for use in HTTP requests and MUST NOT be used in HTTP responses. It is a single HTTP header field value as defined in Section 3.2 of [RFC7230], which MUST NOT have a list of values or occur multiple times in a request.

2.3. Client-Cert-Chain HTTP Header Field

In the context of a TLS terminating reverse proxy deployment, the proxy MAY make the certificate chain used for validation of the end-entity certificate available to the backend application with the Client-Cert-Chain HTTP header field. This field contains certificates used by the proxy to validate the certificate used by the client in the TLS handshake. These certificates might or might not have been provided by the client during the TLS handshake.

Client-Cert-Chain is a List Structured Header [RFC8941]. Each item in the list MUST be a Byte Sequence (Section 3.3.5 of [RFC8941]) encoded as described in Section 2.1.

The header's ABNF is:

```
Client-Cert-Chain = sf-list
```

The Client-Cert-Chain header field is only for use in HTTP requests and MUST NOT be used in HTTP responses. It MAY have a list of values or occur multiple times in a request. For header compression purposes, it might be advantageous to split lists into multiple instances.

The first certificate in the list SHOULD directly certify the end-entity certificate provided in the Client-Cert header; each following certificate SHOULD directly certify the one immediately preceding it. Because certificate validation requires that trust anchors be distributed independently, a certificate that specifies a trust anchor MAY be omitted from the chain, provided that the server is known to possess any omitted certificates.

However, for maximum compatibility, servers SHOULD be prepared to handle potentially extraneous certificates and arbitrary orderings.

2.4. Processing Rules

This section outlines the applicable processing rules for a TLS terminating reverse proxy (TTRP) that has negotiated a mutually-authenticated TLS connection to convey the client certificate from that connection to the backend origin servers. Use of the technique is to be a configuration or deployment option and the processing rules described herein are for servers operating with that option

enabled.

A TTRP negotiates the use of a mutually-authenticated TLS connection with the client, such as is described in [RFC8446] or [RFC5246], and validates the client certificate per its policy and trusted certificate authorities. Each HTTP request on the underlying TLS connection are dispatched to the origin server with the following modifications:

1. The client certificate is placed in the Client-Cert header field of the dispatched request, as described in Section 2.2.

2. If so configured, the validation chain of the client certificate is placed in the Client-Cert-Chain header field of the request, as described in Section 2.3.

3. Any occurrence of the Client-Cert or Client-Cert-Chain header fields in the original incoming request MUST be removed or overwritten before forwarding the request. An incoming request that has a Client-Cert or Client-Cert-Chain header field MAY be rejected with an HTTP 400 response.

Requests made over a TLS connection where the use of client certificate authentication was not negotiated MUST be sanitized by removing any and all occurrences of the Client-Cert and Client-Cert-Chain header fields prior to dispatching the request to the backend server.

Backend origin servers may then use the Client-Cert header field of the request to determine if the connection from the client to the TTRP was mutually-authenticated and, if so, the certificate thereby presented by the client.

Forward proxies and other intermediaries MUST NOT add the Client-Cert or Client-Cert-Chain header fields to requests, or modify an existing Client-Cert or Client-Cert-Chain header field. Similarly, clients MUST NOT employ the Client-Cert or Client-Cert-Chain header field in requests.

When the value of the Client-Cert request header field is used to select a response (e.g., the response content is access-controlled), the response MUST either be uncacheable (e.g., by sending Cache-Control: no-store) or be designated for selective reuse only for subsequent requests with the same Client-Cert header value by sending a Vary: Client-Cert response header. If a TTRP encounters a response with a client-cert field name in the Vary header field, it SHOULD prevent the user agent from caching the response by transforming the value of the Vary response header field to *.

3. Deployment Considerations

3.1. Header Field Compression

If the client certificate header field is generated by an intermediary on a connection that compresses fields (e.g., using HPACK [RFC7541] or QPACK [I-D.ietf-quic-qpack]) and more than one client's requests are multiplexed into that connection, it can reduce compression efficiency significantly, due to the typical size of the field value and its variation between clients. Recipients that anticipate connections with these characteristics can mitigate the efficiency loss by increasing the size of the dynamic table. If a recipient does not do so, senders may find it beneficial to always send the field value as a literal, rather than entering it into the dynamic table.

