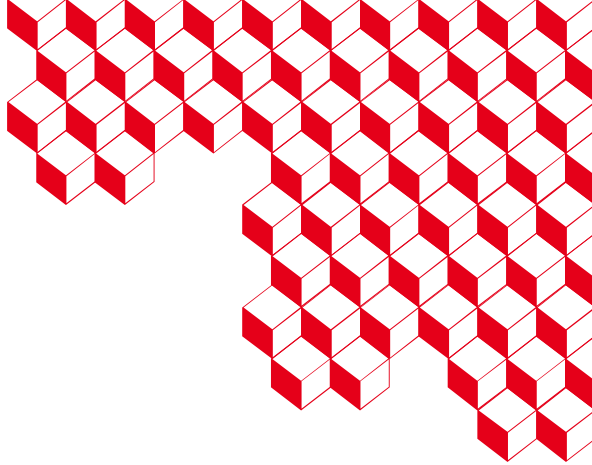
LCIS
Laboratoire de Conception et d'Intégration des Systèmes

UGA
Université Grenoble Alpes



Duskfuzz: Encoding Side-Channel information to improve blackbox fuzzing

Ulysse Vincenti (CEA/LCIS), Thomas Hiscock (CEA), David Hely (LCIS)

GOAL: Find bugs in the most systems!

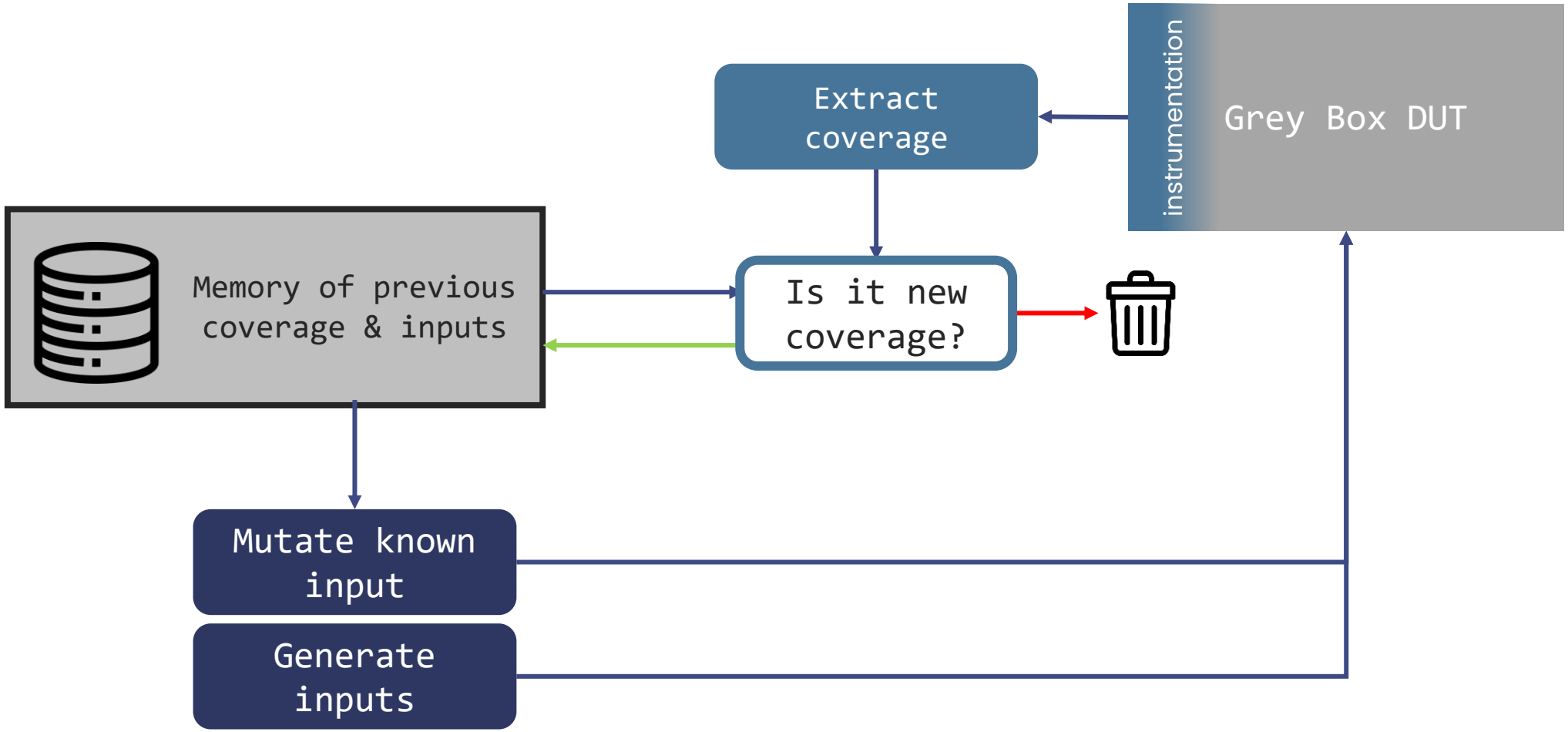
(The more code we cover, the more chances we have to trigger a bug)

Grey Box Fuzzing (example: AFL)

Grey Box fuzzing proved its efficiency

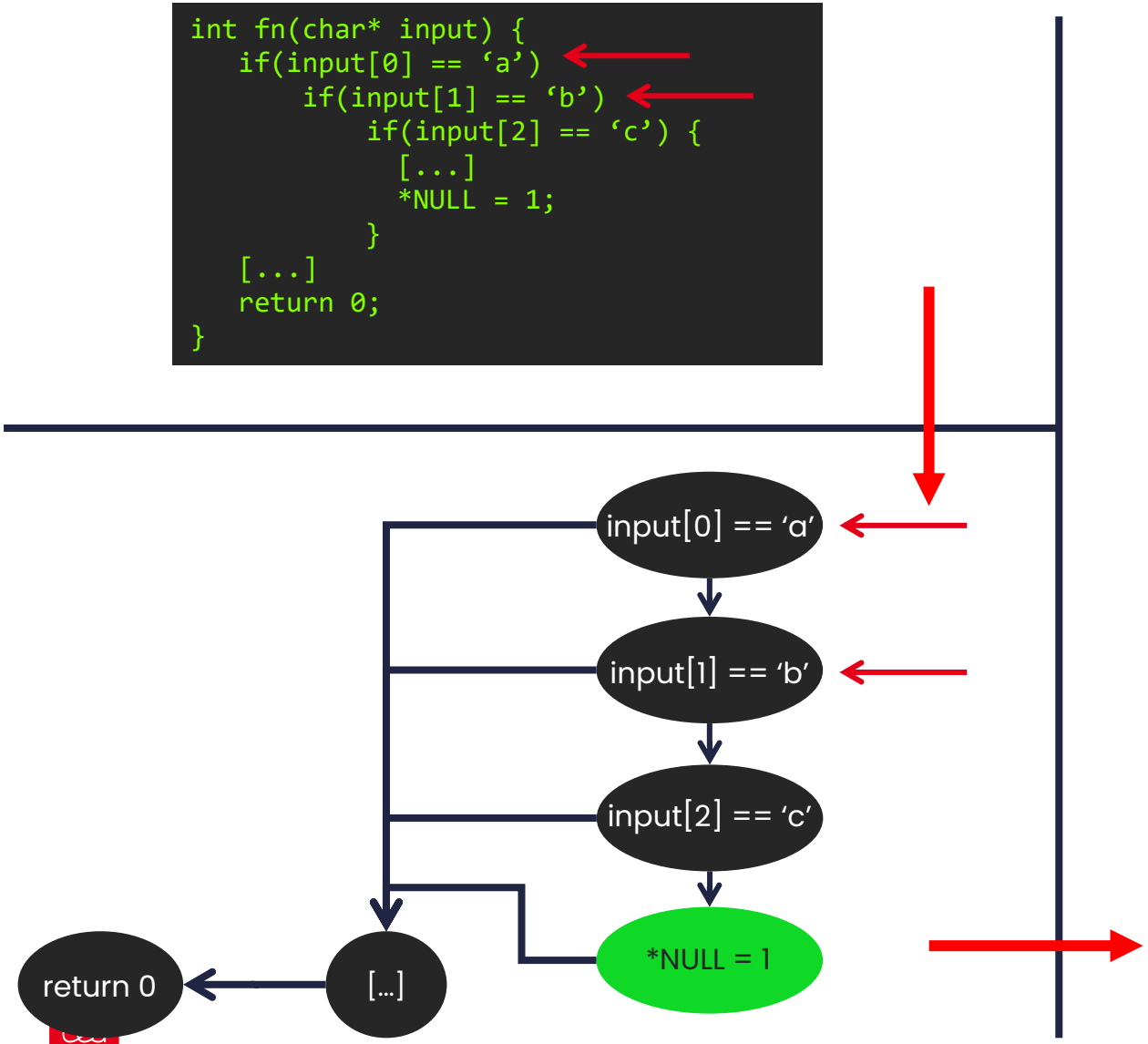
- CVE-2019-14438: VLC heap-based buffer over-read
- CVE-2014-6277: Bash not properly parse function definitions
- CVE-2020-8036: tcpdump exploit
- CVE-2021-3156: sudoedit heap-based buffer overflow
- And a few hundred more

