Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: July 11, 2016

A. Popov, Ed.
M. Nystroem
Microsoft Corp.
D. Balfanz
A. Langley
Google Inc.

January 8, 2016

The Token Binding Protocol Version 1.0
draft-ietf-tokbind-protocol-04

Abstract

This document specifies Version 1.0 of the Token Binding protocol. The Token Binding protocol allows client/server applications to create long-lived, uniquely identifiable TLS [RFC5246] bindings spanning multiple TLS sessions and connections. Applications are then enabled to cryptographically bind security tokens to the TLS layer, preventing token export and replay attacks. To protect privacy, the TLS Token Binding identifiers are only transmitted encrypted and can be reset by the user at any time.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction 2
1.1. Requirements Language 3
2. Token Binding Protocol Overview 3
3. Token Binding Protocol Message 4
4. Establishing a TLS Token Binding 7
5. TLS Token Binding ID Format 7
6. Security Token Validation 8
7. IANA Considerations 8
8. Security Considerations 10
8.1. Security Token Replay 10
8.2. Downgrade Attacks 10
8.3. Privacy Considerations 10
8.4. Token Binding Key Sharing Between Applications 11
8.5. Triple Handshake Vulnerability in TLS 1.2 and Older TLS Versions 11
9. Acknowledgements 11
10. References 11
10.1. Normative References 11
10.2. Informative References 13
Authors' Addresses 13

1. Introduction

Servers generate various security tokens (e.g. HTTP cookies, OAuth tokens) for applications to access protected resources. Any party in possession of such token gains access to the protected resource. Attackers export bearer tokens from the user's machine, present them to the servers, and impersonate authenticated users. The idea of Token Binding is to prevent such attacks by cryptographically binding security tokens to the TLS layer.

A TLS Token Binding is established by the user agent generating a private-public key pair (possibly within a secure hardware module, such as TPM) per target server, and proving possession of the private key on every TLS connection to the target server. The proof of possession involves signing the exported keying material [RFC5705] for the TLS connection with the private key. The corresponding public key is included in the TLS Token Binding identifier structure (described in the "TLS Token Binding ID Format" section of this document). TLS Token Bindings are long-lived, i.e. they encompass multiple TLS connections and TLS sessions between a given client and server. 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.

When issuing a security token to a client that supports TLS Token Binding, a server includes the client's TLS Token Binding ID in the token. Later on, when a client presents a security token containing a TLS Token Binding ID, the server makes sure the ID in the token matches the ID of the TLS Token Binding established with the client. In the case of a mismatch, the server discards the token.

In order to successfully export and replay a bound security token, the attacker needs to also be able to export the client's private key, which is hard to do in the case of the key generated in a secure hardware module.

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. Token Binding Protocol Overview

The client and server use the Token Binding Negotiation TLS Extension [I-D.ietf-tokbind-negotiation] to negotiate the Token Binding protocol version and the parameters (signature algorithm, length) of the Token Binding key. This negotiation does not require additional round-trips.

The Token Binding protocol consists of one message sent by the client to the server, proving possession of one or more client-generated asymmetric keys. This message is only sent if the client and server agree on the use of the Token Binding protocol and the key parameters. The Token Binding message is sent with the application protocol data in TLS application_data records.

A server receiving the Token Binding message verifies that the key parameters in the message match the Token Binding parameters negotiated via [I-D.ietf-tokbind-negotiation], and then validates the signatures contained in the Token Binding message. If either of these checks fails, the server terminates the connection, otherwise the TLS Token Binding is successfully established with the ID contained in the Token Binding message.

When a server supporting the Token Binding protocol receives a bound token, the server compares the TLS Token Binding ID in the security token with the TLS Token Binding ID established with the client. If the bound token came from a TLS connection without a Token Binding, or if the IDs don’t match, the token is discarded.

This document defines the format of the Token Binding protocol message, the process of establishing a TLS Token Binding, the format of the Token Binding ID, and the process of validating a security token. Token Binding Negotiation TLS Extension [I-D.ietf-tokbind-negotiation] describes the negotiation of the Token Binding protocol and key parameters. Token Binding over HTTP [I-D.ietf-tokbind-https] explains how the Token Binding message is encapsulated within HTTP/1.1 [RFC7230] or HTTP/2 [RFC7540] messages. [I-D.ietf-tokbind-https] also describes Token Binding between multiple communicating parties: User Agent, Identity Provider and Relying Party.

