

# EL3XIR: Fuzzing COTS Secure Monitors

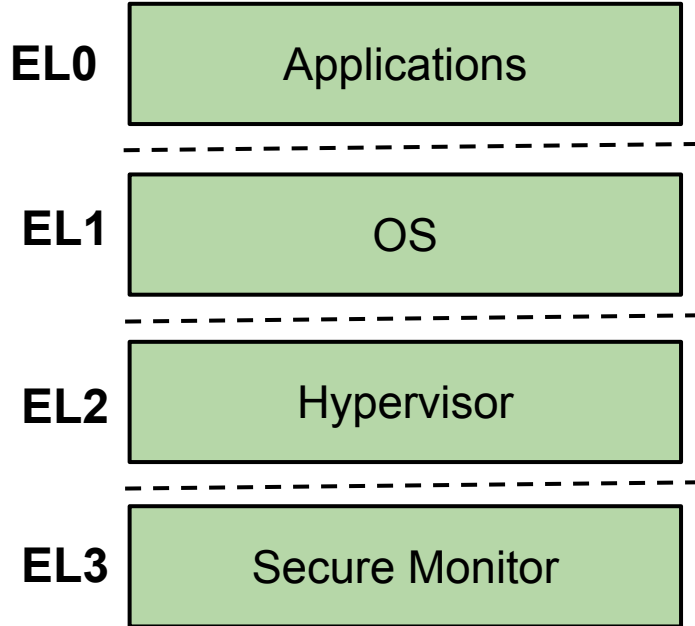
Marcel Busch & Christian Lindenmeier

[christian.lindenmeier@fau.de](mailto:christian.lindenmeier@fau.de)  
[marcel.busch@epfl.ch](mailto:marcel.busch@epfl.ch)



# This Talk

## ARM Cortex-A



Fuzzing



Rehosting



# \$ whoarewe

**Marcel Busch** X @0ddc0de

- PhD graduate from FAU Erlangen-Nuremberg
- PostDoc with HexHive @ EPFL
- Works on
  - Embedded systems security
  - Android TEEs
  - Fuzzing

**Christian Lindenmeier** X @\_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

ARM-software / arm-trusted-firmware

Code Pull requests 11 Actions Projects Wiki Security Insights

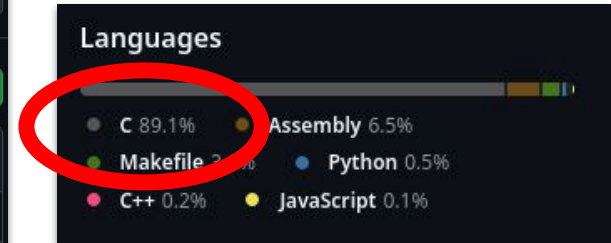
arm-trusted-firmware Public Watch 277

master 5 Branches 125 Tags

Go to file Add file Code

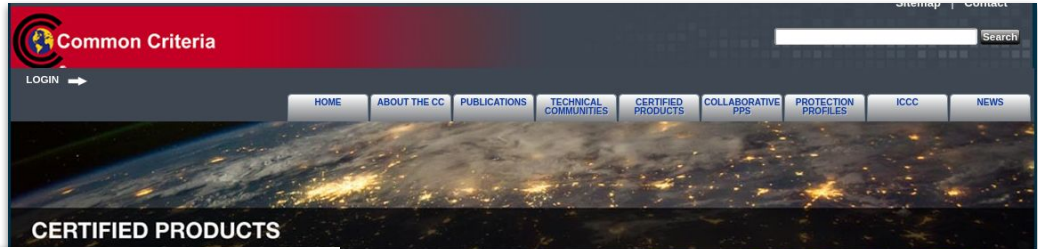
ManishVB-Arm and TrustedFirmware Code Revi... Merge changes from topic ... 742d0e6 · 2 days ago 15,465 Commits

Commit	Message	Time
.husky	build(npm): adhere to Husky deprecation notice	7 months ago
bl1	fix(cpus): modify the fix for Cortex-A75 erratum 764081	2 weeks ago
bl2	Merge changes from topic "early_console" into integration	5 months ago
bl2u	build: remove MAKE_BUILD_STRINGS function	6 months ago
bl31	feat(cm): handle asymmetry for FEAT_TCR2	last month



# 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 sha256: 0cb293569ae1a33477f27b0acd58bd455c4d88bee53af3584facb2bc6e2f74c5
TEE binary reference	Samsung Secure OS Release Version 5.0.0.0 sha256: df6e524d7accacbc722776660ffd698b023a6c55ba8cbe3b0e5cacb5b1d789db
ATF binary reference	Zagreb-SP1A-2124V1-2125R1 sha256: 0f78b256b193620bca0f861b550371c14ac5e3b7c40d049fe6ff63a27d3d0512
TEE binary developer	Samsung Electronics Co., Ltd.