Current State of the Art: AFL



Coverage Map ➔ how AFL extracts behaviors?

```
int fn(char* input) {
   if(input[0] == 'a')
      if(input[1] == 'b')
         if(input[2] == 'c') {
           [...]
           *NULL = 1;
         }
   [...]
   return 0;
}
```



0	0	0	0	0	0
0	1	0	1	0	0
0	0	0	0	0	0
0	1	0	1	0	0
0	0	0	0	1	0
0	0	0	0	0	0

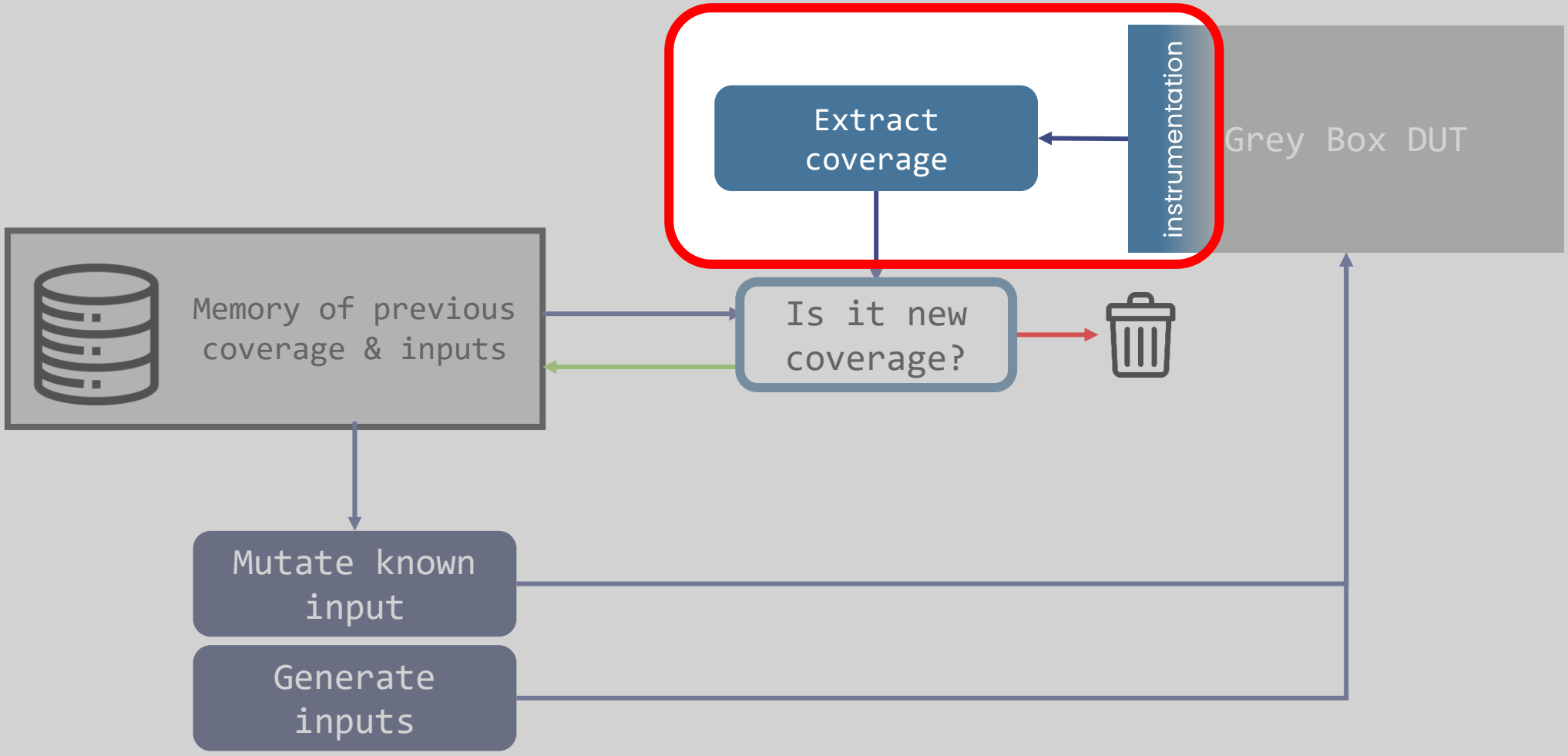
AFL Map (vector in reality)

Coverage Map ➔ how AFL finds novelty ?

0	0	0	0	0	0
0	1	0	1	0	0
0	0	0	0	0	0
0	1	0	1	0	0
0	0	0	0	1	0
0	0	0	0	0	0

0	0	1	0	0	0
0	3	0	1	0	0
0	0	0	0	0	0
0	0	0	1	0	0
1	0	0	0	2	0
0	0	0	0	0	0

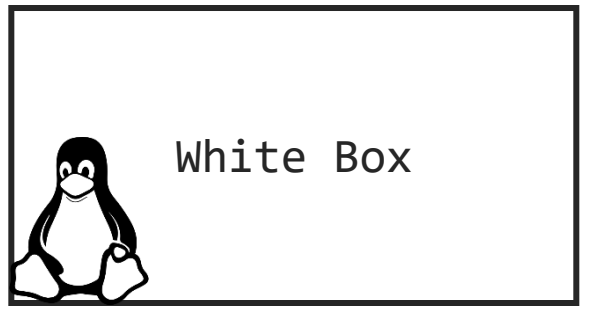
Grey Box Fuzzing principle



Context

Our Goal

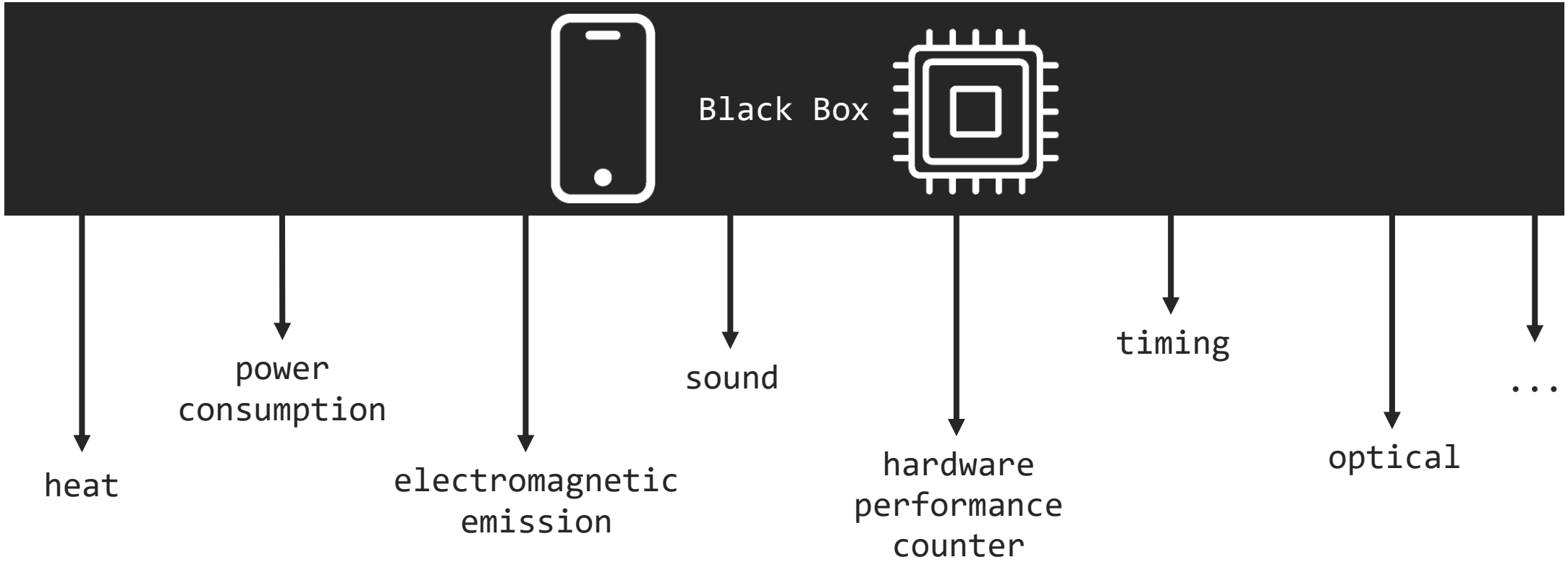
Grey Box



- no source-code
- no instrumentation
- no binary
- limited interactions

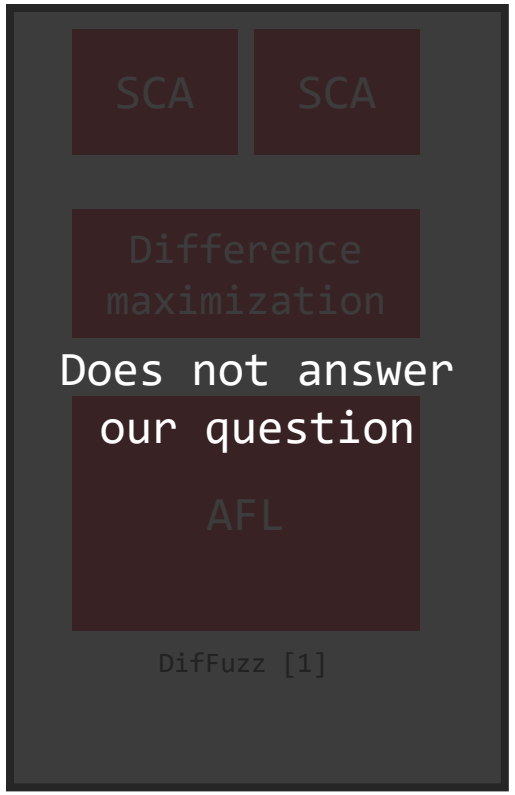
- source-code
- instrumentation
- binary
- full interactions stack

