Token Binding Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 30, 2017

N. Harper
Google Inc.

June 28, 2017

Token Binding for 0-RTT TLS 1.3 Connections
draft-ietf-tokbind-tls13-0rtt-02

Abstract

This document describes how Token Binding can be used in the 0-RTT data of a TLS 1.3 connection. This involves a new TLS extension to negotiate and indicate the use of Token Binding in 0-RTT data. A TokenBindingMessage sent in 0-RTT data has different security properties than one sent after the TLS handshake has finished, which this document also describes.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on December 30, 2017.

Table of Contents

1. Introduction

Token Binding ([I-D.ietf-tokbind-protocol]) cryptographically binds security tokens (e.g. HTTP cookies, OAuth tokens) to the TLS layer on which they are presented. It does so by signing an [RFC5705] exporter value from the TLS connection. TLS 1.3 introduces a new mode that allows a client to send application data on its first flight. If this 0-RTT data contains a security token, then a client using Token Binding would want to prove possession of its Token Binding private key so that the server can verify the binding. The [RFC5705]-style exporter provided by TLS 1.3 cannot be run until the handshake has finished. TLS 1.3 also provides an exporter that can be used with 0-RTT data, but it requires that the application explicitly specify that use. This document specifies how to use the early_exporter_secret with Token Binding in TLS 1.3 0-RTT data.

Using Token Binding in 0-RTT data involves two main changes to Token Binding. The first is the use of a new TLS extension

"early_token_binding" to indicate whether a TLS session ticket can be used with Token Binding in 0-RTT data, and to indicate whether an attempted 0-RTT connection is using Token Binding in 0-RTT data. The second change is one that applies only if Token Binding in 0-RTT data is in use, which changes the definition of the TokenBinding.signature field to use TLS 1.3's early_exporter_secret.

If a client does not send any 0-RTT data, or if the server rejects the client's 0-RTT data, then the client MUST use the 1-RTT exporter, as defined in [I-D.ietf-tokbind-protocol].

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2. TokenBinding Signature Definition

In [I-D.ietf-tokbind-protocol], the signature field of the TokenBinding struct is defined to be the signature of a concatentation that includes the EKM value. Depending on the circumstances, the exporter value in section 7.3.3 of [I-D.ietf-tls-tls13] is computed using either exporter_secret or early_exporter_secret as the Secret.

When early_exporter_secret is used as the Secret, the client MUST indicate this use so the server knows which secret to use in signature verification. This indication is done through a new Token Binding extension, "early_exporter" (with extension type TBD). This extension always has 0-length data, so the full Extension struct is the bytes {0xTBD, 0x00, 0x00}. The early_exporter extension MUST be present in every TokenBinding struct where the exporter that is signed uses the early_exporter_secret, and it MUST NOT be present in any other TokenBinding structs.

2.1. Selecting Which Exporter Secret to Use

A client which is not sending any 0-RTT data on a connection MUST use the exporter defined in [I-D.ietf-tls-tls13] (using exporter_secret as the Secret) for all TokenBindingMessages on that connection so that it is compatible with [I-D.ietf-tokbind-protocol].

When a client sends a TokenBindingMessage in 0-RTT data, it must use the early_exporter_secret. If the server accepts the 0-RTT data, the client must continue to use the early_exporter_secret for the rest of the connection. If the server rejects 0-RTT data, the client must use the exporter_secret.

3. Negotiation

3.1. Indicating Use of 0-RTT Token Binding

The TLS extension "early_token_binding" (extension type TBD) is used in the TLS ClientHello and EncryptedExtensions to indicate use of 0-RTT Token Binding on the current connection. It is also used in a NewSessionTicket message to indicate that 0-RTT Token Binding may be used on a connection resumed with that ticket. In all cases, the "extension_data" field of this extension is empty, so the entire encoding of this extension is 0xTBD 0xTBD 0x00 0x00.

3.2. Token Binding Negotiation TLS Extension

In TLS 1.3, the "token_binding" extension is sent by a server in EncryptedExtensions, whereas in previous versions of TLS this extension was sent in the ServerHello message. On a 1-RTT connection (whether it be a new connection or resumption), no application data is sent in either direction before the "token_binding" TLS extension in the EncryptedExtensions, and the choice of Token Binding version and key parameter is up to the server based on what the client sent and what the server's preferences are, following the same processing rules as in [I-D.ietf-tokbind-negotiation].

