# EL3XIR: Fuzzing COTS Secure Monitors

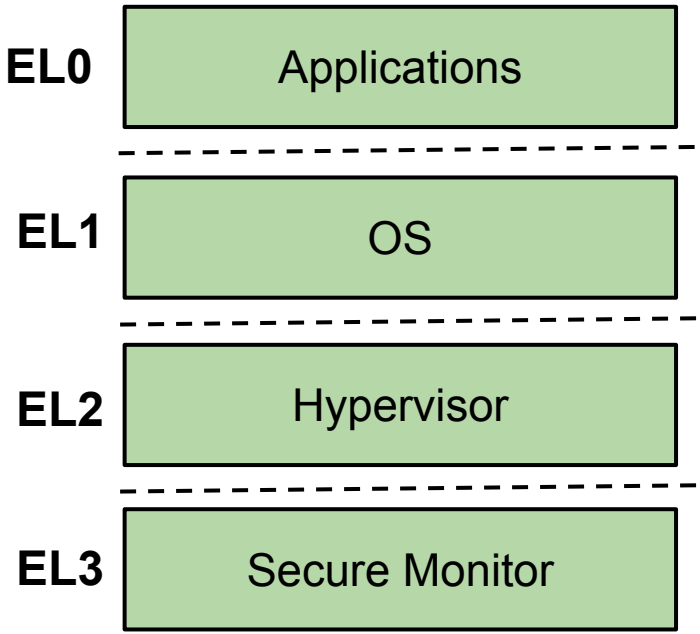
**Marcel Busch & Christian Lindenmeier**

christian.lindenmeier@fau.de
marcel.busch@epfl.ch

FAU EPFL

# This Talk

**ARM Cortex-A**






Fuzzing


Rehosting

# $ whoarewe

**Marcel Busch** @0ddc0de

- PhD graduate from FAU Erlangen-Nuremberg
- PostDoc with HexHive @ EPFL

- Works on
  - Embedded systems security
  - Android TEEs
  - Fuzzing

**Christian Lindenmeier** @_chli_

- PhD student at FAU Erlangen-Nuremberg

- Works on
  - Rehosting and fuzzing firmware
  - Android digital forensics



# Devices Relying on EL3


[1]

[2]

[3]

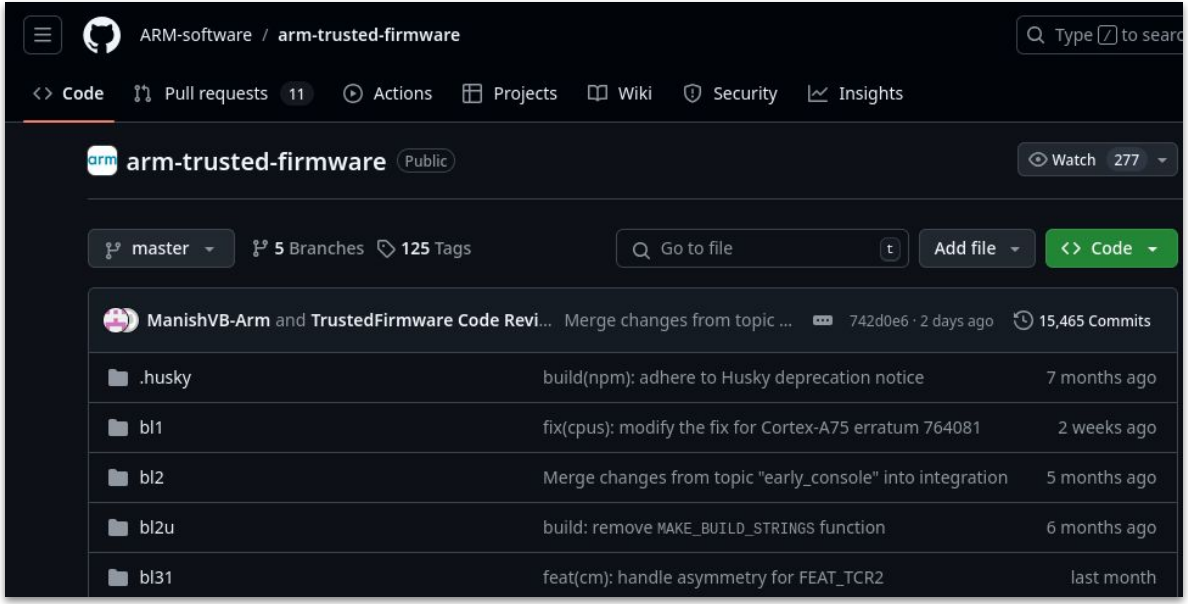

[4]


[5]

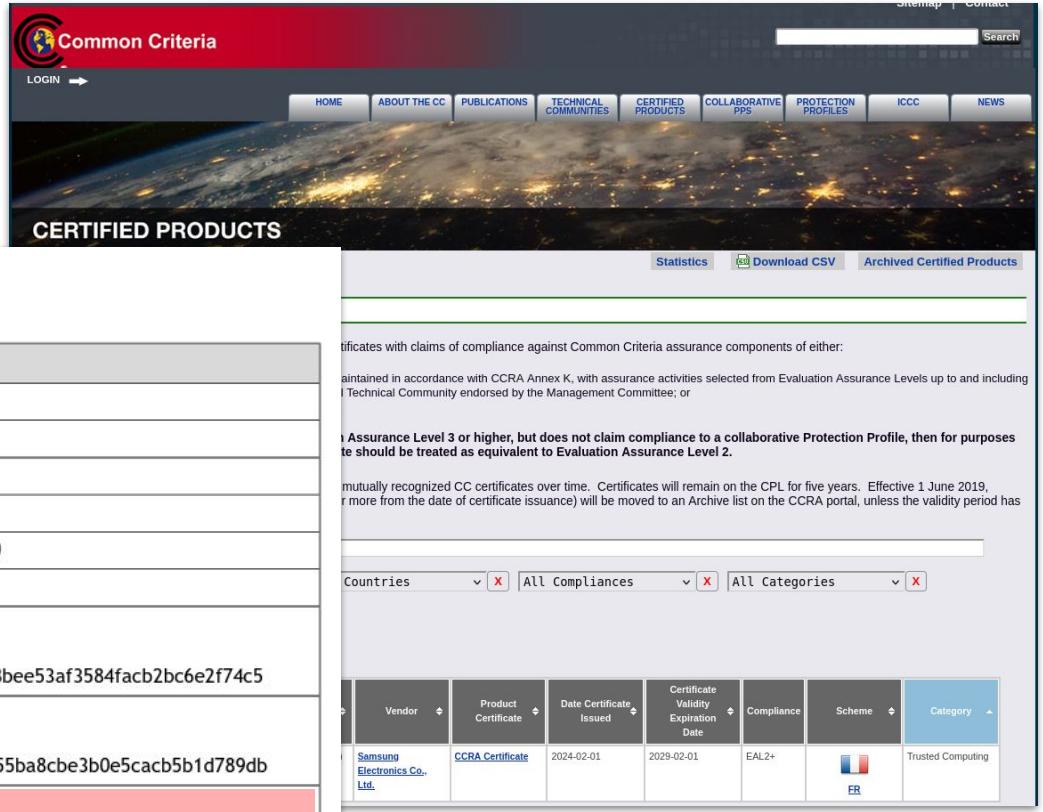
[6]

[7]

# EL3 Typically Implemented in C




# Why should I care?



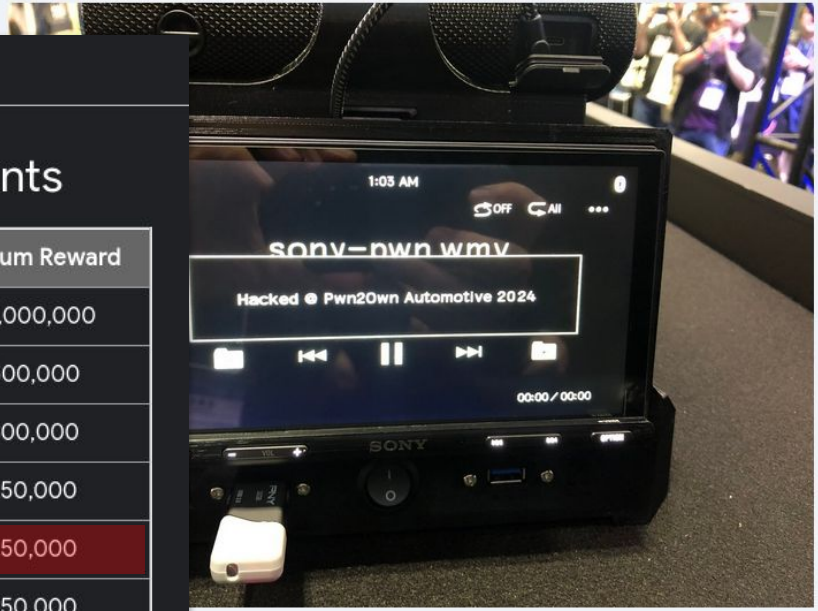

## 2.2.2 TOE References

The following table provides TOE identification information.

