Deployment Models for Backup Certificate Systems

Eric Rescorla

ekr@rtfm.com

NIST Online Trust Workshop 2013

Problem Statement

- We are concerned only with SSL/TLS servers
- The primary source of SSL/TLS identity will continue to be certificates
- Some certificates are incorrectly issued
  - CAs make mistakes
  - Sometimes CAs are compromised (or misbehave)
- Focus is minimizing the impact of misissuance

Lots of work in this area

- DANE [RFC6698]
- Certificate Transparency [draft-laurie-pki-sunlight]
- HPKP [draft-ietf-websec-key-pinning] and TACK [draft-perrin-tls-tack]
- Perspectives, Sovereign Keys, Convergence
- A bunch of survey-type ideas

What is it going to take to get widespread deployment?

Questions for Analysis

- Who needs to change their behavior? (RPs, servers, CAs, ...)
- What are the benefits?
- Who gets the benefits?
- What other technology does this depend on?
- What are the downside risks?

DANE Overview (usages 0 and 1)

- Server operators publish TLSA records in DNS
  - Records can contain:
    * A CA certificate/key that must be in the path
    * An EE certificate/key that must be used by the server
  - Records MUST be authenticated via DNSSEC
- When client visits `www.example.com`
  - Tries to resolve a TLSA record for `_443._tcp.www.example.com`
  - If present, then do *both* the PKIX checks and the DANE checks
  - If absent, do *just* the PKIX checks

DANE Deployment Summary

Changes needed	Browser, server, server's DNS
Benefits	Prevention
Scope	When server and client both deploy
Dependencies	DNSSEC deployment at clients, servers, *and intermediaries*
Risks	Self-DoS via incorrect TLSA records DNSSEC increases rate of ordinary resolution failure "False positives" because of broken intermediaries

What happens if something goes wrong with DNSSEC?

- Example: RRset is supposed to be signed but no RRsig present
  - RP cannot tell whether a TLSA record should be present
- How can this happen?
  - Server error
  - Broken intermediaries (e.g., filter DNSSEC records)
  - An active attack
- Options
  - Assume attack and terminate the connection
  - Assume everything is OK and proceed

What does the specification say?

"An attacker who is able to divert a user to a server under his control is also likely to be able to block DNS requests from the user or DNS responses being sent to the user. Thus, in order to achieve any security benefit from certificate usage 0 or 1, an application that sends a request for TLSA records needs to get either a valid signed response containing TLSA records or verification that the domain is insecure or indeterminate. If a request for a TLSA record does not meet one of those two criteria but the application continues with the TLS handshake anyway, the application has gotten no benefit from TLSA and SHOULD NOT make any internal or external indication that TLSA was applied." [RFC 6698; §4.1]

This is not plausible in most UAs; they must either succeed or fail.

How common are DNSSEC failures?

- DANE requires that endpoints validate DNSSEC
- In 2010 many consumer routers didn't properly proxy DNSSEC resolution [Dietrich 2010]
  - 16 out of 33 had some DNSSEC support
  - Only 9 worked with packets $>$ MTU
- All routers worked if you bypassed DNSSEC proxy
- Unclear how much has changed in 3 years
- How can client tell what he is behind?
- What about ISP behavior?

