# Fuzzing results

Seyed Behnam Andarzian

Cristian Daniele

Erik Poll

Digital Security group

Radboud University Nijmegen

# Fuzzing – case studies

| | SUT | input formats |
|---|---|---|
| 1 | Blurhash | jpeg, png, webp, gif |
| 2 | Gifdiff | gif |
| 3 | ExrTools | OpenEXR |
| 4 | OpenJPH | jpeg2000 |
| 5 | pngquant | png |
| 6 | Fleet-protocol | ??? |
| 7 | libemf2svg | EMF |
| 8 | pdf2htmlEx | pdf |
| 9 | nsxiv | image |
| 10 | cJSON | json |
| 11 | collapse | image |
| 12 | PDFio | pdf |
| 14 | WaveCollapseFunction | png |
| 16 | Godot | .godot .tcsn .escn .pack .tres |
| 17 | OpenJPG | jpeg |
| 20 | Poppler | pdf |
| 22 | Gifsicle | gif |

## Fuzzing – tools used

| | SUT | tools |
|---|---|---|
| 1 | Blurhash | afl++, honggfuzz, radamsa |
| 2 | Gifdiff | afl++, honggfuzz, zzuf |
| 3 | ExrTools | afl++, honggfuzz, zzuf, radamsa |
| 4 | OpenJPH | afl++, honggfuzz, zzuf |
| 5 | pngquant | afl++, zzuf, honggfuzz, Radamsa |
| 6 | Fleet-protocol | afl++ with custom mutator |
| 7 | libemf2svg | afl++, honggfuzz, zzuf |
| 8 | pdf2htmlEx | afl++, honggfuzz, zzuf |
| 9 | nsxiv | afl++, formatfuzzer, honggfuzz |
| 10 | cJSON | afl++, radamsa, zzuf |
| 11 | collapse | afl++, honggfuzz, zzuf |
| 12 | PDFio | afl++, honggfuzz,zzuf |
| 14 | WaveCollapseFunction | afl++, honggfuzz |
| 16 | Godot | afl++, honggfuzz, zzuf, OSSfuzz |
| 17 | OpenJPG | afl++, honggfuzz, afl |
| 20 | Poppler | afl++, honggfuzz, zzuf, libfuzzer |
| 22 | Gifsicle | afl++, honggfuzz, zzuf |

## Fuzzing – results

| | SUT | tools | tools that found crashes |
|---|---|---|---|
| 1 | Blurhash | afl++, honggfuzz, radamsa | afl++, honggfuzz, radamsa? |
| 2 | Gifdiff | afl++, honggfuzz, zzuf | afl++, honggfuzz |
| 3 | ExrTools | afl++, honggfuzz, zzuf, radamsa | - |
| 4 | OpenJPH | afl++, honggfuzz, zzuf | afl++, honggfuzz, zzuf |
| 5 | pngquant | afl++, zzuf, honggfuzz, Radamsa | honggfuzz with MSan |
| 6 | Fleet-protocol | afl++ with custom mutator | - |
| 7 | libemf2svg | afl++, honggfuzz, zzuf | afl++, honggfuzz, zzuf |
| 8 | pdf2htmlEx | afl++, honggfuzz, zzuf | afl++, honggfuzz, zzuf |
| 9 | nsxiv | afl++, formatfuzzer, honggfuzz | afl++, format fuzzer |
| 10 | cJSON | afl++, radamsa, zzuf | - |
| 11 | collapse | afl++, honggfuzz, zzuf | afl++, honggfuzz |
| 12 | PDFio | afl++, honggfuzz,zzuf | zzuf |
| 14 | wcf | afl++, honggfuzz | afl++ , honggfuzz |
| 16 | Godot | afl++, honggfuzz, zzuf, OSSfuzz | afl++, honggfuzz, OSS-Fuzz |
| 17 | OpenJPG | afl++, honggfuzz, afl | afl++, honggfuzz, afl |
| 20 | Poppler | afl++, honggfuzz, zzuf, libfuzzer | - |
| 22 | Gifsicle | afl++, honggfuzz, zzuf | zuff |

## Fuzzing – new flaws found

| | SUT | input formats |
|---|---|---|
| 1 | Blurhash | |
| 2 | Gifdiff | |
| 3 | ExrTools | |
| 4 | OpenJPG | |
| 5 | pngquant | |
| 6 | Fleet-protocol | yes |
| 7 | libemf2svg | yes |
| 8 | pdf2htmlEx | yes |
| 9 | nsxiv | yes? |
| 10 | cJSON | |
| 11 | collapse | yes |
| 12 | PDFio | yes |
| 14 | WaveCollapseFunction | |
| 16 | Godot | |
| 17 | OpenJPG | yes? |
| 20 | Poppler | |
| 22 | Gifsicle | yes? |