| TOE Identification | |
|---|---|
| SoC reference | s5e9925 |
| Device reference | Samsung Galaxy S22, S22+, S22 Ultra |
| Model reference | SM-S901B, SM-S906B, SM-S908B |
| DRAM reference | Samsung LPDDR5 8G/12G 3200MHz |
| Commercial name(s) | Samsung Secure OS (TEEgris) version 5.0.0.0 |
| Main developer | Samsung Electronics |
| ROM/Boot code reference | Zagreb-SP1A-1727<br>sha256:<br>0cb293569ae1a33477f27b0acd58bd455c4d88bee53af3584facb2bc6e2f74c5 |
| TEE binary reference | Samsung Secure OS Release Version 5.0.0.0<br>sha256:<br>df6e524d7accacbc722776660ffd698b023a6c55ba8cbe3b0e5cacb5b1d789db |
| ATF binary reference | Zagreb-SP1A-2124V1-2125R1<br>sha256:<br>0f78b256b193620bca0f861b550371c14ac5e3b7c40d049fe6ff63a27d3d0512 |
| TEE binary developer | Samsung Electronics Co., Ltd. |

ZERO DAY INITIATIVE

**SUCCESS** - Gary Li Wang used a stack-based buffer overflow against the Sony XAV-AX5500. He wins $20,000 and 4 Master of Pwn Points.




Google Bug Hunters

## Code execution reward amounts

| Description | Maximum Reward |
|---|---|
| Pixel Titan M with Persistence, Zero click | Up to $1,000,000 |
| Pixel Titan M without Persistence, Zero click | Up to $500,000 |
| Local App to Pixel Titan M without Persistence | Up to $300,000 |
| Secure Element | Up to $250,000 |
| Trusted Execution Environment | Up to $250,000 |
| Kernel | Up to $250,000 |
| Privileged Process | Up to $100,000 |


SECURE

# What does it do?


Secure Monitor Features

- Replay-Protected Memory Block (RPMB)
- Power Management
- World Switching
- Crypto Hardware
- Secure Boot
- …