Statistics [Download CSV](#) [Archived Certified Products](#)

---

ifications with claims of compliance against Common Criteria assurance components of either:

maintained in accordance with CCRA Annex K, with assurance activities selected from Evaluation Assurance Levels up to and including Technical Community endorsed by the Management Committee; or

Assurance Level 3 or higher, but does not claim compliance to a collaborative Protection Profile, then for purposes it should be treated as equivalent to Evaluation Assurance Level 2.

mutually recognized CC certificates over time. Certificates will remain on the CPL for five years. Effective 1 June 2019, (more from the date of certificate issuance) will be moved to an Archive list on the CCRA portal, unless the validity period has

Countries  All Compliances  All Categories

Vendor	Product Certificate	Date Certificate Issued	Certificate Validity Expiration Date	Compliance	Scheme	Category
Samsung Electronics Co., Ltd.	CCRA Certificate	2024-02-01	2029-02-01	EAL2+	FR	Trusted Computing



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





# 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

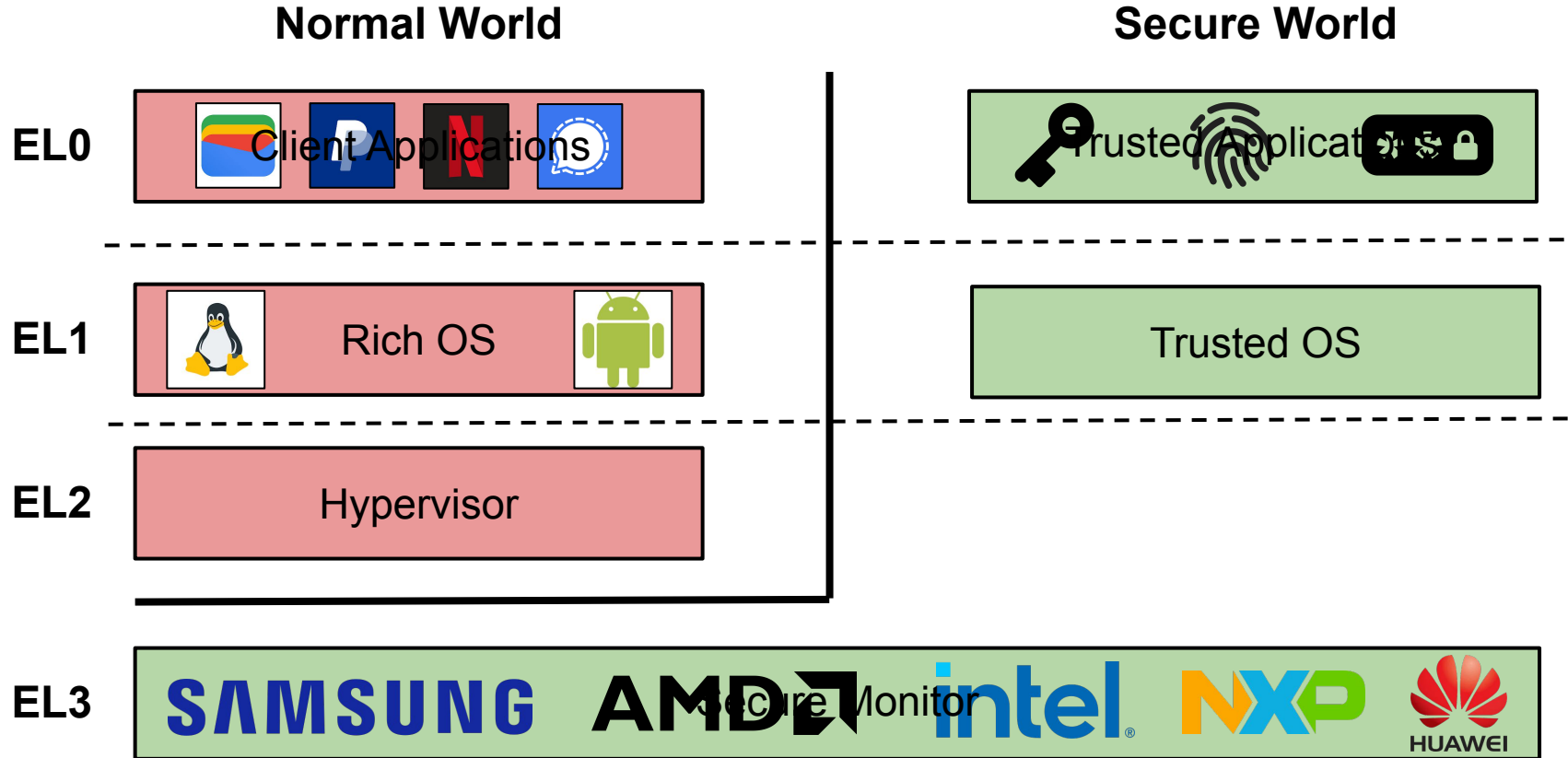
⇒ Fragmentation

⇒ (Potentially) less audited code

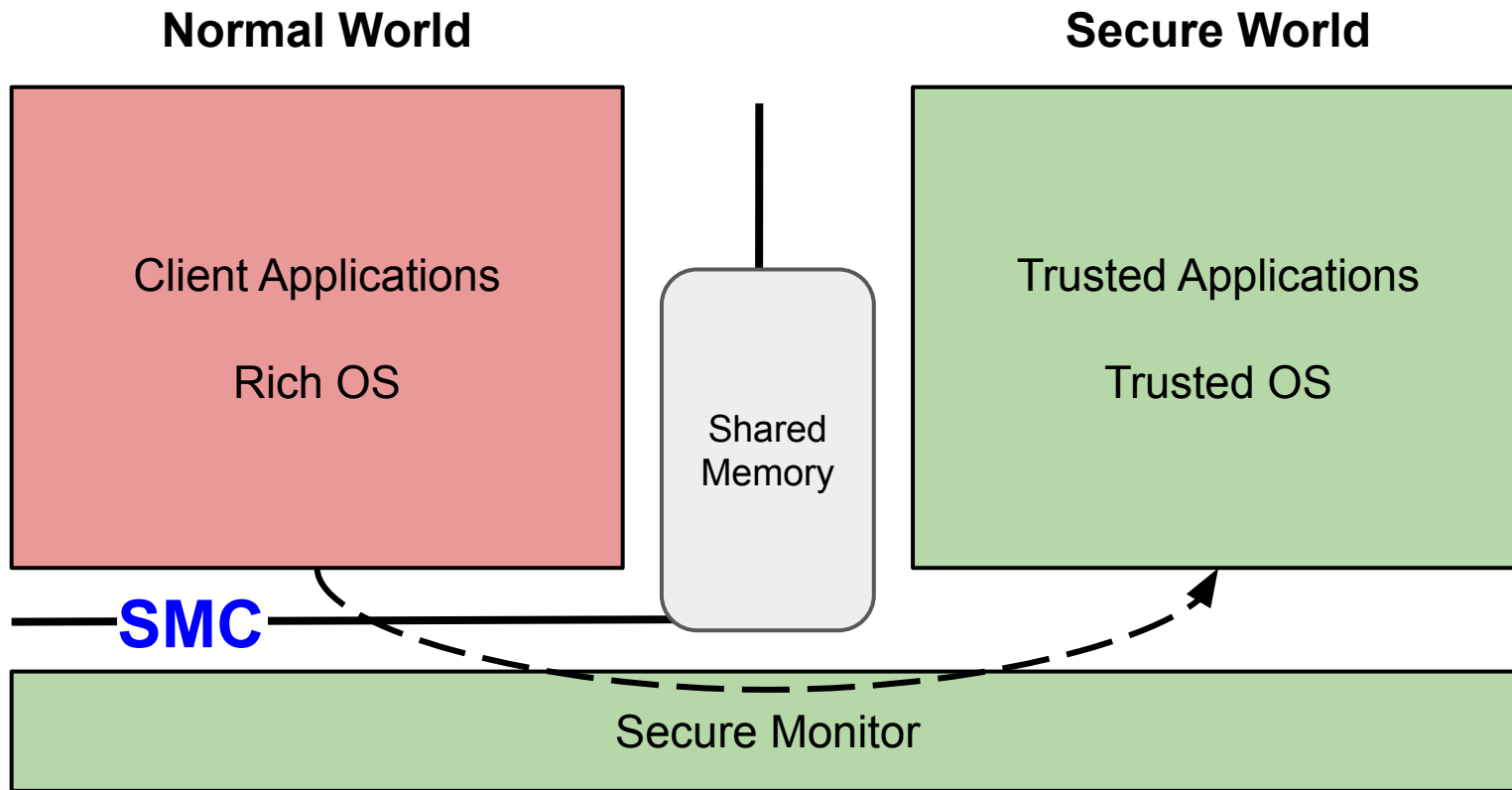
Extensions for ARM Trusted Firmware typically closed-source



