Go or No Go: Differential Fuzzing of Native and C Libraries

Alessandro Sorniotti* Michael Weissbacher^ Anil Kurmus*

***IBM Research Europe – Zurich** **^Block, Inc.**

Motivation

- You are a Go developer writing an Anti-Virus (AV) engine
- The AV engine needs to decompress incoming files
- Which library to use?

C

Go or No Go?

- Conventional wisdom:
  - C library: Fast!
  - Go library: Memory safe!

- “Use Go for more security”?
- What about non-memory corruption vulnerabilities?

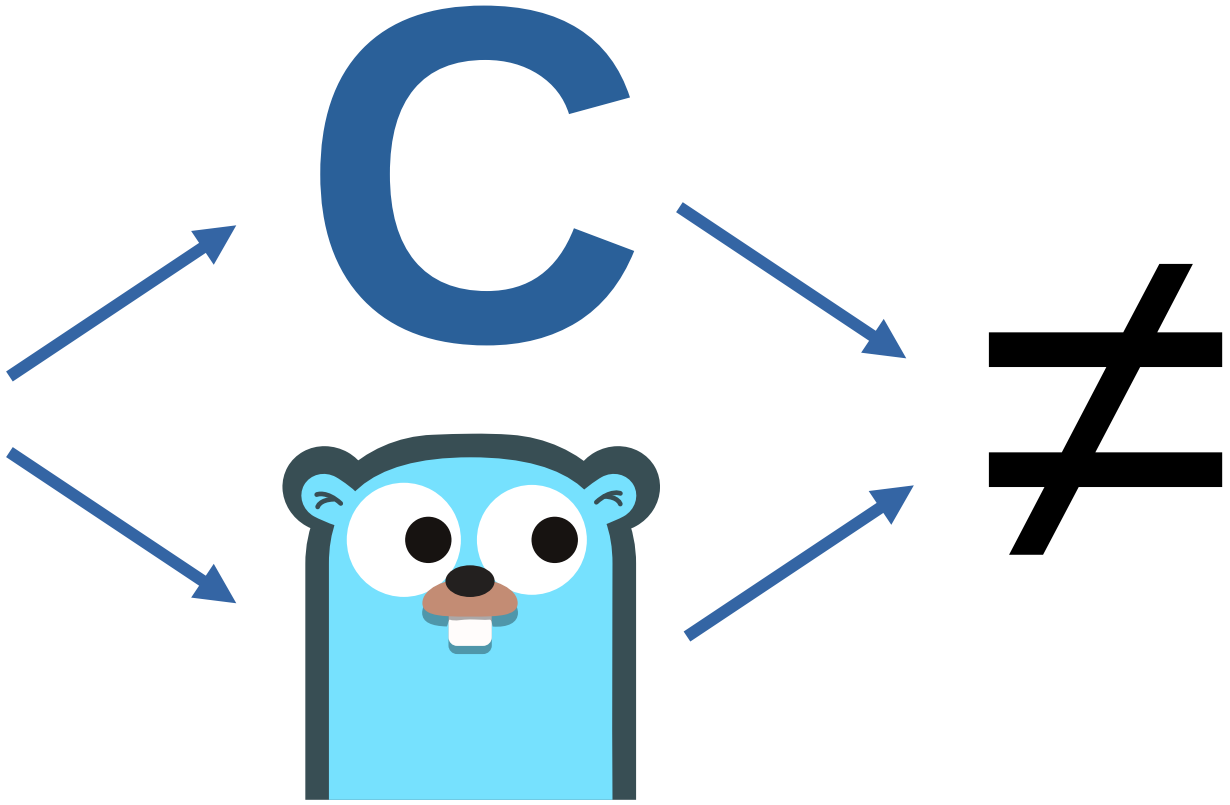
Parser tree differentials

- Two parsers for the same language may differ
- The differences may have security consequences
  - Parser may be overly liberal or strict in accepting input
  - Specification may be under-specified

