Signed Certificate Timestamp: A Never-Failing Promise?

Luis Wengenmair

Supervisor: Prof. Dr. Kevin Borgolte

Thesis Colloquium 10.11.2024

Content

- Introduction
- Problem
- Methodology
  - CT log on local machine
  - sctchecker
  - Own Python code
- Results
- What have we learned?

Introduction

- TLS is the de-facto standard to communicate encrypted
  - Digital certificates are needed
- Certificate Authorities give out certificates
- Digital certificates can be cryptographically verified through signatures
  - CAs themselves not
- Several misbehaving CAs showed need for measures

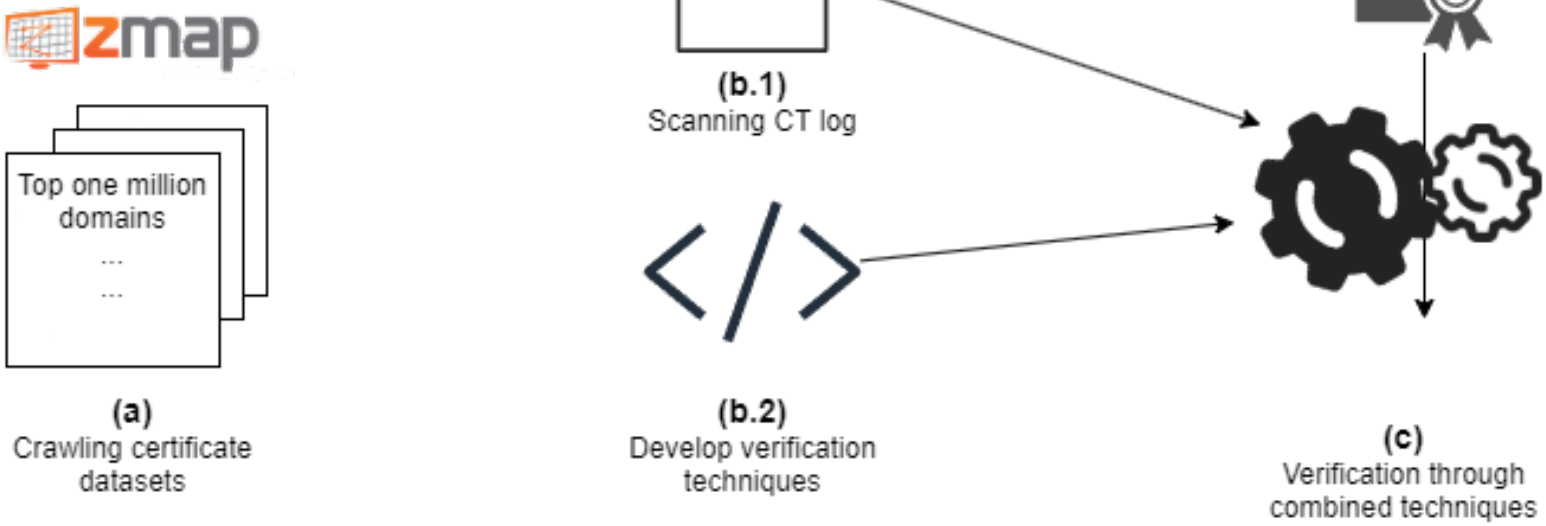
Introduction

- CT logs emerged as a solution to track CA-activities
  - Misissued certificates attract attention
- These logs are huge lists which consist of „Merkle Trees“
- Anyone can check for a given certificate if it is included in a log using the API