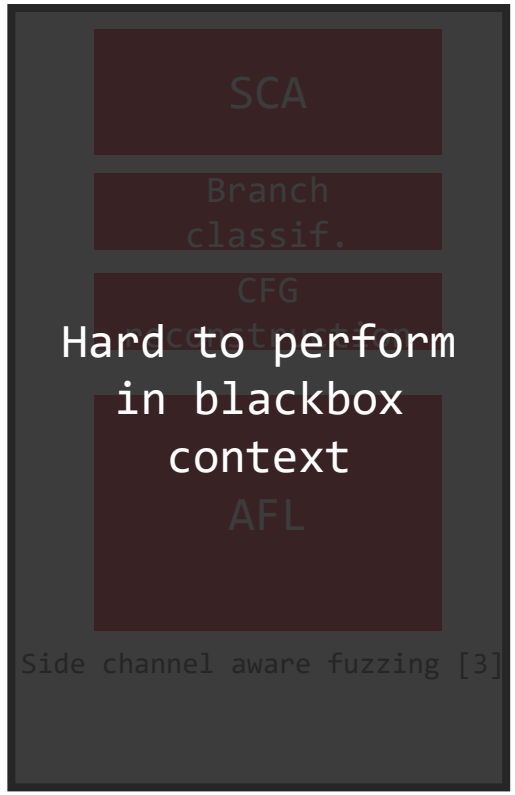
Side-channel analysis



Assumption : physical access + no code access + no debug interface

How do fuzzers and side-channel coexist ?





[1] Nilizadeh, Shirin, Yannic Noller, et Corina S. Pasareanu. « DifFuzz: Differential Fuzzing for Side-Channel Analysis ». In *2019 IEEE/ACM 41st International Conference on Software Engineering*
[2] Riscure. « Black box fuzzing with side channels ». Hardweare.io, 2022.
[3] Sperl, Philip, et Konstantin Böttinger. « Side-Channel Aware Fuzzing ». In *Computer Security – ESORICS 2019: 24th European Symposium on Research in Computer Security*
[4] Barredo, J., Petke, J., Clark, D., Blackwell, D., Eceiza, M., Flores, J.L., Iturbe, M.: GAFLERNA ahoy! integrating EM side-channel analysis into traditional fuzzing workflows (2025).
[5] Su, Kai, Mark Giraud, Anne Borcherding, Jonas Krautter, Philipp Nenninger, et Mehdi Tahoori. « Fuzz Wars: The Voltage Awakens – Voltage-Guided Blackbox Fuzzing on FPGAs ». In *2024 IEEE 42nd VLSI Test Symposium*

How we would like a fuzzer and SCA to coexist?

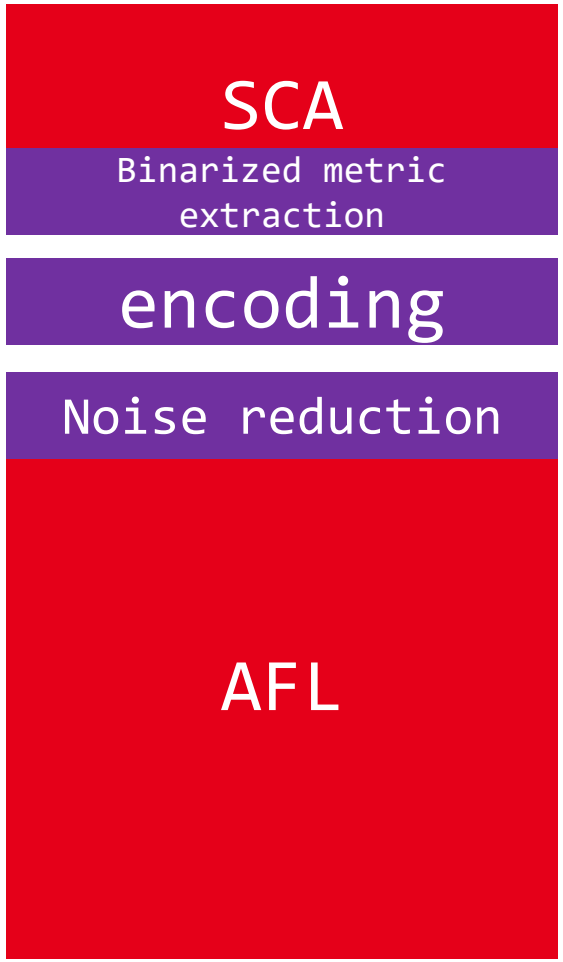
SCA

encoding
clustering

AFL

- Simple
- Deterministic
- Performance efficient
- Easy to setup in Blackbox context

Research questions



Which metrics are relevant?

How can we encode side-channel information for a fuzzer to understand?

How to reduce noise in side-channel measurements?

The main contributions

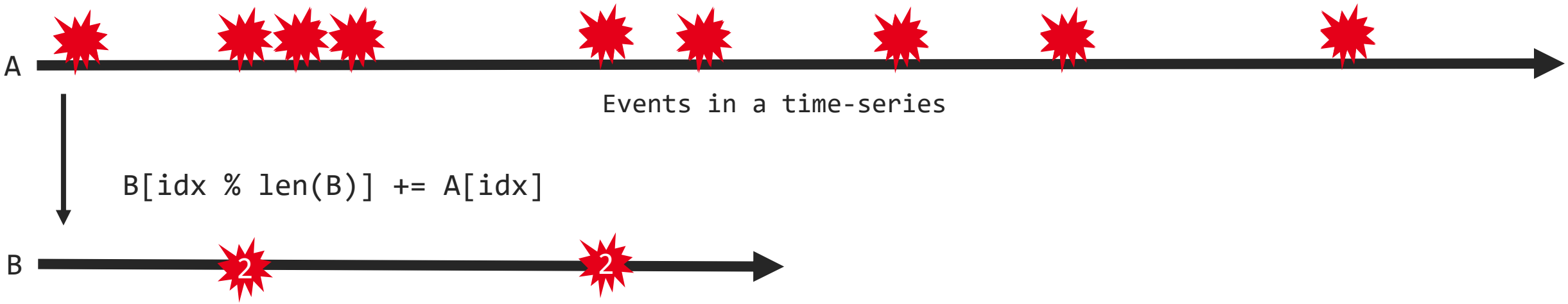
Encode SC information for a greybox fuzzing engine to understand

