Removing TLS from RPKI Provisioning Protocol

Rob Austein <sra@isc.org>

Maastricht, July 2010
Executive Summary

We added TLS to solve a problem (message replay)

TLS as we’re using it creates real operational headaches

There are better ways to solve the problem

So let’s remove TLS and do one of the better things
Replay example

Child requests issuance with key A
Attacker captures copy of child’s request
Server issues with key A
Time passes
Key A is compromised
Child requests reissuance with new key B
Server reissues with key B
Child requests revoke of all certs with key A
Server revokes all certs with key A
Attacker replays saved request
Server reissues with old compromised key A
Oops

Notes

A and B are RPKI keys
BPKI key not compromised
In theory, long-lived TLS session would prevent replay here

In practice, TLS does prevent replay here, but almost by accident
   There is no long-lived session, TLS or otherwise, we’re using HTTPS
   Encryption makes capture hard for attacker
   Client TLS cert makes impersonating client hard for attacker

In theory, TLS just uses same BPKI keys and certs as CMS does

In practice, early testbed experience with TLS has been wretched
   TLS requires extra config due to virtual hosting problem
   TLS Server Name Indication requires DNS hackery
   TLS configuration oops is most single common failure
   TLS configuration oops is nightmare to debug
Other Issues

Our use of TLS relies on client certificates

Across organizational boundaries
Few real-world examples of this

Massive duplication between CMS and TLS

...Except where TLS is worse
We need CMS anyway, for audit trail
All authorization is done based on CMS (audit again)
CMS could do encryption too if we needed that (we don’t)
Easier Replay Protection

Trivial: CMS timestamps
   Already present
   Just insist that it increase monotonically
   Good enough for attack described above

Epsilon more work: serial numbers
   Add field to XML header
   Insist that serial be one greater than last recorded serial number
   Handles sub-second granularity problem
   Need reset mechanism, probably just a timeout
   Not obvious what to do if one detects a sequence gap

My preference: just CMS timestamp, at least for now
   Minimal change and solves the known problem
   "Never test for an error condition you don’t know how to handle"
Summary and Desired Outcome

Proposed solution
  Remove TLS from provisioning protocol
  Add CMS timestamp check to protocol
  Declare victory and move on

Discussed on mailing list April 2010
  Response generally favorable
  But no definite conclusion

Chance to simplify an IETF protocol does not occur very often
  Seize the moment

Thank you!