6.824 Certificate Transparency FAQ

Q: What's the difference between Monitors and Auditors?

A: An "Auditor" runs in each web browser. When the browser visits an https web site, and receives a certificate from the https server, the browser asks the relevant log server(s) for a proof that the certificate is present in the log. The browser also stores (persistently) the last STH (signed tree head) it has seen, and asks the log server for a consistency proof that the server's newest STH describes a log that's an extension of the last log the browser has an STH for.

A "Monitor" checks that the names the monitor knows about have only correct certificates in the log. For example, MIT could run a monitor that periodically looks in the CT logs for all certificates for *.mit.edu, and checks against a list of valid certificates that the monitor knows about. Since MIT is running the monitor, it's reasonable for the monitor to have a list of all MIT certificates (or at least know which CA should have issued them).

Another possibility is for each CA to run a monitor, that periodically checks the logs to make sure that DNS names for which the CA has issued certificates do not suddenly appear to have certificates from some other CA.

A Monitor fetches the entire log, and knows (for some DNS names) how to decide if a certificate is legitimate. A monitor also checks that the STH from the log server matches the log content provided by the log server.

An Auditor (browser) only checks that the certificates it uses are in the log, and that each new version of the log contains the previous version it knew about as a prefix. So an Auditor doesn't have the complete log contents, only some STHs, some proofs of inclusion, and some proofs of log consistency.

If Auditors and Monitors see the same log contents, then the fact that Monitors are checking for bogus certificates affords Auditors (browsers) protection against bogus certificates. If a bogus web server gives a bogus certificate (e.g. claiming falsely to be mit.edu) to a browser, and the browser only uses the certificate if it's in the CT log, and MIT's monitor sees the same log content as the browser, then MIT's monitor will detect the bogus certificate.

Much of the detailed design (the Merkle trees and proofs) are about ensuring that Auditors and Monitors do indeed see the same log contents, and that log servers cannot change log contents between when Auditors look and when Monitors look.

Q: Why is it important for CT to ensure that the log is immutable and append-only -- that certificates cannot be dropped from the log?

A: Suppose a malicious CA creates a bogus certificate for gmail.com. It creates a web site that looks like gmail.com and tricks your browser into going there. It conspires with a corrupt CT log operator to convince your browser that the bogus gmail.com certificate is in the log. Now your browser will accept the bad web site, which may capture your password, log into the real gmail as you, fetch and record your mail, and show you your inbox as if it were real.

Afterwards, the malicious CA and CT log operator need to cover their

tracks. If they can cause the bogus certificate for gmail.com to disappear from the CT log without anyone noticing, then they can probably get away with their crime.

However, since the CT system makes it perhaps impossible to delete a certificate from a log once anyone has seen it, this attack is likely to be detected.

Q: How can someone impersonate a website with a fake SSL certificate?

A: Suppose an attacker wants to steal Ben Bitdiddle's gmail password.

Let's assume that the attacker has a correct-looking certificate for gmail.com, signed by a legitimate CA. This is the scenario CT is aimed at. So let's assume CT does not exist.

One piece of the attack is to trick Ben's browser into connecting to the attacker's web server instead of the real gmail.com's servers. One way to do this is to generate fake DNS replies when Ben's browser asks the DNS system to look up gmail.com -- replies that indicate the IP address of the attacker's servers rather than the real gmail.com servers. DNS is only modestly security in this respect; there have been successful attacks along these lines in the past. The attacker could snoop on Ben's local WiFi network or the MIT campus net, intercept Ben's DNS requests, and supply incorrect replies.

Once Ben's browser has connected to the attacker's web server, using https, the attacker's web server will send the browser the bogus gmail.com certificate. Since it's signed by a legitimate CA, Ben's browser will accept it. The attacker's web server can produce a page that looks just like a gmail.com login page, and Ben will type his password.

Q: What do monitors do when they find a suspicious certificate?

A: I think in practice some human has to look into what happened -- whether the certificate is actually OK, or is evidence of some CA's employee or an entire CA being either malicious or poorly run, or was just a one-off accident. Most situations are likely to be benign (e.g. someone legitimately got a new certificate for their web site). Even an incorrectly-issued certificate may well be just an accident, and once detected should be placed on a certificate revocation list so browsers will no longer accept it. If the problem seems to be deeper (malice or serious negligence at a CA) you'd then report the problem to the major browser vendors, who would talk to the CA in question, and if they didn't get a satisfactory promise to improve, might drop the CA's public key from the browser list of acceptable CAs.

Q: Is the existing CA strategy for authentication still necessary? It seems like the log could serve as the authority itself.

