draft-popov-token-binding-00

Andrei Popov, Microsoft Corp.

The Token Binding Protocol

- The client generates an asymmetric Token Binding key per target server.
- The client proves possession of the Token Binding key on every TLS connection, by signing the tls_unique value [RFC5929] with the private key.
- The Token Binding is identified by the corresponding public key.
- Token Bindings are long-lived, i.e. they encompass multiple TLS connections and TLS sessions between a given client and server.
- Token Binding private key MUST be strongly protected (e.g. using a secure hardware module).

Preventing Token Theft

- When issuing a security token to a client that supports Token Binding, a server includes the client's Token Binding ID in the token.
- Later on, when a client presents a security token containing a Token Binding ID, the server makes sure the ID in the token matches the ID of the 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 a strongly-protected key (e.g. generated in a secure hardware module).

Privacy

- Different Token Binding keys SHOULD be used by the client for connections to different servers, according to the token scoping rules of the application protocol (e.g. eTLD for HTTP).
- Token Binding identifiers are never transmitted in clear text.
- Token Binding keys can be deleted by the user at any time, e.g. when clearing browser cookies.

Open Issue: Negotiating Token Binding Key Parameters

The current draft uses ALPN protocol IDs [RFC7301] to negotiate the use of the Token Binding key parameters (signature algorithm, length):

ALPN ID	Protocol
h2_tb_p256	HTTP/2 with Token Binding using ECDSA key and NIST P256 curve
h2_tb_rsa2048	HTTP/2 with Token Binding using 2048-bit RSA key
http/1.1_tb_p256	HTTP/1.1 with Token Binding using ECDSA key and NIST P256 curve
http/1.1_tb_rsa2048	HTTP/1.1 with Token Binding using 2048-bit RSA key

Pros:

- No TLS protocol changes;
- No additional round-trips.

Cons:

- Cartesian explosion of ALPN IDs;
- TLS ClientHello grows large, triggers interop issues with certain middle boxes;
- Protocols other than HTTP will need to register a separate set of ALPN IDs.

Should we consider a new TLS extension to negotiate Token Binding key parameters?

Links And Contact Information

- The Token Binding Protocol Version 1.0: http://tools.ietf.org/html/draft-popov-token-binding-00
- Token Binding over HTTP: http://tools.ietf.org/html/draft-balfanz-https-token-binding-00
- On GitHub: https://github.com/TokenBinding/Internet-Drafts

- Dirk Balfanz balfanz@google.com
- Vinod Anupam vanupam@google.com
- Andrei Popov andreipo@microsoft.com

The Token Binding Protocol Message Format

```
struct {
    ExtensionType extension_type;
    opaque extension_data<0..2^16-1>;
} Extension;
struct {
    TokenBindingID tokenbindingid;
    opaque signature<0..2^16-1>;// Signature over hashed ("token binding", tls_unique)
    Extension extensions<0..2^16-1>;
} TokenBinding;
struct {
    TokenBinding tokenbindings<0..2^16-1>;
} TokenBindingMessage;
```

Token Binding ID Format

```
enum {
    provided_token_binding(0), referred_token_binding(1), (255)
} TokenBindingType;
struct {
    TokenBindingType tokenbinding_type;
    SignatureAndHashAlgorithm algorithm;
    select (algorithm.signature) {
        case rsa: RSAPublicKey rsapubkey;
        case ecdsa: ECDSAParams ecdsaparams;
    }
} 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.