# Overheads of ASan and MSan (group 5)

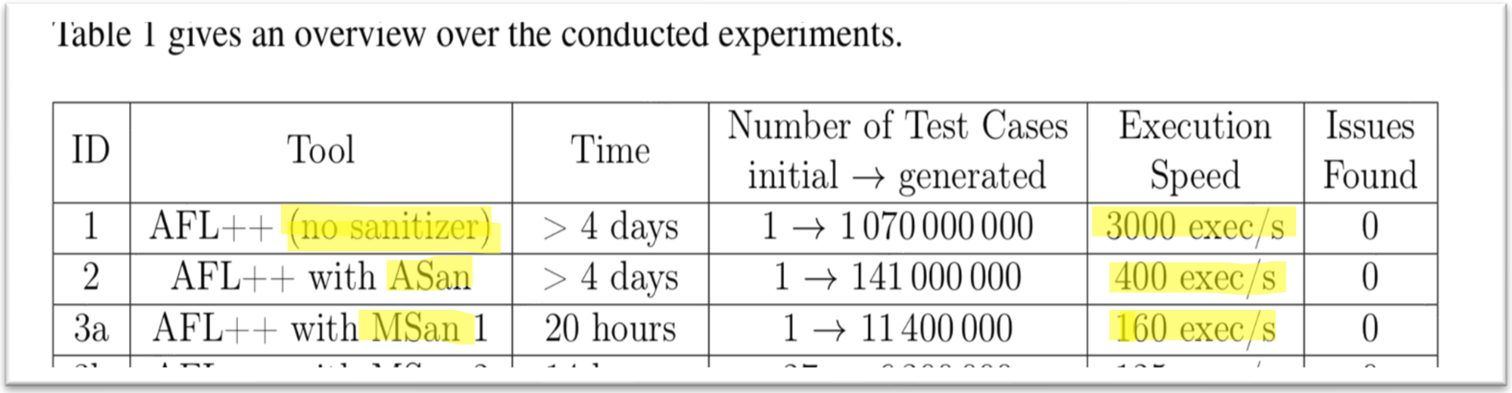


Table 1 gives an overview over the conducted experiments.

| ID | Tool | Time | Number of Test Cases initial → generated | Execution Speed | Issues Found |
|---|---|---|---|---|---|
| 1 | AFL++ (no sanitizer) | > 4 days | 1 → 1 070 000 000 | 3000 exec/s | 0 |
| 2 | AFL++ with ASan | > 4 days | 1 → 141 000 000 | 400 exec/s | 0 |
| 3a | AFL++ with MSan 1 | 20 hours | 1 → 11 400 000 | 160 exec/s | 0 |

