TOKEN BINDING OVER HTTP [RFC 8473]

ABSTRACT

Token Binding is a protocol that allows client/server applications to create long-lived, uniquely identifiable TLS bindings spanning multiple TLS sessions and connections.

RFC 8473 "Token Binding over HTTP" describes how to allow HTTP servers to cryptographically bind security tokens (like cookies and OAuth tokens) to TLS connections in both first-party and federated scenarios, preventing their use by unauthorized parties who may come into possession of those tokens.

SECURITY TOKENS AND BEARER TOKENS

One of the most common methods for protecting access to a certain resource is issuing security tokens to authorized users (possibly after authentication) that will present it to the server responsible for the resource to access it.

They can take the name "bearer tokens" because just being in possession of the token allows access to the associated resources, meaning that anyone who holds the token can use it. This also means that if a token is exfiltrated or stolen in any way, they can be used by an attacker without any other protection in place.

There have been attempts to prevent token exfiltration, for example for cookies

- the `secure` flag prevents them from being sent over an insecure connection
- the `httponly` flag prevents exfiltration via cross-site scripting

Token binding aims to prevent the use of the tokens even if they happen to be stolen.

TOKENS USE CASES

Security tokens are mainly used in two different settings. In a single server-client relationship with a single TLS session, then you are in the “First-Party” use case (as defined in RFC 8473). In that setting, Token Bindings are used to protect security tokens issued by the server, like cookies, while they are on the client. Since the cookie, for example, is bound to the TLS session, a stolen cookie cannot be used.

On the other hand, you have “Federation” use cases (as defined in RFC 8473) which is a bit more complex. In this setting, you have three systems:

- the client (sometimes called user-agent),
- the Token Issuer (also called identity provider, or authorization server),
- the Token Consumer (also sometimes called web application, relying party or client).

As its name implies, the Token Issuer is the one that will issue tokens to be consumed by the Token Consumer. These tokens can either be authentication tokens, to prove users' identity to the Token Consumer, or authorization tokens, to allow the client to use some resources of the Token Consumer. In this case, the Token Binding is more complex since the token will be issued on another TLS connection than the one on which it will be used.

TOKEN BINDING OVERVIEW

The main idea behind Token Binding is that it enables a long-lived binding of cookies or other security tokens to a client generated private-public key pair.

- The use of Token Binding is negotiated during the TLS handshake via the Token Binding TLS extension
  - Token Binding can also be performed during TLS Renegotiation
  - TLS renegotiation must only occur between a client request and a server response
- Possession of the private key is proven by signing the Exported Key Material [RFC 5705] of the current TLS connection with the server and sending this data in an HTTP header with every request
- Cookies and tokens can be bound to the key, thus preventing them from being used by anyone who doesn't possess the private key
- The key pair is scoped to the level below the effective Top-Level Domain (eTLD) (for example, example.ch or example.co.uk are valid but co.uk is not), i.e. the client uses different keys for different sites.
- Keys are long-lived and span multiple TLS connections

THE TOKEN BINDING PROTOCOL [RFC 8471]

Before diving into how Token Binding is used over http connection, it can be useful to look at how the Token Binding messages are defined in RFC 8471.
Token messages can contain more than one Token Binding, with each containing:

- the Token Binding type, which can either be
  - provided, used to establish a Token Binding when connecting to the server
  - referred, used when requesting tokens that are intended to be presented to a different server
- the Token Binding ID, which describes the public key
- the signature over
  - the Token Binding type
  - the Token Binding ID
  - the EKM of the TLS session with the server
- some not yet defined extensions.

Note that Token Binding messages may not be serialized in any human-readable format.

THE SEC-TOKEN-BINDING HTTP REQUEST HEADER FIELD

If a client and a server have negotiated the Token Binding Protocol, for each HTTP request the client needs to include exactly one `Sec-Token-Binding` header field, with its single value being the base64 encoding of the Token Binding Message with all trailing padding characters (i.e. "=") omitted.

- The Token Binding Message must contain exactly one Token Binding of type provided which must be signed with the key
- If the validation of the signature fails and the Token Binding is rejected, any associated tokens must be rejected
- If the HTTP request contains invalid tokens the request must be rejected (for example with error 400 Bad Request)

The Sec prefix should impede the header from being used more than once in a single request and prevents cross-origin modification, safeguarding Token Binding integrity.

The scope of the Token Binding Key pair on HTTPS must be no wider than the domain that was created for, which in practice means that the scope of the key pair is no larger than the scope of a cookie for that domain

In turn the browser may narrow the scope of the cookies as well as of the key pair

TOKEN BINDING USE CASES

In a First-Party scenario, the Token Binding protocol is mainly used to secure authentication cookies. On the first connection, the client negotiates the usage and the parameter of the Token binding protocol at the same time as the TLS handshake. When the user authenticates to the server, the issued token will be bound to the Token Binding ID. Since this ID is persistent, the client can just provide the token in a new TLS connection to be authenticated.