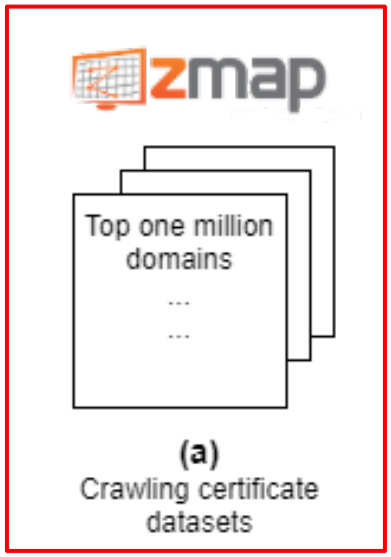
Problem

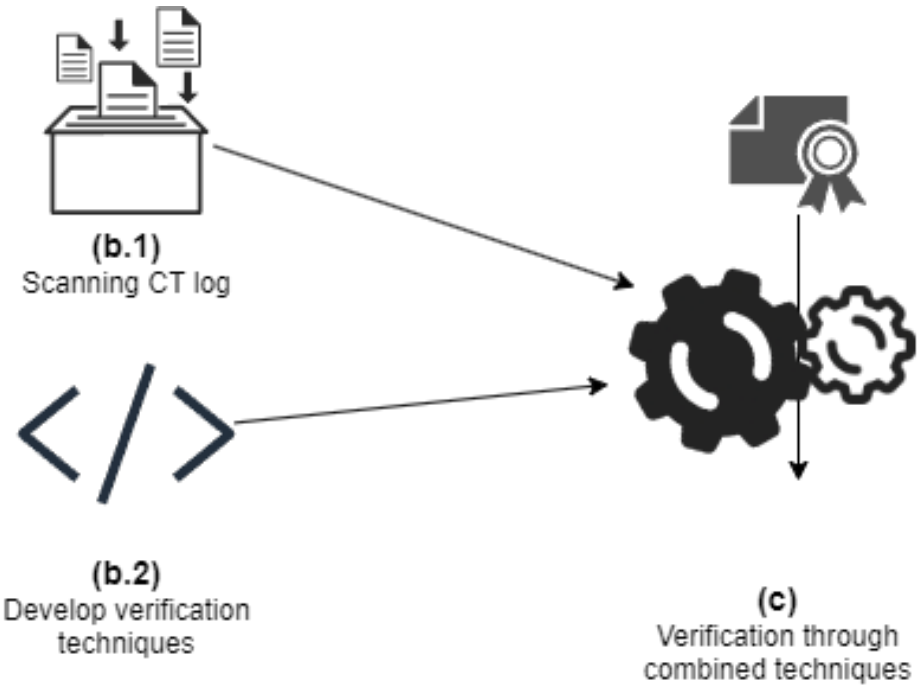
- Instant inclusion into CT logs is not always possible
- „Promises“ for inclusion (so called Signed Certificate Timestamps) are given out by CT log providers
- There exist no research regarding the reliance of SCTs
  - It is not tested on a large-scale whether the existence not only promises the inclusion but also proofs it
- In this thesis we want to verify for a large number of certificates if the promise of inclusion is kept