## Unexpected outliers (group 5)

**honggfuzz with MSan finding bugs in less than a minute**

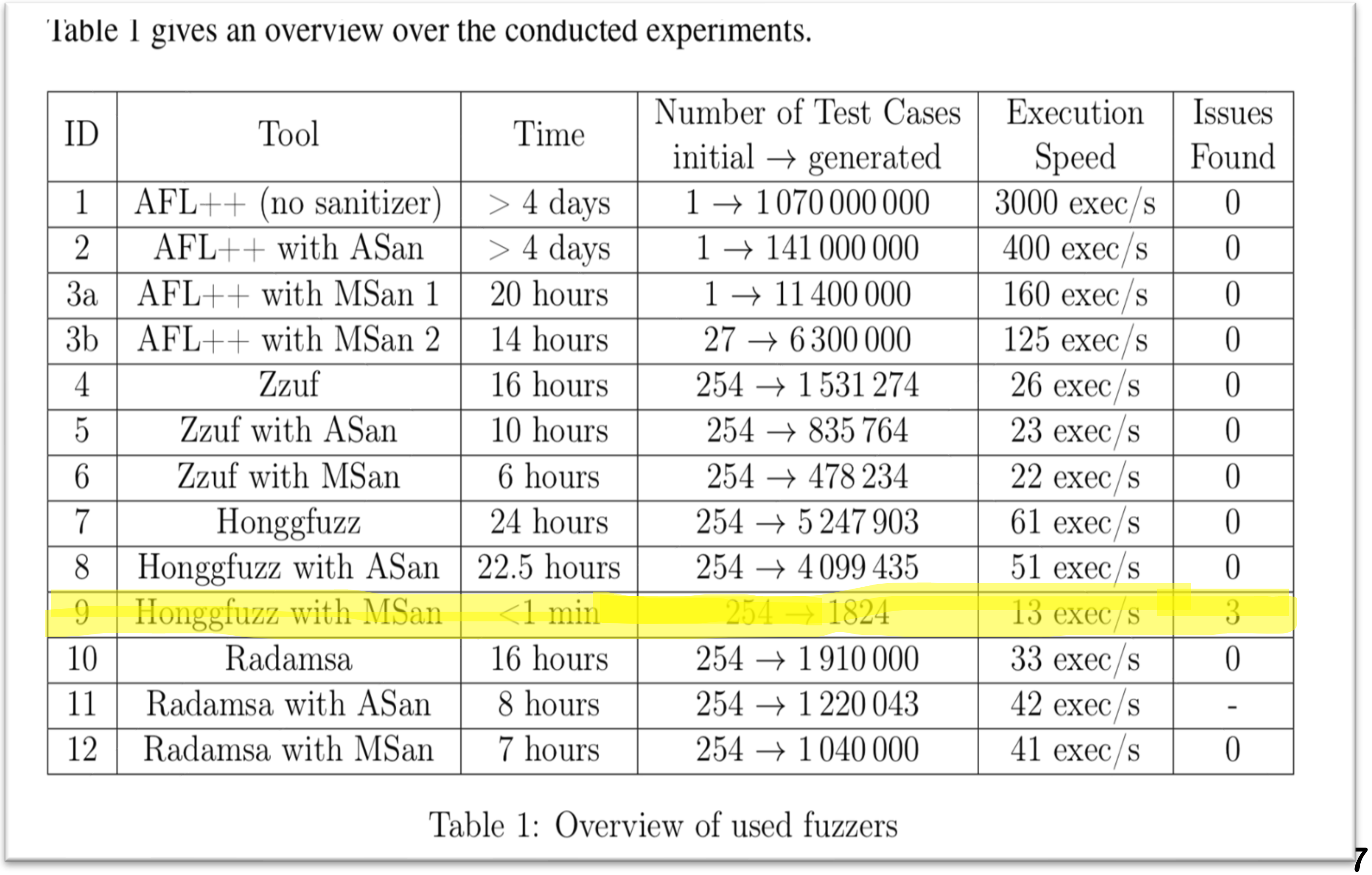


Table 1 gives an overview over the conducted experiments.

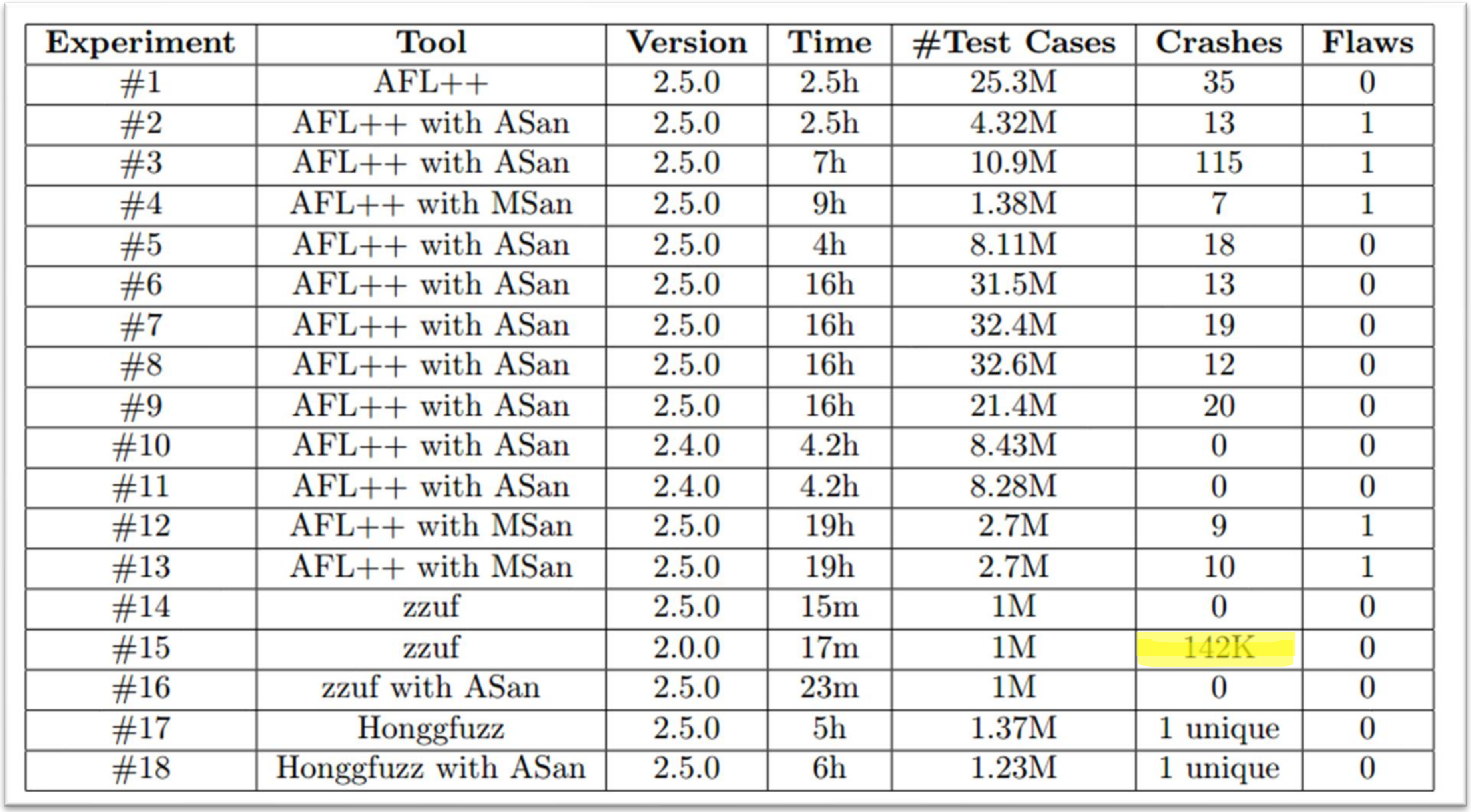
| ID | Tool | Time | Number of Test Cases initial → generated | Execution Speed | Issues Found |
|---|---|---|---|---|---|
| 1 | AFL++ (no sanitizer) | > 4 days | 1 → 1 070 000 000 | 3000 exec/s | 0 |
| 2 | AFL++ with ASan | > 4 days | 1 → 141 000 000 | 400 exec/s | 0 |
| 3a | AFL++ with MSan 1 | 20 hours | 1 → 11 400 000 | 160 exec/s | 0 |
| 3b | AFL++ with MSan 2 | 14 hours | 27 → 6 300 000 | 125 exec/s | 0 |
| 4 | Zzuf | 16 hours | 254 → 1 531 274 | 26 exec/s | 0 |
| 5 | Zzuf with ASan | 10 hours | 254 → 835 764 | 23 exec/s | 0 |
| 6 | Zzuf with MSan | 6 hours | 254 → 478 234 | 22 exec/s | 0 |
| 7 | Honggfuzz | 24 hours | 254 → 5 247 903 | 61 exec/s | 0 |
| 8 | Honggfuzz with ASan | 22.5 hours | 254 → 4 099 435 | 51 exec/s | 0 |
| 9 | Honggfuzz with MSan | <1 min | 254 → 1824 | 13 exec/s | 3 |
| 10 | Radamsa | 16 hours | 254 → 1 910 000 | 33 exec/s | 0 |
| 11 | Radamsa with ASan | 8 hours | 254 → 1 220 043 | 42 exec/s | - |
| 12 | Radamsa with MSan | 7 hours | 254 → 1 040 000 | 41 exec/s | 0 |

