Workgroup: TLS
Internet-Draft:
draft-kampanakis-tls-scas-latest-03
Published: 5 January 2023
Intended Status: Experimental
Expires: 9 July 2023
Authors: P. Kampanakis
AWS
C. Bytheway
AWS
B.E. Westerbaan
Cloudflare
M. Thomson
Mozilla

Suppressing CA Certificates in TLS 1.3

Abstract

A TLS client or server that has access to the complete set of published intermediate certificates can inform its peer to avoid sending certificate authority certificates, thus reducing the size of the TLS handshake.

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 9 July 2023.

Copyright Notice

Table of Contents

1. Introduction
2. Terms and Definitions
3. Suppress CA Certificates Flag
  3.1. Client
  3.2. Server (mutual TLS authentication)
4. Security Considerations
5. IANA Considerations
6. Acknowledgements
7. References
  7.1. Normative References
  7.2. Informative References
Authors' Addresses

1. Introduction

The most data heavy part of a TLS handshake is authentication. It usually consists of a signature, an end-entity certificate and Certificate Authority (CA) certificates used to authenticate the end-entity to a trusted root CA. These chains can sometime add to a few kB of data which could be problematic for some use cases. [EAPTLSCERT] and [EAP-TLS13] discuss the issues big certificate chains in EAP authentication. Additionally, it is known that IEEE 802.15.4 [IEEE802154] mesh networks and Wi-SUN [WISUN] Field Area Networks often notice significant delays due to EAP-TLS authentication in constrained bandwidth mediums.

To alleviate the data exchanged in TLS [RFC8879] shrinks certificates by compressing them. [CBOR-CERTS] uses different certificate encodings for constrained environments. On the other hand, [CTLS] proposes the use of certificate dictionaries to omit sending CA certificates in a Compact TLS handshake.

In a post-quantum context [I-D.hoffman-c2pq][NIST_PQ] [I-D.ietf-tls-hybrid-design], the TLS authentication data issue is exacerbated. [CONEXT-PQTLS13SSH][NDSS-PQTLS13] show that post-quantum certificate chains exceeding the initial TCP congestion window (10MSS [RFC6928]) will slow down the handshake due to the extra round-trips they introduce. [PQTLS] shows that big certificate chains (even smaller than the initial TCP congestion window) will slow down the handshake in lossy environments. [TLS-SUPPRESS] quantifies the post-quantum authentication data in QUIC and TLS and shows that even the leanest post-quantum signature algorithms will impact QUIC and TLS. [CL-BLOG] also shows that 9-10 kilobyte certificate chains (even with 30MSS initial TCP congestion window)

will lead to double digit TLS handshake slowdowns. What's more, it shows that some clients or middleboxes cannot handle chains larger than 10kB. [QUIC-CERTS] also shows discusses how classical RSA certificate chains often exceed the QUIC amplification, an issue which will happen almost always with post-quantum certicates.

Mechanisms like [RFC8879][CBOR-CERTS] would not alleviate the issue with post-quantum certificates as the bulk of the certificate size is in the post-quantum public key or signature which is incompressible.

Thus, this document introduces a backwards-compatible mechanism to shrink the certificate data exchanged in TLS 1.3. In some uses of public key infrastructure (PKI), intermediate CA certificates sign end-entity certificates. In the web PKI, clients require that certificate authorities disclose all intermediate certificates that they create. Although the set of intermediate certificates is large, the size is bounded. Additionally, in some use cases the set of communicating peers is limited.

For a client or server that has the necessary intermediates, receiving them during the TLS handshake, increases the data transmission unnecessarily. This document defines a signal that a client or server can send to inform its peer that it already has the intermediate CA certificates. A peer that receives this signal can limit the certificate chain it sends to just the end-entity certificate, saving on handshake size.

This mechanism is intended to be complementary with certificate compression [RFC8879] in that it further reduces the size of the handshake especially for post-quantum certificates.

It is worth noting that [RFC7924] attempted to address the issue by omitting all certificates in the handshake if the client or server had cached the peer certificate. This standard has not seen wide adoption and could allow for TLS session correlation. Additionally, the short lifetime certificates used today and the large size of peers in some use cases make the peer certificate cache update and maintenance mechanism challenging -- not the least because of privacy concerns. The mechanism proposed in this document is not susceptible to these challenges.