Methodology



Methodology





IPv4-Scan

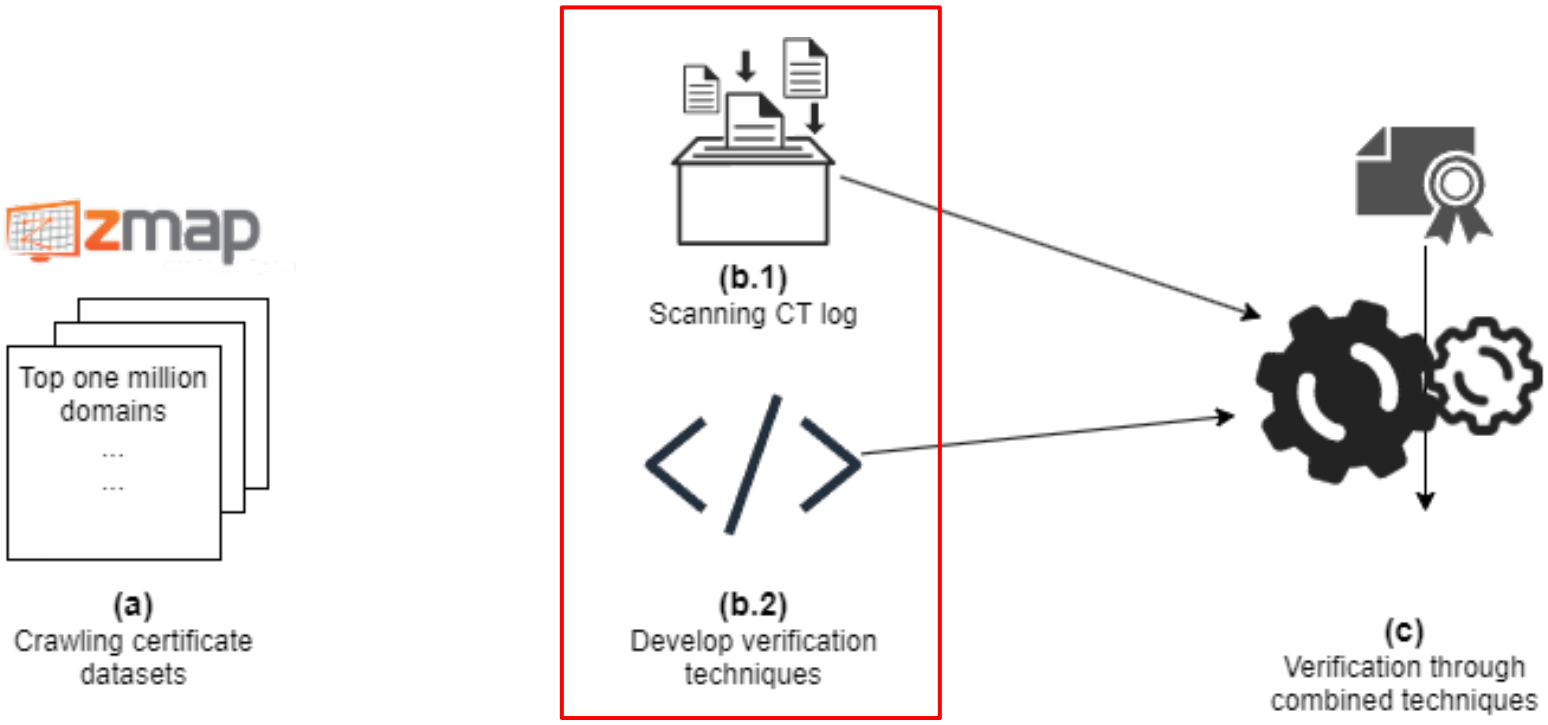
RUB

- Utilization of the ZMap-Project
- ZMap to identify all hosts in the IPv4 space with Port 443 not closed
- With ZGrab try to build up TLS-connection with those hosts
  - download the corresponding digital certificate

Tranco

- Tranco is a „Research-Oriented Top Sites Ranking Hardened Against Manipulation”
  - List of the top one million domains
- Utilizing OpenSSL, download the digital certificate from all these websites

Methodology



CT log on local machine

- Tool from Google repository to scan one whole CT log
- Verification process for one certificate is exceptionally fast
- Preceding extensive time and space requirements

