TRANS
Internet-Draft
Intended status: Experimental
Expires: September 10, 2015

L. Nordberg
NORDUnet
D. Gillmor
ACLU

March 09, 2015

Gossiping in CT
draft-linus-trans-gossip-ct-01

Abstract

This document describes two gossiping mechanisms for Certificate Transparency [RFC6962]; SCT feedback and STH gossip. In order for HTTPS clients to share SCTs with CT auditors in a privacy-preserving manner they send SCTs to originating HTTP servers which in turn share the SCTs with CT auditors. CT auditors and monitors share STHs among each other.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 10, 2015.

Copyright Notice

Table of Contents

1. Problem

Public append-only untrusted logs have to be monitored for consistency, i.e. that they should never rewrite history. Monitors and other log clients need to exchange information about monitored logs in order to be able to detect a partitioning attack.

A partitioning attack is when a log serves different views of the log to different clients. Each client would be able to verify the append-only nature of the log while in the extreme case being the only client seeing this particular view.

Gossiping about what's known about logs helps solving the problem of detecting malicious or compromised logs mounting such a partitioning attack. We want some side of the partitioned tree, and ideally both sides, to see the other side.

Disseminating known information about a log poses a potential threat

to the privacy of end users. Gossiping about data which is linkable to a specific log entry and by that to a specific site has to take privacy considerations into account in order not to leak sensitive information.

2. Who should gossip

- HTTPS clients and servers (SCT feedback)
- HTTPS servers and CT auditors (SCT feedback)
- CT auditors and monitors (STH gossip)

3. What to gossip about and how

There are two separate gossip streams:

- SCT feedback, transporting SCTs from clients to auditors
- STH gossip, sharing STHs between auditors/monitors

3.1. SCT feedback

The goal of SCT feedback is for clients to share SCTs and certificate chains with CT auditors and monitors in a privacy-preserving manner.

HTTPS clients store SCTs and certificate chains they see and later send them to originating HTTPS servers by posting them to a .well-known URL. This is described in Section 3.1.1.

HTTPS servers store SCTs and certificate chains received from clients and later share them with CT auditors by either posting them or making them available on a .well-known URL. This is described in Section 3.1.2.

HTTPS clients MAY send SCTs and cert chains directly to auditors. Note that there are privacy implications of doing so, outlined in Section 4.1.1.

3.1.1. HTTPS client to server

An HTTPS client connects to an HTTPS server for a particular domain. The client receives a set of SCTs as part of the TLS handshake. The client MUST discard SCTs that are not signed by a known log and SHOULD store the remaining SCTs together with the corresponding certificate chain for later retrieval.

When the client later reconnects to any HTTPS server for the same domain it again receives a set of SCTs. The client MUST update its store of SCTs for the domain and MUST send to the server the ones in its store that were not received from that server.

Note that the SCT store also contains SCTs received in certificates.

The client MUST NOT send the same set of SCTs to the same server more often than TBD.

An SCT MUST NOT be sent to any other HTTPS server than one serving the domain that the certificate signed by the SCT refers to.

SCTs and corresponding certificates are POSTed to the originating HTTPS server at the well-known URL:

https://<domain>/.well-known/ct/v1/sct-feedback

The data sent in the POST is defined in Section 3.1.3.

HTTPS servers MUST perform a number of sanity checks on SCTs from clients before storing them:

1. if a bit-wise compare of the SCT matches one already in the store, discard

2. if the SCT can't be verified to be a valid SCT for the accompanying leaf cert, issued by a known log, discard

3. if the leaf cert is not for a domain that the server is authoritative for, discard

Check number 1 is a pure optimisation. Check number 2 is to prevent spamming and attacks where an adversary can fill up the store prior to attacking a client. Check number 3 is to help misbehaving clients from leaking what sites they visit.

3.1.2. HTTPS server to auditors

HTTPS servers receiving SCTs from clients SHOULD share SCTs and certificate chains with CT auditors by either providing the well-known URL:

https://<domain>/.well-known/ct/v1/sct-gossip

or by HTTPS POSTing them to a number of preconfigured auditors.

The data received in a GET of the well-known URL or sent in the POST is defined in Section 3.1.3.

HTTPS servers SHOULD share all SCTs and certificate data they see that pass the checks above, but MAY as an optimisation chose to not share SCTs that the operator consider legitimate. An example of a legitimate SCT might be one that was received from a CA as part of

acquisition of a certificate. Another example is an SCT received directly from a CT log when submitting a certificate chain.

HTTPS servers MUST NOT share any other data that they may learn from the submission of SCTs by HTTP clients.

Auditors SHOULD provide the following URL accepting HTTPS POSTing of SCT feedback data:

https://<auditor>/ct/v1/sct-gossip

Auditors SHOULD regularly poll HTTPS servers at the well-known sct-feedback URL. How to determine which domains to poll is outside the scope of this document but the selection MUST NOT be influenced by potential HTTPS clients connecting directly to the auditor.