Use emulated memory instructions execution as fuzzing feedback

Use EM side-channel to memory activity as fuzzing feedback

1. How to encode Side-Channel into a fuzzer?

A time-series to an AFL map

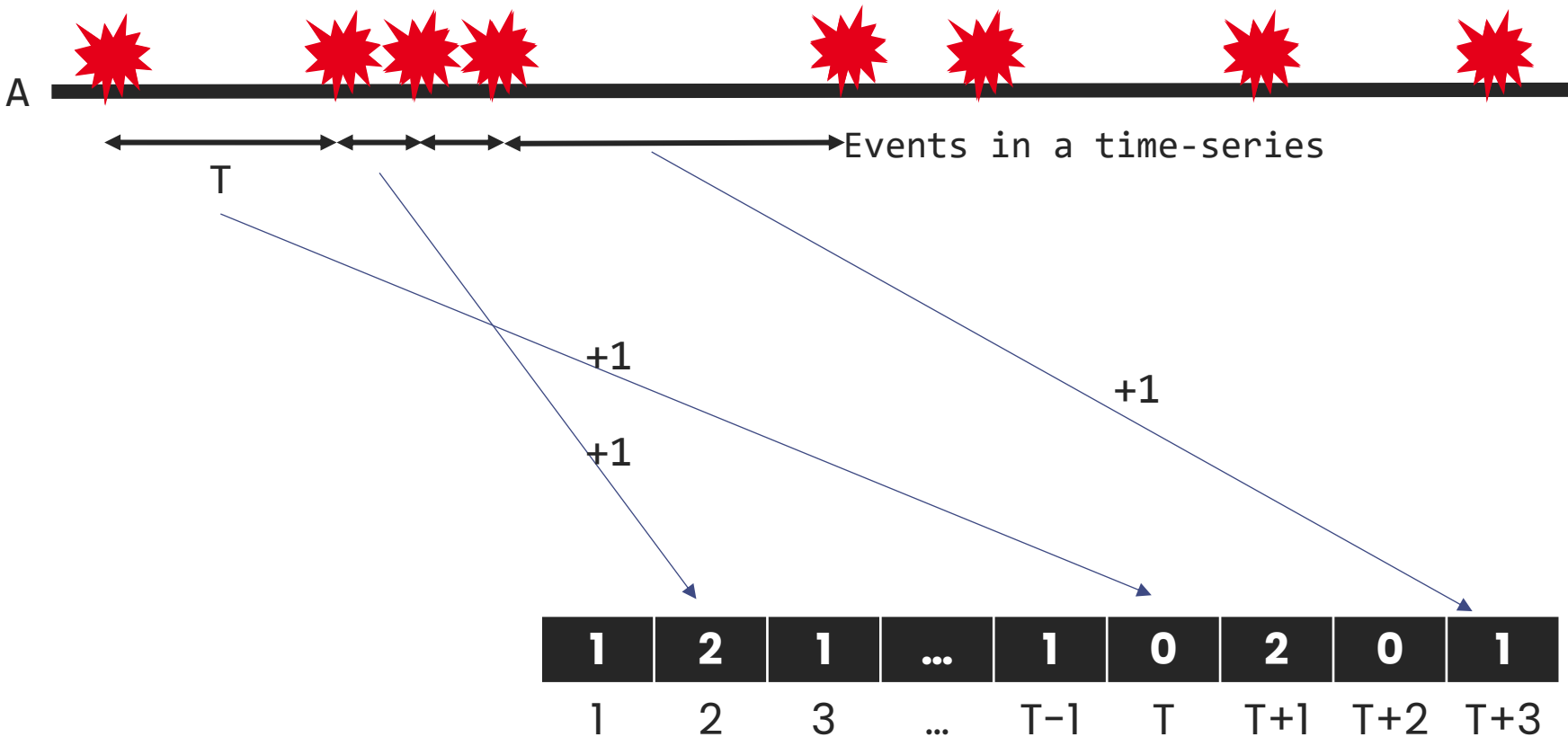


Events accumulated on a smaller representation of the time series

1	2	1	1	1	0	2	0	1

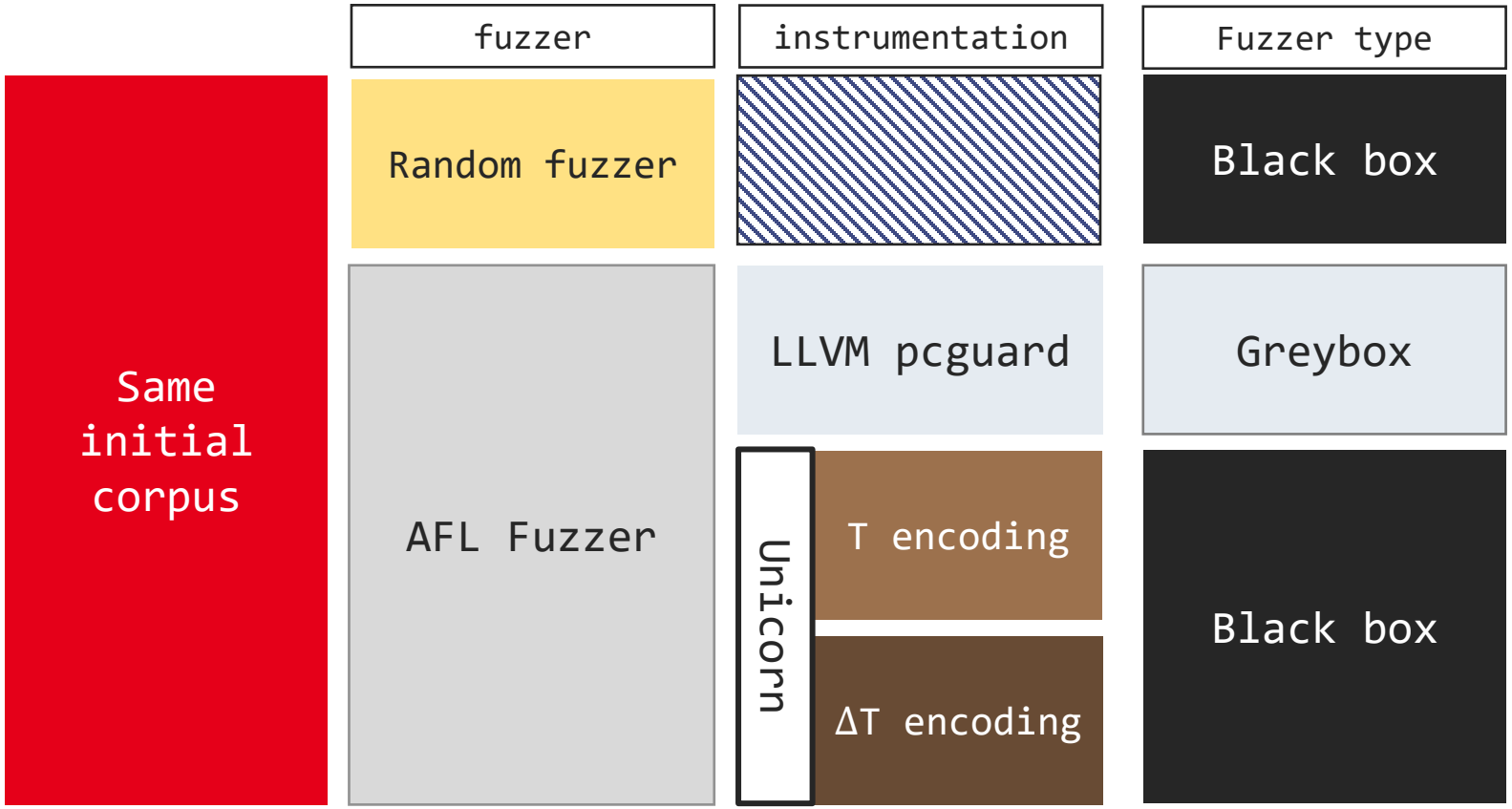
Same time-series but represented in table

A time-series to an AFL map



Same event time-series but represented in their relative distance

How effective is this fuzzer theoretically?



How effective is this fuzzer theoretically?

	cJSON	zlib	libjpeg
Random fuzzer	100%	100%	100%
LLVM pcguard	143%	139%	264%
T encoding	105%	133%	156%
ΔT encoding	109%	133%	165%

How can we encode this information for a fuzzer to understand ?

How effective is this encoding + information theoretically ?

➔ Use of generic encoding

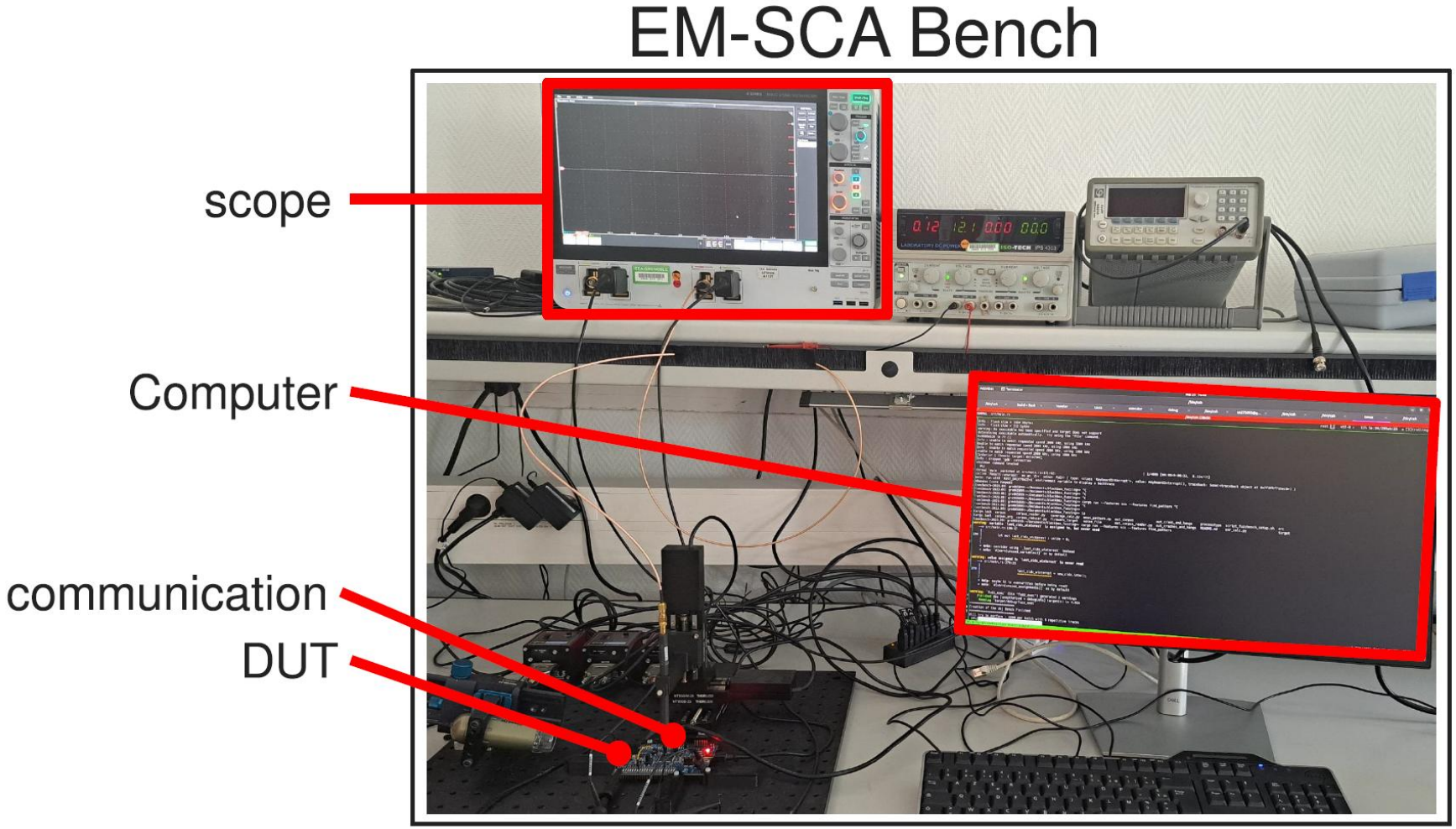
➔ Can reach up to 65% more branches than the random fuzzer

2. Which metrics are relevant?

What microarchitectural event could be used?