3.2. Header Block Size

A server in receipt of a larger header block than it is willing to handle can send an HTTP 431 (Request Header Fields Too Large) status code per Section 5 of [RFC6585]. Due to the typical size of the field values containing certificate data, recipients may need to be configured to allow for a larger maximum header block size. An intermediary generating client certificate header fields on connections that allow for advertising the maximum acceptable header block size (e.g. HTTP/2 [RFC7540] or HTTP/3 [I-D.ietf-quic-http]) should account for the additional size of header block of the requests it sends vs. requests it receives by advertising a value to its clients that is sufficiently smaller so as to allow for the addition of certificate data.

3.3. TLS Session Resumption

Some TLS implementations do not retain client certificate information when resuming. Providing inconsistent values of Client-Cert and Client-Cert-Chain when resuming might lead to errors, so implementations that are unable to provide these values SHOULD either disable resumption for connections with client certificates or initially omit a Client-Cert or Client-Cert-Chain field if it might not be available after resuming.

4. Security Considerations

The header fields described herein enable a TTRP and backend or origin server to function together as though, from the client's perspective, they are a single logical server side deployment of HTTPS over a mutually-authenticated TLS connection. Use of the header fields outside that intended use case, however, may undermine the protections afforded by TLS client certificate authentication. Therefore, steps MUST be taken to prevent unintended use, both in sending the header field and in relying on its value.

Producing and consuming the Client-Cert and Client-Cert-Chain header fields SHOULD be configurable options, respectively, in a TTRP and backend server (or individual application in that server). The default configuration for both should be to not use the header fields thus requiring an "opt-in" to the functionality.

In order to prevent field injection, backend servers MUST only accept the Client-Cert and Client-Cert-Chain header fields from a trusted TTRP (or other proxy in a trusted path from the TTRP). A TTRP MUST sanitize the incoming request before forwarding it on by removing or overwriting any existing instances of the fields. Otherwise, arbitrary clients can control the field values as seen and used by the backend server. It is important to note that neglecting to prevent field injection does not "fail safe" in that the nominal functionality will still work as expected even when malicious actions are possible. As such, extra care is recommended in ensuring that proper field sanitation is in place.

The communication between a TTRP and backend server needs to be secured against eavesdropping and modification by unintended parties.

The configuration options and request sanitization are necessarily functionally of the respective servers. The other requirements can be met in a number of ways, which will vary based on specific deployments. The communication between a TTRP and backend or origin server, for example, might be authenticated in some way with the insertion and consumption of the Client-Cert and Client-Cert-Chain header fields occurring only on that connection. Alternatively the network topology might dictate a private network such that the backend application is only able to accept requests from the TTRP and the proxy can only make requests to that server. Other deployments that meet the requirements set forth herein are also possible.

5. IANA Considerations

5.1. HTTP Field Name Registrations

Please register the following entries in the "Hypertext Transfer Protocol (HTTP) Field Name Registry" defined by [I-D.ietf-httpbis-semantics]:

* Field name: Client-Cert
* Status: permanent
* Specification document: Section 2 of [this document]
* Field name: Client-Cert-Chain
* Status: permanent
* Specification document: Section 2 of [this document]

6. References

6.1. 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, <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>.

[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, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/rfc/rfc5280>.

[ITU.X690.1994] International Telecommunications Union, "Information

Technology - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)", ITU-T Recommendation X.690, 1994.

[RFC8941] Nottingham, M. and P-H. Kamp, "Structured Field Values for HTTP", RFC 8941, DOI 10.17487/RFC8941, February 2021, <https://www.rfc-editor.org/rfc/rfc8941>.

6.2. Informative References

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

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

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

[RFC8740] Benjamin, D., "Using TLS 1.3 with HTTP/2", RFC 8740, DOI 10.17487/RFC8740, February 2020, <https://www.rfc-editor.org/rfc/rfc8740>.

[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, <https://www.rfc-editor.org/rfc/rfc7230>.

[RFC7468] Josefsson, S. and S. Leonard, "Textual Encodings of PKIX, PKCS, and CMS Structures", RFC 7468, DOI 10.17487/RFC7468, April 2015, <https://www.rfc-editor.org/rfc/rfc7468>.

[RFC7541] Peon, R. and H. Ruellan, "HPACK: Header Compression for HTTP/2", RFC 7541, DOI 10.17487/RFC7541, May 2015, <https://www.rfc-editor.org/rfc/rfc7541>.

