Abstract

A malicious Certificate Transparency (CT) log can offer a modified tree to a client in a "split view" attack. This document proposes to extend CT by submitting Signed Tree Heads (STH) into another log, run by a different operator. Auditors and monitors can use these cross logged STHs to detect split view attacks by the log.

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 https://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 May 18, 2018.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.
1. Introduction

A possible attack for Merkle tree-based logs are "split view" attacks. In such an attack, the log presents a modified version of the tree to a subset of its client base. The modified version violates the rules for correct log operation by omitting, inserting, or manipulating a log entry. This powerful attack warrents an appropriate defense.

The CT Threat Analysis [I-D.ietf-trans-threat-analysis] recognises the need to address split view attacks. This is also acknowledged in Section 11.3. of CTv2 [I-D.ietf-trans-rfc6962-bis].

The protection mechanism described in this document extends the functionality of the CT log, auditor, and monitor to provide detection of split view attacks. By piggybacking on other logs, it allows the participants to verify that they all are presented with the same view on the log’s Merkle tree.

One comparable mechanism in CT is Gossip [I-D.ietf-trans-gossip]. This protocol uses TLS servers as exchange mechanism for the view on the log. The security of the gossip mechanisms is analysed in [Chase16] and [Chuat15].

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].
2. Architecture

The threat model addressed here consists of a log executing a split view attack. Any second log used is assumed to be benevolent. The defense is considered successful, if the attack can be detected.

A log is "bound" to a version of its Merkle tree by submitting a Signed Tree Head (STH) into another log, dubbed "witnessing" log. This process is called "cross logging". The bound log cannot manipulate the tree of the witnessing log. It is committed to the version of the tree represented by the submitted STH, because publishing an inconsistent STH can be detected by requesting a consistency proof.

An auditor establishes the _presence_ of a given STH in the witnessing log. It further verifies its consistency with another known STH. Note that this does not yet ensure that only consistent STHs were cross logged.

The monitor verifies the _consistency_ of all logged STHs with each other. To this end, it observes all entries of both the bound log and the witnessing log. For the entries in the witnessing log that are STHs of the bound log, it establishes that these are consistent by computing or querying for consistency proofs.

Which witnessing logs are used is a matter of policy requiring agreement between bound log, witnessing log, and clients.

3. STH submission

3.1. Bound log

The submission of the STH MUST take place at a fixed frequency which is checked by the monitor. The submission is done by the bound log. Failure to submit in time is considered attributable misbehaviour.

The public key of the bound log must be a trust anchor of the witnessing log.

The log API is extended by an additional call offering an STH of the bound log together with inclusion promises, Signed Certificate Timestamps (SCT), issued by the witnessing log.

GET https://<log server>/ct/v2/get-sth-sct

No inputs.

Outputs:
sth: A base64 encoded "TransItem" of type "signed_tree_head_v2", signed by this log, that is no older than the log’s Maximum Merge Delay (MMD).

sct: An array of zero or more base64 encoded "TransItem" of type "x509_sct_v2" corresponding to "sth".

[[ TBD: define separate SCT type? Move into "get-sth" for herd immunity? ]]

3.2. Witnessing log

The log submission API described in Section 5.1. of CTv2 [I-D.ietf-trans-rfc6962-bis] is extended to accept inputs of type "signed_tree_head_v2". The chain in this case has length zero.

The computation of the SCT as described in Section 4.8. of CTv2 [I-D.ietf-trans-rfc6962-bis] is extended. The signature may now also be computed over a "TransItem" structure of type "signed_tree_head_v2".

4. Auditor

The auditor verifies that its own view on the bound log is present in the witnessing log. This extends checking the append-only property and consistency presented to all query sources defined in Section 8.3 of CTv2 [I-D.ietf-trans-rfc6962-bis].

When retrieving an STH from the bound log, the auditor opts for the variant supplying cross logging SCTs. Other than that, auditing of the bound log is done per usual.

The witnessing log is also audited for the append-only property. The inclusion of the bound log STHs into the witnessing log is verified by using the SCT retrieved with them.

Note that the MMD of the bound and witnessing logs accumulate with respect to verifying inclusion of an entry in the bound log. In case of multiple witnessing logs, the MMD of the slowest witnessing log applies.

5. Monitor

The monitor follows the steps in Section 8.2. of CTv2 [I-D.ietf-trans-rfc6962-bis] per usual. Step 4 allows for additional checks on log entries.
The following procedure constitutes an additional check and is performed as step 4:

1. If the entry is not an STH of the bound log, stop processing this entry.

2. Verify the signature. If the entry does not have a valid signature, stop processing this entry.

3. Verify consistency of the STH with the bound log by requesting a consistency proof or recomputing the hash using the log entries available locally.

To make it impossible for the bound log to just stop cross logging when executing a split view attack, another check is required. At fixed time intervals, e.g., the minimum STH publishing interval of the bound log, the monitor MUST verify that the last observed STH entry is no older than the Maximum Merge Delay (MMD) of the bound log.

This check must incorporate a tolerance for the MMD of the witnessing log.

6. IANA Considerations

[[ TBD ]]”

7. Security Considerations

The system detects split view attacks, as long as at least one witnessing log or the bound log is not malicious. If all logs used for cross logging are malicious, the system provides no security.

8. References

8.1. Normative References

[I-D.ietf-trans-rfc6962-bis]

8.2. Informative References


Author’s Address

Benjamin Hof
Technical University of Munich

Email: hof@in.tum.de