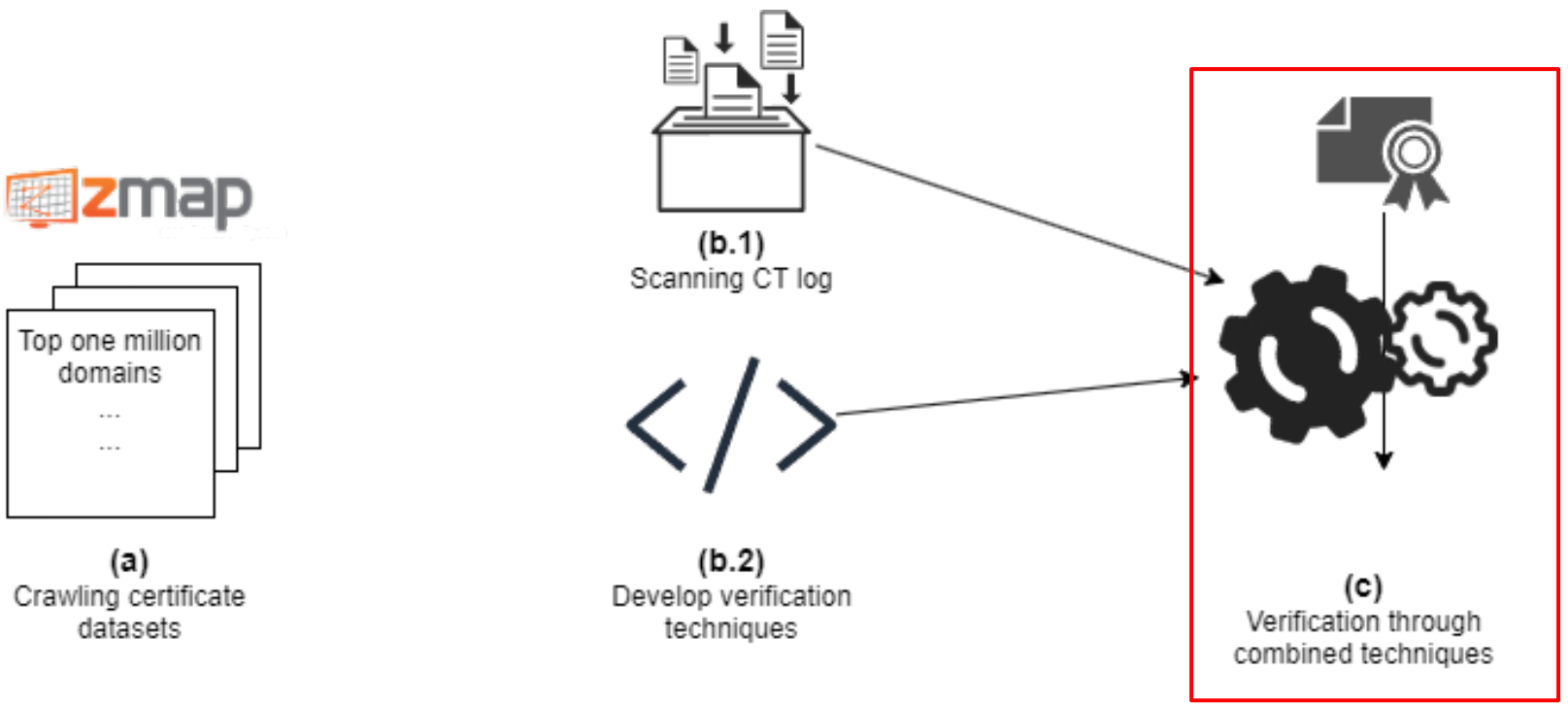
sctchecker

- Tool from Google for verifying the validity of SCTs embedded in a given certificate
- Extensive and sophisticated code base
- Output needs to be further processed
- Relatively slow

Own Python code

- Utilization of different libraries to request proof of inclusion
  - also able to utilize the local CT log
- Exactly developed for our own demand
  - Statistics and minimal function
- Has not been tested extensively
  - Highly likely to not cover every edge case
  - May have (severe) bugs

Methodology



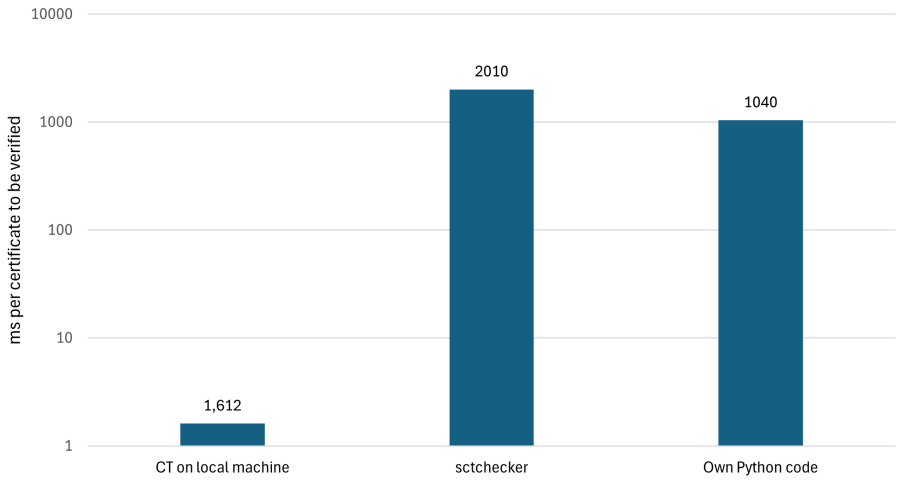
Combining techniques

- We utilize the advantages of each of the techniques
  - reducing the individual disadvantages
- First, use own Python code as well as the local CT log
- Certificates, not successfully verified in first instance, are given to the sctchecker tool from Google