3.3. Client Processing Rules

A client that supports Token Binding in 0-RTT data and receives a NewSessionTicket containing the "early_token_binding" extension must store with the ticket the Token Binding version and key parameter of the connection in which the ticket was issued.

A client that wishes to send a Token Binding message in 0-RTT data may only do so if the TLS connection in which the 0-RTT data is being sent is being resumed from a ticket which included the "early_token_binding" extension. Assuming the ticket included this extension, the client sends a ClientHello containing the "token_binding" extension, "early_data" extension, and "early_token_binding" extensions. The client must include in its "psk_key_exchange_modes" extension psk_dhe_ke.

The contents of the "token_binding" extension SHOULD be the same as they would be on a connection without "early_token_binding" to allow for the client and server to negotiate new Token Binding parameters if the early data is rejected. The Token Binding message sent in the 0-RTT data MUST be sent assuming that the same Token Binding version and key parameter from the connection where the ticket was received will also be negotiated on this connection. If the server includes the "early_data" extension in EncryptedExtensions in response to a

ClientHello with "early_token_binding", but the server does not include "early_token_binding" in EncryptedExtensions, or if the server's "token_binding" extension does not match the values of the connection where the ticket was received, then the client MUST terminate the TLS connection with an illegal_parameter alert.

It is valid for a client to send a ClientHello that contains both the "early_data" and "token_binding" extensions, but without the "early_token_binding" extension. This combination means that the client is attempting to resume a connection and is sending early data, but the client is not using Token Binding on this resumed connection (if the server accepts the early data). The presence of the "token_binding" extension is so the client can negotiate the use of Token Binding for this connection if the server rejects early data.

3.4. Server Processing Rules

When a server issues a NewSessionTicket on a connection where Token Binding was negotiated, and the NewSessionTicket includes an "early_data" extension indicating that the ticket may be used to send 0-RTT data, the server may also include the "early_token_binding" extension in the NewSessionTicket to indicate that this ticket can be used for a future connection with Token Binding in 0-RTT data. If the server includes the "early_token_binding" extension in the NewSessionTicket, the server MUST store with the ticket the Token Binding version and key parameter used for the connection in which the ticket was issued. The "early_token_binding" extension can appear in a NewSessionTicket message only if the "early_data" extension also appears in that message.

If a server receives a ClientHello with the "early_token_binding" extension and supports Token Binding in 0-RTT data, it MUST perform the following checks:

- If either the "early_data" or "token_binding" extensions are missing from the ClientHello, terminate the TLS connection with an illegal_parameter alert.

- If the ticket used for resumption is missing either of the "early_data" or "early_token_binding" extensions, reject the early data.

- Process the "token_binding" extension as if it were received on a 1-RTT connection and compute the Token Binding version and key parameter to use. If either of these values do not match the values that were negotiated on the connection where the ticket used for resumption was sent, reject the early data.

- Perform any other checks to decide whether to accept early data. If the server chooses to accept early data, include in EncryptedExtensions the "early_data" extension, "early_token_binding" extension, and "token_binding" extension with the same version and key parameter from the previous connection.

If a server accepts early data on a connection where "early_token_binding" was offered, it MUST use PSK with (EC)DHE key establishment.

The "early_token_binding" extension must be present in EncryptedExtensions exactly when both "early_data" and "token_binding" are present. A server that receives a ClientHello with "early_token_binding" cannot reject Token Binding and also accept early data at the same time. Said server may reject early data but still negotiate Token Binding.

A server might receive a ClientHello that includes both the "early_data" and "token_binding" extensions, but no "early_token_binding" extension. In this case, the server has three options:

1. Accept early data and continue the connection with no Token Binding,

2. Reject early data and negotiate the use of Token Binding for this connection, or

3. Reject early data and do not negotiate Token Binding for this connection.

The behavior for the "token_binding" extension in 0-RTT is similar to that of ALPN and SNI: the client predicts the result of the negotiation, and if the actual negotiation differs, the server rejects the early data.