3. Token Binding Protocol Message

The Token Binding message is sent by the client and proves possession of one or more private keys held by the client. This message MUST be sent if the client and server successfully negotiated the use of the Token Binding protocol via [I-D.ietf-tokbind-negotiation], and MUST NOT be sent otherwise. This message MUST be sent in the client’s first application protocol message. This message MAY also be sent in subsequent application protocol messages, proving possession of other keys by the same client, to facilitate token binding between more than two communicating parties. Token Binding over HTTP [I-D.ietf-tokbind-https] specifies the encapsulation of the Token Binding message in the application protocol messages, and the scenarios involving more than two communicating parties. The Token Binding message format is defined using TLS specification language:

```
enum {
    rsa2048_pkcs1.5(0), rsa2048_pss(1), ecdsap256(2), (255)
} TokenBindingKeyParameters;

struct {
    opaque modulus<1..2^16-1>;
    opaque publicexponent<1..2^8-1>;
} RSAPublicKey;

struct {
    opaque point <1..2^8-1>;
} ECPoint;

enum {
    provided_token_binding(0), referred_token_binding(1), (255)
} TokenBindingType;

struct {
    TokenBindingKeyParameters key_parameters;
    select (key_parameters) {
        case rsa2048_pkcs1.5:
        case rsa2048_pss:
            RSAPublicKey rsapubkey;
        case ecdsap256:
            ECPoint point;
    }
} TokenBindingID;

enum {
    (255)                      // No initial ExtensionType registrations
} ExtensionType;

struct {
    ExtensionType extension_type;
    opaque extension_data<0..2^16-1>;
} Extension;

struct {
    TokenBindingType tokenbinding_type;
    TokenBindingID tokenbindingid;
    opaque signature<0..2^16-1>;// Signature over the exported keying material v
alue
    Extension extensions<0..2^16-1>;
} TokenBinding;

struct {
    TokenBinding tokenbindings<0..2^16-1>;
} TokenBindingMessage;
```

The Token Binding message consists of a series of TokenBinding structures containing the type of the token binding, the TokenBindingID, a signature over the exported keying material (EKM) value, optionally followed by Extension structures.

This document defines two token binding types: provided_token_binding used to establish a Token Binding when connecting to a server, and referred_token_binding used when requesting tokens to be presented to a different server. Token Binding over HTTP [I-D.ietf-tokbind-https] describes Token Binding between multiple communicating parties: User Agent, Identity Provider and Relying Party.

When an rsa2048_pkcs1.5 or rsa2048_pss key is used, TokenBinding.signature contains the signature generated using, respectively, the RSASSA-PKCS1-v1_5 or RSASSA-PSS signature scheme defined in [RFC3447]. RSAPublicKey.modulus and RSAPublicKey.publicexponent contain the length-prefixed modulus and exponent of the RSA public key represented in big-endian format.

When an ecdsap256 key is used, TokenBinding.signature contains a pair of integers, R followed by S, as defined in [ANSI.X9-62.2005]. R and S are encoded in big-endian format. ECPoint.point contains the X coordinate followed by the Y coordinate. The X and Y coordinates are unsigned integers encoded in big-endian format. Future specifications may define Token Binding keys using other elliptic curves with their corresponding signature and point formats.

The EKM is obtained using the Keying Material Exporters for TLS defined in [RFC5705], by supplying the following input values:

- o Label: The ASCII string "EXPORTER-Token-Binding" with no terminating NUL.

- o Context value: NULL (no application context supplied).

- o Length: 32 bytes.

An implementation MUST ignore any unknown extensions. Initially, no extension types are defined. One of the possible uses of extensions envisioned at the time of this writing is attestation: cryptographic proof that allows the server to verify that the Token Binding key is hardware-bound. The definitions of such Token Binding protocol extensions are outside the scope of this specification.

At least one TokenBinding MUST be included in the Token Binding message. The signature algorithm and key length used in the TokenBinding MUST match the parameters negotiated via [I-D.ietf-tokbind-negotiation]. The client SHOULD generate and store

Token Binding keys in a secure manner that prevents key export. In order to prevent cooperating servers from linking user identities, different keys SHOULD be used by the client for connections to different servers, according to the token scoping rules of the application protocol.

4. Establishing a TLS Token Binding

The triple handshake vulnerability in TLS 1.2 and older TLS versions affects the security of the Token Binding protocol, as described in the "Security Considerations" section below. Therefore, the server MUST NOT negotiate the use of the Token Binding protocol with these TLS versions, unless the server also negotiates Extended Master Secret [RFC7627] and Renegotiation Indication [RFC5746] TLS extensions.

