Dynamic Public Key Pinning

Overview

Google

Server identities tend to be long-lived, but clients have to **re-establish the server's identity on every TLS session**.

On **each connection**, any trusted signing certificate (root or intermediary) could **sign for any identity, true or fake**.

To make Chrome resilient against this problem for Google properties, we built in "preloaded" fingerprints for the known public keys in the certificate chains of Google properties.

We thereby **exposed the false *.google.com certificate** DigiNotar signed.

Scaling Up

Google

We would like to provide this risk mitigation technique to everyone, to protect more users and more web sites.

Preloading does not scale, so we need something dynamic.

- Could use an HTTP header
- Could use an X.509 extension or "superfluous" certificate
- Other means?

For now, we are exploring headers and superfluous certificates. The current Internet-Draft defines the Public-Key-Pins header.

Server Deployment

Google

The header-based solution is easy to deploy. Take the SHA1 or SHA256 hash of the Subject Public Key Info structure of the X.509 certificate (code in the Internet-Draft), base-64 encode it, and configure the web server to put it in a header.

```
Header add Public-Key-Pins "max-age=10000; pin-sha1=\"
ObT42aoSpAqWdY9WfRfL7i0HsVk=\"; pin-sha1=\"
hvfkN/qlp/zhXR3cuerq6jd2Z7g=\""
```

You can pin to any SPKI in the signature chain: end-entity, intermediary, root. Select your degree of precision.

Client Behavior

Google

Clients require servers to serve at least one **live pin** (a hash of an SPKI in the current cert chain) and at least one **backup pin** (a hash of an SPKI **not** in the current cert chain).

Clients remember the **most-recently-seen** set of pins for *max-age* seconds after it was most recently seen.

Clients drop TLS connections for which the set of SPKIs in the chain does not intersect with the set of remembered pins!

Limitations

Google

Pinning does not protect against compromise or abuse of any of the private keys or signing powers corresponding to any pinned SPKIs.

Dynamic pinning still suffers from the bootstrap problem, but it is significantly better than the status quo (which is "bootstrap every time") and significantly more scalable than preloading pins.

Servers might inadvertantly "brick" themselves (pin for a long time to an SPKI which is later lost, for example). That's why we require a backup pin, although there is no fool-proof solution.

More Information

palmer@google.com
cevans@google.com

http://www.ietf.org/id/draft-evans-palmer-key-pinning-00.txt