Token Binding in TLS 1.3

Nick Harper IETF 99

Token Binding for 1-RTT TLS 1.3

TBNEGO is limited to TLS 1.2 and earlier, but changes to support TLS 1.3 are minimal. Proposed changes for a new draft:

- Server puts token_binding extension in EncryptedExtensions
- Define signed value in terms of TLS 1.3 section 7.5 (Exporters) instead of RFC 5705
- Define an interaction with 0-RTT that a server MUST NOT negotiate token_binding and early_data on the same connection (unless updated by another draft)

draft-ietf-tokbind-tls13-0rtt: Changes in -02 from -01

- Use one exporter value for entire connection*
- Restrict to using PSKs issued from NewSessionTicket and used with (EC)DHE key exchange mode
- Added new TLS extension to negotiate and indicate use of 0-RTT TB
- Removed replay indication TLS extension

*Needs further discussion

0-RTT TB negotiation: initial handshake

ClientHello			
+ token_binding			
+ psk_key_exchange_modes			
+ key_share	>		
Key_share	/		
		ServerHello	
		{EncryptedExtensions}	
		+token_binding	
		{Certificate}	
		{CertificateVerify}	
		{Finished}	
{Finished}	>		
	<	[NewSessionTicket]	
		+early_data	
		<pre>+early_token_binding</pre>	
[Application Data]	<>	[Application Data]	

0-RTT or TB negotiation (client does not support 0-RTT TB)

 $(-1)^{-1}$

ClientHello			
+ early_data			
+ token_binding			
+ key_share			
<pre>+ psk_key_exchange_modes</pre>			
+ pre_shared_key			
(Application Data*)	>		
		ServerHello	
		<pre>+ pre_shared_key</pre>	
		+ key_share*	
		<pre>{EncryptedExtensions}</pre>	
		+token_binding	
		{Finished}	
	<	[Application Data*]	
{Finished}	>		
[Application Data]	<>	[Application Data]	
[Ubbitcarion para]		[Apprecation Data]	

0-RTT TB negotiation: resumption handshake (early data accepted)

```
ClientHello
+ early_data
+ early_token_binding
+ token_binding
+ key share
+ psk_key_exchange_modes
+ pre shared key
(Application Data*)
                         _ _ _ _ >
                                                ServerHello
                                           + pre shared key
                                               + key share*
                                      {EncryptedExtensions}
                                               +early_data
                                      +early_token_binding
                                            +token_binding
                                                 {Finished}
                          <----
                                       [Application Data*]
{Finished}
                          ---->
[Application Data]
                         <---->
                                         [Application Data]
```

Switching Exporters: Background

Current design (no switching exporters) is convenient for HTTP

Other application protocols might send a message in early data without any tokens, and want to send bound tokens after TLS handshake

Do we need application profiles on how to use protocols with 0-RTT TB?

Switching Exporters: Options

- Always use early exporter: security roughly equivalent to client certs used with resumption
- Require normal exporter used for all TokenBindingMessage structs sent after handshake completes: security is as close to TBPROTO as possible
- Some sort of middle ground
 - Previously suggested: client switches ASAP; server has no way of enforcing switch
 - Another option: client SHOULD switch post handshake; server soft fails (e.g. sends 4NN Too Early HTTP status code) if wrong exporter used
 - Other options?