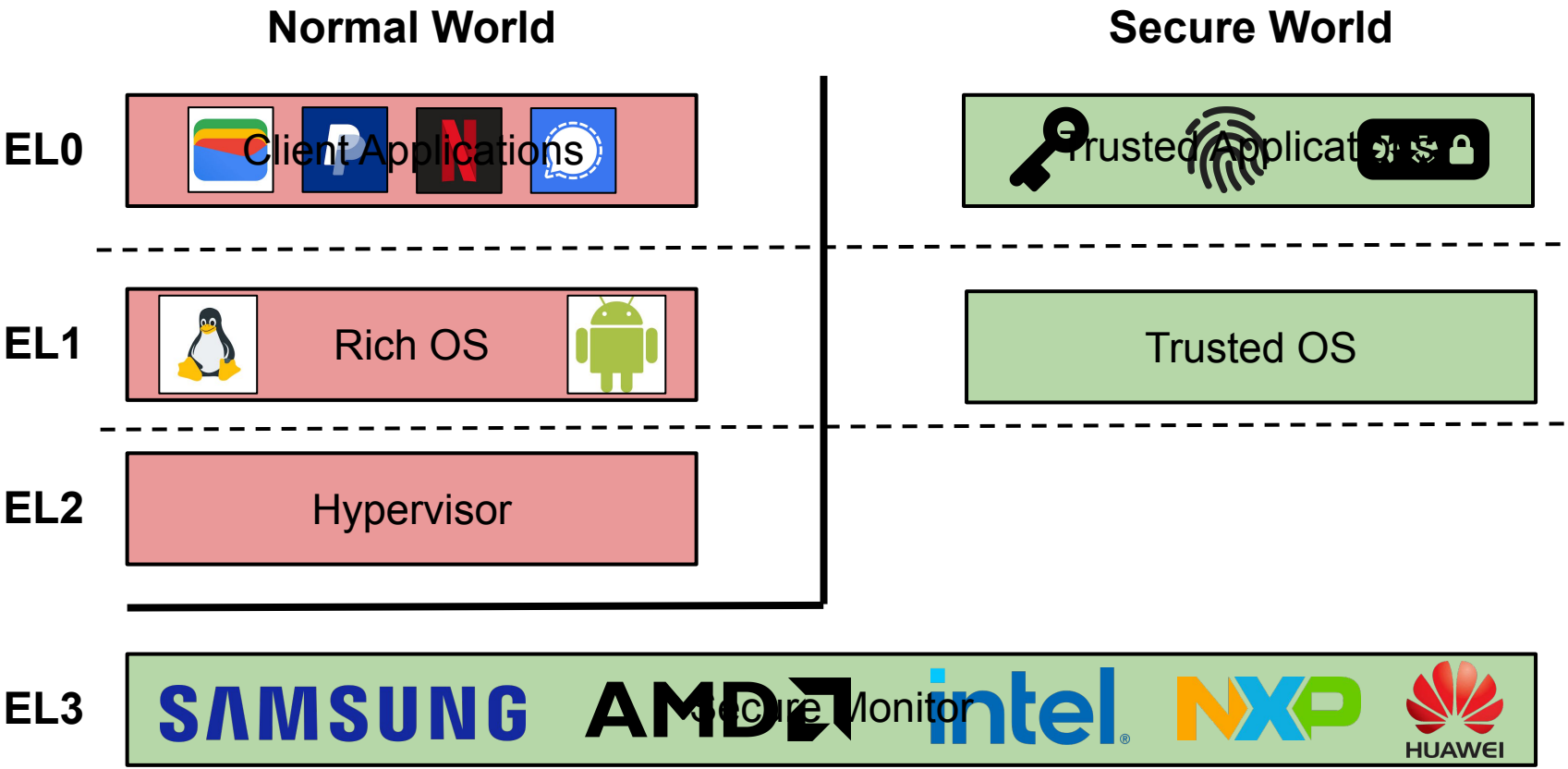
System-on-Chip-specific features

⇒ Fragmentation

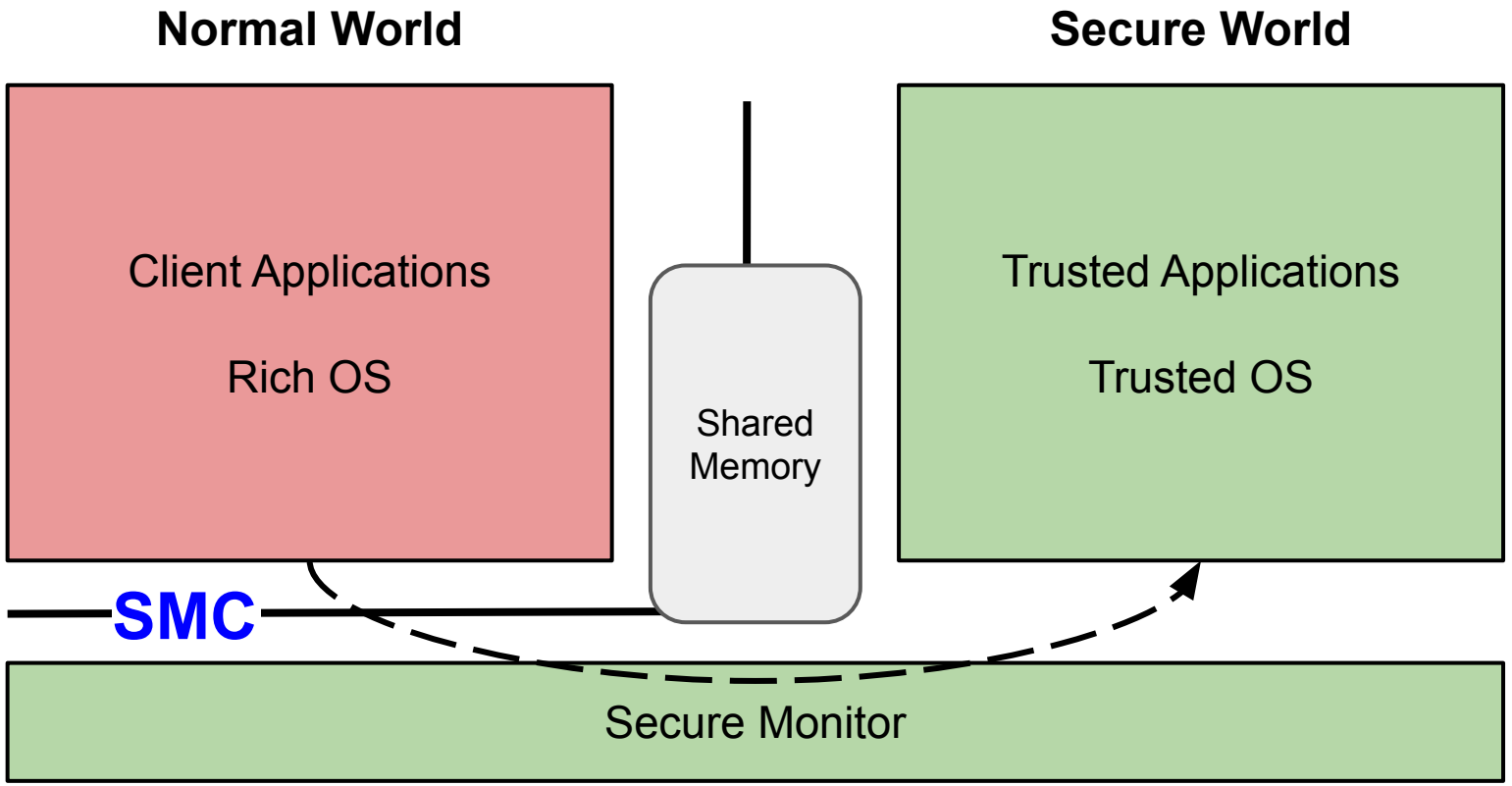
⇒ (Potentially) less audited code

Extensions for ARM Trusted Firmware typically closed-source

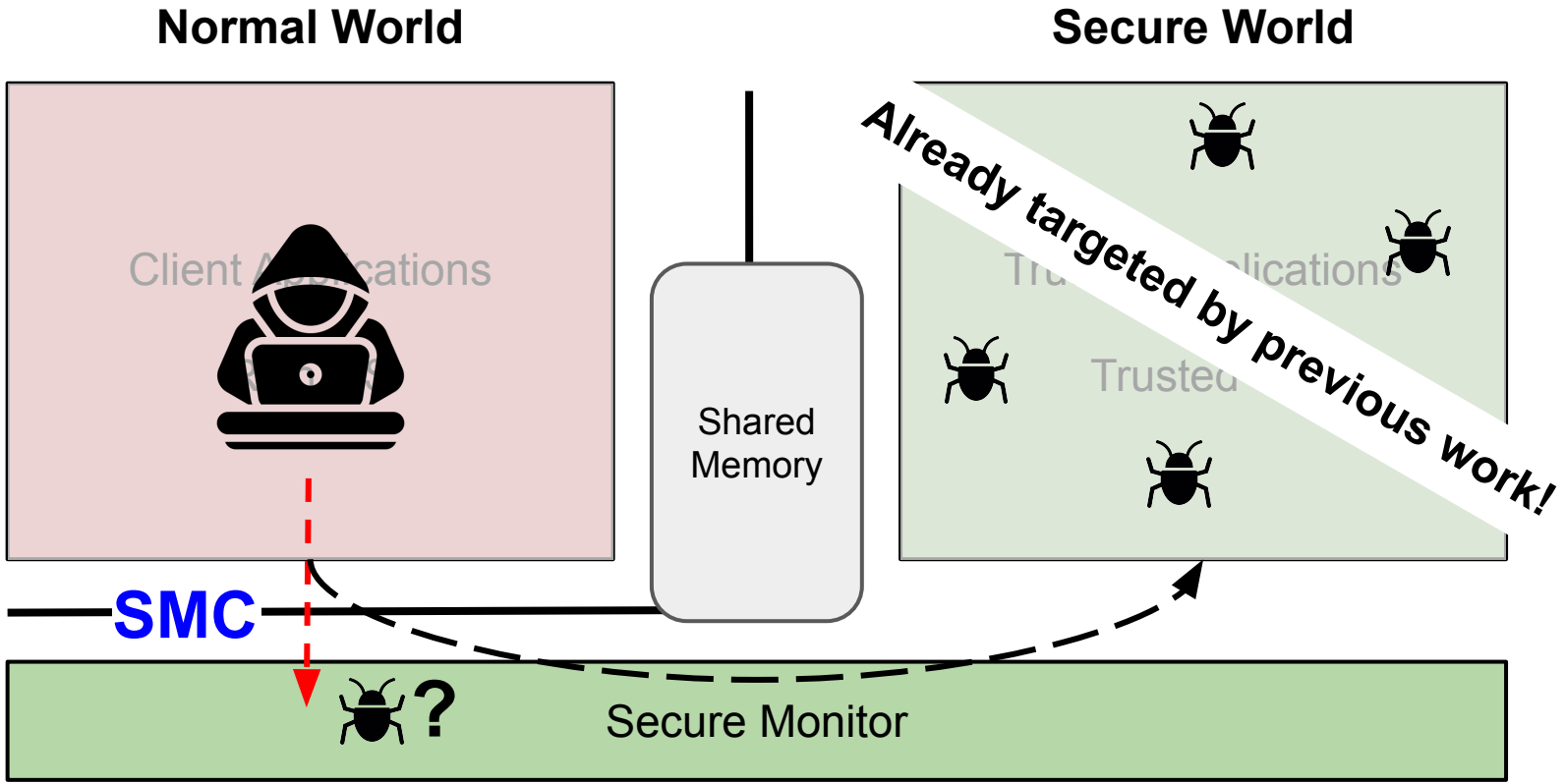
# ARMv8-A Privilege Levels and Execution Modes



# Interacting with TrustZone



# Secure Monitor as Target?



# Previous Work (non-exhaustive)





USENIX
Analysis
Using E

INFILTRATE 2019: EL3
The Ultimate Privilege c
Phone by **Guanxing W**

INFILTRATE SECURITY CONFERENCE

vices

2019 20

2024

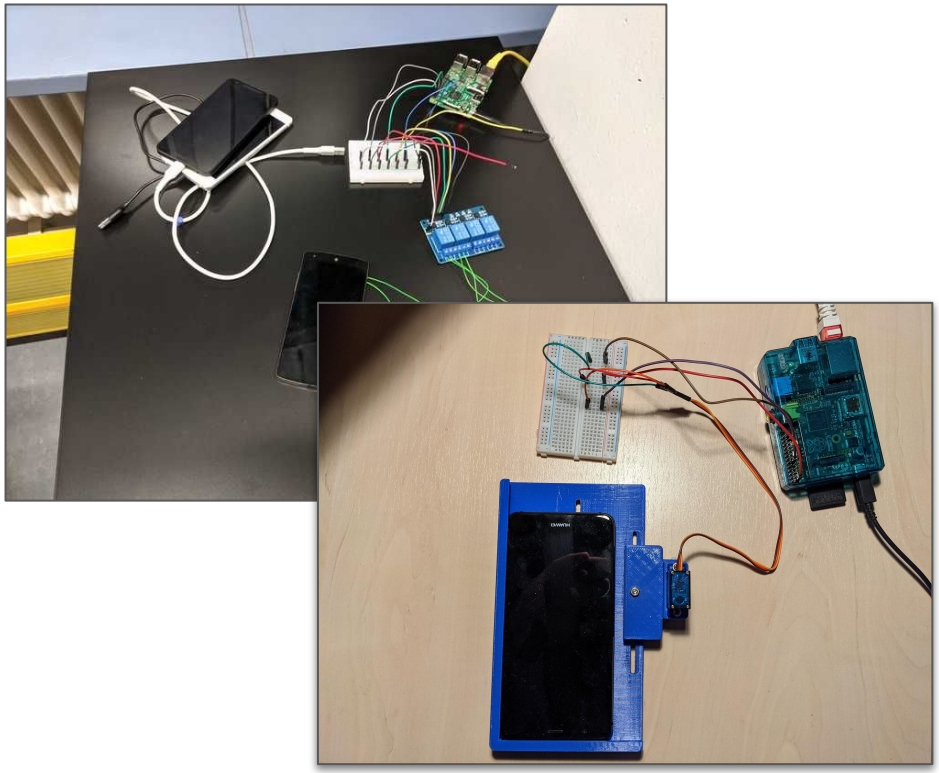
S&P 2020: SoK
Prevailing Secu
Trustzone-assis
**Cerdeira et al.**

n 2024: Attacking the
alaxy A * Boot Chain by
**ssi Bellom et al.**

SIVECON

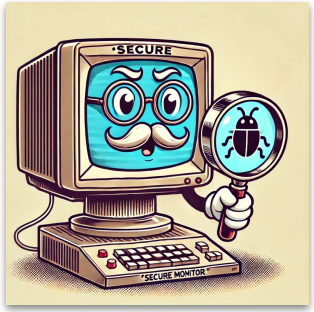
OTA Fix For BootROM Vulnerabilities
by **Szabo et al.**

# Fuzzing on Hardware is hard…




Experience with TEEzz (see Hardwear.io 2023)

