Better IPSec
Security Association Resolution

Netconf 2006
Tokyo

James Morris
jmorris@namei.org
Problem

a) Outbound packet

b) Security policy db entry match

c) No security association in kernel

• Most of the time, we return EAGAIN to app or drop packet if forwarding.

• We kick the key manager, and usually have an SA available for next packet.
Problem...

- It actually kind of works for one case: blocking sendmsg() of datagrams.
- Process is scheduled in a loop until SA resolved. See xfrm_lookup().
- Does not work for connect(2), so ping and many UDP apps just get EAGAIN.
Solution

• General solution for all protocols and contexts:
  - `connect(2)`
  - `sendmsg(2)`
  - forwarding path (tunnel endpoint)
  - various kernel-generated packets
  - blocking and non-blocking modes
Solution...

• Ideally, we'd like connect(2) to follow Posix semantics, for non-blocking this is:
  - Return EINPROGRESS first
  - Return EALREADY until SA resolved

• For non-blocking sockets in general, it'd be nice to make sure poll(2) works as expected.
  - even for datagram protocols, as IPSec adds a kind of session underneath.
Solution...

• `sendmsg(2)` should return EAGAIN for non-blocking case

• For tunnel end point, we probably need to queue packets in a resolution queue.

• This may also be useful for non-blocking socket case.

• Herbert has suggested larval dst to go with larval SA.
Status

• Current patch contains a lot of instrumentation and some initial changes:
  - Make connect(2) work for the blocking case, hooking into ip_route_connect()
  - Propagate new flags down to xfrm_lookup() to control behavior:
    • Kick the key manager?
    • Sleep until resolved?
Ongoing work

• Continue to develop code to handle all cases and protocols
• Probably involve some code consolidation
• Determine how much of the problem to solve
Issues

• Not clear on all of the use-cases for this:
  - Opportunistic encryption
  - Complex/large scale policy where pro-active SA negotiation overhead would be too high
  - Others?