Results

Performance

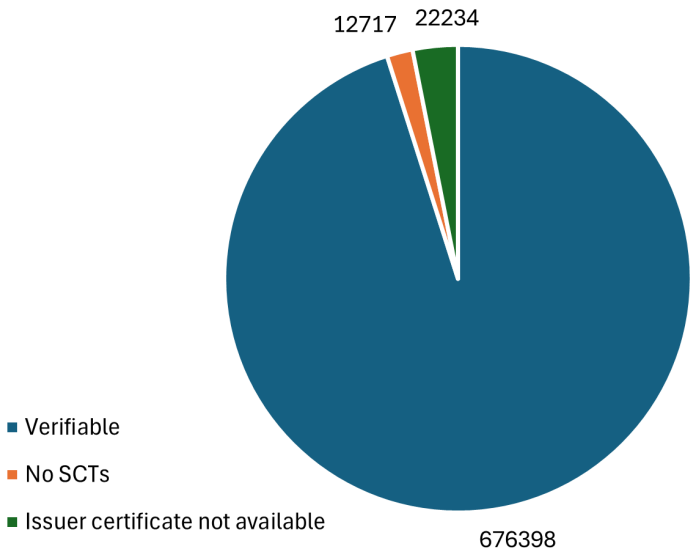
- To showcase the efficiency of our pipeline, we present the average time needed to verify one single certificate
- Python code (only with API) takes about one second
- sctchecker takes about two seconds
- Local CT log takes less then two milliseconds



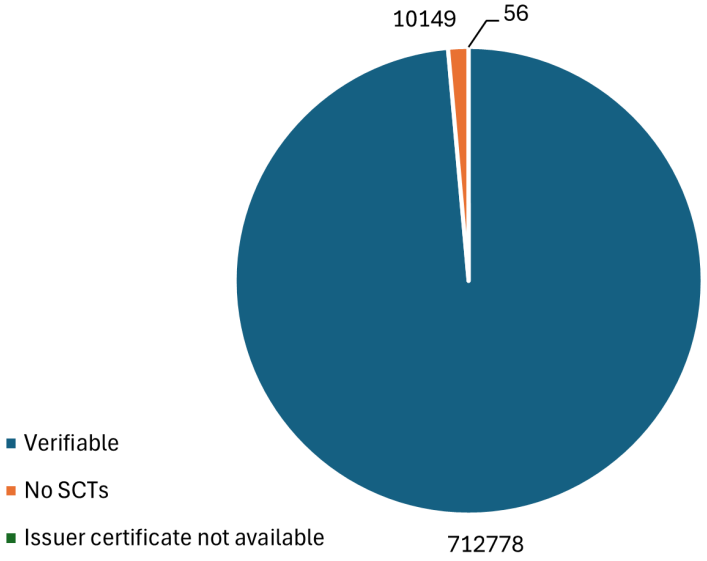
Results

Verifiable certificates

IPv4-scan (gathered 711349 in total)