The server MUST terminate the connection if the use of the Token Binding protocol was not negotiated, but the client sends the Token Binding message. If the Token Binding type is "provided_token_binding", the server MUST verify that the signature algorithm (including elliptic curve in the case of ECDSA) and key length in the Token Binding message match those negotiated via [I-D.ietf-tokbind-negotiation]. In the case of a mismatch, the server MUST terminate the connection. As described in [I-D.ietf-tokbind-https], Token Bindings of type "referred_token_binding" may have different key parameters than those negotiated via [I-D.ietf-tokbind-negotiation].

If the Token Binding message does not contain at least one TokenBinding structure, or the signature contained in a TokenBinding structure is invalid, the server MUST terminate the connection. Otherwise, the TLS Token Binding is successfully established and its ID can be provided to the application for security token validation.

5. TLS Token Binding ID Format

The ID of the TLS Token Binding established as a result of Token Binding message processing is a binary representation of the following structure:

```
struct {
    TokenBindingKeyParameters key_parameters;
    select (key_parameters) {
        case rsa2048_pkcs1.5:
        case rsa2048_pss:
            RSAPublicKey rsapubkey;
        case ecdsap256:
            ECPoint point;
    }
} TokenBindingID;
```

TokenBindingID contains the key parameters negotiated via [I-D.ietf-tokbind-negotiation]. TLS Token Binding ID can be obtained from the TokenBinding structure described in the "Token Binding Protocol Message" section of this document by discarding the token binding type, signature and extensions. TLS Token Binding ID will be available at the application layer and used by the server to generate and verify bound tokens.

6. Security Token Validation

Security tokens can be bound to the TLS layer either by embedding the Token Binding ID in the token, or by maintaining a database mapping tokens to Token Binding IDs. The specific method of generating bound security tokens is application-defined and beyond the scope of this document.

Upon receipt of a security token, the server attempts to retrieve TLS Token Binding ID information from the token and from the TLS connection with the client. Application-provided policy determines whether to honor non-bound (bearer) tokens. If the token is bound and a TLS Token Binding has not been established for the client connection, the server MUST discard the token. If the TLS Token Binding ID for the token does not match the TLS Token Binding ID established for the client connection, the server MUST discard the token.

7. IANA Considerations

This document establishes a registry for Token Binding type identifiers entitled "Token Binding Types" under the "Token Binding Protocol" heading.

Entries in this registry require the following fields:

- o Value: The octet value that identifies the Token Binding type (0-255).

- o Description: The description of the Token Binding type.

- o Specification: A reference to a specification that defines the Token Binding type.

This registry operates under the "Expert Review" policy as defined in [RFC5226]. The designated expert is advised to encourage the inclusion of a reference to a permanent and readily available specification that enables the creation of interoperable implementations using the identified Token Binding type.

An initial set of registrations for this registry follows:

Value: 0

Description: provided_token_binding

Specification: this document

Value: 1

Description: referred_token_binding

Specification: this document

This document establishes a registry for Token Binding extensions entitled "Token Binding Extensions" under the "Token Binding Protocol" heading.

Entries in this registry require the following fields:

- o Value: The octet value that identifies the Token Binding extension (0-255).

- o Description: The description of the Token Binding extension.

- o Specification: A reference to a specification that defines the Token Binding extension.

This registry operates under the "Expert Review" policy as defined in [RFC5226]. The designated expert is advised to encourage the inclusion of a reference to a permanent and readily available specification that enables the creation of interoperable implementations using the identified Token Binding extension. This document creates no initial registrations in the "Token Binding Extensions" registry.

This document uses "Token Binding Key Parameters" registry originally created in [I-D.ietf-tokbind-negotiation]. This document creates no new registrations in this registry.

8. Security Considerations

8.1. Security Token Replay

The goal of the Token Binding protocol is to prevent attackers from exporting and replaying security tokens, thereby impersonating legitimate users and gaining access to protected resources. Bound tokens can still be replayed by the malware present in the User Agent. In order to export the token to another machine and successfully replay it, the attacker also needs to export the corresponding private key. Token Binding private keys are therefore high-value assets and SHOULD be strongly protected, ideally by generating them in a hardware security module that prevents key export.