A: It's an attractive idea! One thing people are worried about is that, without CAs to act as a filter, the CT system could be overwhelmed with garbage certificate submissions. Another is that the CAs do perform some verification on certificate requests. Another is that browsers and web servers already use certificates as they exist today; switching to a different scheme would be a giant pain. I suspect that just as the CA scheme ultimately had more serious problems than most people imagined with it was introduced in the 1990s, so would CT if we had to entirely rely on it.

Q: How can a Monitor know which certificates in the CT log are
legitimate, and which are bogus?

A: One way or another a Monitor has to have special knowledge about
the set of DNS names for which it checks for bogus certificates.

Some companies (e.g. Google and Facebook, probably others) run their
own Monitors that look for certificates that mention DNS names that
they own. Such monitors are supplied with a list of the certificates
that the owner considers legitimate, or of the CAs that are authorized
to issue certificates for the owner.

A CA can run a Monitor that checks, for each DNS name the CA has
issued certificates for, that no other CA has issued a certificate. So
that the CA's monitor is protecting the CA's customers.

There are also monitoring services. I think you tell them what
certificates or CAs are legitimate for your DNS names, and the service
periodically checks the CT logs for bogus certificates for your names.

https://www.facebook.com/notes/protect-the-graph/introducing-our-certificate-transparency-monitoring-tool/1811919779048165/

https://blog.cloudflare.com/introducing-certificate-transparency-monitoring/

https://sslmate.com/certspotter/howithelps

Q: How does the browser/Auditor learn the index of the certificate for
which it needs an inclusion proof?

A: The browser (Auditor) doesn't actually supply the index, it
supplies a hash of the certificate it wants to look up. The CT log
server replies with a proof of inclusion (plus the certificate's index
in the log). So the CT log server must have a big side-table that maps
hashes of certificates to their position in the log.

See section 4.5 of this document for details:

https://tools.ietf.org/html/rfc6962

Q: How can I check if a given web site's certificate is in the CT logs?

A: https://transparencyreport.google.com/https/certificates

Q: How can I watch the browser checking CT logs?

A: Here are directions for Chrome:

  http://www.certificate-transparency.org/certificate-transparency-in-chrome

Q: Has CT ever actually caught a bogus certificate?

A: Here's a success story:

https://security.googleblog.com/2015/09/improved-digital-certificate-security.html

Q: Won't checking the CT logs be a performance bottleneck?

A: I do now know how much a problem performance is.

Replicas of a given log can be served by many servers to increase throughput. A log operator would have a master copy of the log, and push updates to the replicas. I don't know if CT implementations actually do this.

Once a browser (Auditor) fetches a log inclusion proof for an https site's certificate once, the browser can cache the proof forever.

Here's a post that touches on some performance measures for CT:

https://blog.cloudflare.com/a-tour-through-merkle-town-cloudflares-ct-ecosystem-dashboard/

Q: Isn't it possible for a log server to "fork" its log, i.e. show different logs to different viewers?

A: That is indeed a possible attack, at least temporarily.

The way to think about this is that the CT design aims to improve the current CA system, while still being practical to deploy. It doesn't aim for perfection (which no-one really knows how to achieve). In the pre-CT system, there was basically no way to ever discover bogus certificates. The ones that were discovered were found accidentally. With CT, you can't spot them right away, so they can be used, but monitors will eventually (perhaps within a day) realize either that there's a bogus certificate in the logs, or that a log operator has forked their log.

Q: How are incorrectly-issued certificates revoked?

A: https://www.ssl.com/article/how-do-browsers-handle-revoked-ssl-tls-certificates/

Q: Are the different CT logs supposed to be identical?

A: Different CT logs are likely to be different. To achieve its goals, CT only needs a certificate to be in a single well-behaved log. The certificate (really the "SCT") contains the ID of the log it is in, so browsers know which log service to talk to. Since browsers insist that the certificate be in the log, and monitors inspect all logs, it's likely that a bogus certificate that's in even a single log will be quickly detected.

Because any given log service may fail or become malicious, there are a modest number of independent log services (dozens), and CAs typically insert each new certificate in multiple logs (e.g. five).

Q: Gossipping doesn't seem to be defined or implemented. Is CT still secure even without gossip?

A: Without gossip, a conspiring CA and CT log could fork a browser and trick it into using a bogus certificate (which can happen even with gossip), and perhaps never be noticed (which gossip could prevent). But, because CT is fork-consistent, the malicious CA/CT would have to forever maintain and extend the log they show to the victim, to include all certificates the victim expects. This would have to be a special and unique log for that victim. If the malicious CA/CT didn't continue to do this, the victim would notice that many certificates that work for other people don't work for the victim, and perhaps the victim would realize something was wrong. This is a somewhat weak story, but it does seem to be the case that exploiting lack of gossip would be awkward and risky.