[I-D.ietf-quic-qpack]
Krasic, C. '., Bishop, M., and A. Frindell, "QPACK: Header Compression for HTTP/3", Work in Progress, Internet-Draft, draft-ietf-quic-qpack-21, 2 February 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-quic-qpack-21>.

[RFC6585] Nottingham, M. and R. Fielding, "Additional HTTP Status Codes", RFC 6585, DOI 10.17487/RFC6585, April 2012, <https://www.rfc-editor.org/rfc/rfc6585>.

[I-D.ietf-quic-http]
Bishop, M., "Hypertext Transfer Protocol Version 3 (HTTP/3)", Work in Progress, Internet-Draft, draft-ietf-quic-http-34, 2 February 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-quic-http-34>.

[I-D.ietf-httpbis-semantics]
Fielding, R. T., Nottingham, M., and J. Reschke, "HTTP Semantics", Work in Progress, Internet-Draft, draft-ietf-httpbis-semantics-19, 12 September 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-httpbis-semantics-19>.

[RFC7239] Petersson, A. and M. Nilsson, "Forwarded HTTP Extension", RFC 7239, DOI 10.17487/RFC7239, June 2014, <https://www.rfc-editor.org/rfc/rfc7239>.

[RFC8705] Campbell, B., Bradley, J., Sakimura, N., and T. Lodderstedt, "OAuth 2.0 Mutual-TLS Client Authentication and Certificate-Bound Access Tokens", RFC 8705, DOI 10.17487/RFC8705, February 2020, <https://www.rfc-editor.org/rfc/rfc8705>.

Appendix A. Example

In a hypothetical example where a TLS client presents the client and intermediate certificate from Figure 1 when establishing a mutually-authenticated TLS connection with the TTRP, the proxy would send the Client-Cert field shown in {#example-header} to the backend. Note that line breaks and whitespace have been added to the field value in Figure 2 for display and formatting purposes only.


```
-----BEGIN CERTIFICATE-----
MIIBqDCCAU6gAwIBAgIBBzAKBggqhkjOPQQDAjA6MRswGQYDVQQKDBJMZXQncyBB
dXRoZW50aWNhdGUxGzAZBgNVBAMMEkxBIEludGVybWVkaWF0ZSBDQTAeFw0yMDAx
MTQyMjU1MzNaFw0yMTAxMjMyMjU1MzNaMA0xCzAJBgNVBAMMAkJDMFkwEwYHKoZI
zj0CAQYIKoZIzj0DAQcDQgAE8YnXXfaUgmnMtOXU/IncWalRhebrXmckC8vdgJ1p
5Be5F/3YC8OthxM4+k1M6aEAEFcGzkJiNy6J84y7uzo9M6NyMHAwCQYDVR0TBAIw
ADAfBgNVHSMEGDAWgBRm3WjLa38lbEYCuiCPct0ZaSED2DAOBgNVHQ8BAf8EBAMC
BsAwEwYDVR0lBAwwCgYIKwYBBQUHAwIwHQYDVR0RAQH/BBMwEYEPYmRjQGV4YW1w
bGUuY29tMAoGCCqGSM49BAMCA0gAMEUCIBHda/r1vaL6G3VliL4/Di6YK0Q6bMje
SkC3dFCOOB8TAiEAx/kHSB4urmiZ0NX5r5XarmPk0wmuydBVoU4hBVZ1yhk=
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
MIIB5jCCAYugAwIBAgIBFjAKBggqhkjOPQQDAjBWMQswCQYDVQQGEwJVUzEbMBkG
A1UECgwSTGV0J3MgQXV0aGVudGljYXRlMSowKAYDVQQDDCFMZXQncyBBdXRoZW50
aWNhdGUgUm9vdCBBdXRob3JpdHkwHhcNMjAwMTE0MjEzMjMwWhcNMzAwMTExMjEz
MjMwWjA6MRswGQYDVQQKDBJMZXQncyBBdXRoZW50aWNhdGUxGzAZBgNVBAMMEkxB
IEludGVybWVkaWF0ZSBDQTBZMBMGByqGSM49AgEGCCqGSM49AwEHA0IABJf+aA54
RC5pyLAR5yfXVYmNpgd+CGUTDp2KOGhc0gK91zxhHesEYkdXkpS2UN8Kati+yHtW
```

```
CV3kkhCngGyv7RqjZjBkMB0GA1UdDgQWBBRm3WjLa38lbEYCuiCPct0ZaSED2DAf
BgNVHSMEGDAWgBTEA2Q6eecKu9g9yb5glbkhhVINGDASBgNVHRMBAf8ECDAGAQH/
AgEAMA4GA1UdDwEB/wQEAwIBhjAKBggqhkjOPQQDAgNJADBGAiEA5pLvaFwRRkxo
mIAtDIwg9D7gC1xzxBl4r28EzmSO1pcCIQCJUShpSXO9HDIQMUgH69fNDEMHXD3R
RX5gP7kuu2KGMg==
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
MIICBjCCAaygAwIBAgIJAKS0yiqKtlhoMAoGCCqGSM49BAMCMFYxCzAJBgNVBAYT
AlVTMRswGQYDVQQKDBJMZXQncyBBdXRoZW50aWNhdGUxKjAoBgNVBAMMIUxldCdz
IEF1dGhlbnRpY2F0ZSBSb290IEF1dGhvcml0eTAeFw0yMDAxMTQyMTI1NDVaFw00
MDAxMDkyMTI1NDVaMFYxCzAJBgNVBAYTAlVTMRswGQYDVQQKDBJMZXQncyBBdXRo
ZW50aWNhdGUxKjAoBgNVBAMMIUxldCdzIEF1dGhlbnRpY2F0ZSBSb290IEF1dGhv
cml0eTBZMBMGByqGSM49AgEGCCqGSM49AwEHA0IABFoaHU+Z5bPKmGzlYXtCf+E6
HYj62fORaHDOrt+yyh3H/rTcs7ynFfGn+gyFsrSP3Ez88rajv+U2NfD0o0uZ4Pmj
YzBhMB0GA1UdDgQWBBTEA2Q6eecKu9g9yb5glbkhhVINGDAfBgNVHSMEGDAWgBTE
A2Q6eecKu9g9yb5glbkhhVINGDAPBgNVHRMBAf8EBTADAQH/MA4GA1UdDwEB/wQE
AwIBhjAKBggqhkjOPQQDAgNIADBFAiEAmAeg1ycKHriqHnaD4M/UDBpQRpkmdcRF
YGMg1Qyrkx4CIB4ivz3wQcQkGhcsUZ1SOImd/lq1Q0FLf09rGfLQPWDc
-----END CERTIFICATE-----
```