Table 1: Overview of used fuzzers

## Overheads of ASan & MSan (group 17)

| Experiment | Tool | Version | Time | #Test Cases | Crashes | Flaws |
|---|---|---|---|---|---|---|
| #1 | AFL++ | 2.5.0 | 2.5h | 25.3M | 35 | 0 |
| #2 | AFL++ with ASan | 2.5.0 | 2.5h | 4.32M | 13 | 1 |
| #3 | AFL++ with ASan | 2.5.0 | 7h | 10.9M | 115 | 1 |
| #4 | AFL++ with MSan | 2.5.0 | 9h | 1.38M | 7 | 1 |
| #5 | AFL++ with ASan | 2.5.0 | 4h | 8.11M | 18 | 0 |
| #6 | AFL++ with ASan | 2.5.0 | 16h | 31.5M | 13 | 0 |
| #7 | AFL++ with ASan | 2.5.0 | 16h | 32.4M | 19 | 0 |
| #8 | AFL++ with ASan | 2.5.0 | 16h | 32.6M | 12 | 0 |
| #9 | AFL++ with ASan | 2.5.0 | 16h | 21.4M | 20 | 0 |
| #10 | AFL++ with ASan | 2.4.0 | 4.2h | 8.43M | 0 | 0 |
| #11 | AFL++ with ASan | 2.4.0 | 4.2h | 8.28M | 0 | 0 |
| #12 | AFL++ with MSan | 2.5.0 | 19h | 2.7M | 9 | 1 |
| #13 | AFL++ with MSan | 2.5.0 | 19h | 2.7M | 10 | 1 |
| #14 | zzuf | 2.5.0 | 15m | 1M | 0 | 0 |
| #15 | zzuf | 2.0.0 | 17m | 1M | 142K | 0 |
| #16 | zzuf with ASan | 2.5.0 | 23m | 1M | 0 | 0 |
| #17 | Honggfuzz | 2.5.0 | 5h | 1.37M | 1 unique | 0 |
| #18 | Honggfuzz with ASan | 2.5.0 | 6h | 1.23M | 1 unique | 0 |