4. Implementation Considerations

4.1. Not Implementing Token Binding on 0-RTT Connections

This spec has been designed so that both clients and servers can support Token Binding on some connections and 0-RTT data on other connections without needing to support Token Binding on 0-RTT connections.

A client that wishes to support both without supporting Token Binding on 0-RTT connections can function by completely ignoring the

"early_token_binding" TLS extension. When resuming a connection with early data, the client can still advertise support for Token Binding, providing the server the opportunity to accept early data (without Token Binding) or to reject early data and negotiate Token Binding. By always including the "token_binding" extension in its ClientHello, the client can prioritize Token Binding over 0-RTT.

A server can support both Token Binding and 0-RTT data without supporting Token Binding on 0-RTT connections by never minting NewSessionTickets containing the "early_token_binding" extension. Such a server that never mints NewSessionTickets with "early_token_binding" can ignore that extension in a ClientHello as it would only appear if the client is not spec compliant. On connections where a server negotiates Token Binding, the server SHOULD NOT include the "early_data" extension in a NewSessionTicket.

4.2. Adding Support for Token Binding on 0-RTT Connections

A server that supports early data but not Token Binding may wish to add support for Token Binding (and Token Binding on 0-RTT connections) at a later time. For a client to learn that a server supports Token Binding, the server must reject early data to send the "token_binding" extension.

4.3. Implementation Challenges

The client has to be able to modify the message it sends in 0-RTT data if the 0-RTT data gets rejected and needs to be retransmitted in 1-RTT data. Even if the Token Binding integration with 0-RTT were modified so that Token Binding never caused a 0-RTT reject that required rewriting a request, the client still has to handle the server rejecting the 0-RTT data for other reasons.

HTTP2 allows for requests to different domains to share the same TLS connection if the SAN of the cert covers those domains. If one.example.com supports 0-RTT and Token Binding, but two.example.com only supports Token Binding as defined in [I-D.ietf-tokbind-protocol], those servers cannot share a cert and use HTTP2.

5. IANA Considerations

This document defines a new TLS extension "early_token_binding" with code point TBD which needs to be added to IANA's TLS "ExtensionType Values" registry.

This document defines a new Token Binding extension "early_exporter", which needs to be added to the IANA "Token Binding Extensions" registry.

6. Security Considerations

Token Binding messages that use the 0-RTT exporter have weaker security properties than with the [RFC5705] exporter. If either party of a connection using Token Binding does not wish to use 0-RTT token bindings, they can do so: a client can choose to never send 0-RTT data on a connection where it uses token binding, and a server can choose to reject any 0-RTT data sent on a connection that negotiated token binding.

0-RTT data in TLS 1.3 has weaker security properties than other kinds of TLS data. Specifically, TLS 1.3 does not guarantee non-replayability of data between connections. Token Binding has similar replayability issues when in 0-RTT data, but preventing replay of Token Binding and preventing replay of 0-RTT data are two separate problems. Token Binding is not designed to prevent replay of 0-RTT data, although solutions for preventing the replay of Token Binding might also be applicable to 0-RTT data.

6.1. Proof of Possession of Token Binding Key

When a Token Binding signature is generated using the exporter with early_exporter_secret, the value being signed is under the client's control. An attacker with temporary access to the Token Binding private key can generate Token Binding signatures for as many future connections as it has NewSessionTickets for. An attacker can construct these to be usable at any time in the future up until the NewSessionTicket's expiration. Section 4.6.1 of [I-D.ietf-tls-tls13] requires that a NewSessionTicket be valid for a maximum of 7 days.

Unlike in [I-D.ietf-tokbind-protocol], where the proof of possession of the Token Binding key proves that the client had possession at the time the TLS handshake finished, 0-RTT Token Binding only proves that the client had possession of the Token Binding key at some point after receiving the NewSessionTicket used for that connection.

6.2. Exporter Replayability

The exporter specified in [I-D.ietf-tokbind-protocol] is chosen so that a client and server have the same exporter value only if they are on the same TLS connection. This prevents an attacker who can read the plaintext of a TokenBindingMessage sent on that connection from replaying that message on another connection (without also having the token binding private key). The 0-RTT exporter only

covers the ClientHello and the PSK of the connection, so it does not provide this guarantee.