# Fuzzing EL3 – A Wishlist




Introspection

- Detecting bugs
- Coverage-guided fuzzing
- Triaging bugs


Reset target state

- Reproducibility
- Determinism


Scalability


The Rise of Rehosting

# Simple Plan

# Full vs. Partial Rehosting?

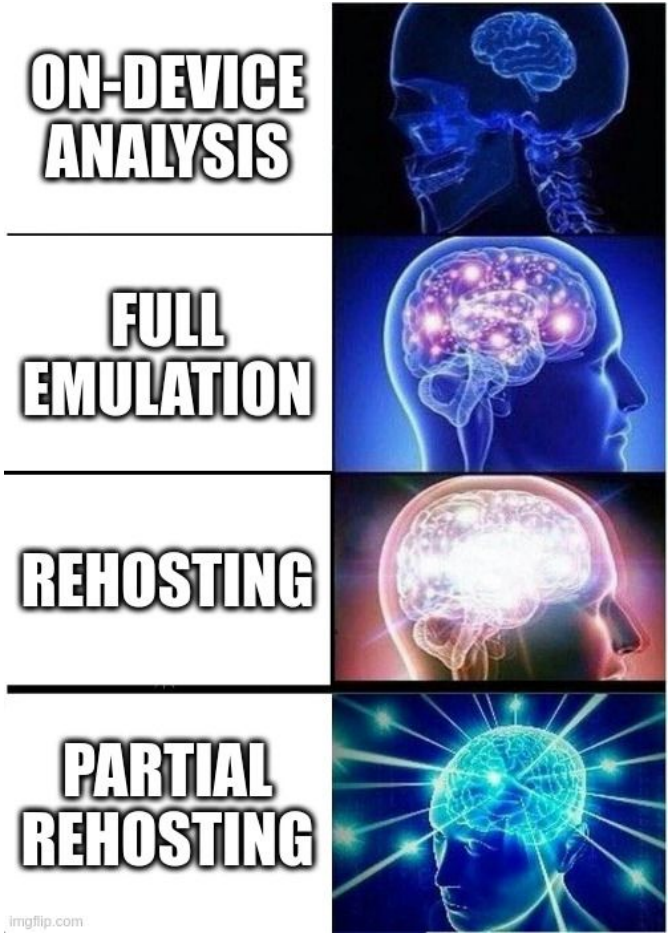
We cannot:

1. Work with hardware

2. Build a fully-fledged Android emulator

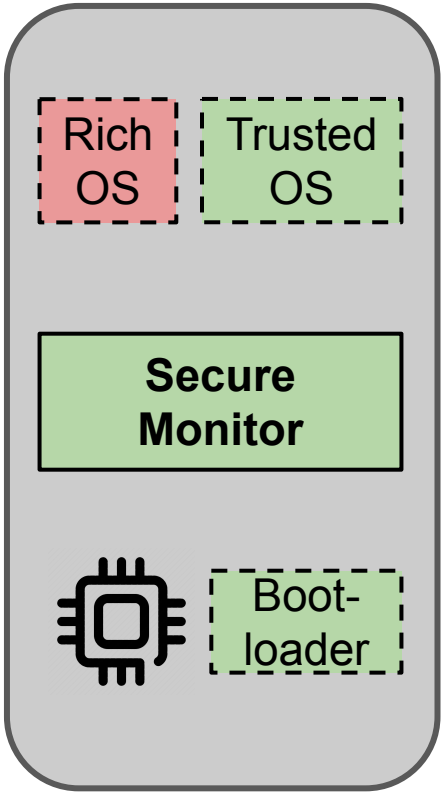
3. Implement a lot of unnecessary peripherals in QEMU

We only run the Secure Monitor!