## Dumb fuzzers being dumb (group 17)



| Experiment | Tool | Version | Time | #Test Cases | Crashes | Flaws |
|---|---|---|---|---|---|---|
| #1 | AFL++ | 2.5.0 | 2.5h | 25.3M | 35 | 0 |
| #2 | AFL++ with ASan | 2.5.0 | 2.5h | 4.32M | 13 | 1 |
| #3 | AFL++ with ASan | 2.5.0 | 7h | 10.9M | 115 | 1 |
| #4 | AFL++ with MSan | 2.5.0 | 9h | 1.38M | 7 | 1 |
| #5 | AFL++ with ASan | 2.5.0 | 4h | 8.11M | 18 | 0 |
| #6 | AFL++ with ASan | 2.5.0 | 16h | 31.5M | 13 | 0 |
| #7 | AFL++ with ASan | 2.5.0 | 16h | 32.4M | 19 | 0 |
| #8 | AFL++ with ASan | 2.5.0 | 16h | 32.6M | 12 | 0 |
| #9 | AFL++ with ASan | 2.5.0 | 16h | 21.4M | 20 | 0 |
| #10 | AFL++ with ASan | 2.4.0 | 4.2h | 8.43M | 0 | 0 |
| #11 | AFL++ with ASan | 2.4.0 | 4.2h | 8.28M | 0 | 0 |
| #12 | AFL++ with MSan | 2.5.0 | 19h | 2.7M | 9 | 1 |
| #13 | AFL++ with MSan | 2.5.0 | 19h | 2.7M | 10 | 1 |
| #14 | zzuf | 2.5.0 | 15m | 1M | 0 | 0 |
| #15 | zzuf | 2.0.0 | 17m | 1M | 142K | 0 |
| #16 | zzuf with ASan | 2.5.0 | 23m | 1M | 0 | 0 |
| #17 | Honggfuzz | 2.5.0 | 5h | 1.37M | 1 unique | 0 |
| #18 | Honggfuzz with ASan | 2.5.0 | 6h | 1.23M | 1 unique | 0 |