In a federated scenario, since clients use different Token Bindings key pairs with different servers, one server cannot issue a security token bound to a TLS connection between the client and a different server.

The approach taken in RFC 8473 is based on redirect interaction between Token Consumers and Token Providers:

0. The client issues a request to the Token Consumer to access a protected resource
1. The Token Consumer replies with a redirect code pointing the client towards the Token Provider
    a. The redirect message includes a header field Include-Referred-Token-Binding-ID set to true
2. The Client establishes a TLS connection with the Token Provider, negotiating the Token Binding protocol keys
    a. The client authenticates itself if it was not authenticated already
    b. The client sends a GET or a POST request including in the Sec-Token-Binding header field:
        i. The Provided Token Binding (used with the Token Provider)
        ii. The Referred Token Binding (used with the Token Consumer)
3. The Token Provider issues a security token bound to the Token Binding ID of the referred Token Binding

TECHNICALITIES

Below are reported some technical information about the use of Token Binding in the federated scenario:

- The referred Token Binding sent to the Token Provider include the signature of the EKM of the connection between the Client and the Token Provider using the key established with the Token Consumer
- The referred Token Binding is included iff the header field is present and set to true
- The Client includes a referred Token Binding even if both Token Provider and Token Consumer are under the same level below eTLD
- The referred Token Binding is sent only in the initial request
    - Any further redirects do not include the original referred Token Binding (others may be included)
- The client must ignore the header field if
    - Is received with a response to a request that did not include a Sec-Token-Binding field
    - the response status code from the Token Consumer is not a redirect code (300-399)

SECURITY CONSIDERATIONS

The goal of the Token Binding protocol is to prevent attackers from exporting and replaying security tokens and from thereby impersonating legitimate users and gaining access to protected resources. Bound tokens can still be replayed by malware present in User Agents.

To export a bound security token, the attacker would also need to export the private key, which should be intrinsically more difficult than exporting the token itself. But if the attacker comes into possession of both a bound token and the private key of a client associated with that token, he could establish a TLS connection to the Token Consumer, negotiate the Token Binding protocol using the clients key and then successfully access client resources by presenting the bound token

If the attacker has knowledge of the EKM could craft a Sec-Token-Binding header with his own key. If he manages to trick the user into sending his header with his key, it will appear to the server as if the client controls a certain key when in fact he does not.
Furthermore, the attacker has a pre-existing relationship with the server, he can trick the user to also send a cookie that belongs to the attacker, making the client log in as the attacker on the server, which could lead to loss of privacy if the server allows for activity tracking

Security Considerations under the Federated Scenario

Consider the following assumptions:

- The client has an authentication token with the Token Provider bound to its connection
- The client wants to establish a secure authenticated session with the Token Consumer
- A man-in-the-middle is allowed to intercept on one or both connection of the client

Consider now a MITM between the Client and the Token Provider. The attacker can either:

1. Leave the Sec-Token-Binding field unchanged
    a. But the token Provider will not be able to verify the signature on the EKM
2. Change the Sec-Token-Binding header field with one that matches his connection with the provider
    a. But then the Token Binding ID in the token will not match the one in the header (keep in mind the token is integrity protected)

Consider now a MITM between the Client and the Token Provider.

1. The goal of the man-in-the-middle is to trick the Token Provider into issuing a token bound to its Token Binding ID and not to the Token Binding ID of the legitimate client.
2. Since the attacker controls the redirect URL and can tamper with any redirect URL issued by the Token Consumer, the client's Referred Token Binding ID must be communicated to the Token Provider in a manner that cannot be affected by the man-in-the-middle
3. But the Token Binding ID with the Token Consumer is not included in any application-level redirect message, hence the attacker cannot modify it

Privacy Considerations

Tokens Bindings IDs are long-lived, since they do not have an expiration time, and unique to each client. This is the definition of a tracker. The client (mostly browsers) designers should be aware of this and allow the user to remove their Token Bindings IDs with the same agility as they do for cookies. In addition, they should also discard them when they automatically discard cookies, for example after a private browsing session.

When used with an HTTP cookie, the Bindings must also have the same scope as the cookie. If the scope of the binding is too small, then the missing servers should reject the cookie. On the other hand, with a larger scope, then the Token Binding ID might be used to track a user across multiple applications (which all share the same domain below an eTLD).

Conclusions

Even if it was standardized in 2018, only the Microsoft ecosystem currently supports Token Binding. Firefox still has it as a pending bug[1], while Chrome decided to remove this feature[2] which was never enabled by default. It still is supported by Google and Microsoft and has been mentioned as one of the important security features for FIDO[3].

[1] https://bugzilla.mozilla.org/show_bug.cgi?id=1363987 (Last visited on the 09.03.2025)
[2] https://chromestatus.com/feature/5097603234529280 (Last visited on 09.03.2025)
[3] https://fidoalliance.org/fido-technote-the-growing-role-of-token-binding/ (Last visited on 09.03.2025)