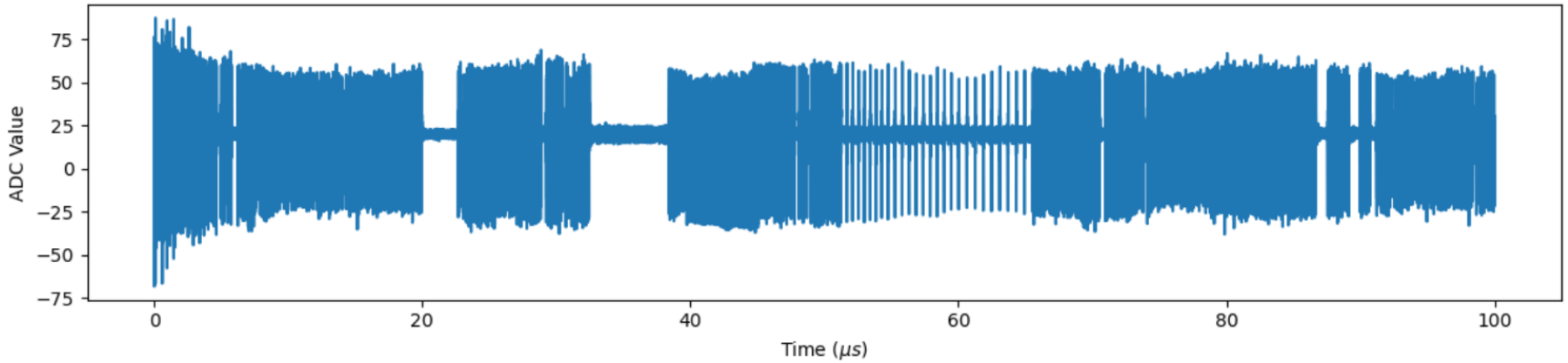
Cache miss/hits
TLB miss/hits
Specific instructions execution : Branch, Ld, Str …
Specific hardware block (FPU, Accelerators, SIMD…)
Bus events
Interrupts
Performance counters
…

Experimental setup

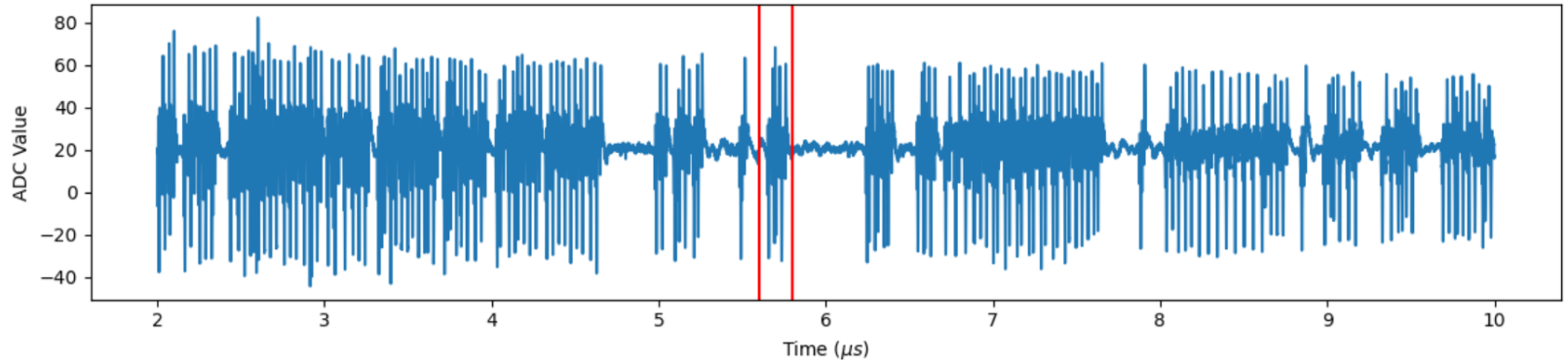


Analysis on a CJSON Decoder

EM trace of a CJSON decoder running on a STM32F4

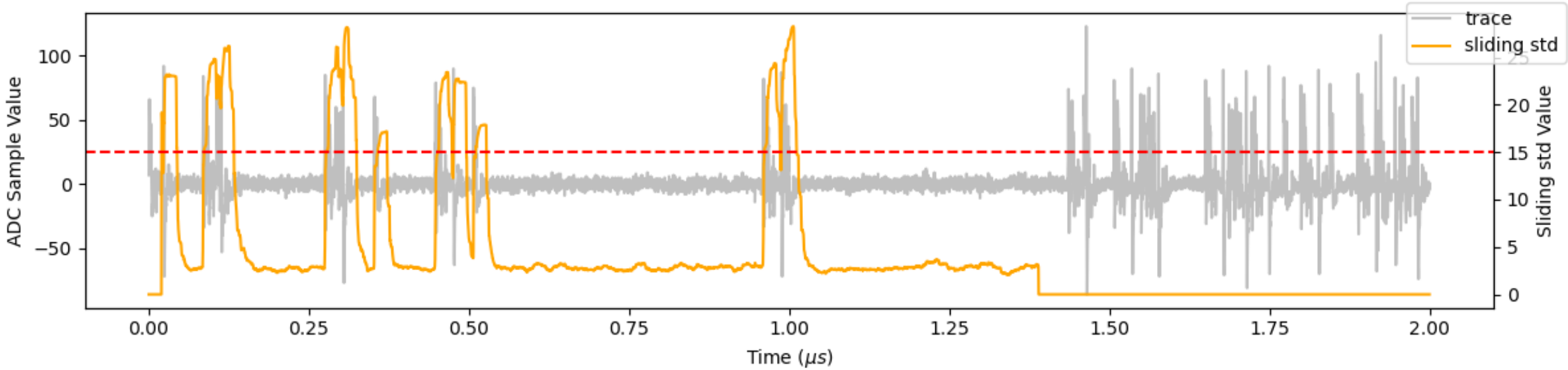


Zoomed view:



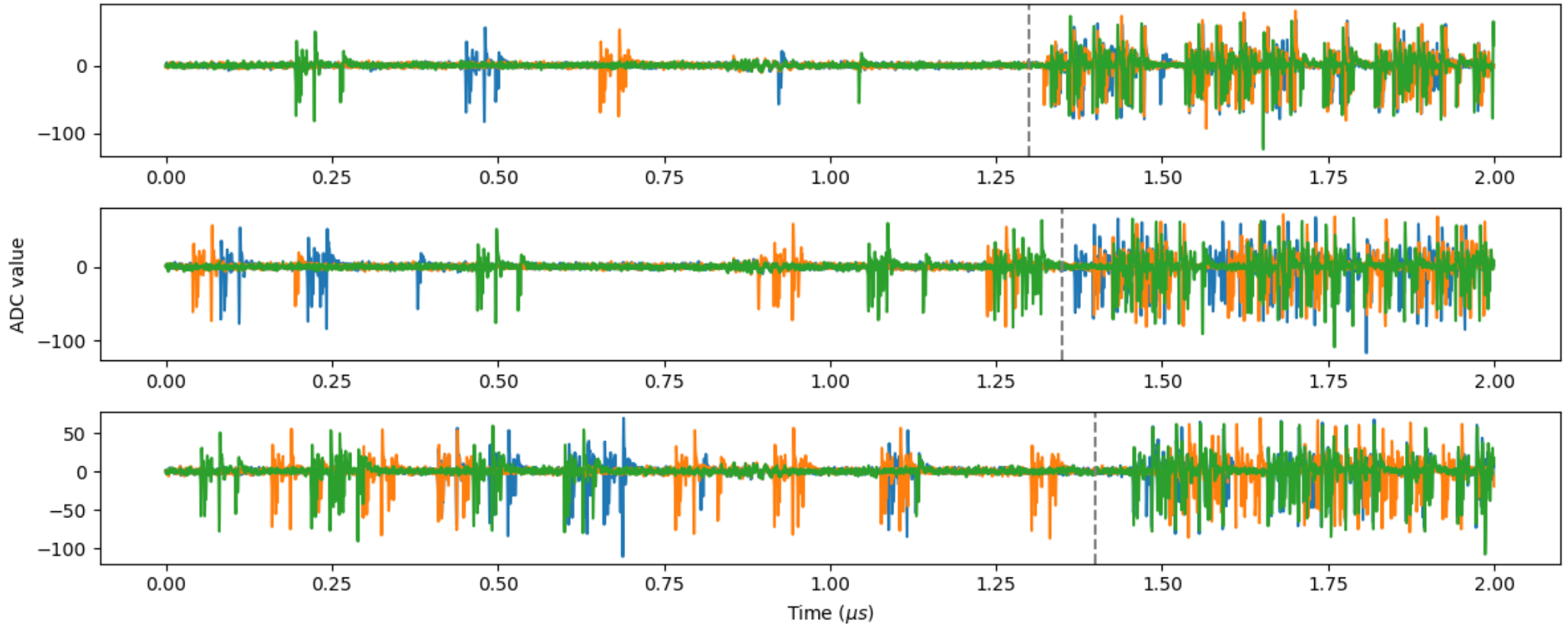
Our guess => the patterns are due to cache misses

A metric to control them all.