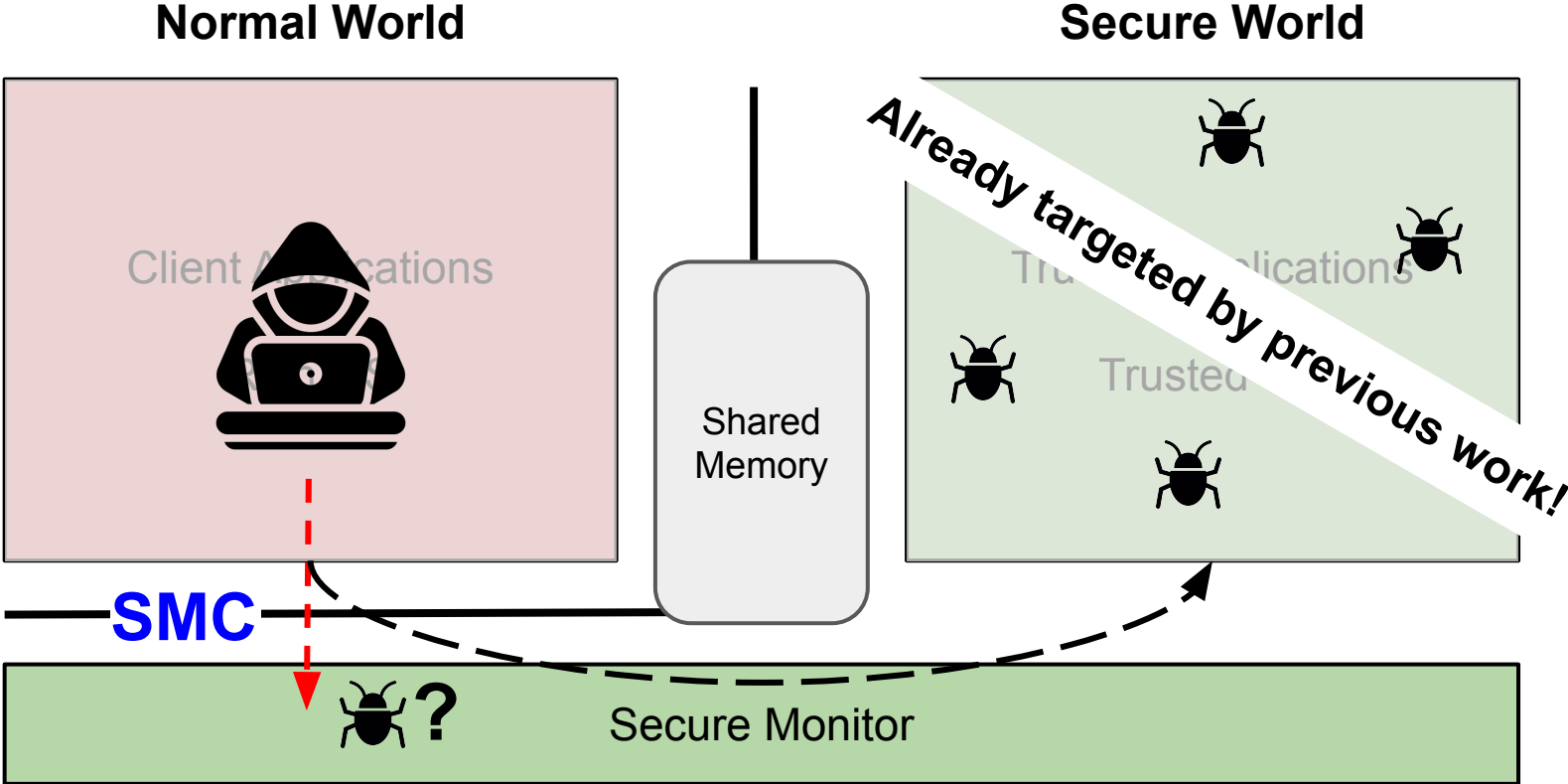
# ARMv8-A Privilege Levels and Execution Modes