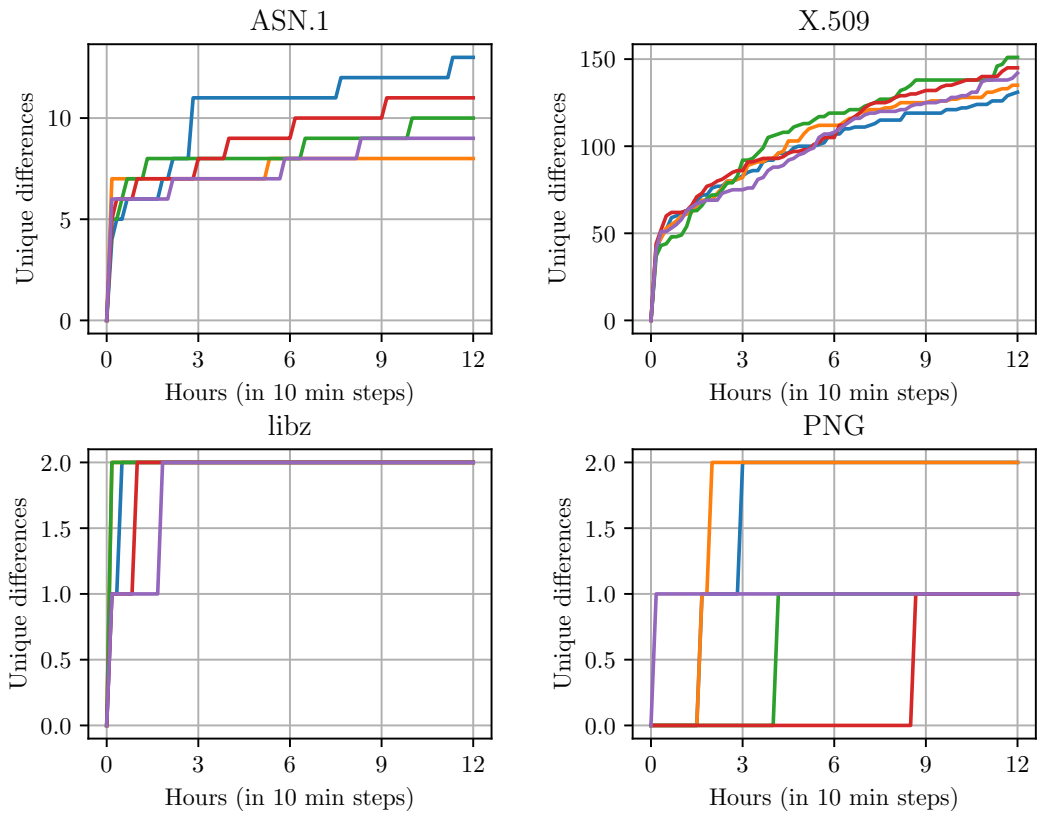
Differential fuzzing

Input

789c 8b30 f557 0c50…



Evaluation



- Library choice criteria:
  - Parsing
  - Security sensitive
  - Widely used C library
  - Easy to write harnesses
- Our differential fuzzer is effective
  - Less than ~1 hour to first bug

Use case: libz

Compliant:

789c 8b30 f557 0c50…

Go libz uncompress: OK

C zlib uncompress: OK

C

Mutated:

f81f 8b30 f557 0c50…

Go libz uncompress: OK

C zlib uncompress: Fail

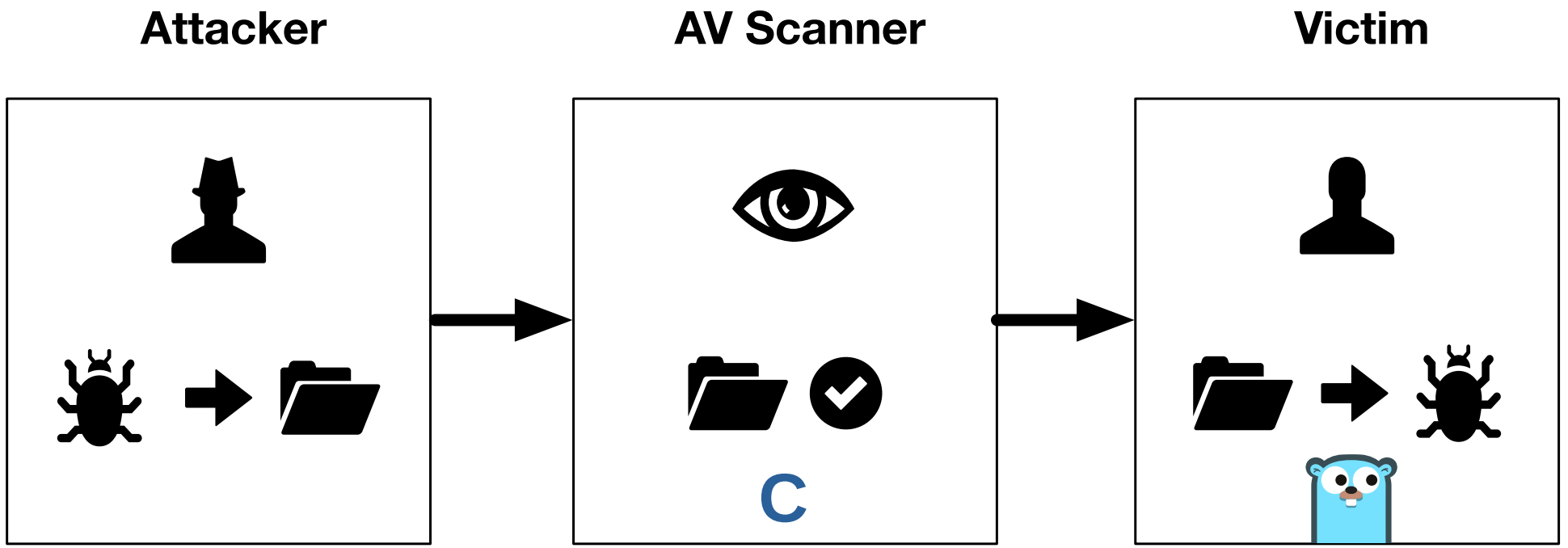
Security impact: libz

Compliant:

789c 8b30 f557 0c50…

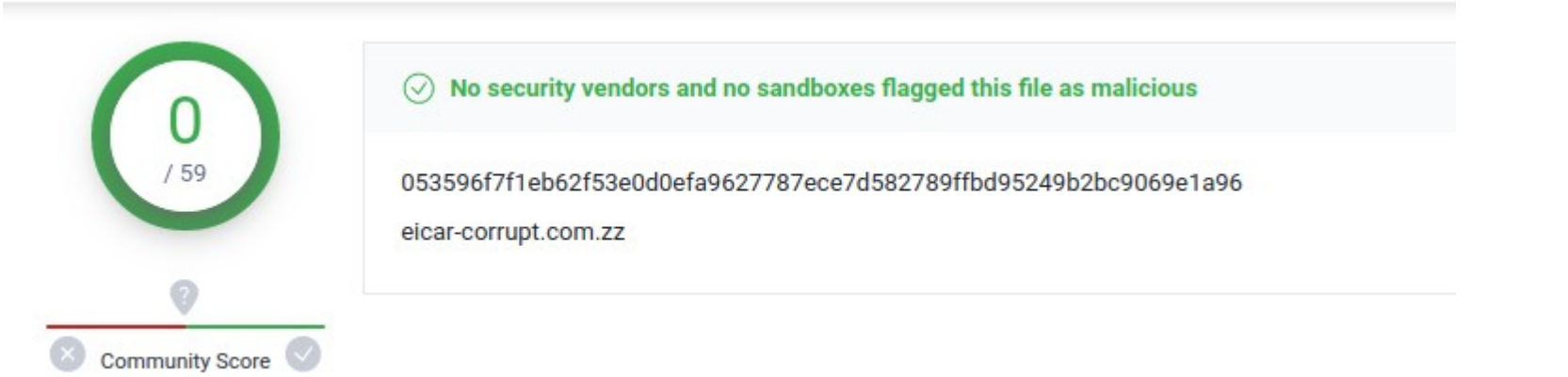
Mutated:

f81f 8b30 f557 0c50…

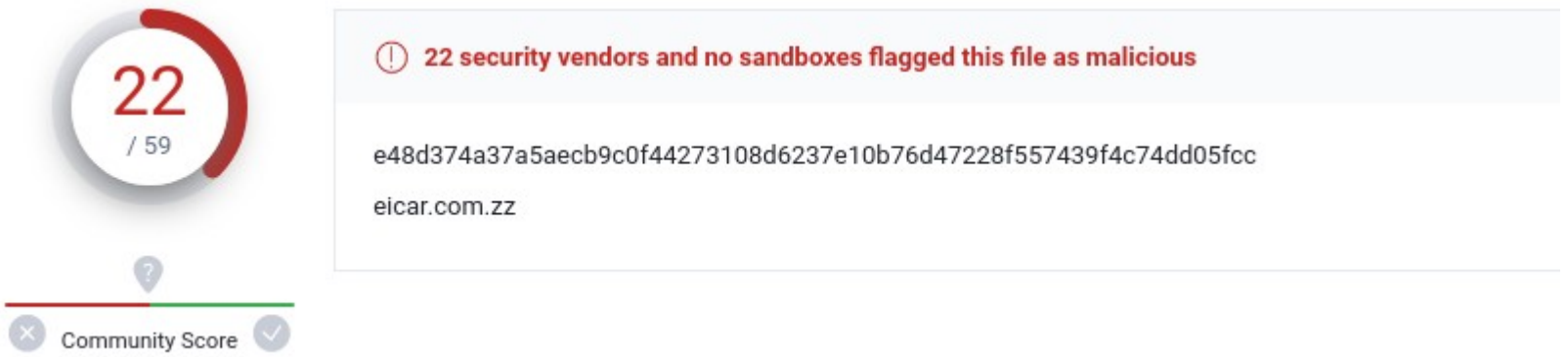


Bypassing AVs

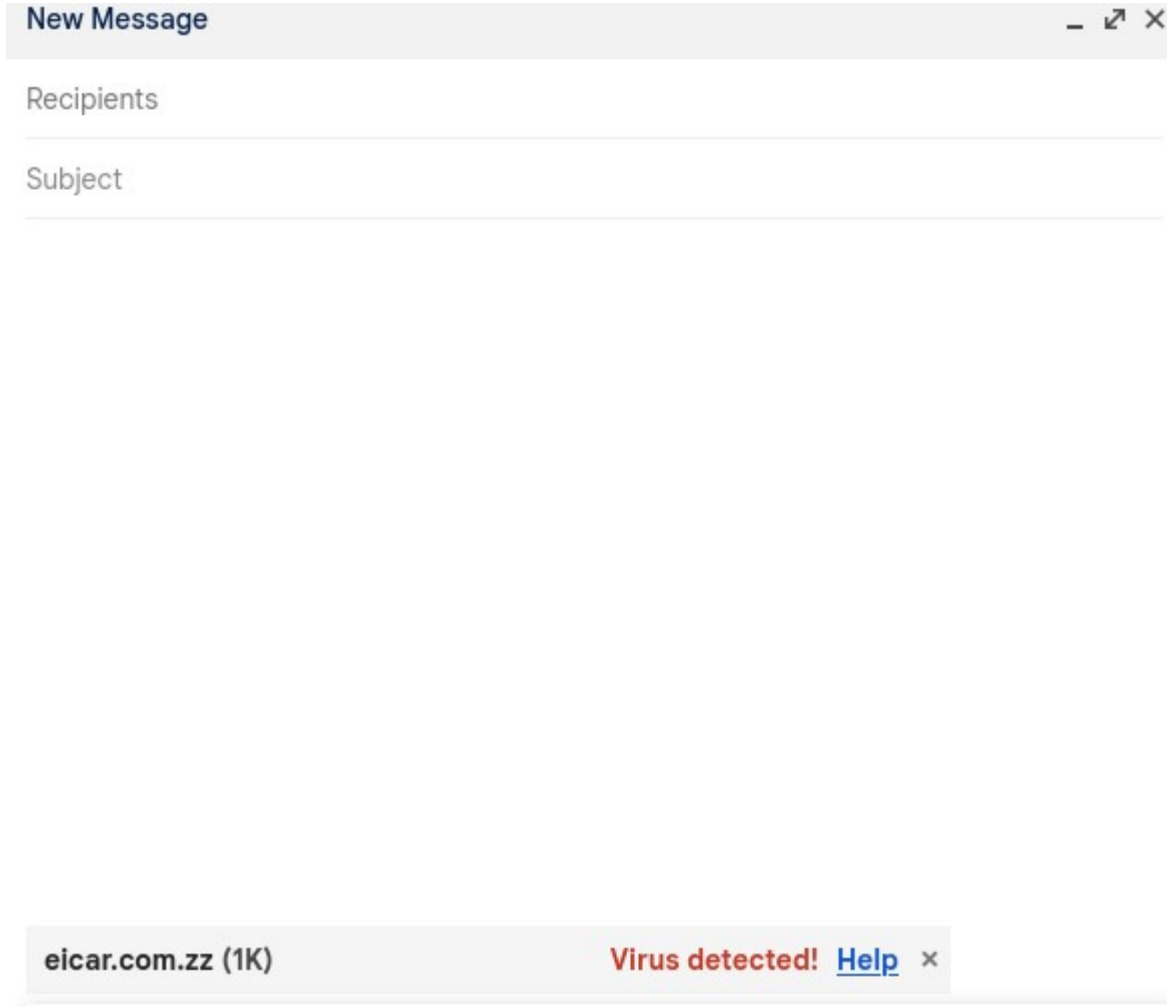
f81f 8b30 f557 0c50…



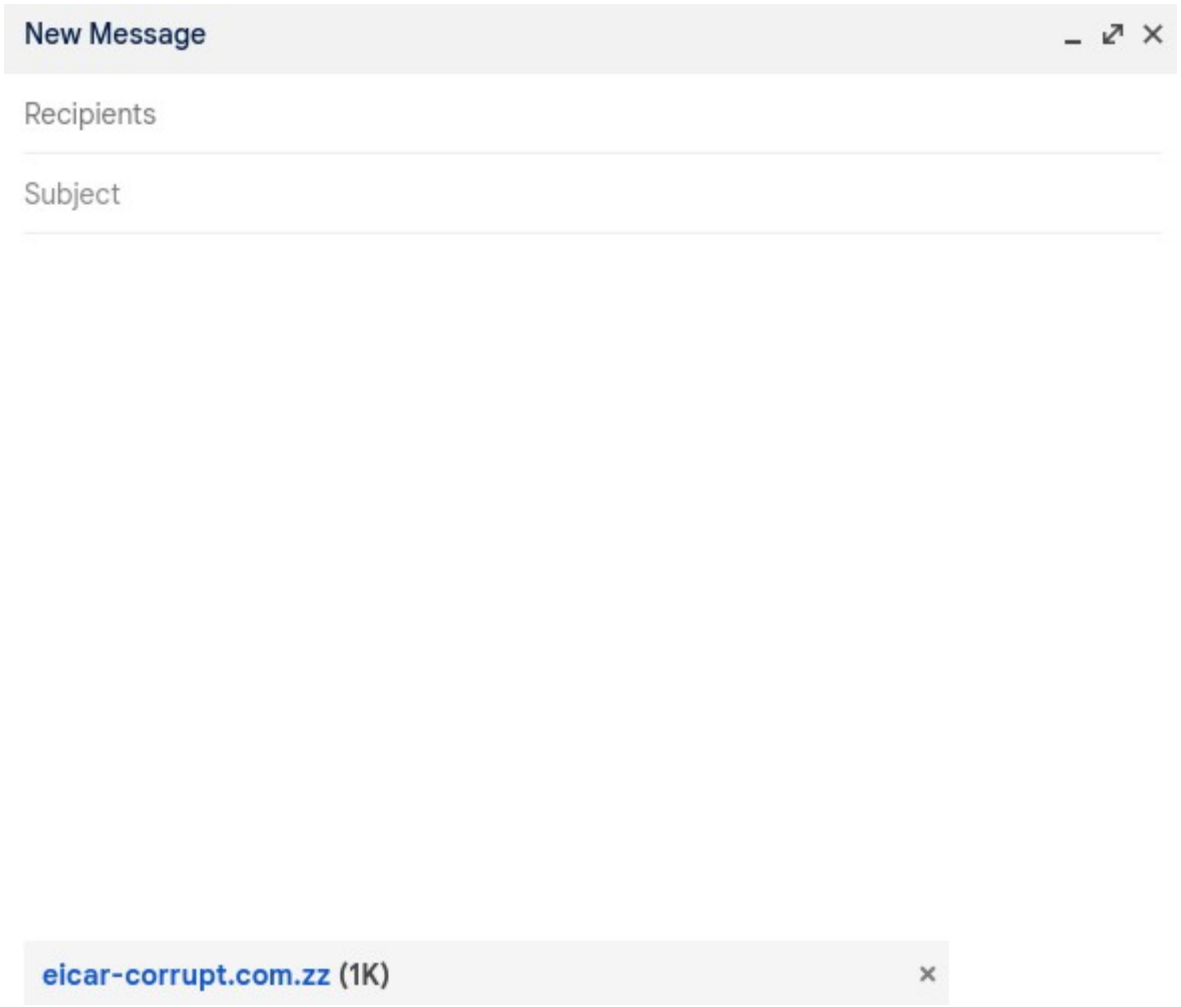
789c 8b30 f557 0c50…



Bypassing AVs



789c 8b30 f557 0c50…



f81f 8b30 f557 0c50…

Takeaways

As a library user:

- Native libraries are not necessarily "more secure"
- Danger in settings where two distinct parsers are used

As a library writer:

- Use differential fuzzing to find bugs in your code!