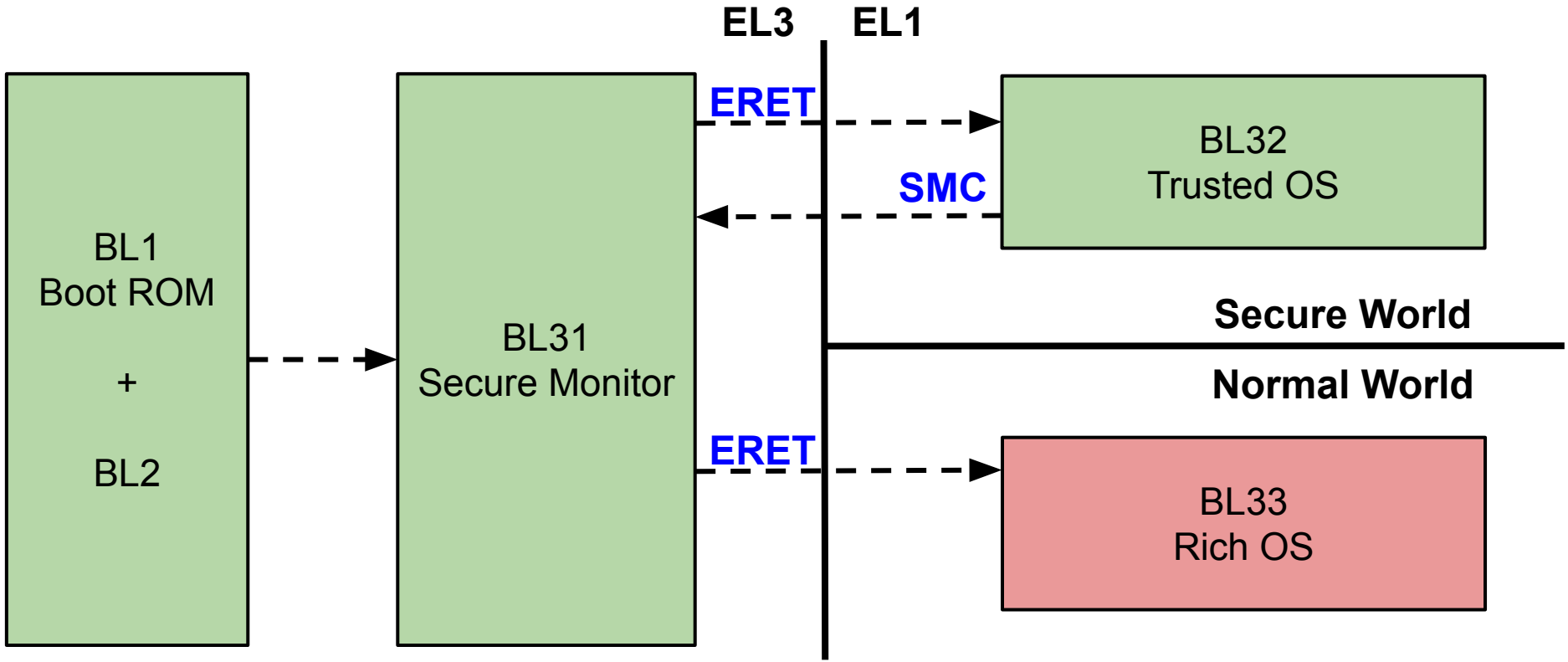


# Partial Rehosting of Secure Monitors


- Dependency on software components
- Missing hardware interactions



# Software Dependencies



# Software Dependencies - Bootloader

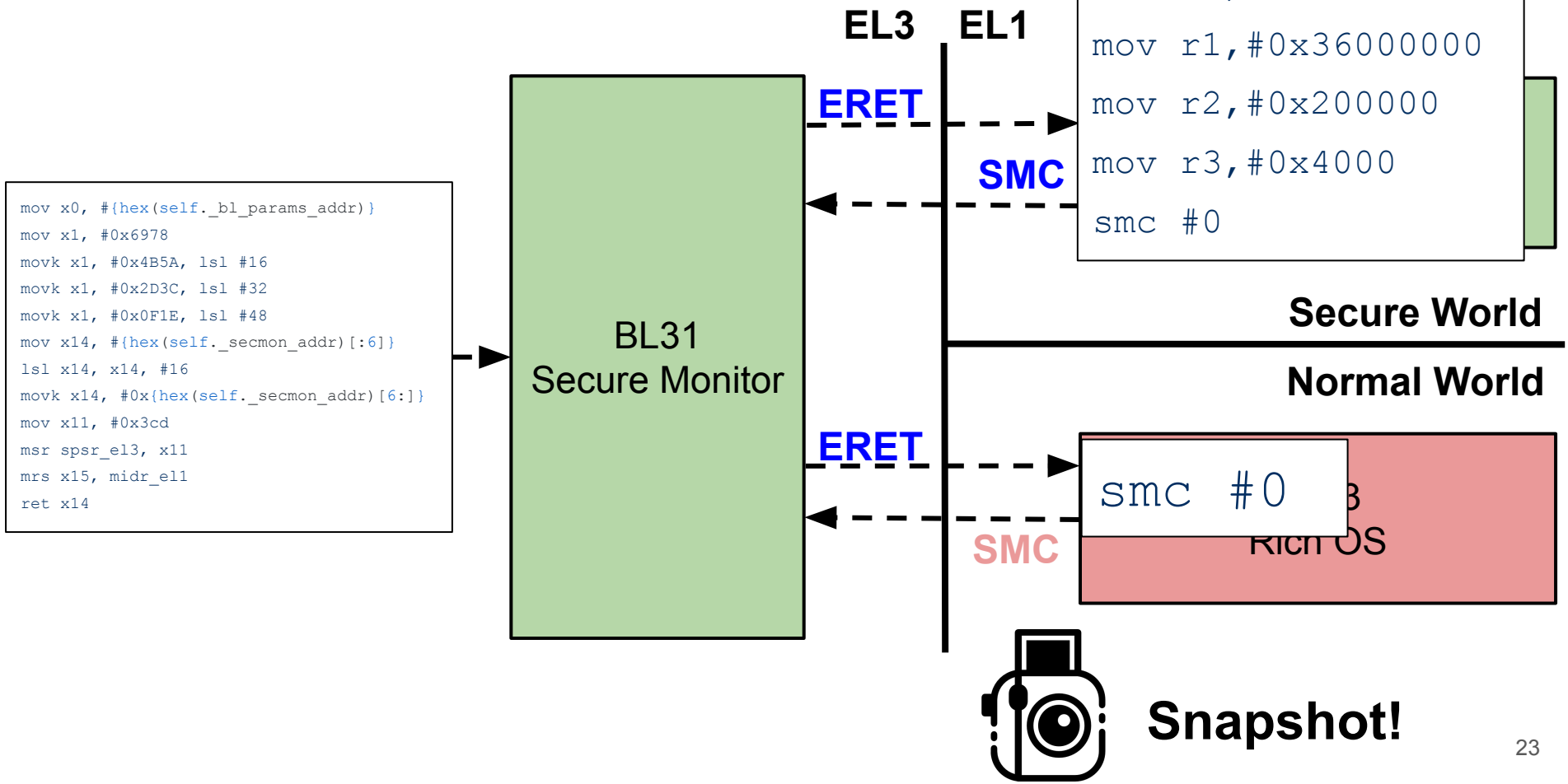
```
mov x0, #{hex(self._bl_params_addr)}
mov x1, #0x6978
movk x1, #0x4B5A, lsl #16
movk x1, #0x2D3C, lsl #32
movk x1, #0x0F1E, lsl #48
mov x14, #{hex(self._secmon_addr)[:6]}
lsl x14, x14, #16
movk x14, #0x{hex(self._secmon_addr)[6:]}
mov x11, #0x3cd
msr spsr_el3, x11
mrs x15, midr_el1
ret x14
```

```
typedef struct entry_point_info {
      param_header_t h;
      unintptr_t pc;
      unint32_t spsr;
      aapcs64 _params_t args;
} entry_point_info_t;
```

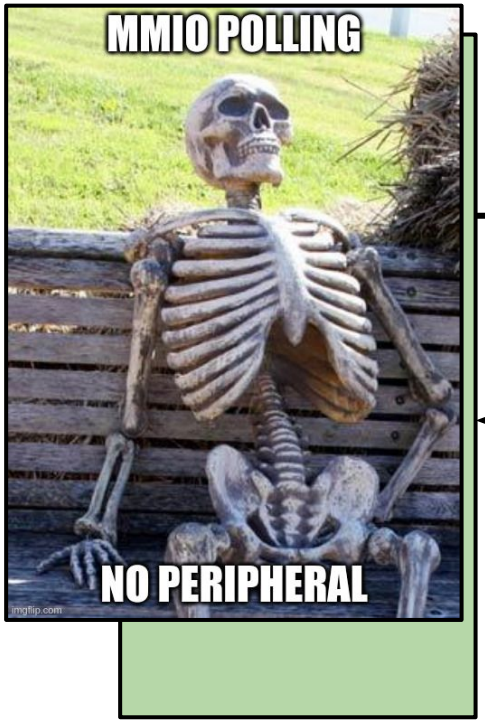
prepare entry addr

configure CPU state

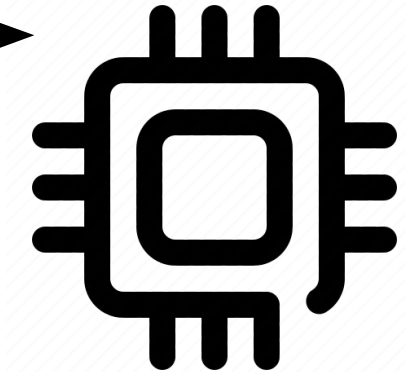
# Software Dependencies



# MMIO Interactions for Configuration




mmio_write(addr, config_value)



```
while(true)
        if(mmio_read(addr) == OK)
                break;
```

# Handling Hardware Dependencies


```
write_register("x0", 0x3)
```

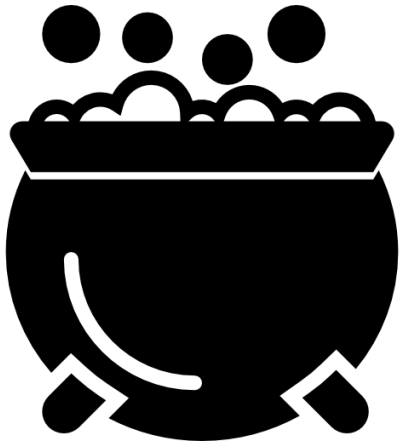
```
write_register("x0", 0x1)
```

```
execute_command("human-monitor-command",
{"command-line": "savevm booted"}))
```

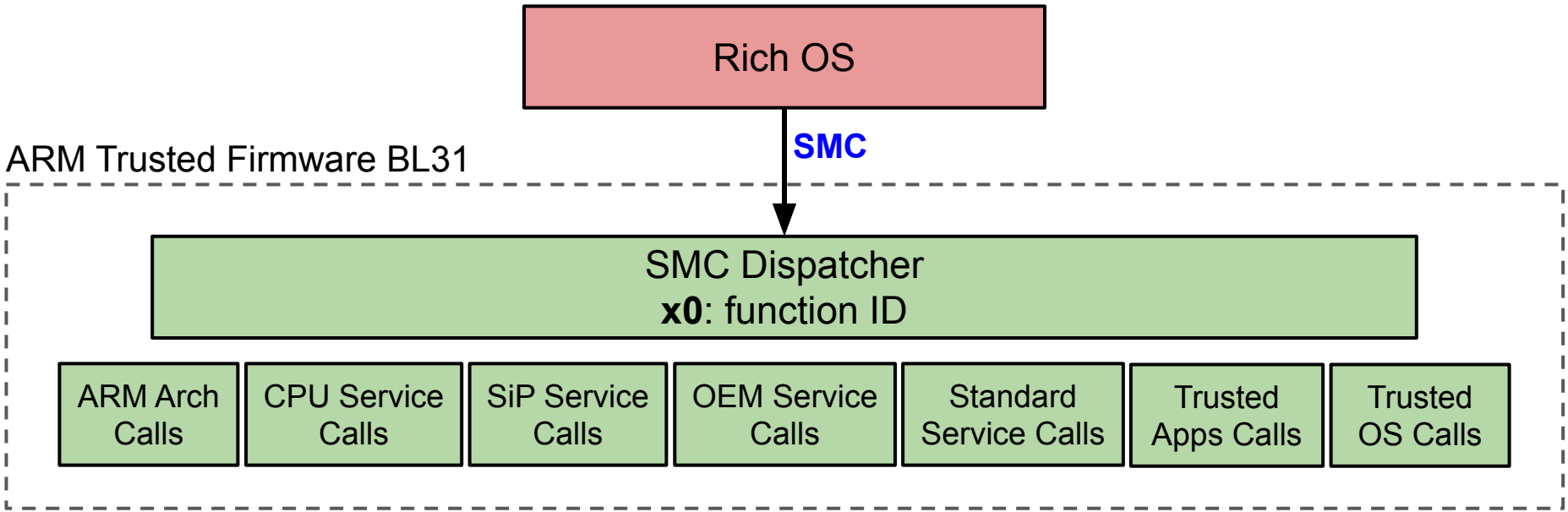
```
to make snapshot
    0x480000: "making after SecMon is finished booting -> first address in NW",
}
```