# 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**



2019

20



S&P 2020: Sok  
Prevailing Sec  
Trustzone-assis  
**Cardeira et al.**



imgflip.com

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

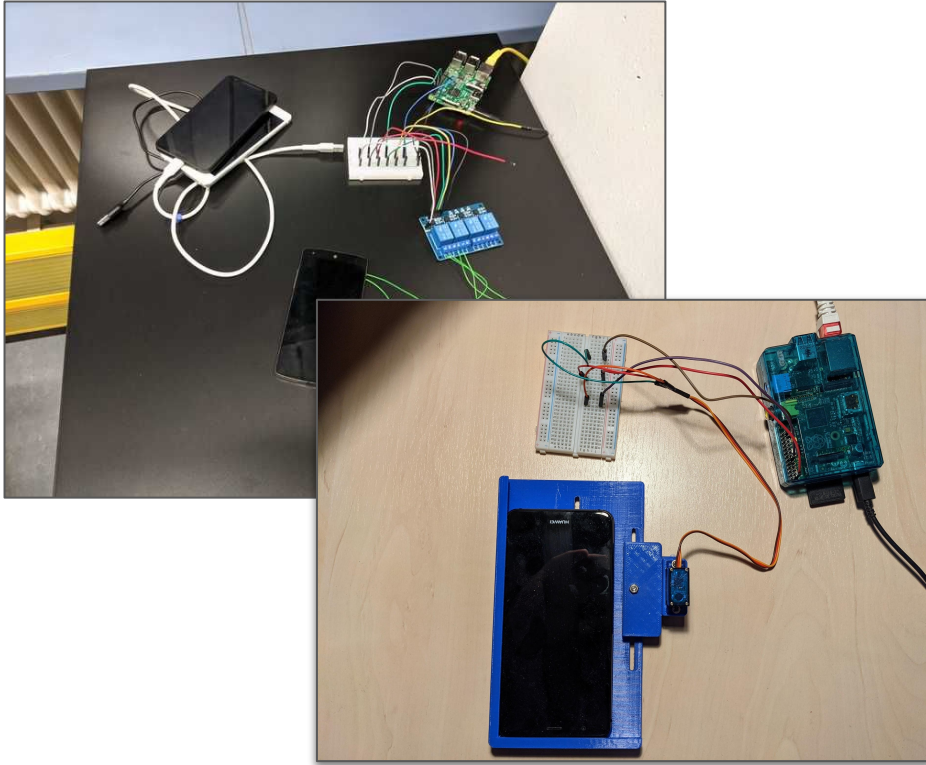


2024

in 2024: Attacking the  
Galaxy A \* Boot Chain by  
**ssi Bellom et al.**

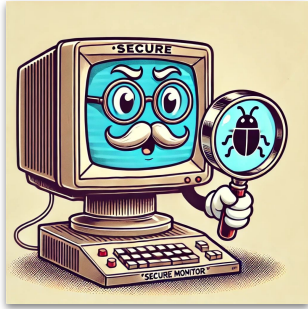


# Fuzzing on Hardware is hard...



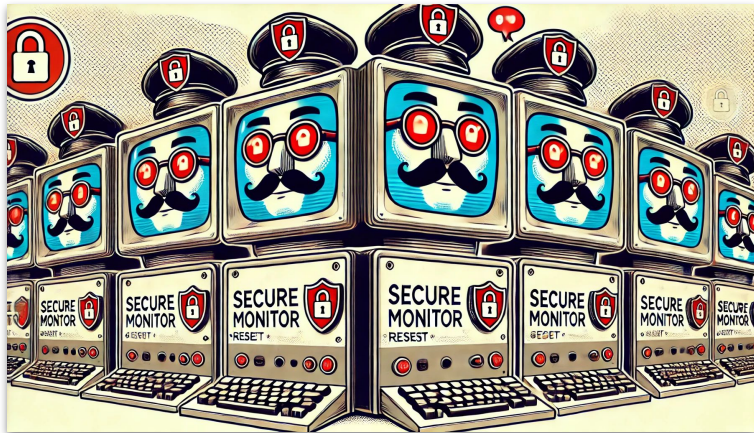
Experience with TEEzz (see [Hardwear.io](https://www.hardwear.io) 2023)