Impact of using DNSSEC at all

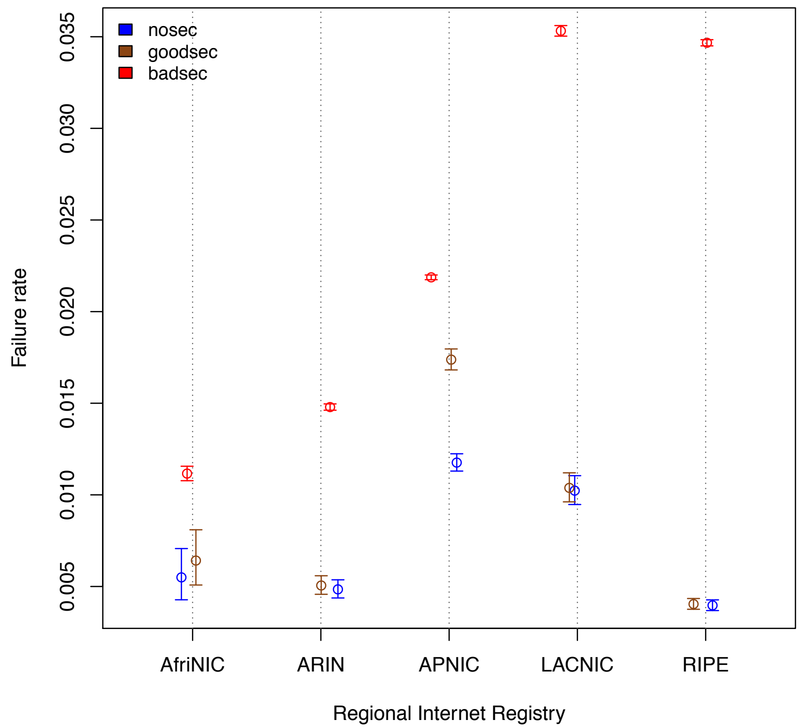

Figure 7: Failure rates broken down by resolver IP RIR. Error bars indicate a 95 percent binomial proportion confidence interval.

Source: Lian et al. (in submission, 2013)

User Agent Vendor Incentives?

- UAs must decide whether to implement and rely on DANE
  - Users rarely change the system defaults
  - And this has real costs
- UA vendors have no control over the user's network environment
  - And this environment can change when the user moves
- The network environment changes very slowly
  - Even for much more compelling applications like voice and video
  - ... which still need extensive NAT/firewall traversal mechanisms

HPKP Overview

- Server can provide a `Public-Key-Pins` or `Public-Key-Pins-Report-Only` HTTP header
  - Lists hashes of public keys which must appear in the server's certificate chain
  - Client remembers these hashes ("pins" for future use)
- Once pinned client does *both* PKIX checks and verifies that one of the pinned keys is present
- If pin check fails
  - `Public-Key-Pins` → fail
  - `Public-Key-Pins-Report-Only` → report error to `report-uri`
- TACK is conceptually similar but operates at the TLS layer not the HTTP layer

HPKP Deployment Summary

Changes needed	Browser (already in Chrome, under development in Firefox), server
Benefits	Prevention (or detection)
Scope	When server and client both deploy
Dependencies	None
Risks	Attack on first use Self-DoS via incorrect pinning

Self-DoS via Incorrect Pinning

- What happens if a host has key X and advertises a pin for key Y
  - This would create a self-DoS for the duration of the pin
- Complete PIN failures are unrecoverable for pin lifetime
  - Pin lifetimes need to be long in order to work
- HPKP has three mechanisms to prevent this
  - Current connection must be valid with proposed pin
  - Must advertise multiple keys ("backup pin")
  - Report-only mode allows server to discover all keys currently in use

HPKP Server Incentives

- Publishing a pin provides security with set of pin-verifying clients
  - Currently abouut 20-30% of user's browsers
- Primary risk is self-DoS
- Currently very few sites publish pins
  - About 300 static pins in Chromium
  - Unknown how many published pins (but rumor is it is small)
    - Less than 1000 HSTS sites [Ristic 2013]
- Why is this number so low?

Certificate Transparency Overview

- Participating CAs publish all certificates they issue
  - Provide servers with proofs*
- Clients check whether certificates have a proof of publication
  - Assuming they are supposed to
  - Any certificate which should have a proof but does not is rejected
- Server operators (or some service) can check for certificates which should not exist
  - Did someone else obtain a certificate for my domain?
  - Actually dealing with misissued certificates is out of scope

*Insert crypto magic here

CT Deployment Summary

Changes needed	Browser, server, notary service, CA
Benefits	Detection
Scope	Participating CAs and servers who check
Dependencies	Robust revocation (currently nonexistent)
Risks	Breakage of non-participating CAs (whenever CT is required)

Limits of CT Detection

- CT detects misissuance by *participating CAs*
  - Does nothing about non-participating CAs
- Server isn't primarily worried about misissuance by *his* CA
  - ... but he is worried about other CAs
  - And it can't control them
- Attacker can pick any CA to attack
  - Attacker difficulty is security of the weakest non-participating CA
  - Poorly run CAs seem likely not to participate

CT Deployment Incentives

- Requires participation by CAs
  - In principle servers can self-publish
  - ... but clients need to know when proofs are expected
- CAs have little incentive to participate
  - Unless browsers require proofs from all CAs
  - ... which breaks the world
- Browsers can only require proofs once nearly all CAs already publish
- Classic collective action problem

Summary

- None have seen widespread deployment
- All have severe collective action problems
- Minimally, need support on both clients and servers
  - CT and DANE both require support elsewhere
- Hard to deploy any of these without breaking stuff

Questions?