Figure 1: Certificate Chain (with client certificate first)


```
Client-Cert: :MIIBqDCCAU6gAwIBAgIBBzAKBggqhkjOPQQDAjA6MRswGQYDVQQKDBJ
 MZXQncyBBdXRoZW50aWNhdGUxGzAZBgNVBAMMEkxBIEludGVybWVkaWF0ZSBDQTAeFw0
 yMDAxMTQyMjU1MzNaFw0yMTAxMjMyMjU1MzNaMA0xCzAJBgNVBAMMAkJDMFkwEwYHKoZ
 Izj0CAQYIKoZIzj0DAQcDQgAE8YnXXfaUgmnMtOXU/IncWalRhebrXmckC8vdgJ1p5Be
 5F/3YC8OthxM4+k1M6aEAEFcGzkJiNy6J84y7uzo9M6NyMHAwCQYDVR0TBAIwADAfBgN
 VHSMEGDAWgBRm3WjLa38lbEYCuiCPct0ZaSED2DAOBgNVHQ8BAf8EBAMCBsAwEwYDVR0
 lBAwwCgYIKwYBBQUHAwIwHQYDVR0RAQH/BBMwEYEPYmRjQGV4YW1wbGUuY29tMAoGCCq
 GSM49BAMCA0gAMEUCIBHda/r1vaL6G3VliL4/Di6YK0Q6bMjeSkC3dFCOOB8TAiEAx/k
 HSB4urmiZ0NX5r5XarmPk0wmuydBVoU4hBVZ1yhk=:
```