## Uniqueness

***Does unique really mean unique?***

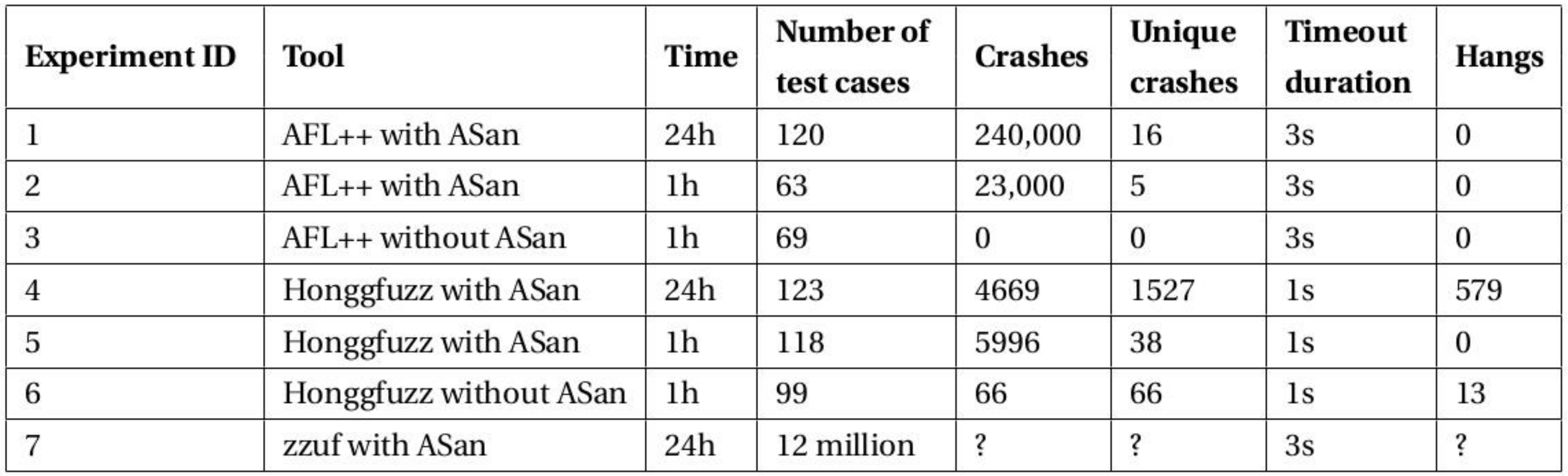
**Often not!**

## Hangs / time-outs

***Are hangs/time-outs really hangs?***

**Often not!**

# Time-outs (group 22)

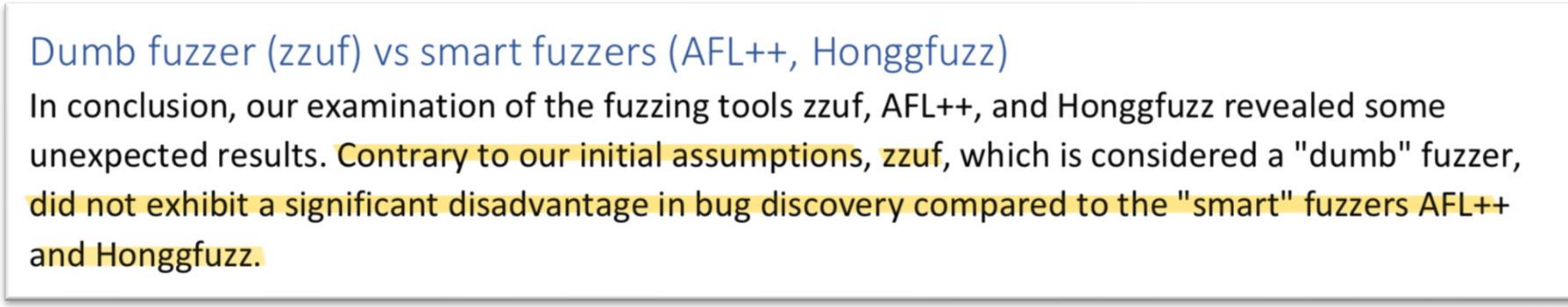


| Experiment ID | Tool | Time | Number of test cases | Crashes | Unique crashes | Timeout duration | Hangs |
|---|---|---|---|---|---|---|---|
| 1 | AFL++ with ASan | 24h | 120 | 240,000 | 16 | 3s | 0 |
| 2 | AFL++ with ASan | 1h | 63 | 23,000 | 5 | 3s | 0 |
| 3 | AFL++ without ASan | 1h | 69 | 0 | 0 | 3s | 0 |
| 4 | Honggfuzz with ASan | 24h | 123 | 4669 | 1527 | 1s | 579 |
| 5 | Honggfuzz with ASan | 1h | 118 | 5996 | 38 | 1s | 0 |
| 6 | Honggfuzz without ASan | 1h | 99 | 66 | 66 | 1s | 13 |
| 7 | zzuf with ASan | 24h | 12 million | ? | ? | 3s | ? |

**My guess: time-out too short for honggfuzz**

## zuff not worse than ‘smart’ fuzzers

group 4

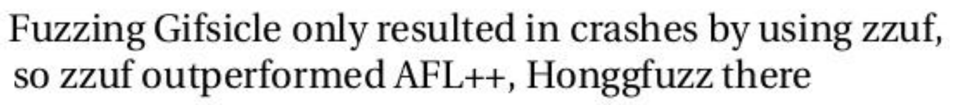


### Dumb fuzzer (zzuf) vs smart fuzzers (AFL++, Honggfuzz)

In conclusion, our examination of the fuzzing tools zzuf, AFL++, and Honggfuzz revealed some unexpected results. Contrary to our initial assumptions, zzuf, which is considered a "dumb" fuzzer, did not exhibit a significant disadvantage in bug discovery compared to the "smart" fuzzers AFL++ and Honggfuzz.

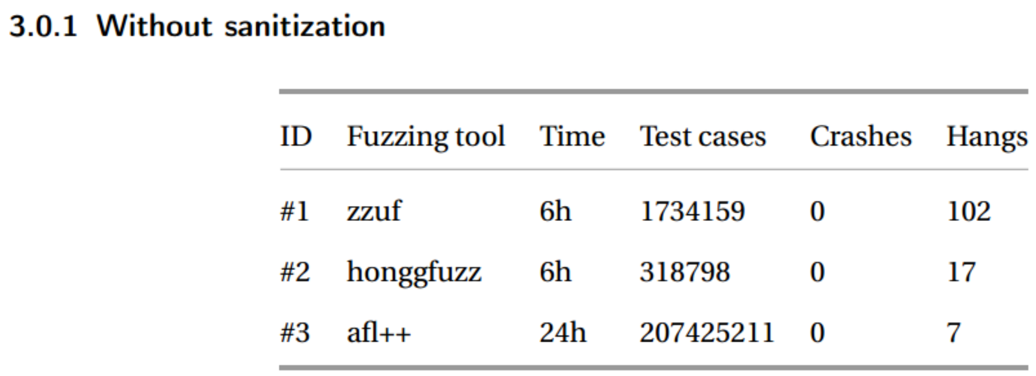
# zuff better than ‘smart’ fuzzers!

**group 22**

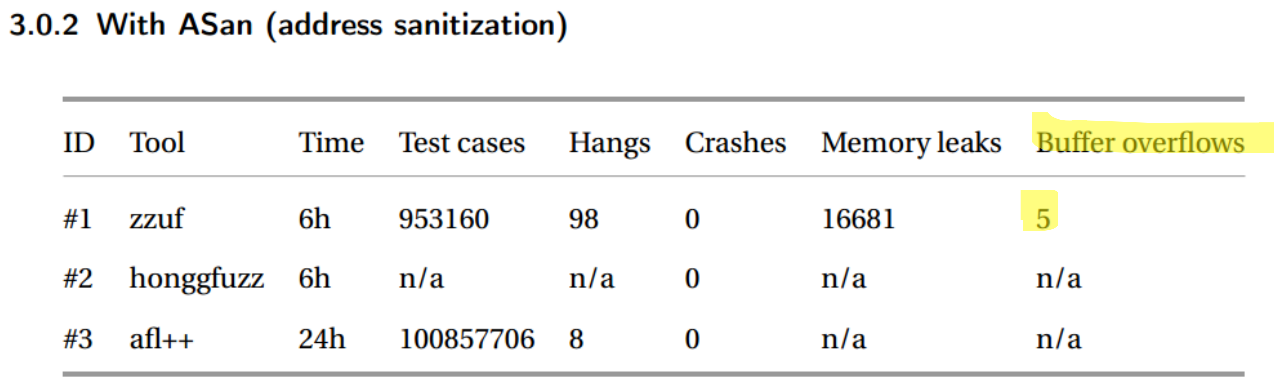

Fuzzing Gifsicle only resulted in crashes by using zzuf, so zzuf outperformed AFL++, Honggfuzz there

