BTNS API proposal overview

Michael Richardson
<mcr@xelerance.com>
Nicolas Williams
<Nicolas.Williams@sun.com>

Three objects

- pToken – “protection Token”
  - deals with details of one session (IPsec SA)
- iToken – what identity to use
  - translates to/from phase 1 ID
- cToken - “credentials Token”
  - what credential (private keys) to use. May be a smart card, etc. (optional aspect of iToken)

Connected "sockets"

- TCP, SCTP
- UDP sockets that call connect()
- “initiator” => end that calls connect(), and likely becomes IKE initiator, after connect().
- “acceptor” => end that calls accept(), and therefore becomes IKE responder before accept().

Use Case 1

simple use case for initiators and/or acceptors

1. connect(2) (initiator) or accept(2) (acceptor)

2. get pToken from "fd"

3. get iToken from pToken

=> initiator identity and credential determined by system policy (PAD/SPD)

=> authorization based on peer ID evaluated by application after connection establishment

Use Case 2

initiator only

1. desired_acceptor_iToken = get_new_iToken("bob");
2. pToken = get_new_pToken(/* who am I*/ DEFAULT_INITIATOR_CTOKEN,
   /* who I want to talk to */desired_acceptor_iToken);
3. set pToken on fd.
4. connect(2)

=> initiator identity and credential determined by system policy

=> initiator specifies desired acceptor identity a priori

=> acceptor just like use case 1 or use case 4

Use Case 3

initiator only

1. desired_acceptor_iToken = get_new_iToken("bob");

2. i_iToken= get_new_iToken("alice");

3. desired_initiator_cToken= get_new_cToken(i_iToken);

4. desired_initiator_cToken= get_new_cToken(desired_initiator_cToken,
pkcs11_session);

5. pToken = get_new_pToken(/* who am I*/ desired_initiator_cToken,
/* who I want to talk to */desired_acceptor_iToken);

4. set pToken on fd.

5. connect(2)

=> initiator identity and credential determined by application

=> acceptor identity selected by initiator appliction (or could have been as in use 1)

=> acceptor application just as in use case 1 or use case 4

Use Case 4

this is acceptor side only

1. a_iToken = get_new_iToken("bob");
2. desired_acceptor_cToken = get_new_cToken(a_iToken, /*location of private credentials*/...);
3. set cToken on "fd"
4. accept(2)
5. step 2 and 3 from use case 1

Unconnected "sockets" (datagrams)

Use Case 5

simple use acceptors

1. recvmsg(...,&pToken);

2. get iToken from pToken

=> initiator identity and credential determined by system policy (PAD/SPD)

=> authorization based on peer ID evaluated by application after connection establishment

=> initiator identity and credential determined by system policy

=> initiator specifies desired acceptor identity a priori

=> acceptor just like use case 1 or use case 4

Use Case 6

Use Case 6 (initiator only)

1. desired_acceptor_iToken = get_new_iToken("bob");
2. pToken =

   get_new_pToken(/* who am I*/ DEFAULT_INITIATOR_CTOKEN,
   /* who I want to talk to */desired_acceptor_iToken);
3. sendmsg(...,pToken);

Similarities to GSSAPI

SEE RFC2743, section 2.2.1. GSS_Init_sec_context()
claimant_cred_handle and targ_name arguments.

(targ_name is optional in BTNS API --- the system can determine it.
But it is required in GSSAPI, because the system has no default).

RFC2743, section 2.2.2. GSS_Accept_sec_context()
acceptor_cred_handle.

iToken is similar to GSS "NAME" object

cToken is similar to GSS "CREDENTIAL HANDLE"

pToken is similar to GSS "CONTEXT HANDLE"

Use Case 5 and Use Case 6 is ***not*** easily implemented for systems using connection-latching-01 section 2.2: "Latching through PAD manipulations (and extensions)"

easily done with section 2.1: "Using Intimate Interfaces Between ULPs and IPsec"