Figure 2: Header Field in HTTP Request to Origin Server

If the proxy were configured to also include the certificate chain, it would also include this header:

```
Client-Cert-Chain: :MIIB5jCCAYugAwIBAgIBFjAKBggqhkjOPQQDAjBWMQsw
 CQYDVQQGEwJVUzEbMBkGA1UECgwSTGV0J3MgQXV0aGVudGljYXRlMSowKAYDVQQ
 DDCFMZXQncyBBdXRoZW50aWNhdGUgUm9vdCBBdXRob3JpdHkwHhcNMjAwMTE0Mj
 EzMjMwWhcNMzAwMTExMjEzMjMwWjA6MRswGQYDVQQKDBJMZXQncyBBdXRoZW50a
 WNhdGUxGzAZBgNVBAMMEkxBIEludGVybWVkaWF0ZSBDQTBZMBMGByqGSM49AgEG
 CCqGSM49AwEHA0IABJf+aA54RC5pyLAR5yfXVYmNpgd+CGUTDp2KOGhc0gK91zx
 hHesEYkdXkpS2UN8Kati+yHtWCV3kkhCngGyv7RqjZjBkMB0GA1UdDgQWBBRm3W
 jLa38lbEYCuiCPct0ZaSED2DAfBgNVHSMEGDAWgBTEA2Q6eecKu9g9yb5glbkhh
 VINGDASBgNVHRMBAf8ECDAGAQH/AgEAMA4GA1UdDwEB/wQEAwIBhjAKBggqhkjO
 PQQDAgNJADBGAiEA5pLvaFwRRkxomIAtDIwg9D7gC1xzxBl4r28EzmSO1pcCIQC
 JUShpSXO9HDIQMUgH69fNDEMHXD3RRX5gP7kuu2KGMg==:, :MIICBjCCAaygAw
 IBAgIJAKS0yiqKtlhoMAoGCCqGSM49BAMCMFYxCzAJBgNVBAYTAlVTMRswGQYDV
 QQKDBJMZXQncyBBdXRoZW50aWNhdGUxKjAoBgNVBAMMIUxldCdzIEF1dGhlbnRp
 Y2F0ZSBSb290IEF1dGhvcml0eTAeFw0yMDAxMTQyMTI1NDVaFw00MDAxMDkyMTI
 1NDVaMFYxCzAJBgNVBAYTAlVTMRswGQYDVQQKDBJMZXQncyBBdXRoZW50aWNhdG
 UxKjAoBgNVBAMMIUxldCdzIEF1dGhlbnRpY2F0ZSBSb290IEF1dGhvcml0eTBZM
 BMGByqGSM49AgEGCCqGSM49AwEHA0IABFoaHU+Z5bPKmGzlYXtCf+E6HYj62fOR
 aHDOrt+yyh3H/rTcs7ynFfGn+gyFsrSP3Ez88rajv+U2NfD0o0uZ4PmjYzBhMB0
 GA1UdDgQWBBTEA2Q6eecKu9g9yb5glbkhhVINGDAfBgNVHSMEGDAWgBTEA2Q6ee
 cKu9g9yb5glbkhhVINGDAPBgNVHRMBAf8EBTADAQH/MA4GA1UdDwEB/wQEAwIBh
 jAKBggqhkjOPQQDAgNIADBFAiEAmAeg1ycKHriqHnaD4M/UDBpQRpkmdcRFYGMg
 1Qyrkx4CIB4ivz3wQcQkGhcsUZ1SOImd/lq1Q0FLf09rGfLQPWDc:
```

Figure 3: Certificate Chain in HTTP Request to Origin Server

Appendix B. Considerations Considered

B.1. Field Injection

This draft requires that the TTRP sanitize the fields of the incoming request by removing or overwriting any existing instances of the

Client-Cert and Client-Cert-Chain header fields before dispatching that request to the backend application. Otherwise, a client could inject its own values that would appear to the backend to have come from the TTRP. Although numerous other methods of detecting/preventing field injection are possible; such as the use of a unique secret value as part of the field name or value or the application of a signature, HMAC, or AEAD, there is no common general standardized mechanism. The potential problem of client field injection is not at all unique to the functionality of this draft, and it would therefore be inappropriate for this draft to define a one-off solution. In the absence of a generic standardized solution existing currently, stripping/sanitizing the fields is the de facto means of protecting against field injection in practice today. Sanitizing the fields is sufficient when properly implemented and is a normative requirement of Section 4.