Sliding windowed standard deviation trace*

Analysis of FLASH data load behavior

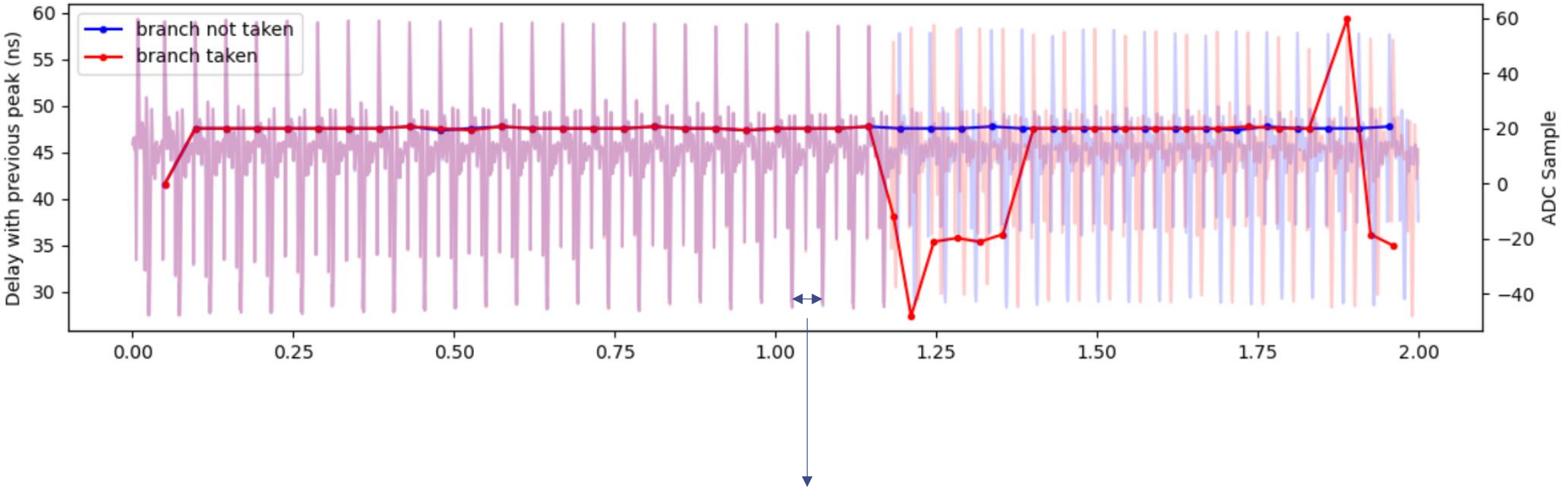


Top: 1, middle: 4, and 10 loads at the bottom

Green, blue and orange traces correspond to 3 executions of the benchmark with different load instruction positions

Analysis of FLASH instruction cache

Analysis of delay between peaks.



Time for a complet $I cache line execution
(128b line -> 8 thumb inst.)

What physical side-channel information can be used to create coverage ?

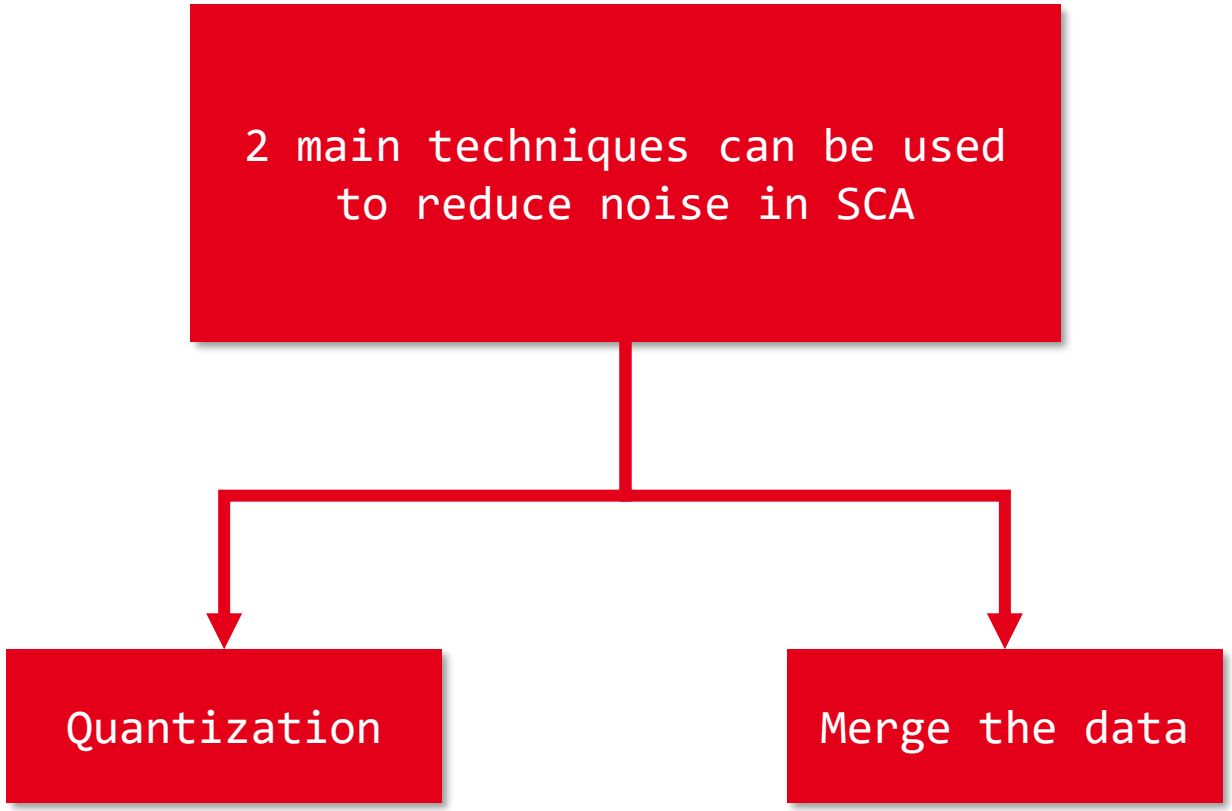
➔ Use of EM Cache Miss leakages

3. How can we reduce noise in the information fed to the fuzzer ?

Coverage Map ➔ what AFL might see when there is noise.

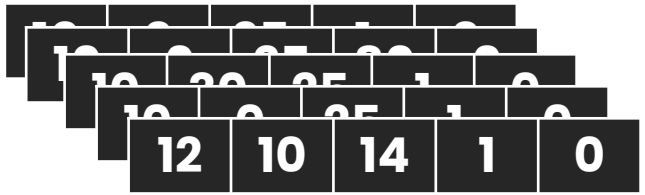
0	0	0	0	0	0
0	1	0	1	0	0
0	0	0	0	0	0
0	1	0	1	0	0
0	0	0	0	1	0
128	0	0	0	0	0

Denoising

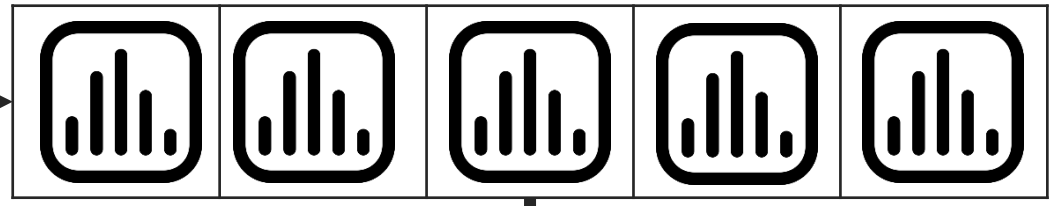


Quantization*: reduction of the range of values

How to Merge data in AFL Maps ?