2. Terms 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. Suppress CA Certificates Flag

The goal is when a client or server has the intermediate CAs to build the certificate chain for the peer it is establishing a TLS connection with, to signal to the peer to not send these certificates. TLS [RFC5246] [RFC8446] allows for the root CA certificate to be omitted from the handshake under the assumption that the remote peer already possesses it in order to validate its peers. Thus, a client or server in possession of the CA certificates would only need the peer end-entity certificate to validate its identity which would alleviate the data flowing in TLS.

This draft assumes that the endpoint can keep as set of ICAs in memory to use them while building certificate chains to authenticate a peer. Most usually the set will be stored locally in non-volatile memory. In constrained devices the intermediates could be cached, kept and updated only in volatile memory especially when the communicating peers' PKI domains are limited.

How CA certificates are identified and stored is dependent on the use case. In some use cases (e.g. WebPKI [ICA-PRELOAD]) the peer may assume that all intermediates are assembled, distributed and updated regularly using an out-of-band mechanism. In other use cases when the communicating peers' PKI domains are limited and not all CA certificates can be stored (i.e., constrained devices), or distributed, intermediates could be cached and updated dynamically using a caching mechanism. Such mechanisms are discussed in [TLS-SUPPRESS].

Although this document uses mechanisms to minimize TLS authentication failures due to stale or incomplete ICA lists, an endpoint is expected to re-attempt a TLS connection if it failed to authenticate a peer certificate after requesting ICA suppression. [EDNOTE: draft-ietf-tls-esni already requires the client to retry a connection when ECH is "securely replaced by the server" or "securely disabled by the server".]

[EDNOTE: To prevent failuers, one additional option could be to use a TLS extension like the one defined in [RFC7924] to include the chain fingerprint so the peer can confirm that he does not need to send the chain because the peer asking for suppression has the correct chain to validate the server. That could prevent inadvertent mistakes where the client thinks it has the intermediates to validate the server, but what it has is wrong. The shortcoming is that could be used as a cookie. Alternatively we could HMAC the chain to make it indistinguisable. Another option is for the server to provide a ticket so client returning visits tell the server that the client has the ICAs and it does not need to send them. These

options require further evaluation only if we think that the complexity is worth the benefit.]

The 0xTBD1 flag used to signal CA suppression can only be sent in a ClientHello or CertificateRequest message as defined below. Endpoints that receive a 0xTBD1 flag with a value of 1 in any other handshake message MUST generate a fatal illegal_parameter alert.

3.1. Client

A client that believes that it has a current, complete set of intermediate certificates to authenticate the server sends the tls_flags extension [TLS-FLAGS] with the 0xTBD1 flag set to 1 in its ClientHello message.

To prevent a failed TLS connection, a client MAY choose not to send the flag if its list of ICAs hasn't been updated in TBD3 time or has any other reason to believe it does not include the ICAs for its peer.

A server that receives a value of 1 in the 0xTBD1 flag of a ClientHello message SHOULD omit all certificates other than the end-entity certificate from its Certificate message that it sends in response. Otherwise if it does not support CA certificate suppression, the server SHOULD ignore the 0xTBD1 flag.

To prevent a failed TLS connection, a server could choose to send its intermediates regardless of the flag from the client, if it has a reason to believe the issuing CAs do not exist in the client ICA list. For example, if the server's certificate chain contains ICAs with technical constraints which are not disclosed, the server SHOULD send the chain back to the client regardless of the suppression flag in the ClientHello.

If the connection still fails because the client cannot build the certificate chain to authenticate the server, the client MUST NOT send the flag in a subsequent connection to the server.

3.2. Server (mutual TLS authentication)

In a mutual TLS authentication scenario, a server that believes that it has a current, complete set of intermediate certificates to authenticate the client, sends the tls_flags extension [TLS-FLAGS] with the 0xTBD1 flag set to 1 in its CertificateRequest message.

To prevent a failed TLS connection, a server MAY choose not to send the flag if its list of ICAs hasn't been updated in TBD3 time or has any other reason to believe it does not include the ICAs for its peer.