# Let’s brew a Secure Monitor Fuzzer



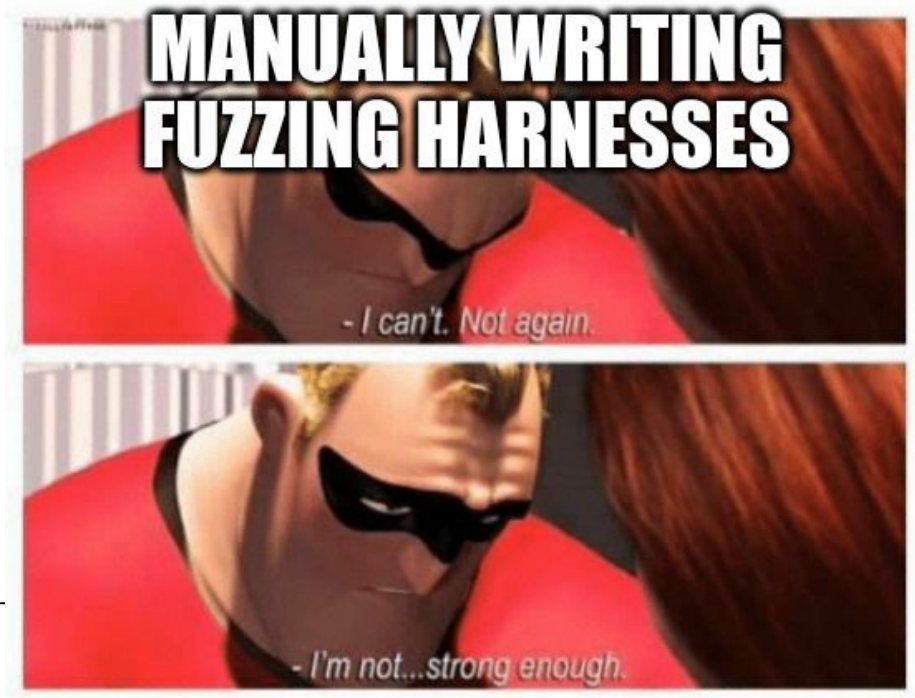
# Fuzzing Secure Monitors - SMC Interface



# Runtime Services

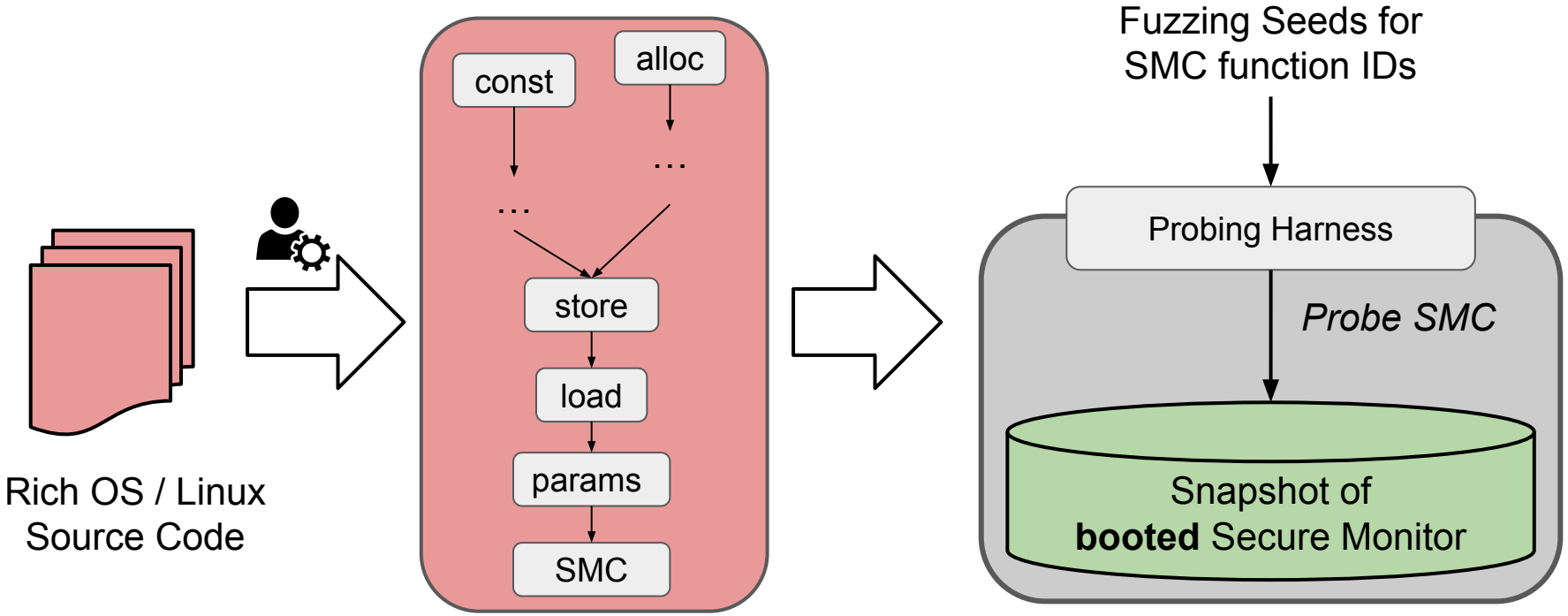
```
typedef uintptr_t (*rt_svc_handle_t)(
    uint32_t smc_fid,
    u_register_t x1,
    u_register_t x2,
    u_register_t x3,
    u_register_t x4,
    void *cookie,
    void *handle,
    u_register_t flags);
```

