GS2: Bridge between SASL and GSS-API

Simon Josefsson <simon@josefsson.org>

- Background, XML source, timeline, etc:
  - http://josefsson.org/sasl-gs2/
- Draft -00 posted Feb-06.
- Draft -01 posted Jun-06.
  - I picked one solution to solve Channel Bindings
    - Section 5
  - No other changes
- On-list discussions after -01 was posted
  - Resulted in simplified channel binding section
  - DO NOT read -01, read the text posted to the list and available in the URL above (and in later slides).

GS2: Bridge between SASL and GSS-API

Simon Josefsson <simon@josefsson.org>

- Open Issue before the WGLC: Channel Binding
  - Design and the text itself – posted to the list
  - Syntax of channel binding data used for TLS
    - In -01 and -02: Use TLS PRF to derive data
    - Alternative: reference draft-ietf-nfsv4-channel-bindings-04.txt
      - However, not ready – no syntax defined
      - Proposes to use client/server finished messages as CB
        - The client/server finished messages is the first encrypted messages sent under TLS encryption.
        - Difficult to implement for me in GNU SASL and GnuTLS
        - Seems less robust from a security analysis point of view.
          - The PRF output is well-defined, disclosing the encrypted client/server finished messages should be review further to make sure it is a good idea.
    - I prefer to reference Nico's draft, if it uses the TLS PRF
  - Update examples with channel bindings

Appendix

- For reference...

- Section 5 – next slide
  - "Channel Bindings"
- Section 5.1 – slide after next slide
  - "Name of TLS Channel For Use As Channel Binding"

5. Channel Bindings

The GS2 mechanism provide its own channel binding mechanism, instead of using the "chan_binding" parameter in the GSS-API context functions. The reason for this is that the GS2 mechanism provide an option to proceed even if the channel bindings does not match. The GSS-API framework specifies that authentication cannot proceed if channel bindings does not match.

Client and servers MUST set the "chan_binding" parameter in the calls to GSS_Init_sec_context to GSS_Accept_sec_context, respectively, to NULL.

Implementations SHOULD set the "client_cbqops" and "server_cbqops" to no security layer and instead depend on the session security afforded by the bound-in channel.

Use of no SASL security layers in combination with channel binding should provide better performance than using SASL security layers over secure channels, and better security characteristics than using no SASL security layers over secure channels without channel binding. For more discussions of channel bindings, see [16].

For TLS, the channel binding data is specified below. For other security layers, channel binding data will have to specified elsewhere, and this specification will have to be updated with explicit considerations.

5.1. Name Of TLS Channel For Use As Channel Binding

The TLS Pseudo-Random Function (PRF) generate, using the constant string "TLS channel binding", and based on the master secret and the random values established during a TLS handshake, a 64 octet string that make up the SASL channel binding data.

Using the terminology of TLS [13], the channel binding data is computed as follows:

```
ChannelBinding =
   PRF(SecurityParameters.master_secret,
      "TLS channel binding",
      SecurityParameters.server_random +
      SecurityParameters.client_random) [0..64];
```

The derived ChannelBinding is intended to be used as a name of the TLS channel that is cryptographically bound to the channel, for use in authentication mechanisms tunneled over TLS.