# Changes in the Core Token Binding I-Ds Since IETF 96

Andrei Popov, Microsoft Corp.

# TBNEGO Changes

- Added information about the temporary IANA code point registration for the token_binding TLS extension (24).
- Clarified that Renegotiation Indication TLS extension is only required if the client or server is configured to initiate or allow renegotiation.

# TBPROTO Changes

- Added TokenBindingID.key_length field so that implementations can skip Token Binding IDs with unknown key parameters:

```
struct {
    TokenBindingKeyParameters key_parameters;
    uint16 key_length;  /* Length (in bytes) of the following
                           TokenBindingID.TokenBindingPublicKey */
    select (key_parameters) {
      case rsa2048_pkcs1.5:
      case rsa2048_pss:
        RSAPublicKey rsapubkey;
      case ecdsap256:
        ECPoint point;
    } TokenBindingPublicKey;
  } TokenBindingID;
```

- Added context information to the TokenBinding.signature:

```
opaque signature<0..2^16-1>;  /* Signature over the concatenation of
                                 tokenbinding_type, key_parameters and
                                 exported keying material (EKM) */
```

# TBPROTO Changes

- Added a recommendation that Token Binding protocol implementations SHOULD make Token Binding IDs available to the application as opaque byte sequences.
- Specified that the RSASSA-PKCS1-v1_5 and ECDSA signature schemes use SHA256 hash and the RSASSA-PSS signature scheme uses MGF1 with SHA256 and 32 bytes of salt.
- Various editorial changes.

# HTTPSTB Changes

- Clarified Sec-Token-Binding header encoding:

The header field name is "Sec-Token-Binding" and its value is a base64url encoding of the TokenBindingMessage defined in [I-D.ietf-tokbind-protocol] using the URL- and filename-safe character set 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.

- Explicitly stated that federation can be done within the same eTLD+1.

When a client receives the Include-Referred-Token-Binding-ID header, it includes the referred token binding even if both the Token Provider and the Token Consumer fall under the same eTLD+1 and the provided and referred token binding IDs are the same.

# HTTPSTB Changes

- Added implementation considerations for multi-party use cases other than HTTP redirect and applications other than Web browsers (e.g. “native apps”).
- A new privacy considerations section discussing the potential for correlation based on Token Binding IDs.
- Various editorial changes.

# Implementation Status

- Chrome 55.0.2868.0 and later support TB10 (beta, dev, and canary).

In Chrome, TB needs to be explicitly enabled, either at chrome://flags/#enable-token-binding or on the command line with --enable-features=token-binding.

- IE, Edge and IIS support TB10 (next preview of Windows 10).

IE and Edge have TB enabled by default; IIS requires configuration in the registry:

Key: HKLM\System\CurrentControlSet\Services\Http\Parameters
Value: EnableSslTokenBinding REG_DWORD 1

# Open Issue: Renegotiation and EKM

- TBPROTO-10 section 3.3 says that the TokenBinding.signature is computed over the Exported Keying Material (EKM) value obtained from the current TLS connection (among other things).
- TLS renegotiation means that multiple TLS sessions may be established during the lifetime of one TLS connection, each with its own master secret and EKM value.
- Should the TokenBinding contain the signature over the current/latest EKM, or the original/stale EKM of the first TLS session established on the connection?

# If We Sign the Current/Latest EKM

1. The client will prove possession of the TB key in each TLS session and connection.
2. A bound token cannot be replayed on a new TLS session, even if this new session has been established on an existing TLS connection (or something that looks like an existing TLS connection to the server).
3. Server application can be stateless/request-oriented, does not need to track connections.
4. The client and server need to ensure that the correct EKM is used when verifying a TB message. The TLS stack knows which master secret and EKM correspond to each application_data record, but API changes may be required to convey this information to the application (depending on the design).

# If We Sign the Original/Stale EKM

1. An application can query the EKM value when a connection is established, cache it and use for the lifetime of the connection to generate or validate Token Bindings. This may be easier to implement (depending on the design).
2. Fewer signatures to perform and validate ➔ some CPU savings.
3. A bound token can be replayed in a renegotiated TLS session, but maybe this is not a security issue?
4. The server application needs to track TLS connections, and cannot be purely stateless/request-oriented.

# Links And Contact Information

- TLS Extension for Token Binding Negotiation: https://datatracker.ietf.org/doc/draft-ietf-tokbind-negotiation/
- The Token Binding Protocol Version 1.0: https://datatracker.ietf.org/doc/draft-ietf-tokbind-protocol/
- Token Binding over HTTP: https://datatracker.ietf.org/doc/draft-ietf-tokbind-https/
- GitHub: https://github.com/TokenBinding/Internet-Drafts


- Dirk Balfanz balfanz@google.com
- Andrei Popov andreipo@microsoft.com
- Jeff Hodges Jeff.Hodges@paypal.com

# The Token Binding Protocol Message Format

```
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 concatenation of
                                     tokenbinding_type, key_parameters and
                                     exported keying material (EKM) */
    Extension extensions<0..2^16-1>;
} TokenBinding;
struct {
    TokenBinding tokenbindings<0..2^16-1>;
} TokenBindingMessage;
```

# Token Binding ID Format

```
struct {
    TokenBindingKeyParameters key_parameters;
    uint16 key_length;  /* Length (in bytes) of the following
                                TokenBindingID.TokenBindingPublicKey */
    select (key_parameters) {
       case rsa2048_pkcs1.5:
       case rsa2048_pss:
          RSAPublicKey rsapubkey;
       case ecdsap256:
          ECPoint point;
    } TokenBindingPublicKey;
 } TokenBindingID;
```

- Provided_token_binding is used to establish a Token Binding when connecting to a server.
- Referred_token_binding is used when requesting tokens to be presented to a different server.