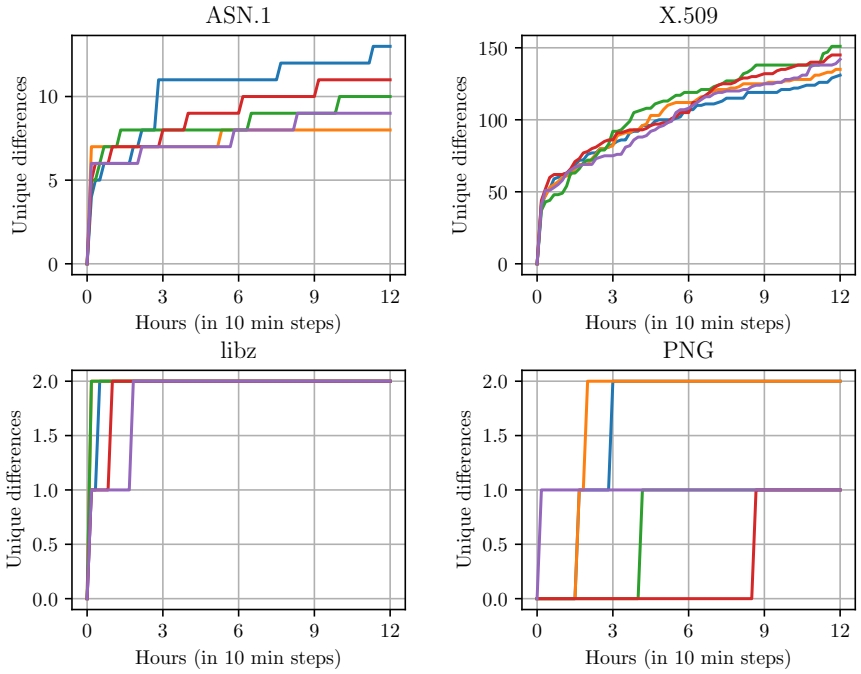
Additional results

- Other bug-inducing differences
- Certificate transparency use case
- Practical implementation challenges
- A study of prevalence of C/S code in Go programs

Check out the paper!

Go or No Go: Differential Fuzzing of Native and C Libraries

Alessandro Sorniotti, Michael Weissbacher, Anil Kurmus



Evolution of fuzzer-found differences over time for four Go/C libraries.
Each color represents a distinct fuzzer run.

- Design and implementation of a **differential fuzzer** to uncover security relevant parsing differences in libraries used in Go programs.
- Evaluation on unique differences and analyse root cause and potential impact.
- Two case studies where discovered **parser differences lead to security impact**. One study affecting **Certificate Transparency** and one **bypass for 19 AV systems** tested on Virus Total, plus Gmail.

Get the paper here: