                        Token Binding over HTTP
                   draft-balfanz-https-token-binding-00

Abstract

   This document describes a collection of mechanisms that allow HTTP
   servers to cryptographically bind authentication tokens (such as
   cookies and OAuth tokens) to a TLS [RFC5246] connection.

   We describe both _first-party_ as well as _federated_ scenarios. In
   a first-party scenario, an HTTP server issues a security token (such
   as a cookie) to a client, and expects the client to send the security
   token back to the server at a later time in order to authenticate.
   Binding the token to the TLS connection between client and server
   protects the security token from theft, and ensures that the security
   token can only be used by the client that it was issued to.

   Federated token bindings, on the other hand, allow servers to
   cryptographically bind security tokens to a TLS [RFC5246] connection
   that the client has with a _different_ server than the one issuing
   the token.

   This Internet-Draft is a companion document to The Token Binding
   Protocol [DraftPopov]

This Internet-Draft will expire on April 16, 2015.

Copyright Notice

Table of Contents

1. Introduction

   The Token Binding Protocol [DraftPopov] defines a Token Binding ID
   for a TLS connection between a client and a server. The Token
   Binding ID of a TLS connection is related to a private key that the
   client proves possession of to the server, and is long-lived (i.e.,
   subsequent TLS connections between the same client and server have
   the same Token Binding ID). When issuing a security token (e.g. an
   HTTP cookie or an OAuth token) to a client, the server can include
   the Token Binding ID in the token, thus cryptographically binding the

token to TLS connections between that particular client and server,
and inoculating the token against theft by attackers.

While the Token Binding Protocol [DraftPopov] defines a message
format for establishing a Token Binding ID, it doesn't specify how
this message is embedded in higher-level protocols. The purpose of
this specification is to define how TokenBindingMessages are embedded
in HTTP (both versions 1.1 [RFC2616] and 2 [I-D.ietf-httpbis-http2]).
Note that TokenBindingMessages are only defined if the underlying
transport uses TLS. This means that Token Binding over HTTP is only
defined when the HTTP protocol is layered on top of TLS (commonly
referred to as HTTPS).

HTTP clients establish a Token Binding ID with a server by including
a special HTTP header in HTTP requests. The HTTP header value is a
TokenBindingMessage.

TokenBindingMessages allow clients to establish multiple Token
Binding IDs with the server, by including multiple TokenBinding
structures in the TokenBindingMessage. By default, a client will
establish a _provided_ Token Binding ID with the server, indicating a
Token Binding ID that the client will persistently use with the
server. Under certain conditions, the client can also include a
referred Token Binding ID in the TokenBindingMessage, indicating a
Token Binding ID that the client is using with a _different_ server
than the one that the TokenBindingMessage is sent to. This is useful
in federation scenarios.

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. The Token-Binding Header

Once a client and server have negotiated the Token Binding Protocol
with HTTP/1.1 or HTTP/2 (see The Token Binding Protocol
[DraftPopov]), clients MUST include the following header in their
HTTP requests:

 Token-Binding: EncodedTokenBindingMessage

The EncodedTokenBindingMessage is a web-safe Base64-encoding of the
TokenBindingMessage as defined in the TokenBindingProtocol
[DraftPopov].

The TokenBindingMessage MUST contain a TokenBinding with
TokenBindingType provided_token_binding, which MUST be signed with
the Token Binding key used by the client for connections between
itself and the server that the HTTP request is sent to (clients use
different Token Binding keys for different servers). The Token
Binding ID established by this TokenBinding is called a _Provided
Token Binding ID_

In HTTP/2, the client SHOULD use Header Compression
[I-D.ietf-httpbis-header-compression] to avoid the overhead of
repeating the same header in subsequent HTTP requests.

3. Federation Use Cases

3.1. Introduction

For privacy reasons, clients use different private keys to establish
Provided Token Binding IDs with different servers. As a result, a
server cannot bind a security token (such as an OAuth token or an
OpenID Connect identity token) to a TLS connection that the client
has with a different server. This is, however, a common requirement
in federation scenarios: For example, an Identity Provider may wish
to issue an identity token to a client and cryptographically bind
that token to the TLS connection between the client and a Relying
Party.

In this section we describe mechanisms to achieve this. The common
idea among these mechanisms is that a server (called the _Token
Consumer_ in this document) gives the client permission to reveal the
Provided Token Binding ID that is used between the client and itself,
to another server (called the _Token Provider_ in this document).
Also common across the mechanisms is how the Token Binding ID is
revealed to the Token Provider: The client uses the Token Binding
Protocol [DraftPopov], and includes a TokenBinding structure in the
Token-Binding HTTP header defined above. What differs between the
various mechanisms is _how_ the Token Consumer grants the permission
to reveal the Token Binding ID to the Token Provider.