**but… different seed files were used**

**group 12**

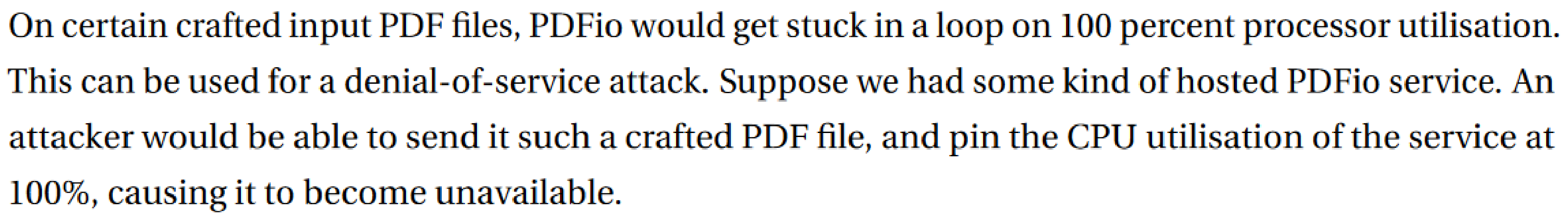

### 3.0.1 Without sanitization

| ID | Fuzzing tool | Time | Test cases | Crashes | Hangs |
|---|---|---|---|---|---|
| #1 | zzuf | 6h | 1734159 | 0 | 102 |
| #2 | honggfuzz | 6h | 318798 | 0 | 17 |
| #3 | afl++ | 24h | 207425211 | 0 | 7 |


### 3.0.2 With ASan (address sanitization)

| ID | Tool | Time | Test cases | Hangs | Crashes | Memory leaks | Buffer overflows |
|---|---|---|---|---|---|---|---|
| #1 | zzuf | 6h | 953160 | 98 | 0 | 16681 | 5 |
| #2 | honggfuzz | 6h | n/a | n/a | 0 | n/a | n/a |
| #3 | afl++ | 24h | 100857706 | 8 | 0 | n/a | n/a |

## Flaws programs vs inherently dangerous inputs?

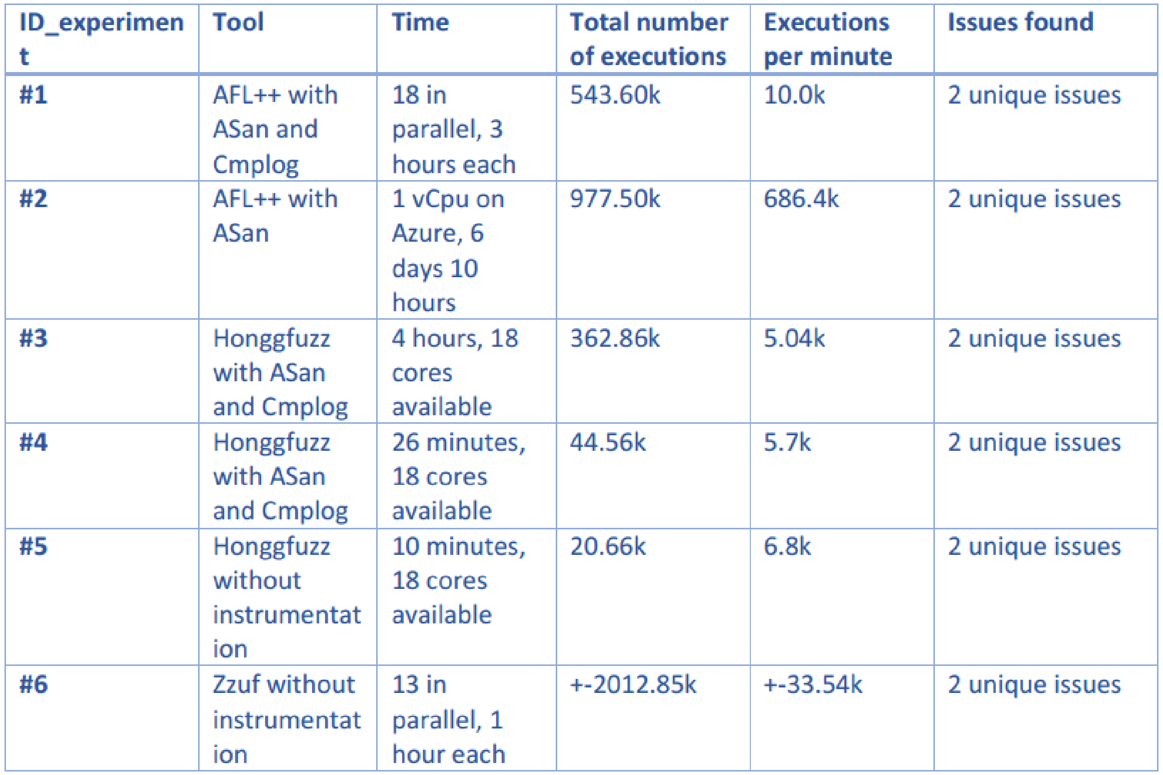

On certain crafted input PDF files, PDFio would get stuck in a loop on 100 percent processor utilisation. This can be used for a denial-of-service attack. Suppose we had some kind of hosted PDFio service. An attacker would be able to send it such a crafted PDF file, and pin the CPU utilisation of the service at 100%, causing it to become unavailable.

