

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: December 31, 2017

B. Campbell
Ping Identity
June 29, 2017

HTTPS Token Binding with TLS Terminating Reverse Proxies
draft-campbell-tokbind-ttrp-00

Abstract

This document defines common HTTP header fields that enable a TLS terminating reverse proxy to convey information about the validated Token Binding Message sent by the client to a backend server, which enables that backend server to bind, or verify the binding of, cookies and other security tokens to the client's Token Binding key.

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 31, 2017.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
1.1. Requirements Notation and Conventions	3
2. HTTP Header Fields and Processing Rules	3
2.1. Token Binding ID HTTP Header Fields	3
2.2. Processing Rules	4
2.3. Examples	5
2.3.1. Provided Token Binding ID	5
2.3.2. Provided and Referred Token Binding IDs	6
3. Security Considerations	6
4. IANA Considerations	7
4.1. HTTP Message Header Field Names Registration	7
5. References	8
5.1. Normative References	8
5.2. Informative References	9
Appendix A. Acknowledgements	9
Appendix B. Open Issues	9
Appendix C. Document History	10
Author's Address	10

1. Introduction

Token Binding over HTTP [I-D.ietf-tokbind-https] provides a mechanism that enables HTTP servers to cryptographically bind cookies and other security tokens to a key held by the browser or other HTTP client, possession of which is proven on the TLS [RFC5246] connections over which the tokens are used. When Token Binding is negotiated in the TLS handshake [I-D.ietf-tokbind-negotiation] the client sends an encoded Token Binding Message [I-D.ietf-tokbind-protocol] as a header in each HTTP request, which proves possession of one or more private keys held by the client. The public portion of the keys are represented in the Token Binding IDs of the Token Binding Message and for each one there is a signature over some data, which includes the exported keying material [RFC5705] of the TLS connection. An HTTP server issuing cookies or other security tokens can associate them with the Token Binding ID, which ensures those tokens cannot be used successfully over a different TLS connection or by a different client than the one to which they were issued.

A fairly common deployment architecture for HTTPS applications is to have the backend HTTP application servers sit behind a reverse proxy that terminates TLS. The proxy is accessible to the internet and dispatches client requests to the appropriate backend server within a private or protected network. The backend servers are not directly accessible outside the private network and are only reachable through the reverse proxy. The details of such deployments are typically opaque to clients who make requests to the proxy server and see

responses as though they originated from the proxy server itself. TLS connections for HTTPS are established between each client and the reverse proxy server.

Token Binding facilitates a binding of security tokens to a key held by the client by way of the TLS connection between that client and the server. In a deployment where TLS is terminated by a reverse proxy, however, the TLS connection is between the client and the proxy while the backend server is likely the system that will issue cookies or other security tokens. Additional steps are therefore needed to enable the use of Token Binding in such deployment architectures. In the absence of a standardized approach, different implementations will address it differently, which will make interoperability between implementation difficult or impossible without complex configurations or custom integrations.

This document standardizes HTTP header field names that a TLS terminating reverse proxy (TTRP) adds to requests that it sends to the backend servers. The headers contain the information from the validated Token Binding Message sent by the client to the proxy with the "Sec-Token-Binding" header, thus enabling the backend server to bind, or verify the binding of, cookies and other security tokens to the client's Token Binding key. The usage of the headers, both the reverse proxy adding it and the application server using them to bind cookies or other tokens, are to be configuration options of the respective systems as they will not always be applicable.

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 RFC 2119 [RFC2119].

2. HTTP Header Fields and Processing Rules

2.1. Token Binding ID HTTP Header Fields

The Token Binding Protocol [I-D.ietf-tokbind-protocol] recommends that implementations make Token Binding IDs available to the application as opaque byte sequences, enabling those applications to use the Token Binding IDs when generating and verifying bound tokens. In the context of a TLS terminating reverse proxy (TTRP) deployment, the provided and referred Token Binding IDs are made available to the backend application as the "Provided-Token-Binding-ID" and "Referred-Token-Binding-ID" HTTP headers respectively. The value of both headers is an "EncodedTokenBindingID", for which the ABNF [RFC5234] syntax is shown in Figure 1 below. "EncodedTokenBindingID" 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. An "EncodedTokenBindingID" is only for use in HTTP requests and MUST NOT to be used in HTTP responses.

```
EncodedTokenBindingID = *( DIGIT / ALPHA / "-" / "_" )

DIGIT = <Defined in Section B.1 of [RFC5234]>
ALPHA = <Defined in Section B.1 of [RFC5234]>
```

Figure 1: Encoded Token Binding ID Header ABNF

The value of an "EncodedTokenBindingID" is a base64url encoding of the TokenBindingID byte sequence (see section 3 of [I-D.ietf-tokbind-protocol]) using the URL and filename safe alphabet described in Section 5 of [RFC4648], with all trailing pad characters '=' omitted and without the inclusion of any line breaks, whitespace, or other additional characters.

