

CPU Vulnerabilities

Finding Bugs Where Software Ends

Michael Schwarz | FOSAD Summer School 2026

Lab #1:

Your Side Channel



`https://fosad.cpu.wtf`



Setup: Overview



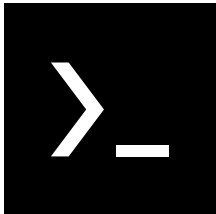
Termux



- ARM device → Android phone, Raspberry Pi (requires Android, Linux, or Windows + WSL)
- or get a remote machine:
`https://fosad-lab.cpu.wtf`
- or team up with people if you don't have one available



Setup: Termux Shell



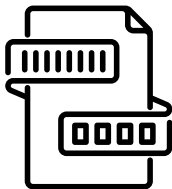
- Install Termux via GitHub (**Not from Play Store**)
- Then in Termux:

```
pkg update && pkg upgrade
```

```
pkg install croc wget
```
- You can transfer files via croc: `croc send <file>`



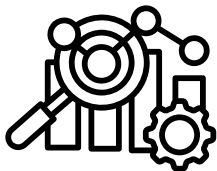
Setup: Download the Binaries



- Download the prebuilt binaries
- `wget` and `chmod +x`
- `exfilstate`: Fuzzer, discovering timer-less side channels
- `repro-runner`: Reproduce discovered primitives
- Both **static** binaries (no dependencies required)



Task: Run the fuzzer



- Get sheet:
`https://fabianthomas.de/pdfs/exfilstate-sheet.pdf`
- Set up device
- Run the fuzzer: `./exfilstate`
- **Questions:**
 - How many cores are detected?
 - Which core types (microarchitectures) are detected?
 - How many memory instructions are detected?

ARM is not one device



Different cores, different vendor modifications, and many configurations



Task: Take a look at your results

- Fuzzer produces reproducers (yaml files)
- Run (a few) reproducers with: `./repro-runner <repro-file>`
 - Does it reproduce nicely?
 - How are the evaluation scores?
- Inspect (a few) reproducers:
 - Which core was the reproducer discovered on?
 - Inspect the instruction sequence.
 - How long is it?
 - What do the instructions do? (STXRH [site:developer.arm.com](https://developer.arm.com))
 - Inspect the architectural states.
 - What is the difference? Do both results comply with the ISA?



Session 1 close

We learned to search for a receiver: a path from hidden CPU state to a program-visible observation.

Takeaway

You (hopefully) found your own side channel. Session 2 will provide a bug, and you exploit buggy memory protection.

You survived the cache-state part.



Tomorrow: one program, two implementations, two outputs