An attacker with possession of the PSK secret and a transcript of the ClientHello and early data sent by a client under that PSK can extract the TokenBindingMessage, create a new connection to the server (using the same ClientHello and PSK), and send different application data with the same TokenBindingMessage. Note that the ClientHello contains public values for the (EC)DHE key agreement that is used as part of deriving the traffic keys for the TLS connection, so if the attacker does not also have the corresponding private values, they will not be able to read the server's response or send a valid Finished message in the handshake for this TLS connection. Nevertheless, by that point the server has already processed the attacker's message with the replayed TokenBindingMessage.

This sort of replayability of a TokenBindingMessage is different than the replayability caveat of 0-RTT application data in TLS 1.3. A network observer can replay 0-RTT data from TLS 1.3 without knowing any secrets of the client or server, but the application data that is replayed is untouched. This replay is done by a more powerful attacker who is able to view the plaintext and then spoof a connection with the same parameters so that the replayed TokenBindingMessage still validates when sent with different application data.

6.3. Replay Mitigations

This section presents multiple ways that a client or server can mitigate the replay of a TokenBinding while still using Token Binding with 0-RTT data. Note that even with replay mitigations, 0-RTT Token Binding is vulnerable to other attacks.

6.3.1. Server Mitigations

If a server uses a session cache instead of stateless tickets, it can enforce that a PSK generated for resumption can only be used once. If an attacker tries to replay 0-RTT data (with a TokenBindingMessage), the server will reject it because the PSK was already used.

Preventing all replay of 0-RTT data is not necessary to prevent replay of a TokenBinding. A server could implement a mechanism to prevent a particular TokenBinding from being presented on more than one connection. In cases where a server's TLS termination and application layer processing happen in different locations, this option might be easier to implement, especially when not all requests have bound tokens. This processing can also take advantage of the

structure of the bound token, e.g. a token that identifies which user is making a request could shard its store of which TokenBindings have been seen based on the user ID.

A server can prevent some, but not all, 0-RTT data replay with a tight time window for the ticket age that it will accept. See Section 6.4 for more details.

6.3.2. Client Mitigations

A client cannot prevent a sufficiently motivated attacker from replaying a TokenBinding, but it can make it so difficult to replay the TokenBinding that it is easier for the attacker to steal the Token Binding key directly. If the client secures the resumption secret with the same level of protection as the Token Binding key, then the client has made it not worth the effort of the attacker to attempt to replay a TokenBinding. Ideally the resumption secret (and Token Binding key) are protected strongly and virtually non-exportable.

6.4. Early Data Ticket Age Window

When an attacker with control of the PSK secret replays a TokenBindingMessage, it has to use the same ClientHello that the client used. The ClientHello includes an "obfuscated_ticket_age" in its EarlyDataIndication extension, which the server can use to narrow the window in which that ClientHello will be accepted. Even if a PSK is valid for a week, the server will only accept that particular ClientHello for a smaller time window based on the ticket age. A server should make their acceptance window for this value as small as practical to limit an attacker's ability to replay a ClientHello and send new application data with the stolen TokenBindingMessage.

7. Acknowledgements

The author would like to thank David Benjamin, Steven Valdez, Bill Cox, and Andrei Popov for their feedback and suggestions.

8. Normative References

[I-D.ietf-tls-tls13]
Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", draft-ietf-tls-tls13-20 (work in progress), April 2017.

[I-D.ietf-tokbind-negotiation]
Popov, A., Nystrom, M., Balfanz, D., and A. Langley, "Transport Layer Security (TLS) Extension for Token Binding Protocol Negotiation", draft-ietf-tokbind-negotiation-08 (work in progress), April 2017.

[I-D.ietf-tokbind-protocol]
Popov, A., Nystrom, M., Balfanz, D., Langley, A., and J. Hodges, "The Token Binding Protocol Version 1.0", draft-ietf-tokbind-protocol-14 (work in progress), April 2017.

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

[RFC5705] Rescorla, E., "Keying Material Exporters for Transport Layer Security (TLS)", RFC 5705, DOI 10.17487/RFC5705, March 2010, <http://www.rfc-editor.org/info/rfc5705>.

Author's Address

Nick Harper
Google Inc.

Email: nharper@google.com