Multiple observation of the same input



Select values with highest frequency

mode

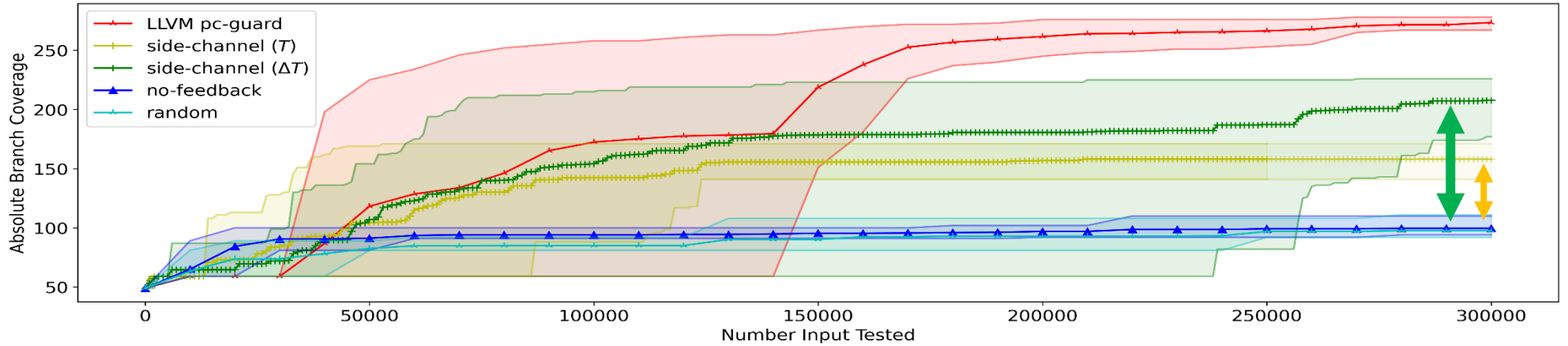
Merge

Merged observation

quantization

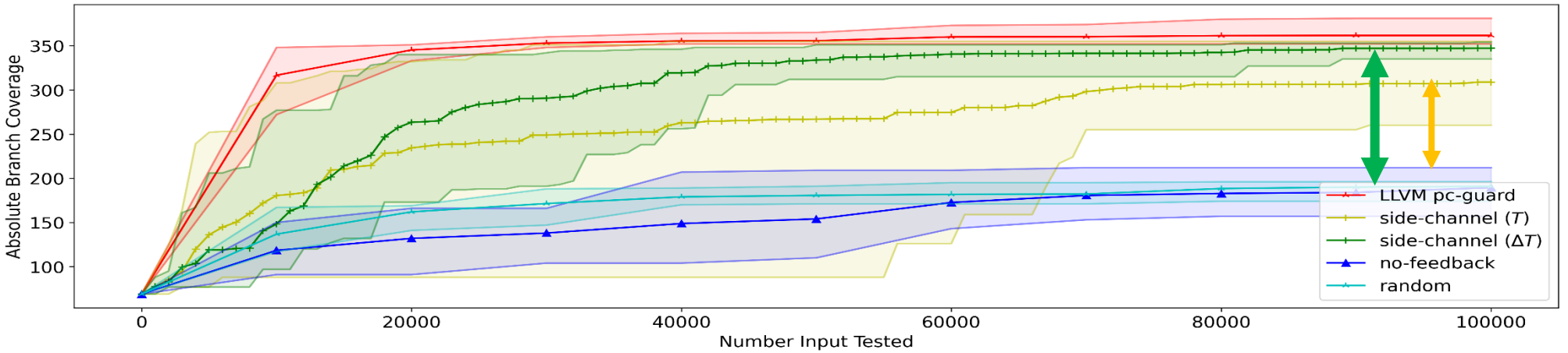
AFL quantization

JPEG



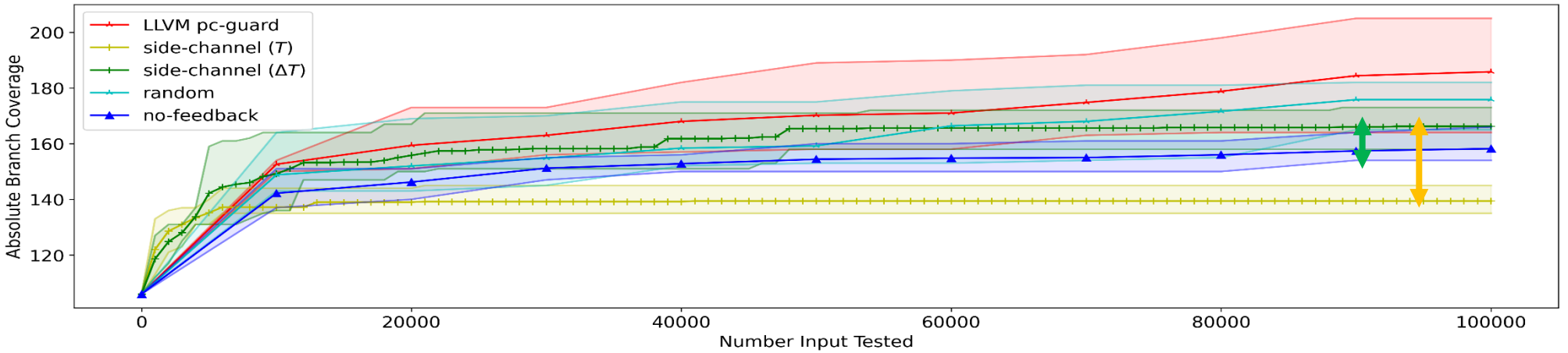
108 pt %
59 pt %

ZLIB



83 pt %
63 pt %

CJSON



5 pt %
-12 pt %

How can we reduce noise ?

➔ Use of trace aggregation and internal AFL quantization

How effective is this encoding + information in reality ?

➔ Can reach up to 108% branches more than a random fuzzer

Conclusion

Fuzzing with side channel offers interesting possibilities for black box systems

- A new way to encode the side-channel information.
- The use of AFL internals for noise reduction
- There still remain a lot of work for effective fuzzing with side-channel feedback

PS : some of this work used the Secbench framework

https://github.com/CEA-Leti/secbench

Thank you for your time

Do you have any questions ?

1 - ulysse.vincenti [at] cea [dot] fr
2 – thomas.hiscock [at] cea [dot] fr
3 – david.hely [at] lcis [dot] grenoble-inp [dot] fr

Icons are from flaticons
Hacker image from pexel

cJSON results explanations: 2 answer sources

Duskfuzz falls in a strategy to go through floating points or ints with different levels of precisions on cJSON

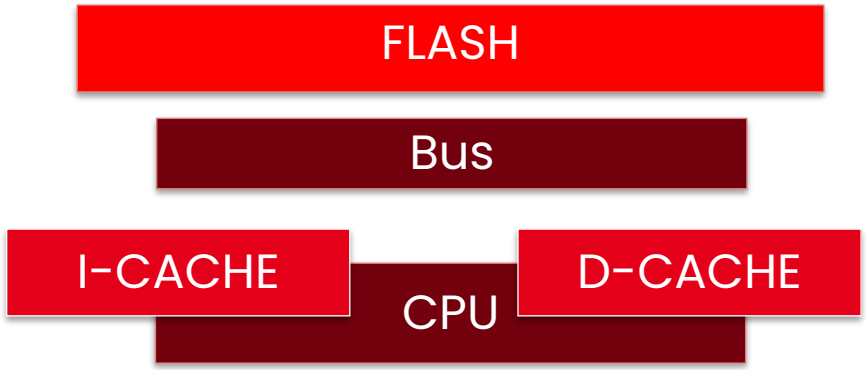
The random fuzzers have trouble to go through a magic number challenge

A performance problem

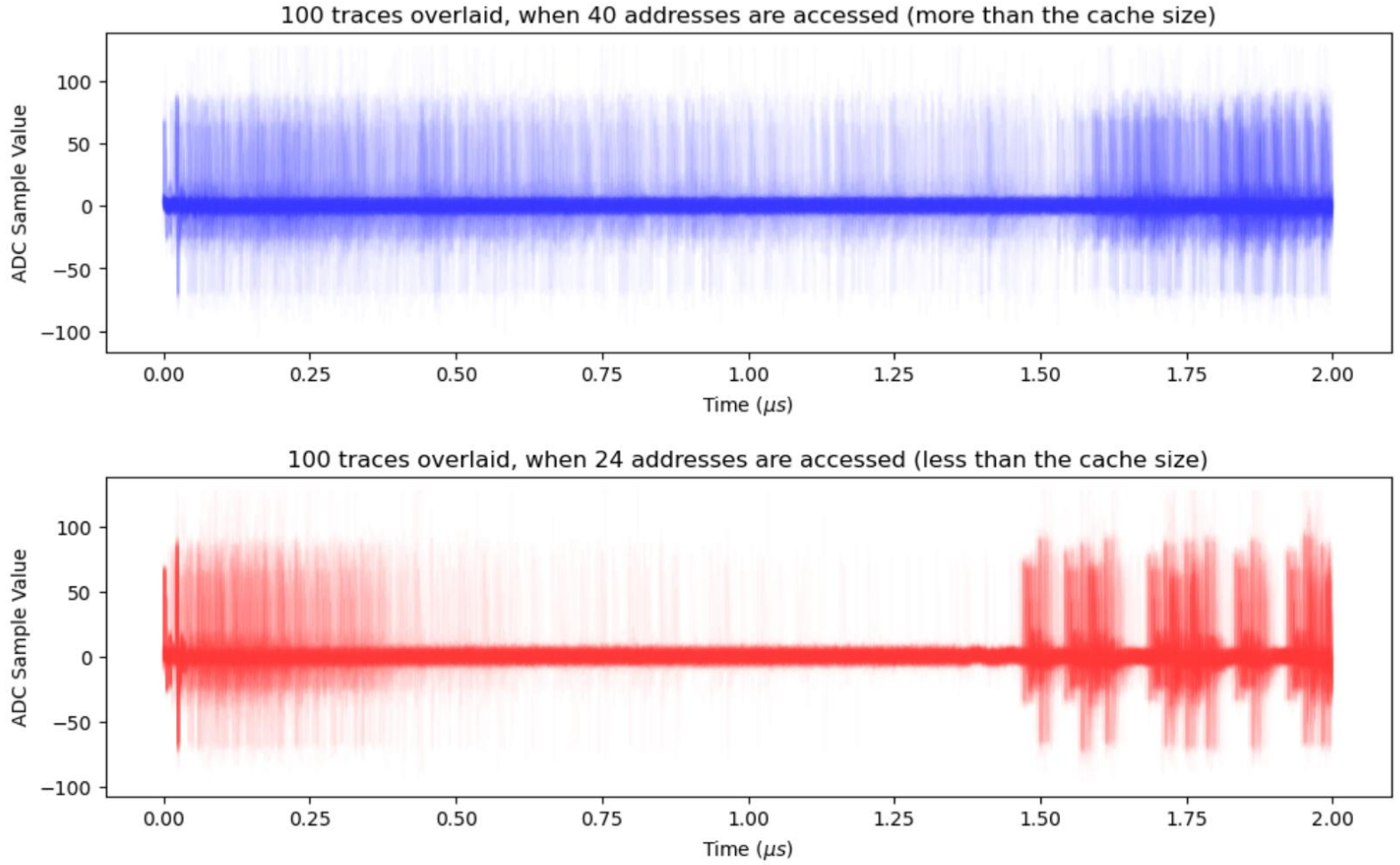
- 3 benchmarks to test
- 5 repetition of each fuzzing campaign on each benchmark
- Each campaign takes ~2 days to perform
- ~5exec/sec on the DUT
- Multiple crash of the equipements