3.2. Overview

In a Federated Sign-On protocol, an Identity Provider issues an
identity token to a client, which sends the identity token to a
Relying Party to authenticate itself. Examples of this include
OpenID Connect (where the identity token is called "ID Token") and
SAML (where the identity token is a SAML assertion).

To better protect the security of the identity token, the Identity
Provider may wish to bind the identity token to the TLS connection

between the client and the Relying Party, thus ensuring that only
said client can use the identity token: The Relying Party will
compare the Token Binding ID in the identity token with the Token
Binding ID of the TLS connection between it an the client.

This is an example of a federation scenario, which more generally can
be described as follows:

o A Token Consumer causes the client to issue a token request to the
   Token Provider. The goal is for the client to obtain a token and
   then use it with the Token Consumer.

o The client delivers the token request to the Token Provider.

o The Token Provider issues the token. The token is issued for the
   specific Token Consumer who requested it (thus preventing
   malicious Token Consumers from using tokens with other Token
   Consumers). The token is, however, typically a bearer token,
   meaning that any client can use it with the Token Consumer, not
   just the client to which it was issued.

o Therefore, in the previous step, the Token Provider may want to
   include the Token Binding ID of the TLS connection between the
   client and the Token Consumer in the token.

o That Token Binding ID must therefore be communicated to the Token
   Provider along with the token request. Communicating a Token
   Binding ID involves proving possession of a private key and is
   described in the Token Binding Protocol [DraftPopov].

The client will perform this last operation (proving possession of a
private key that corresponds to a Token Binding ID between the client
and the Token Consumer while delivering the token request to the
Token Provider) only if the Token Consumer permits the client to do
so.

Below, we will enumerate a number of mechanisms available to Token
Consumers to grant this permission.

3.3. HTTP Redirects

When a Token Consumer redirects the client to a Token Provider as a
means to deliver the token request, it SHOULD include the following
HTTP response header in its HTTP response:

 Include-Referer-Token-Binding-ID: true

Including this response header signals to the client that it should reveal the Token Binding ID used between the client and the Token Consumer to the Token Provider. In the absence of this response header, the client will not disclose any information about the Token Binding used between the client and the Token Consumer to the Token Provider.

This header has only meaning if the HTTP status code is 302 or 301, and MUST be ignored by the client for any other status codes. If the client supports the Token Binding Protocol, and has negotiated the Token Binding Protocol with both the Token Consumer and the Token Provider, it already sends the following header to the Token Provider with each HTTP request (see above):

 Token-Binding: EncodedTokenBindingMessage

The TokenBindingMessage SHOULD contain a TokenBinding with TokenBindingType referred_token_binding. If included, this TokenBinding MUST be signed with the Token Binding key used by the client for connections between itself and the Token Consumer (more specifically, the web origin that issued the Include-Referer-Token-Binding-ID response header). The Token Binding ID established by this TokenBinding is called a _Referred Token Binding ID_.

As described above, the TokenBindingMessage MUST additionally contain a Provided Token Binding ID, i.e., a TokenBinding structure with TokenBindingType provided_token_binding, which MUST be signed with the Token Binding key used by the client for connections between itself and the Token Privider (more specifically, the web origin that the token request sent to).

3.4. Cross-Origin Resource Sharing

When issuing an XML HTTP request across origins to a Token Provider, a Token Consumer can reveal its Token Binding ID through the withRefererTokenBindingID property of the XmlHttpRequest object. Example:

 var xhr = new XMLHttpRequest();
 xhr.withCredentials = true; // send cookies
 xhr.withRefererTokenBindingID = true;
 xhr.open(method, url, true);

The client SHOULD include the Token-Binding: header to the outgoing request as described above if:

o the withRefererTokenBindingID property of the XmlHttpRequest
   object is set to true, and

o the client has negotiated the Token Binding Protocol both with the
   web origin that issued the XmlHttpRequest, and the web origin to
   which the XmlHttpRequest is addressed.

3.5. Negotiated Key Parameters

The Token Binding Protocol [DraftPopov] allows the server and client
to negotiate a signature algorithm used in the TokenBindingMessage.
It is possible that the Token Binding ID used between the client and
the Token Consumer, and the Token Binding ID used between the client
and Token Provider, use different signature algorithms. The client
MUST use the signature algorithm negotiated with the Token Consumer
in the referred_token_binding TokenBinding of the
TokenBindingMessage, even if that signature algorithm is different
from the one negotiated with the origin that the header is sent to.