8.2. Downgrade Attacks

The Token Binding protocol is only used when negotiated via [I-D.ietf-tokbind-negotiation] within the TLS handshake. TLS prevents active attackers from modifying the messages of the TLS handshake, therefore it is not possible for the attacker to remove or modify the Token Binding Negotiation TLS Extension used to negotiate the Token Binding protocol and key parameters. The signature algorithm and key length used in the TokenBinding of type "provided_token_binding" MUST match the parameters negotiated via [I-D.ietf-tokbind-negotiation].

8.3. 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. Some applications offer a special privacy mode where they don’t store or use tokens supplied by the server, e.g. "in private" browsing. When operating in this special privacy mode, applications SHOULD use newly generated Token Binding keys and delete them when exiting this mode, or else SHOULD NOT negotiate Token Binding at all.

In order to prevent cooperating servers from linking user identities, different keys SHOULD be used by the client for connections to different servers, according to the token scoping rules of the application protocol.

A server can use tokens and Token Binding IDs to track clients. Client applications that automatically limit the lifetime of tokens to maintain user privacy SHOULD apply the same validity time limits to Token Binding keys.

8.4. Token Binding Key Sharing Between Applications

Existing systems provide a variety of platform-specific mechanisms for certain applications to share tokens, e.g. to enable single sign-on scenarios. For these scenarios to keep working with bound tokens, the applications that are allowed to share tokens will need to also share Token Binding keys. Care must be taken to restrict the sharing of Token Binding keys to the same group(s) of applications that share the same tokens.

8.5. Triple Handshake Vulnerability in TLS 1.2 and Older TLS Versions

The Token Binding protocol relies on the exported keying material (EKM) to associate a TLS connection with a Token Binding. The triple handshake attack [TRIPLE-HS] is a known vulnerability in TLS 1.2 and older TLS versions, allowing the attacker to synchronize keying material between TLS connections. The attacker can then successfully replay bound tokens. For this reason, the Token Binding protocol MUST NOT be negotiated with these TLS versions, unless the Extended Master Secret [RFC7627] and Renegotiation Indication [RFC5746] TLS extensions have also been negotiated.

9. Acknowledgements

This document incorporates comments and suggestions offered by Eric Rescorla, Gabriel Montenegro, Martin Thomson, Vinod Anupam, Bill Cox, Nick Harper and others.

10. References

10.1. Normative References

[ANSI.X9-62.2005]
American National Standards Institute, "Public Key Cryptography for the Financial Services Industry, The Elliptic Curve Digital Signature Algorithm (ECDSA)", ANSI X9.62, 2005.

[I-D.ietf-tokbind-https]
Popov, A., Nystrom, M., Balfanz, D., and A. Langley, "Token Binding over HTTP", draft-ietf-tokbind-https-02 (work in progress), October 2015.

[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-01 (work in progress), October 2015.

[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>.

[RFC3447] Jonsson, J. and B. Kaliski, "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1", RFC 3447, DOI 10.17487/RFC3447, February 2003, <http://www.rfc-editor.org/info/rfc3447>.

[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, DOI 10.17487/RFC4492, May 2006, <http://www.rfc-editor.org/info/rfc4492>.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, DOI 10.17487/RFC5226, May 2008, <http://www.rfc-editor.org/info/rfc5226>.

[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>.

[RFC5746] Rescorla, E., Ray, M., Dispensa, S., and N. Oskov, "Transport Layer Security (TLS) Renegotiation Indication Extension", RFC 5746, DOI 10.17487/RFC5746, February 2010, <http://www.rfc-editor.org/info/rfc5746>.

[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>.

[RFC7540] Belshe, M., Peon, R., and M. Thomson, Ed., "Hypertext Transfer Protocol Version 2 (HTTP/2)", RFC 7540, DOI 10.17487/RFC7540, May 2015, <http://www.rfc-editor.org/info/rfc7540>.

[RFC7627] Bhargavan, K., Ed., Delignat-Lavaud, A., Pironti, A., Langley, A., and M. Ray, "Transport Layer Security (TLS) Session Hash and Extended Master Secret Extension", RFC 7627, DOI 10.17487/RFC7627, September 2015, <http://www.rfc-editor.org/info/rfc7627>.

10.2. Informative References

[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 (editor)
Microsoft Corp.
USA

Email: andreipo@microsoft.com

Magnus Nystroem
Microsoft Corp.
USA

Email: mnystrom@microsoft.com

Dirk Balfanz
Google Inc.
USA

Email: balfanz@google.com

Adam Langley
Google Inc.
USA

Email: agl@google.com