STM32F4 FLASH Caches

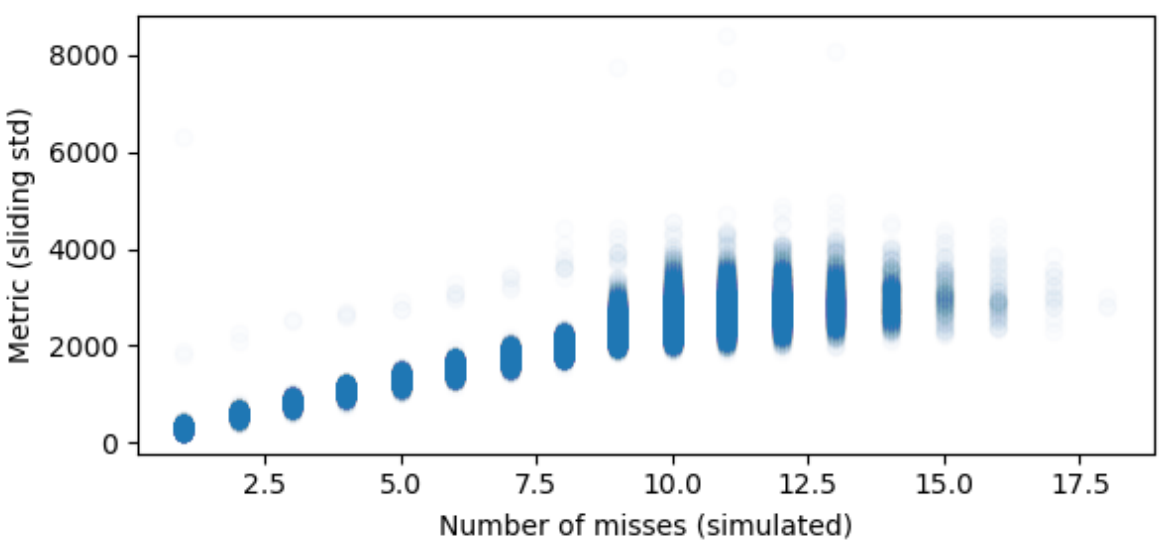
- According to the reference manual:
  - Data cache: 8x128 bit lines, fully associative, with LRU
  - Instruction cache: 64x128 bit lines, fully associative, with LRU, and sequential prefetcher.

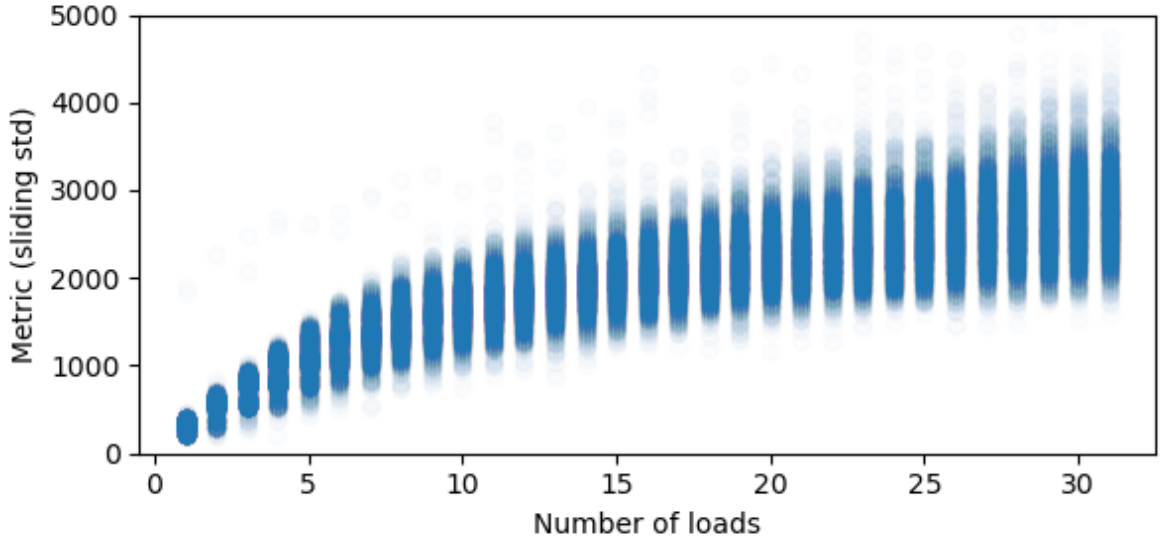


Analysis of FLASH data load behavior



Use of flash data cache-miss information



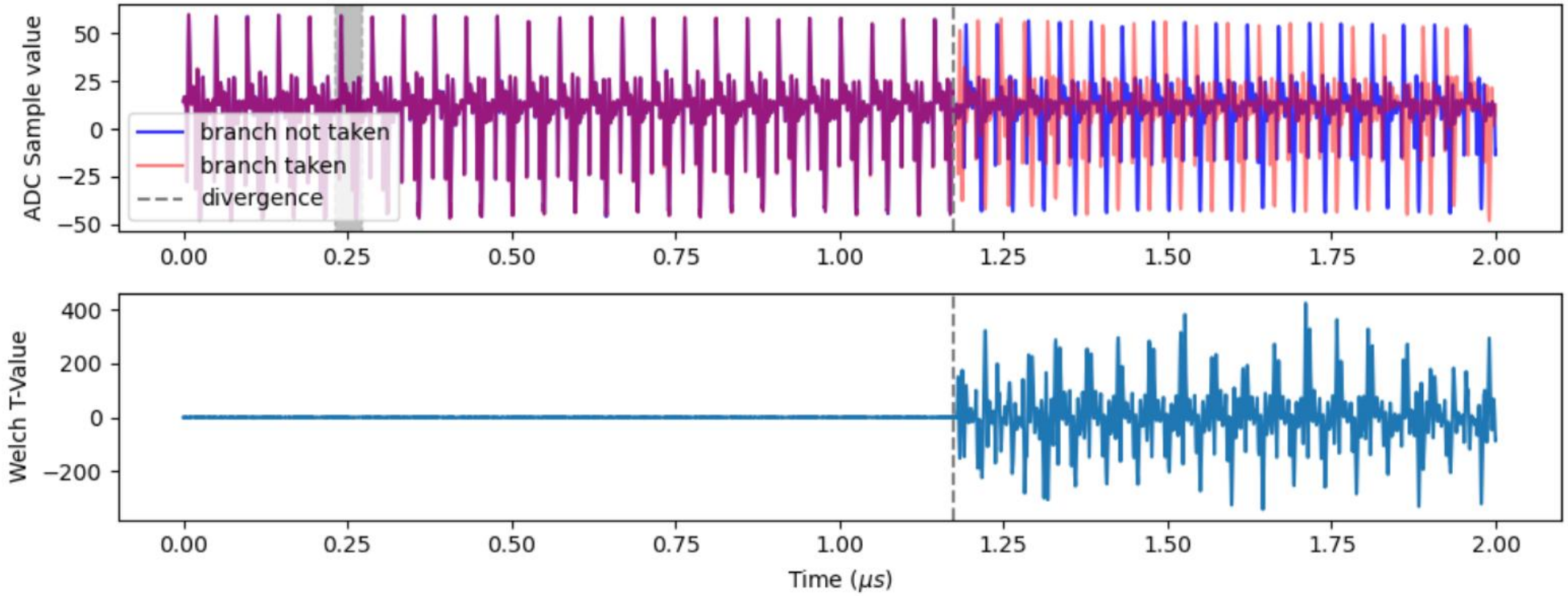


Link between cache miss simulation and sliding std. Number of cache miss(left), number of loads(right)

Metric	Pearson correlation
Number of load vs sliding std	0.86
Number of DCM vs sliding std	0.93

Analysis of FLASH instruction cache

Comparison of traces with branch taken and not taken.



Black Box Fuzzing principle