A client that receives a value of 1 in the 0xTBD1 flag in a CertificateRequest message SHOULD omit all certificates other than the end-entity certificate from the Certificate message that it sends in response. Otherwise if it does not support CA certificate suppression, the client SHOULD ignore the 0xTBD flag.

To prevent a failed TLS connection, a client could choose to send its intermediates regardless of the flag from the server, if it has a reason to believe the issuing CAs do not exist in the server ICA list. For example, if the client's certificate chain contains ICAs with technical constraints which are not disclosed, the client SHOULD send the chain back to the server regardless of the CA suppression flag in the CertificateRequest. [EDNOTE: MSRP 2.8 may require constrained intermediates which would mean this could change for WebPKI.]

If the connection still fails because the server cannot build the certificate chain to authenticate the client, the server MUST NOT send the flag in a subsequent connection from the client. [EDNOTE: There is a challenge with this in that the server needs to keep track of failed client connections.]

4. Security Considerations

This document creates an unencrypted signal in the ClientHello that might be used to identify which clients believe that they have intermediates to build the certificate chain for their peer. Although it does not reveal any additional information about the peers, it might allow clients to be more effectively fingerprinted by peers or any passive observers in the network path. A mitigation against this concern is to encrypt the ClientHello in TLS 1.3 [ESNI] which would hide the CA certificate suppression signal.

Even when the 0xTBD1 flag is encrypted in the handshake, a passive observer could fingerprint the peers by analyzing the TLS handshake data sizes flowing each direction. Widespread adoption of the TLS CA suppression mechanism described in this document will deem the use of the signal for fingerprinting impractical.

5. IANA Considerations

This document registers the 0xTBD1 in the registry created by [TLS-FLAGS].

6. Acknowledgements

We would like to thank Ilari Liusvaara, Ryan Sleevi Filippo Valsorda and for their valuable feedback contributions to this document.

The authors would also like to thank Filippo Valsorda for his feedback regarding ICA lists [FILOSOTTILE].

7. References

7.1. Normative References

[RFC2119] Bradner, S. and RFC Publisher, "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/info/rfc2119>.

[RFC8174] Leiba, B. and RFC Publisher, "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/info/rfc8174>.

[TLS-FLAGS] Nir, Y., "A Flags Extension for TLS 1.3", Work in Progress, Internet-Draft, draft-ietf-tls-tlsflags-10, 26 July 2022, <https://www.ietf.org/archive/id/draft-ietf-tls-tlsflags-10.txt>.

7.2. Informative References

[CBOR-CERTS] Mattsson, J. P., Selander, G., Raza, S., Höglund, J., and M. Furuhed, "CBOR Encoded X.509 Certificates (C509 Certificates)", Work in Progress, Internet-Draft, draft-ietf-cose-cbor-encoded-cert-04, 10 July 2022, <https://www.ietf.org/archive/id/draft-ietf-cose-cbor-encoded-cert-04.txt>.

[CL-BLOG] Westerbaan, B.E., "Sizing Up Post-Quantum Signatures", November 2021, <https://blog.cloudflare.com/sizing-up-post-quantum-signatures/>.

[CONEXT-PQTLS13SSH] Sikeridis, D., Kampanakis, P., and M. Devetsikiotis, "Assessing the Overhead of Post-Quantum Cryptography in TLS 1.3 and SSH", DOI 10.1145/3386367.3431305, ISBN 9781450379489, November 2020, <https://doi.org/10.1145/3386367.3431305>.

[CTLS] Rescorla, E., Barnes, R., Tschofenig, H., and B. M. Schwartz, "Compact TLS 1.3", Work in Progress, Internet-Draft, draft-ietf-tls-ctls-07, 3 January 2023, <https://www.ietf.org/archive/id/draft-ietf-tls-ctls-07.txt>.

[EAP-TLS13] Mattsson, J. P. and M. Sethi, "EAP-TLS 1.3: Using the Extensible Authentication Protocol with TLS 1.3", Work in Progress, Internet-Draft, draft-ietf-emu-eap-tls13-21, 20

October 2021, <https://www.ietf.org/archive/id/draft-ietf-emu-eap-tls13-21.txt>.

[EAPTLSCERT] Sethi, M., Mattsson, J. P., and S. Turner, "Handling Large Certificates and Long Certificate Chains in TLS-Based EAP Methods", Work in Progress, Internet-Draft, draft-ietf-emu-eaptlscert-08, 20 November 2020, <https://www.ietf.org/archive/id/draft-ietf-emu-eaptlscert-08.txt>.