Token Providers SHOULD support all the SignatureAndHashAlgorithms
specified in the Token Binding Protocol [DraftPopov]. If a token
provider does not support the SignatureAndHashAlgorithm specified in
the referred_token_binding TokenBinding in the TokenBindingMessage,
it MUST issue an unbound token.

4. Security Considerations

4.1. Security Token Replay

The goal of the Federated Token Binding mechanisms is to prevent
attackers from exporting and replaying tokens used in protocols
between the client and Token Consumer, thereby impersonating
legitimate users and gaining access to protected resources. Bound
tokens can still be replayed by malware present in the client. In
order to export the token to another machine and successfully replay
it, the attacker also needs to export the corresponding private key.
The Token Binding private key is therefore a high-value asset and
MUST be strongly protected, ideally by generating it in a hardware
security module that prevents key export.

4.2. Privacy Considerations

The Token Binding protocol uses persistent, long-lived TLS Token
Binding IDs. To protect privacy, TLS Token Binding IDs are never
transmitted in clear text and can be reset by the user at any time,
e.g. when clearing browser cookies. Unique Token Binding IDs MUST be
generated for connections to different origins, so they cannot be
used by cooperating servers to link user identities.

4.3. Triple Handshake Vulnerability in TLS

   The Token Binding protocol relies on the tls_unique value to
   associate a TLS connection with a TLS Token Binding. The triple
   handshake attack [TRIPLE-HS] is a known TLS protocol vulnerability
   allowing the attacker to synchronize tls_unique values between TLS
   connections. The attacker can then successfully replay bound tokens.
   For this reason, the Token Binding protocol MUST NOT be negotiated
   unless the Extended Master Secret TLS extension
   [I-D.ietf-tls-session-hash] has also been negotiated.

5. References

5.1. Normative References

   [DraftPopov]
            Popov, A., "The Token Binding Protocol Version 1.0", 2014.

   [I-D.ietf-httpbis-header-compression]
            Peon, R. and H. Ruellan, "HPACK - Header Compression for
            HTTP/2", draft-ietf-httpbis-header-compression-09 (work in
            progress), July 2014.

   [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
            Requirement Levels", BCP 14, RFC 2119, March 1997.

   [RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H.,
            Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext
            Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

   [RFC4492] Blake-Wilson, S., Bolyard, N., Gupta, V., Hawk, C., and B.
            Moeller, "Elliptic Curve Cryptography (ECC) Cipher Suites
            for Transport Layer Security (TLS)", RFC 4492, May 2006.

   [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an
            IANA Considerations Section in RFCs", BCP 26, RFC 5226,
            May 2008.

   [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security
            (TLS) Protocol Version 1.2", RFC 5246, August 2008.

   [RFC5929] Altman, J., Williams, N., and L. Zhu, "Channel Bindings
            for TLS", RFC 5929, July 2010.

   [RFC7301] Friedl, S., Popov, A., Langley, A., and E. Stephan,
            "Transport Layer Security (TLS) Application-Layer Protocol
            Negotiation Extension", RFC 7301, July 2014.

5.2. Informative References

   [I-D.ietf-httpbis-http2]
             Belshe, M., Peon, R., and M. Thomson, "Hypertext Transfer
             Protocol version 2", draft-ietf-httpbis-http2-14 (work in
             progress), July 2014.

   [I-D.ietf-tls-session-hash]
             Bhargavan, K., Delignat-Lavaud, A., Pironti, A., Langley,
             A., and M. Ray, "Transport Layer Security (TLS) Session
             Hash and Extended Master Secret Extension", draft-ietf-
             tls-session-hash-02 (work in progress), October 2014.

   [TRIPLE-HS]
             Bhargavan, K., Delignat-Lavaud, A., Fournet, C., Pironti,
             A., and P. Strub, "Triple Handshakes and Cookie Cutters:
             Breaking and Fixing Authentication over TLS. IEEE
             Symposium on Security and Privacy", 2014.

Authors' Addresses

   Andrei Popov
   Microsoft Corp.
   USA

   Email: andreipo@microsoft.com

   Magnus Nystroem
   Microsoft Corp.
   USA

   Email: mnystrom@microsoft.com

   Dirk Balfanz (editor)
   Google Inc.
   USA

   Email: balfanz@google.com

   Adam Langley
   Google Inc.
   USA

   Email: agl@google.com