2.2. Processing Rules

This section defines the applicable processing rules for a TLS terminating reverse proxy (TTRP) and backend server(s) to provide server side support of Token Binding over HTTP [I-D.ietf-tokbind-https] using the HTTP headers described in Section 2.1. Use of the technique is to be a configuration or deployments option and the processing rules described herein are for servers operating with that option enabled.

A TTRP negotiates the use of Token Binding with the client per [I-D.ietf-tokbind-negotiation] and validates the Token Binding Message as defined in The Token Binding Protocol [I-D.ietf-tokbind-protocol] and Token Binding over HTTP [I-D.ietf-tokbind-https] for each HTTP request on the underlying TLS connection. Requests with a valid Token Binding Message (and meeting any other authorization or policy requirements of the TTRP) are dispatched to the backend server with the following modifications.

1. The "Sec-Token-Binding" header in the original incoming request MUST be removed from the request that is dispatched to the backend server.
2. The Token Binding ID of the provided Token Binding of the Token Binding Message MUST be placed in the "Provided-Token-Binding-ID" header field of the dispatched request using the format defined in Section 2.1.

3. If the Token Binding Message contains a referred Token Binding, the referred Token Binding ID MUST be placed in the "Referred-Token-Binding-ID" header field of the dispatched request using the format defined in Section 2.1. Otherwise, the "Referred-Token-Binding-ID" header field MUST NOT be present in the dispatched request.
4. Any occurrence of the "Provided-Token-Binding-ID" or "Referred-Token-Binding-ID" header in the original incoming request MUST be removed or overwritten before forwarding the request.

Requests made over a TLS connection where the use of Token Binding was not negotiated MUST be sanitized by removing any occurrences of the "Provided-Token-Binding-ID" and "Referred-Token-Binding-ID" header fields prior to dispatching the request to the backend server.

Forward proxies and other intermediaries MUST NOT add the "Provided-Token-Binding-ID" or "Referred-Token-Binding-ID" header to requests.

2.3. Examples

Extra line breaks and whitespace have been added to the following examples for display and formatting purposes only.

2.3.1. Provided Token Binding ID

The following "Sec-Token-Binding" header is from an HTTP request made over a TLS connection between the client and the TTRP where the use of Token Binding has been negotiated (The base64url-encoded representation of the exported keying material, which can be used to validate the Token Binding Message, for that connection is "AYVUayPTP9RmELNpGjF16Ykm2CUx7pUMxe35yb1ldgU"). The encoded Token Binding Message has the provided Token Binding the client uses with the server.

```
Sec-Token-Binding: AIkAAgBBQKzyIrmcY_YCtHVoSHBut69vrGfFdyl_YKTZfFJv
6BjrZsKD9b9FRzSBxDsltwTqnAS7lMlRBumuihhI9xqxXKkAQEtXe4jeUJU0WezxlQ
XWVSBFeHxFMdXRBIH_LKOSAUSMOJ0XEwlQ8DE248qkOiRKzw3KdSNYukYEPmO21bQi
3YYAAA
```

Figure 2: Header in HTTP Request to TTRP

After validating the Token Binding Message, the TTRP removes the "Sec-Token-Binding" header and adds the following "Provided-Token-Binding-ID" header with the provided Token Binding ID to the request that is dispatched to the backend server.

Producing and consuming the headers SHOULD be a configurable option, respectively, in a reverse proxy and backend server (or individual application in that server). The default configuration for both should be to not use the headers thus requiring an "opt-in" to the functionality.

Reverse proxies SHOULD only add the headers to requests that are forwarded to trusted backend servers.

Backend servers MUST only accept the headers from trusted reverse proxies. And reverse proxies MUST sanitize the incoming request before forwarding it on by removing or overwriting any existing instances of the headers. Otherwise arbitrary clients can control the header values as seen and used by the backend server.