**Maybe these PDF files are inherently complex, and would cause problems for other PDF viewers too?**

## CmpLog (group 4)

**Alternative to using branch coverage to guide fuzzing:**

**CmpLog instruments code to record values used in comparisons & then using these values in mutations**



| ID_experiment | Tool | Time | Total number of executions | Executions per minute | Issues found |
|---|---|---|---|---|---|
| **#1** | AFL++ with ASan and Cmplog | 18 in parallel, 3 hours each | 543.60k | 10.0k | 2 unique issues |
| **#2** | AFL++ with ASan | 1 vCpu on Azure, 6 days 10 hours | 977.50k | 686.4k | 2 unique issues |
| **#3** | Honggfuzz with ASan and Cmplog | 4 hours, 18 cores available | 362.86k | 5.04k | 2 unique issues |
| **#4** | Honggfuzz with ASan and Cmplog | 26 minutes, 18 cores available | 44.56k | 5.7k | 2 unique issues |
| **#5** | Honggfuzz without instrumentation | 10 minutes, 18 cores available | 20.66k | 6.8k | 2 unique issues |
| **#6** | Zzuf without instrumentation | 13 in parallel, 1 hour each | +-2012.85k | +-33.54k | 2 unique issues |

## Beware of different goals of instrumentation

Instrumentation is used for two very different purposes in fuzzing:

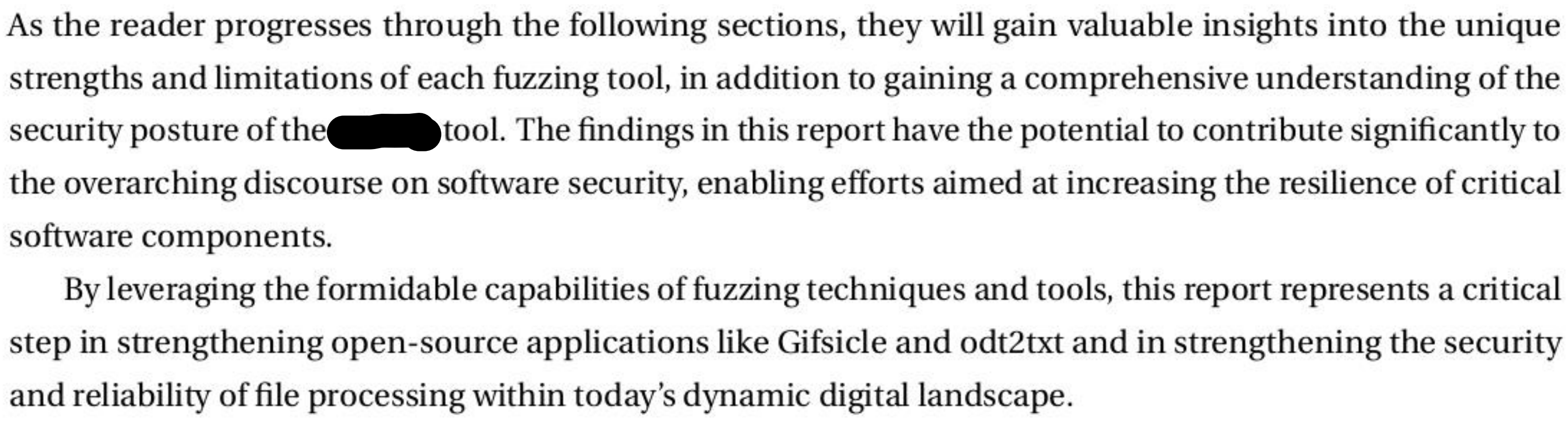
1) to provide feedback to guide the mutation process

   eg afl's standard instrumentation to observe branch coverage and CmpLog

2) to detect bugs

   eg the instrumentation added by sanitisers such as ASan, MSan, UBSan

# Watch your prose



As the reader progresses through the following sections, they will gain valuable insights into the unique strengths and limitations of each fuzzing tool, in addition to gaining a comprehensive understanding of the security posture of the [redacted] tool. The findings in this report have the potential to contribute significantly to the overarching discourse on software security, enabling efforts aimed at increasing the resilience of critical software components.

By leveraging the formidable capabilities of fuzzing techniques and tools, this report represents a critical step in strengthening open-source applications like Gifsicle and odt2txt and in strengthening the security and reliability of file processing within today's dynamic digital landscape.