3.1.3. SCT feedback data format

The data shared between HTTPS clients and servers as well as between HTTPS servers and CT auditors/monitors is a JSON object [RFC7159] with the following content:

- sct_feedback: An array of objects consisting of
  - x509_chain: An array of base64-encoded X.509 certificates. The first element is the end-entity certificate, the second chains to the first and so on.
  - sct_data: An array of objects consisting of
    - sct_version - Version as defined in [RFC6962] Section 3.2, as a number.
    - log_id - LogID as defined in [RFC6962] Section 3.2, as a base64 encoded string.
    - timestamp - The SCT timestamp, as a number.
    - extensions - CtExtensions as defined in [RFC6962] Section 3.2, as a base64 encoded string.
    - signature - The SCT signature, as a base64 encoded string.

The 'x509_chain' element MUST contain at least the leaf certificate and SHOULD contain the full chain to a known root.

3.2. STH gossip

The goal of gossiping about STHs is to detect logs that are presenting different (inconsistent) views of the log to different parties. CT auditors and monitors SHOULD gossip about Signed Tree Heads (STHs) with as many other auditors and monitors as possible.

[TBD gossip about inclusion proofs and consistency proofs too?]

Which STHs to share and how often gossip should happen is regarded as policy and out of scope for this document.

Auditors and monitors SHOULD provide the following URL accepting GET requests returning STHs:

https://<auditor-or-monitor>/ct/v1/sth-gossip

The data returned is a JSON object [RFC7159] with the following content:

o sth_gossip: An array of objects consisting of

  * sth_version - Version as defined in [RFC6962] Section 3.2, as a number. It's the version of the protocol to which the signature conforms.

  * tree_size: The size of the tree, in entries, as a number.

  * timestamp: The timestamp, as a number.

  * sha256_root_hash: The Merkle Tree Hash of the tree, as a base64 encoded string.

  * tree_head_signature: A TreeHeadSignature as defined in [RFC6962] Section 3.5 for the above data, as a base64 encoded string.

  * log_id - LogID as defined in [RFC6962] Section 3.2, as a base64 encoded string.

4. Security considerations

4.1. Privacy considerations

The most sensitive relationships in the CT ecosystem are the relationships between HTTPS clients and HTTPS servers. Client-server relationships can be aggregated into a network graph with potentially

serious implications for correlative de-anonymisation of clients and relationship-mapping or clustering of servers or of clients.

4.1.1. Privacy and SCTs

SCTs contain information that typically links it to a particular web site. Because the client-server relationship is sensitive, gossip between clients and servers about unrelated SCTs is risky.

Therefore, a client with an SCT for a given server should transmit that information in only two channels: to a server associated with the SCT itself; and to a trusted CT auditor, if one exists.

4.1.2. Privacy in SCT feedback

HTTPS clients which allow users to clear history or cookies associated with an origin MUST clear stored SCTs associated with the origin as well.

Auditors should treat all SCTs as sensitive data. SCTs received directly from an HTTPS client are especially sensitive, since the auditor is a trusted by the client to not reveal their associations with servers. Auditors MUST NOT share such SCTs in any way, including sending them to an external log, without first mixing them with multiple other SCTs learned through submissions from multiple other clients. The details of mixing SCTs are TBD.

There is a possible fingerprinting attack where a log issues a unique SCT for targeted log client(s). A colluding log and HTTPS server operator could therefore be a threat to the privacy of an HTTPS client. Given all the other opportunities for HTTPS servers to fingerprint clients - TLS session tickets, HPKP and HSTS headers, HTTP Cookies, etc. - this is acceptable.

The fingerprinting attack described above could be avoided by requiring that logs i) MUST return the same SCT for a given cert chain ([RFC6962] Section 3) and ii) use a deterministic signature scheme when signing the SCT ([RFC6962] Section 2.1.4).

4.1.3. Privacy in STH gossip

Nowhere in this document is it suggested that HTTPS clients deal with STHs but for completeness here's a privacy analysis for STHs. An STH linked to a client indicates the following about that client: - that the client gossips - that the client been using CT at least until the time that the timestamp and the tree size indicate - that the client is talking, possibly indirectly, to the log indicated by the tree hash - which software and software version is being used.

There is a possible fingerprinting attack where a log issues a unique

STH for targeted log auditor(s). This is similar to the fingerprinting attack described in Section 4.1.2, but it is mitigated by the following factors:

- the relationship between auditors and logs is not sensitive in the way that the relationship between clients and servers is.
- because auditors regularly exchange STHs with each other, the re-appearance of a targeted STH from some auditor does not imply that the auditor was the original one targeted by the log.

5. IANA considerations

TBD

6. Contributors

The authors would like to thank Tom Ritter and Magnus Ahltorp for valuable contributions.

7. ChangeLog

7.1. Changes between -00 and -01

- Add the SCT feedback mechanism: Clients send SCTs to originating web server which shares them with auditors.
- Stop assuming that clients see STHs.
- Don't use HTTP headers but instead .well-known URL's - avoid that battle.
- Stop referring to trans-gossip and trans-gossip-transport-https - too complicated.
- Remove all protocols but HTTPS in order to simplify - let's come back and add more later.
- Add more reasoning about privacy.
- Do specify data formats.

8. Normative References

[RFC6962] Laurie, B., Langley, A., and E. Kasper, "Certificate Transparency", RFC 6962, June 2013.

[RFC7159] Bray, T., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, March 2014.

Authors' Addresses

Linus Nordberg
NORDUnet

Email: linus@nordu.net

Daniel Kahn Gillmor
ACLU

Email: dkg@fifthhorseman.net