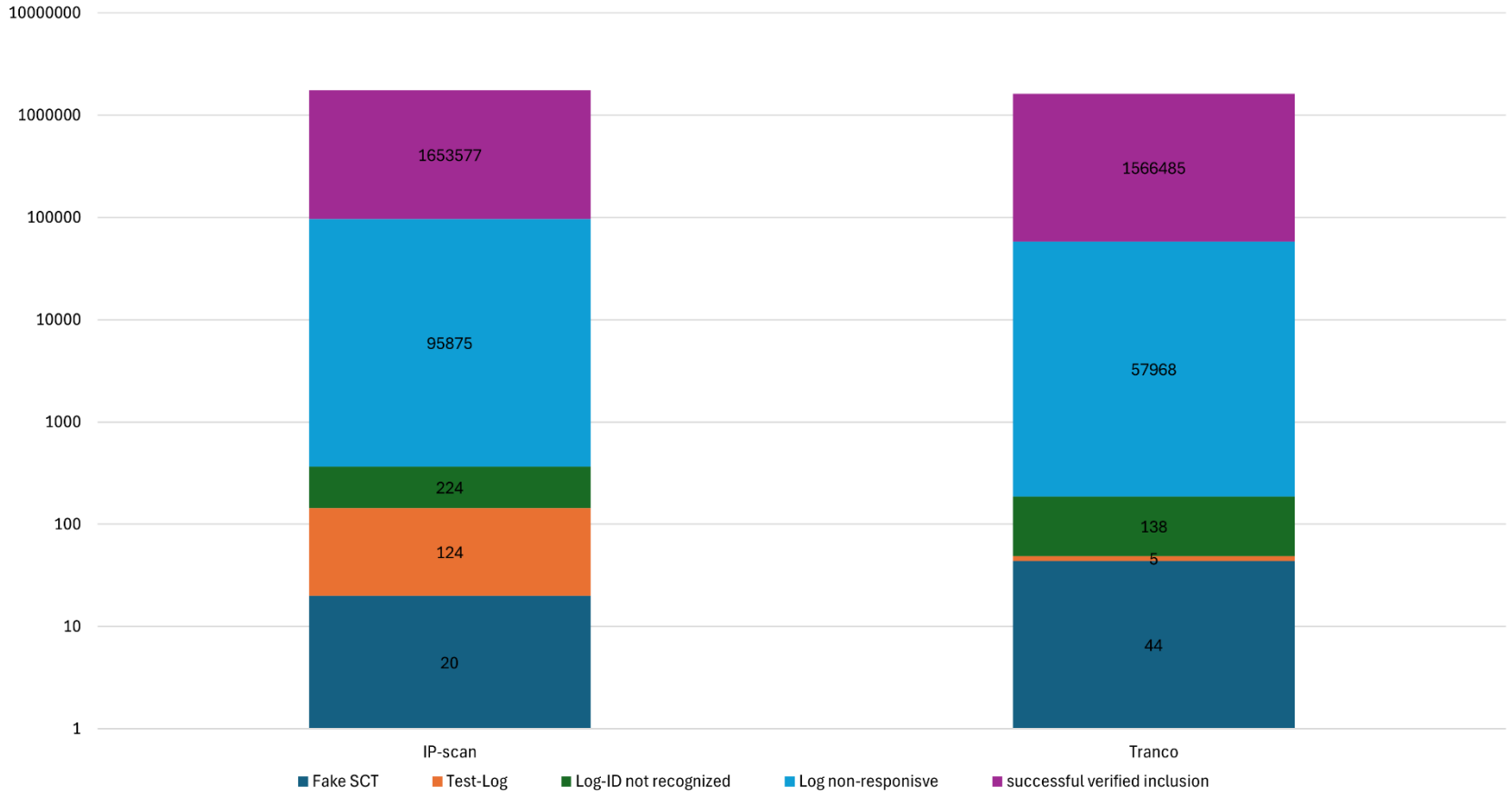
Tranco (gathered 722983 in total)



Results

SCT statistics

- IP-Scan: ~94% successful verification
- Tranco: ~96% successful verification



What have we learned?

RUB

- We found no evidence for misbehaving or non-compliant SCTs
  - Most of the time, we were not able to verify the inclusion of one certificate, external factors played a role
- No definitive proof that SCTs are absolutely reliable
- Further research is needed
  - This can include some larger datasets to test
  - Could optimize verification by enhancing given code

What have we learned?

- Rate limiting and blacklisting makes it harder to audit on a large-scale
  - Rate limiting by CT Logs
  - Blacklisting by issuers' certificate provider
- Checking for delay of inclusion not possible after long period of time
  - Some monitors check timely inclusion

Thank You For Your Attention!
Any Questions?

Thesis Colloquium 10.11.2024

Performance

Amount of work

- All certificates are processed by our own Python code
  - optional: verify through local CT log
- Failed verification results in utilization of sctchecker
- IP-scan
  - ~20% local CT log
  - ~4% sctchecker
- Tranco
  - ~10% local CT log
  - ~2% sctchecker