An Introduction to System Transparency Logging

October 15, 2024

Rasmus Dahlberg

Outline
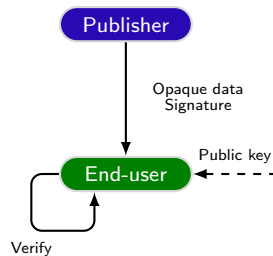
1. Setting and problem
2. Design overview
3. How to get involved
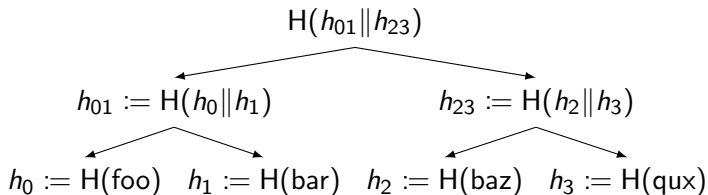
https://creativecommons.org/licenses/by-sa/4.0/

Our starting point

- Data publisher
- End-user
- Assumptions
  - ▶ Public key can be located
  - ▶ Signed data can be located
  - ▶ End-user can install extra tooling

The attacker can compromise the data publisher
The goal is to detect unwanted key-usage

A quick step back—Transparency log crash course

$$H(h_{01}\|h_{23})$$

$$h_{01} := H(h_0\|h_1) \qquad h_{23} := H(h_2\|h_3)$$

$$h_0 := H(\mathsf{foo}) \quad h_1 := H(\mathsf{bar}) \quad h_2 := H(\mathsf{baz}) \quad h_3 := H(\mathsf{qux})$$

- Tree head
- Consistency proof
- Inclusion proof

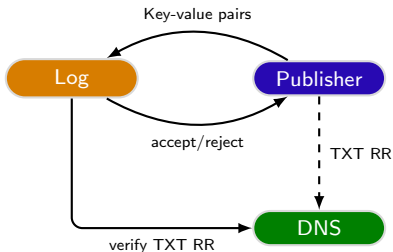The attacker can control the log

Preparing a logging request

- Select a shard hint and checksum
- Sign using your private key

```c
/*
 * The logged Merkle tree leaf data
 */
struct tree_leaf {
    u64 shard_hint;
    u8 checksum[32];
    u8 signature[64];
    u8 key_hash[32];
}
```
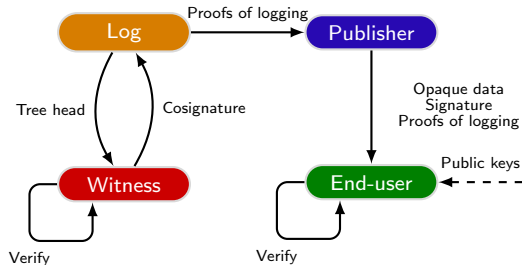
Submitting a logging request

Key-value pairs:

- Shard hint
- Checksum
- Signature
- Public key
- Domain hint

Distributing proofs of public logging

- End-user will not talk to the log
- Proofs of logging
  - ▶ Inclusion proof
  - ▶ Tree head
- Witness cosigning

The attacker can control a threshold of witnesses

Summary and additional details

- Signed checksums
- Sharding
- Preserved data flows
- Anti-spam
- Global consistency
- Few simple parsers
- No cryptographic agility

Get involved

- Feedback on our v0 design[1] and API[2]?

- Is this a service that you would use? Why (not)?

- Want to run an experimental log or witness?

- Implementation and tooling is still early-days

- Reach out via slack[3], IRC[4], GitHub, or DM

---

[1] https://github.com/system-transparency/stfe/blob/design/doc/design.md
[2] https://github.com/system-transparency/stfe/blob/design/doc/api.md
[3] https://communityinviter.com/apps/system-transparency/join
[4] irc/oftc #siglog