Several tens of runtime services with unique APIs…

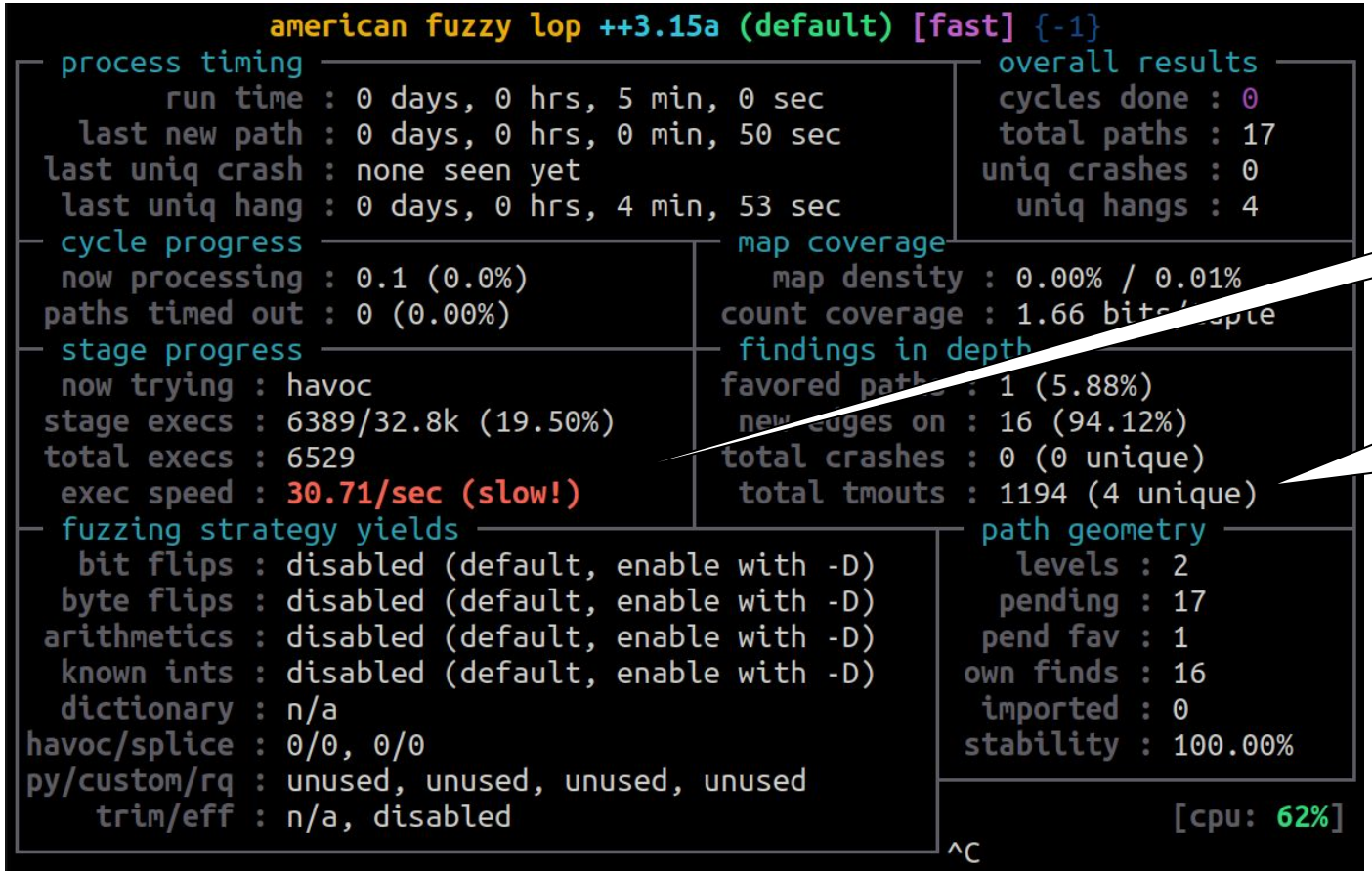


# The Rich OS uses the SMC Interface

# EL3XIR's Fuzzing Harness Synthesis Pipeline



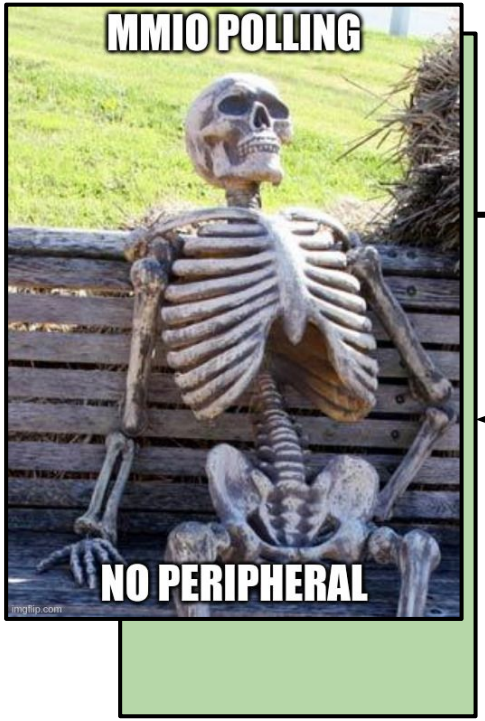
# Ok all set? Let's fuzz!




slow!

A **lot** of hangs and timeouts…

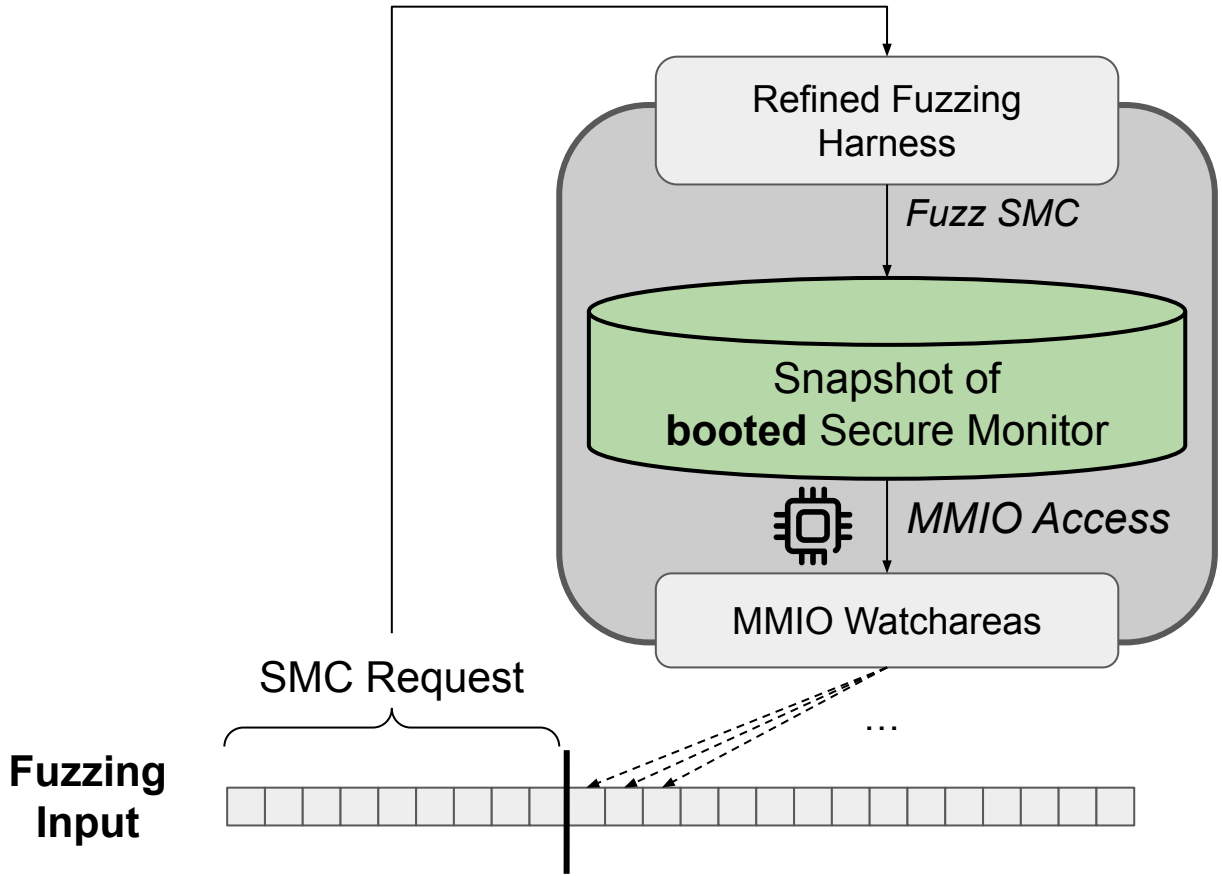
# MMIO (obviously) happens after boot… **a lot**



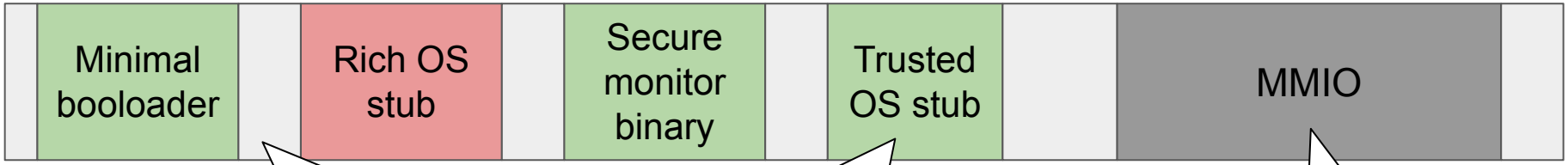

```
mmio_write(addr, value)
```

```
while(true)
    if(mmio_read(addr) == expected)
        break;
```

# EL3XIR's Reflected Peripheral Modeling



# Physical Memory Layout

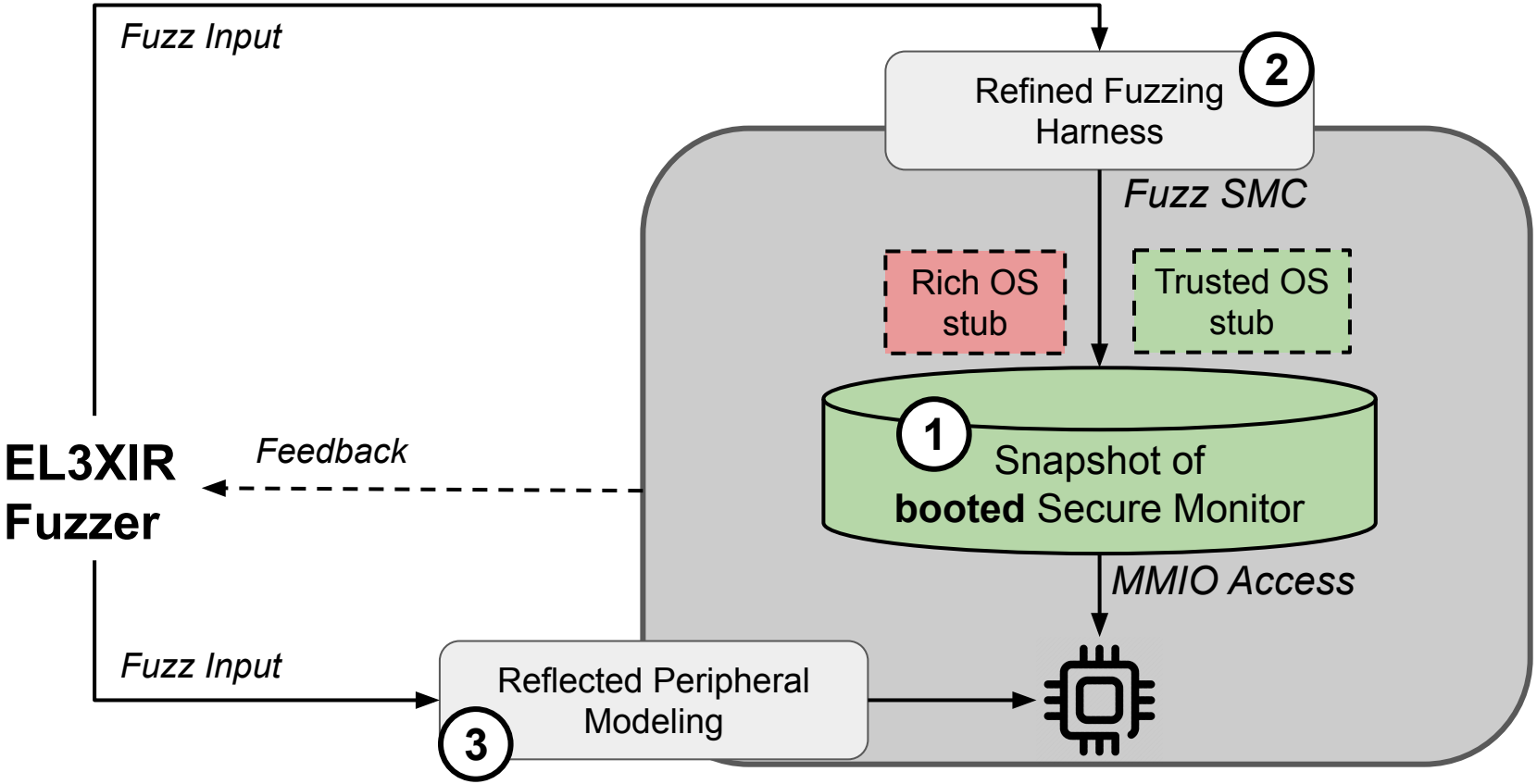



add_memory_range()