The communication between a reverse proxy 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 reverse proxy and backend server, for example, might be over a mutually authenticated TLS with the insertion and consumption headers occurring only on for that connection. Alternatively the network topology might dictate a private network such that the backend application is only able to accept requests from the reverse proxy and the proxy can only make requests to that server. Other deployments that meet the requirements set forth herein are also possible.

4. IANA Considerations

4.1. HTTP Message Header Field Names Registration

This document specifies the following new HTTP header fields, registration of which is requested in the "Permanent Message Header Field Names" registry defined in [RFC3864].

- o Header Field Name: "Provided-Token-Binding-ID"
- o Applicable protocol: HTTP
- o Status: standard
- o Author/change Controller: IETF
- o Specification Document(s): [[this specification]]

- o Header Field Name: "Referred-Token-Binding-ID"
- o Applicable protocol: HTTP
- o Status: standard

- o Author/change Controller: IETF
- o Specification Document(s): [[this specification]]

5. References

5.1. Normative References

- [I-D.ietf-tokbind-https]
Popov, A., Nystrom, M., Balfanz, D., Langley, A., and J. Hodges, "Token Binding over HTTP", draft-ietf-tokbind-https-09 (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>>.
- [RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <<http://www.rfc-editor.org/info/rfc4648>>.
- [RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <<http://www.rfc-editor.org/info/rfc5234>>.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <<http://www.rfc-editor.org/info/rfc5246>>.
- [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>>.

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

5.2. Informative References

[RFC3864] Klyne, G., Nottingham, M., and J. Mogul, "Registration Procedures for Message Header Fields", BCP 90, RFC 3864, DOI 10.17487/RFC3864, September 2004, <<http://www.rfc-editor.org/info/rfc3864>>.

Appendix A. Acknowledgements

The author would like to thank the following people for their various contributions to the specification: Vinod Anupam, Dirk Balfanz, John Bradley, Subodh Iyengar, Leif Johansson, Yoav Nir, Andrei Popov, Eric Rescorla, Piotr Sikora, Martin Thomson, Hans Zandbelt and others (please let me know, if you've contributed and I've forgotten you).

Appendix B. Open Issues

- o During discussions at a side meeting in Chicago (IETF 98) there seemed to be general support for having the TTRP rename the "Sec-Token-Binding" header to something else and pass the full original EncodedTokenBindingMessage to the backend server via a different header (maybe "TTRP-Token-Binding" or something) in addition to the "Provided-Token-Binding-ID" and if applicable the "Referred-Token-Binding-ID" headers defined herein. The idea was largely that the backend server "might need it for something" so pass the whole thing along just in case. However, as I sat down to write this draft, I couldn't bring myself to add it in the main text. On thinking about it more, it feels inefficient/duplicative and rather inelegant. And without the EKM, much of the data not already made available via the Token Binding IDs is meaningless (e.g. the signature value). Data in TokenBinding.extensions, if extensions are present, might be useful to the backend server. But might also only be useful/meaningful at the TTRP where the initial TLS connection is terminated. I really don't know. Perhaps any extensions, if present, should be passed to the backend via different header(s)? Or maybe it would be more appropriate to not attempt to cover TokenBinding.extensions in this document and defer to the definition of individual extensions to say how/if they are to be handled in a TTRP type deployment?

Appendix C. Document History

[[to be removed by the RFC Editor before publication as an RFC]]

draft-campbell-tokbind-ttrp-00

- o Initial draft based on 'consensus to work on the problem' from the Seoul meeting [1][2] and reflecting the consensus approach from discussions at the Chicago meeting [3].

[1] <https://www.ietf.org/proceedings/97/minutes/minutes-97-tokbind-01.txt> (minutes from Seoul)

[2] <https://www.ietf.org/proceedings/97/slides/slides-97-tokbind-reverse-proxies-00.pdf> (slides from Seoul)

[3] https://mailarchive.ietf.org/arch/msg/unbearable/_ZHI8y2Vs5WMP8VMRr7zroo_sNU (summary of discussion)

Author's Address

Brian Campbell
Ping Identity

Email: brian.d.campbell@gmail.com

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.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

"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 concatenation 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:

- o If either the "early_data" or "token_binding" extensions are missing from the ClientHello, terminate the TLS connection with an illegal_parameter alert.
- o If the ticket used for resumption is missing either of the "early_data" or "early_token_binding" extensions, reject the early data.
- o 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.

- o 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