[ESNI] Rescorla, E., Oku, K., Sullivan, N., and C. A. Wood, "TLS Encrypted Client Hello", Work in Progress, Internet-Draft, draft-ietf-tls-esni-15, 3 October 2022, <https://www.ietf.org/archive/id/draft-ietf-tls-esni-15.txt>.

[FILOSOTTILE] Valsorda, F., "filippo.io/intermediates", 2022, <https://github.com/FiloSottile/intermediates>.

[I-D.hoffman-c2pq] Hoffman, P. E., "The Transition from Classical to Post-Quantum Cryptography", Work in Progress, Internet-Draft, draft-hoffman-c2pq-07, 26 May 2020, <https://www.ietf.org/archive/id/draft-hoffman-c2pq-07.txt>.

[I-D.ietf-tls-hybrid-design] Stebila, D., Fluhrer, S., and S. Gueron, "Hybrid key exchange in TLS 1.3", Work in Progress, Internet-Draft, draft-ietf-tls-hybrid-design-05, 28 August 2022, <https://www.ietf.org/archive/id/draft-ietf-tls-hybrid-design-05.txt>.

[ICA-PRELOAD] Keeler, D., "Preloading Intermediate CA Certificates into Firefox", November 2020, <https://blog.mozilla.org/security/2020/11/13/preloading-intermediate-ca-certificates-into-firefox/>.

[IEEE802154] "IEEE Standard for Low-Rate Wireless Networks", DOI 10.1109/IEEESTD.2020.9144691, July 2020, <https://doi.org/10.1109/IEEESTD.2020.9144691>.

[NDSS-PQTLS13] Sikeridis, D., Kampanakis, P., and M. Devetsikiotis, "Post-Quantum Authentication in TLS 1.3: A Performance Study", DOI 10.14722/ndss.2020.24203, February 2020, <https://doi.org/10.14722/ndss.2020.24203>.

[NIST_PQ] NIST, ., "Post-Quantum Cryptography", 2021, <https://csrc.nist.gov/projects/post-quantum-cryptography>.

[PQTLS] Paquin, C., Stebila, D., and G. Tamvada, "Benchmarking Post-Quantum Cryptography in TLS", 2019, <https://ia.cr/2019/1447>.

[QUIC-CERTS] Nawrocki, M., Tehrani, P., Hiesgen, R., Mucke, J., Schmidt, T., and M. Wahlisch, "On the Interplay between TLS Certificates and QUIC Performance", DOI 10.1145/3555050.3569123, November 2022, <https://doi.org/10.1145/3555050.3569123>.

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

[RFC6928] Chu, J., Dukkipati, N., Cheng, Y., Mathis, M., and RFC Publisher, "Increasing TCP's Initial Window", RFC 6928, DOI 10.17487/RFC6928, April 2013, <https://www.rfc-editor.org/info/rfc6928>.

[RFC7924] Santesson, S., Tschofenig, H., and RFC Publisher, "Transport Layer Security (TLS) Cached Information Extension", RFC 7924, DOI 10.17487/RFC7924, July 2016, <https://www.rfc-editor.org/info/rfc7924>.

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

[RFC8879] Ghedini, A., Vasiliev, V., and RFC Publisher, "TLS Certificate Compression", RFC 8879, DOI 10.17487/RFC8879, December 2020, <https://www.rfc-editor.org/info/rfc8879>.

[TLS-SUPPRESS] Kampanakis, P. and M. Kallitsis, "Speeding up post-quantum TLS handshakes by suppressing intermediate CA certificates", 2021, <https://www.amazon.science/publications/speeding-up-post-quantum-tls-handshakes-by-suppressing-intermediate-ca-certificates>.

[WISUN] "WI-SUN Alliance", n.d., <https://wi-sun.org/>.

Authors' Addresses

Panos Kampanakis
AWS

Email: kpanos@amazon.com

Cameron Bytheway
AWS

Email: bythewc@amazon.com

Bas Westerbaan
Cloudflare

Email: bas@cloudflare.com

Martin Thomson
Mozilla

Email: mt@lowentropy.net