add_secure_stub()

add_in_memory_buffer_peripheral()
…
**request_afl_data**(value, size)
…

# EL3XIR

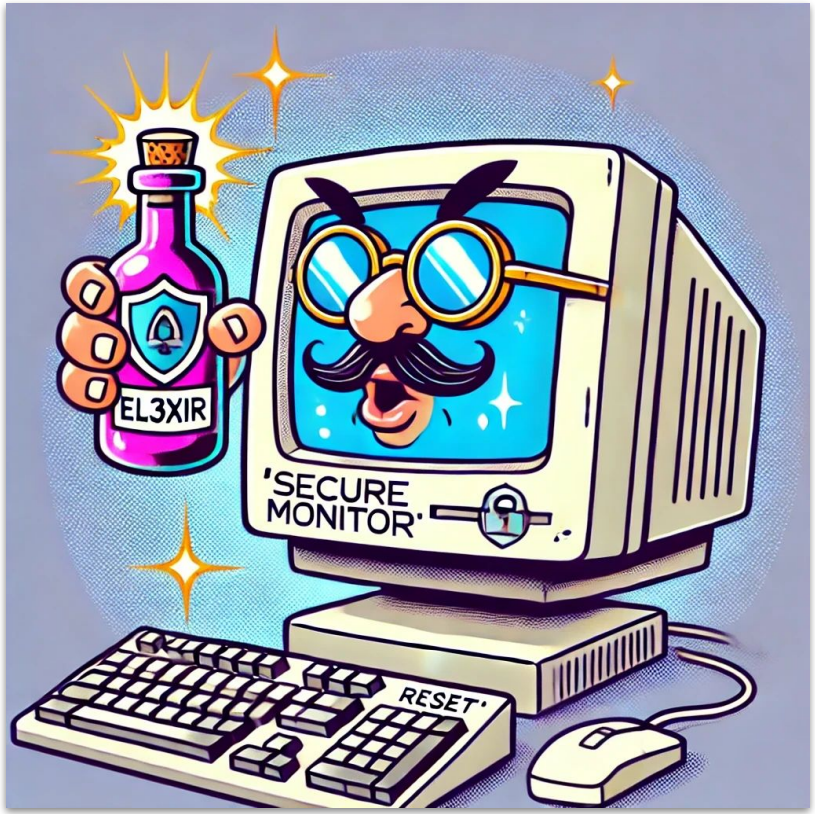

Fuzz Input
Refined Fuzzing Harness
2
Fuzz SMC
Rich OS stub
Trusted OS stub
1
Snapshot of **booted** Secure Monitor
MMIO Access
EL3XIR Fuzzer
Feedback
Fuzz Input
Reflected Peripheral Modeling
3

# Found EL3 Bugs and CVEs

- 7 targets from 6 different vendors
  - 4 open-source, 3 closed-source
- EL3XIR triggered 34 bugs (**17** security relevant) in 5 targets
- Responsible disclosure resulted in 6 CVEs plus 11 confirmed bugs

**CVE-2022-38787, CVE-2023-22327 (5 different bugs), CVE-2023-49614, CVE-2024-22390, CVE-2023-31339, CVE-2023-49100**

# Healing Secure Monitors with EL3XIR




INFILTRATE 2019
GUANXING WEN
EL3 TOUR: GET THE ULTIMATE PRIVILEGE OF ANDROID PHONE
INFILTRATE

# Demo: Rediscover EL3 Bug

- P20 Lite
- Released 2018
- Android 8.0
- HiSilicon Kirin 659 SoC


[8]

# Demo Time!

```
ADRP X1, #0x35C20E98@PAGE
STR X0, [X1, #0x35C20E98@PAGEOFF]
```

*(0x35c20E98) = X0

```
ADRP X1, #0x35C20E98@PAGE
LDR X19, [X1, #0x35C20E98@PAGEOFF]
ADD X20, X19, #6, LSL#12
CBZ X0, loc_35C08C38
LDR X1, [X20, #0xA28]
BLR X1
```

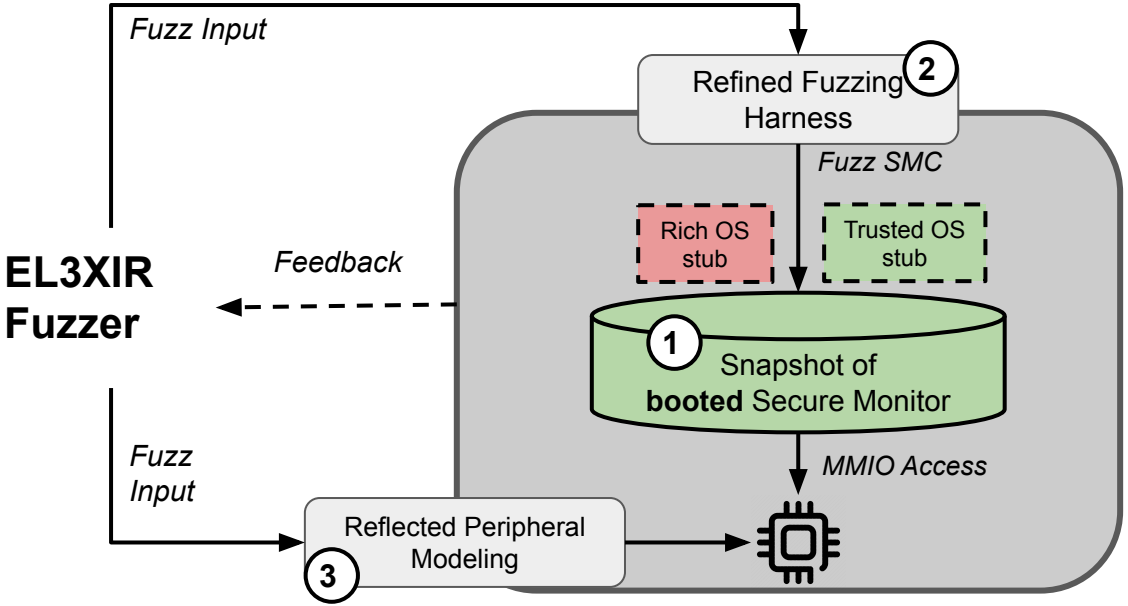
⟹ X19 = *(0x35c20E98)
⟹ X20 = X19 + 0x6000
⟹ if !X0 return
⟹ X1 = *(X20 + 0xa28)
⟹ X1()

**Sploit available @ github.com/HexHive/EL3XIR**

THX Moritz!

Infiltrate 2019 “EL3 Tour” Recording: https://vimeo.com/335948808

# EL3XIR: Fuzzing COTS Secure Monitors




github.com/HexHive/EL3XIR

ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE

ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL

ARTIFACT EVALUATED usenix ASSOCIATION REPRODUCED


christian.lindenmeier@fau.de
𝕏@_chli_

marcel.busch@epfl.ch
𝕏@0ddc0de

# Attribution


[1] https://tradersunion.com/interesting-articles/trading-equipment/mobile-phones/
[2] https://www.goodfon.ru/hi-tech/wallpaper-htc-nexus-9-by-google-tablet.html
[3] https://www.deviantart.com/kcajd/art/Sony-6-D-HD-Smart-T-V-858288157
[4] https://www.donanimhaber.com/Samsungun-SmartThings-uygulamasi-yenilendi--91544
[5] https://www.flipkart.com/sony-xav-ax-8100-android-auto-apple-carplay-hdmi-car-stereo/p/itm8f2672e74acaa
[6] https://djistor.ru/goods/Komplekt-oborudovaniya-sputnikovoj-svyazi-Starlink-gen-2
[7] https://drones.measurusa.com/products/dji-mavic-3-multispectral
[8] https://guide-images.cdn.ifixit.com/igi/BUAZB4YMHDZ3oIll.large