CT Policy Days Breakout Session Notes

CT Enforcement (04/2018) Breakout Discussion

CT Policy Days @Google NYC
Nov 2-3, 2017

100% CT for DV/OV/EV in Chrome

Applies to newly issued certificates

Evangelizing with CAs

- Some Sub CAs concerned about April 2018 being “unreasonable”
- Chrome concerned about publicly-trusted CAs who can’t comply with policy with year+ notice

Sub CAs waiting for Roots to notify them of this requirement.

- Root CAs not necessarily going to do this

Role of Technically constrained Sub CAs

Implementation hidden by flag in earlier release?

- Possible, but not currently scoped as it would need reworking to allow flag to pass effective date (since April 2018 would be in the future)

100% CT for DV/OV/EV in Chrome

Enterprises running publicly-trusted Sub CAs

- Ability to push enterprise policy to disable enforcement for particular CAs
- Many enterprises using publicly-trusted sub CAs should probably migrate to Managed PKI / Enterprise RA solutions

Bypassable Interstitial

- Only for newly issued certificates as of implementation date
- Caching for interstitial decisions. ~1 Week cache
- Same interstitial for no SCT as insufficient / untrusted SCTs
- Ability to see additional details in DevTools/Console
    - Maybe. The value proposition to level of effort is likely low

Enterprise Roots & Configuration

Enterprise configuration to enable disabling CT enforcement for **particular CAs**

- The exact strategy is not entirely concrete
    - Could be just for TCSCs
    - Idea to restrict to OV labels

Log Lifecycle Breakout Discussion

CT Policy Days @Google NYC
Nov 2-3, 2017

Improving the Log Evaluation Period

Currently:

- Log adds Compliance Monitor CA
- Compliance CA issues new certificates and attempts to regularly
  - Logs certs, pre-certs, and pre-certs signed by special pre-cert CA
  - Performs get-entries calls
  - Performs get-sth calls
- Uptime is calculated from successful responses
  - Only get-sth calls affect uptime calculations
  - (# Successful) / (# Probes) = uptime %
- Certain responses are considered the Log’s fault, others are not

Possible Improvements

Firehose Log with Compliance CA certificates

- Bloating Log
- Can allow for re-spin of log after load test

Firehose Log with valid certificates

Build a test suite to exercise every API endpoint for edge cases

Live, interactive exercise trying to run through high-risk scenarios

Create a wiki describing possible failures and post-mortems

Availability measurement starting at day 0 from 90/90 days 100% uptime (going backwards before existence) and then measuring eval period more accurately

How to encourage Community Participation

- Maybe get CAs to submit newly-issued certificates to Logs under evaluation

How to handle Sharded log applications?

Should there be a limit on the number of shards we permit in a single application?

How should we handle adding new temporal shards?

- Allow annual additions for future shards?
- Require batch additions?

Considerations

How to balance Repeat Log Operators in good standing with the need to do consistently strong evaluation with log

- Streamlining process should mitigate this sufficiently. (a few weeks is more reasonable than 90 days, e.g.)

Do we allow respinning logs after firehosing with synthetic certs?

- Becomes Log Operator Evaluation
- Phase 1: firehose with synthetic certs (Maybe 1 week?)
- Phase 2: respin log and change URL, set up shards, and open up to public roots. Announce to CAs that submission is open.
- Phase 3: if log passes eval, announce to forum, log enters queue to become qualified in a Chrome release

Considerations Continued

How large is too large for a log?

- Pilot, Rocketeer, Icarus all >140 Million certificates
- Over half are expired in Icarus

Providing Ceiling on their maximum log size during application

Logs should stay up until the last certificate logged expires

Log URL needs to be unique per key

Multiple UA CT Policy Breakout Discussion

CT Policy Days @Google NYC
Nov 2-3, 2017

Goals

Resiliency - Reduce the chance that log failure will result in certificate distrust

Consistency between various UA policies

Determine what dependencies there are on a robust inclusion checking mechanism

Considerations

Different UAs have different update frequencies

- This can lead to different requirements in a policy
- Mobile vs Desktop
- Browser vs OS

Scope restricted to TLS certificates

Inclusion Proof Checking vs SCT

The one Google, one non-Google policy is, in part, originated in our ability to determine the reliability of at least one log (which is required for all CT-qualified certificates)

If all User Agents adopt an “Our_favorite_log +1” style policy, is that bad?

Considerations Continued

Difference between “Our Favorite Log + 1” and UAs defining a Designated Log Set

- DLS results in greater flexibility, fewer single points of failure for issuance (embedding)

Should the choice of where to log be decided by UAs or CAs?

- Incentive structures differ, to a degree

Can we attain the desired level of assurance from CT absent a robust inclusion checking mechanism?

- In the “Us +1” model, the answer is yes. UAs implicitly trust themselves, clients trust UAs
- In the “Our Favorite Log + 1” model, the UA is delegating explicit trust to the Log Operator. Would likely be backed by contractual agreements / obligations or strong technical controls.
- In the DLS model, contractual agreements do appear to be required, likely resulting in human (traditional) auditing of the logs

Considerations Continued

Requiring CT in a UA versus requiring CT in a Root Program level

- Requiring in Root Program turns issuing a non CT qualified certificate into a misissuance, which forces the issue of redaction or mandating massive overhaul on site operators
- Requiring at the UA (Browser) level allows failure within the UA while not fundamentally altering the trust on the issuing CA

Need for a sensible, consistent approach to deprecations / disruptive changes in the future (Browser vs Root Program enforcement)

How to choose a Designated Log Set?

What should a log in this set look like?

- Demonstrated ability to handle enormous scale, QPS, redundancy safeguards
- Provides global availability to reach users across the world, not restricted to a particular geographic region
- Has the proper incentive structure to responsibly operate a CT Log
- Maintains an open policy for accepting submissions from publicly-trusted CAs

What’s bad about existing policy

One Google Requirement

- Is it bad if this is only in Google policy?

2-4 required SCTs?

Limitation to Stapled OCSP

90 Day Evaluation

Lack of Robust Auditing Mechanism causing policy compromises

Axes of Diversity (From Last CT Policy Days)

Corporate

- Risk of Collusion
- Risk of Incompetence

Geopolitical (jurisdictions)

- Risk of Compulsion

Implementations (Codebase)

- Risk of Bug/Exploit

Infrastructure (AWS, GCP, Etc.)

- Risk of Compulsion
- Risk of Outage

Assurance Level

CT Policy Days Log Usage

CT Policy Days Log Usage

Log Usage

- Expired Certs
- Revoked Certs
- New usages from CAs
- Researcher (large corpus)

Rate Limits

- Rate limit by root
    - May have already done all the work, and rate limit may not help

CT Policy Days Log Usage

Log Operator Acceptance/Rejection

- Are there policies on what logs must accept/reject
    - Example: Policy Mappings
- Tension between log being able to reject certs and clients desire to log certs

Proof of Work requirement to provide rate limiting

Client read limiting

CT Policy Days Log Usage

Preloading

- Should it be required
- Should it be forbidden

Best Practices

Known Issues

Q: What would the risks be if required to log all CAs

Q: What would the risks be if required if sharding was required

Enterprise CT

Enterprise CT

Internal Names, Internal Keys

Q: What’s the threat model

Scenarios:

- Want to only trust a given enterprise cert (e.g. no self-signed)
- Want to have a detection mechanism for ‘rogue’ PKIs (e.g. everything must be publicly logged || privately logged)
- Is it a question about simply wanting to know what clients trust?