B.2. The Forwarded HTTP Extension

The Forwarded HTTP header field defined in [RFC7239] allows proxy components to disclose information lost in the proxying process. The TLS client certificate information of concern to this draft could have been communicated with an extension parameter to the Forwarded field; however, doing so would have had some disadvantages that this draft endeavored to avoid. The Forwarded field syntax allows for information about a full chain of proxied HTTP requests, whereas the Client-Cert and Client-Cert-Chain header fields of this document are concerned only with conveying information about the certificate presented by the originating client on the TLS connection to the TTRP (which appears as the server from that client's perspective) to backend applications. The multi-hop syntax of the Forwarded field is expressive but also more complicated, which would make processing it more cumbersome, and more importantly, make properly sanitizing its content as required by Section 4 to prevent field injection considerably more difficult and error-prone. Thus, this draft opted for a flatter and more straightforward structure.

B.3. The Whole Certificate and Certificate Chain

Different applications will have varying requirements about what information from the client certificate is needed, such as the subject and/or issuer distinguished name, subject alternative name(s), serial number, subject public key info, fingerprint, etc.. Furthermore, some applications, such as [RFC8705], make use of the entire certificate. In order to accommodate the latter and ensure

wide applicability by not trying to cherry-pick particular certificate information, this draft opted to pass the full encoded certificate as the value of the Client-Cert field.

The handshake and validation of the client certificate (chain) of the mutually-authenticated TLS connection is performed by the TTRP. With the responsibility of certificate validation falling on the TTRP, the end-entity certificate is oftentimes sufficient for the needs of the origin server. The separate Client-Cert-Chain field can convey the certificate chain for deployments that require such information.

Appendix C. Acknowledgements

The authors would like to thank the following individuals who've contributed in various ways ranging from just being generally supportive of bringing forth the draft to providing specific feedback or content:

* Evan Anderson
* Annabelle Backman
* Alan Frindell
* Rory Hewitt
* Fredrik Jeansson
* Benjamin Kaduk
* Torsten Lodderstedt
* Kathleen Moriarty
* Mark Nottingham
* Erik Nygren
* Mike Ounsworth
* Matt Peterson
* Eric Rescorla
* Justin Richer
* Michael Richardson

* Joe Salowey
* Rich Salz
* Mohit Sethi
* Rifaat Shekh-Yusef
* Travis Spencer
* Nick Sullivan
* Willy Tarreau
* Martin Thomson
* Peter Wu
* Hans Zandbelt

Appendix D. Document History

To be removed by the RFC Editor before publication as an RFC

draft-ietf-httpbis-client-cert-field-02

* Add a note about cert retention on TLS session resumption
* Say to use only the last one in the case of multiple post-handshake client cert authentications

draft-ietf-httpbis-client-cert-field-01

* Use RFC 8941 Structured Field Values for HTTP
* Introduce a separate header that can convey the certificate chain
* Add considerations on header compression and size
* Describe interaction with caching

* Fill out IANA Considerations with HTTP field name registrations

* Discuss renegotiation

draft-ietf-httpbis-client-cert-field-00

* Initial WG revision

* Mike Bishop added as co-editor

draft-bdc-something-something-certificate-05

* Change intended status of the draft to Informational

* Editorial updates and (hopefully) clarifications

draft-bdc-something-something-certificate-04

* Update reference from draft-ietf-oauth-mtls to RFC8705

draft-bdc-something-something-certificate-03

* Expanded further discussion notes to capture some of the feedback in and around the presentation of the draft in SECDISPATCH at IETF 107 and add those who've provided such feedback to the acknowledgements

draft-bdc-something-something-certificate-02

* Editorial tweaks + further discussion notes

draft-bdc-something-something-certificate-01

* Use the RFC v3 Format or die trying

draft-bdc-something-something-certificate-00

* Initial draft after a time constrained and rushed secdispatch presentation (https://datatracker.ietf.org/meeting/106/materials/slides-106-secdispatch-securing-protocols-between-proxies-and-backend-http-servers-00) at IETF 106 in Singapore with the

recommendation to write up a draft (at the end of the minutes (https://datatracker.ietf.org/meeting/106/materials/minutes-106-secdispatch)) and some folks expressing interest despite the rather poor presentation

Authors' Addresses

Brian Campbell
Ping Identity
Email: bcampbell@pingidentity.com

Mike Bishop (editor)
Akamai
Email: mbishop@evequefou.be