# Fuzzing EL3 – A Wishlist



## Introspection

- Detecting bugs
- Coverage-guided fuzzing
- Triaging bugs



## Reset target state

- Reproducibility
- Determinism



## Scalability



# The Rise of Rehosting

A sunset over a body of water with the title 'The Rise of Rehosting' overlaid in white text. The sun is low on the horizon, creating a bright orange and yellow glow that reflects on the water's surface. The sky transitions from a deep orange near the horizon to a dark, dusky blue at the top. The water is dark with a shimmering path of light leading from the sun to the foreground.

# Simple Plan



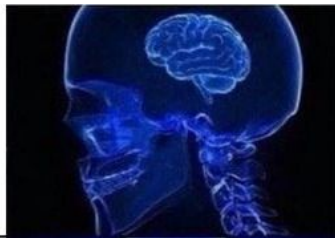
# Full vs. Partial Rehosting?

We cannot:

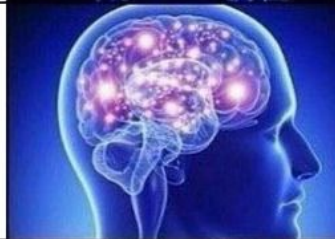
1. Work with hardware
2. Build a fully-fledged Android emulator
3. Implement a lot of unnecessary peripherals in QEMU

We only run the Secure Monitor!

**ON-DEVICE  
ANALYSIS**



**FULL  
EMULATION**



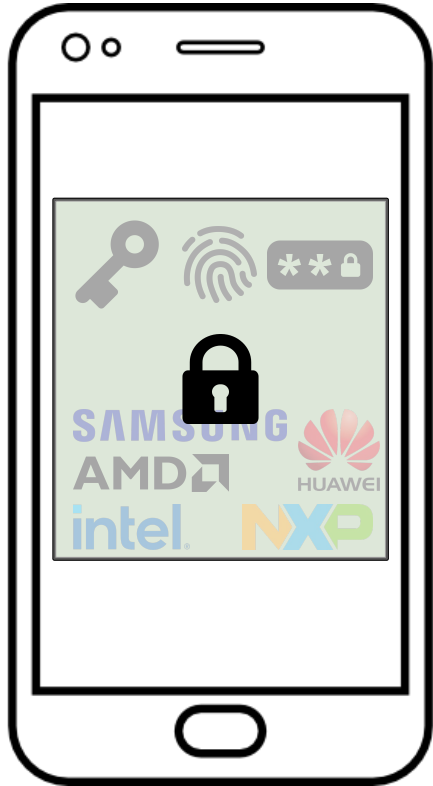
**REHOSTING**



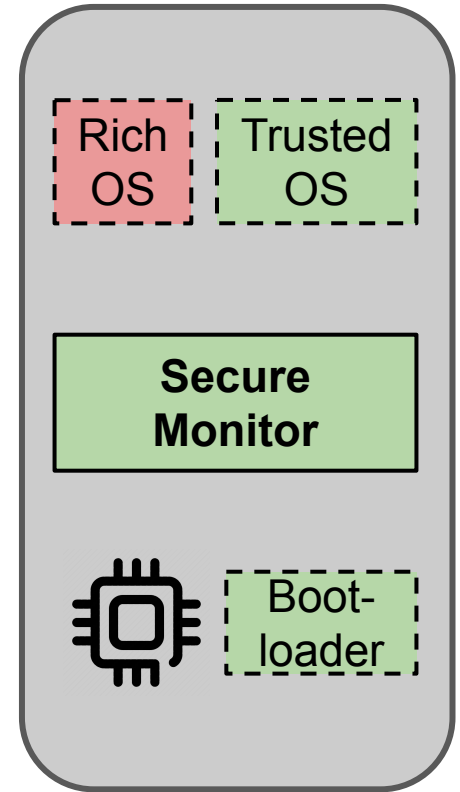
**PARTIAL  
REHOSTING**



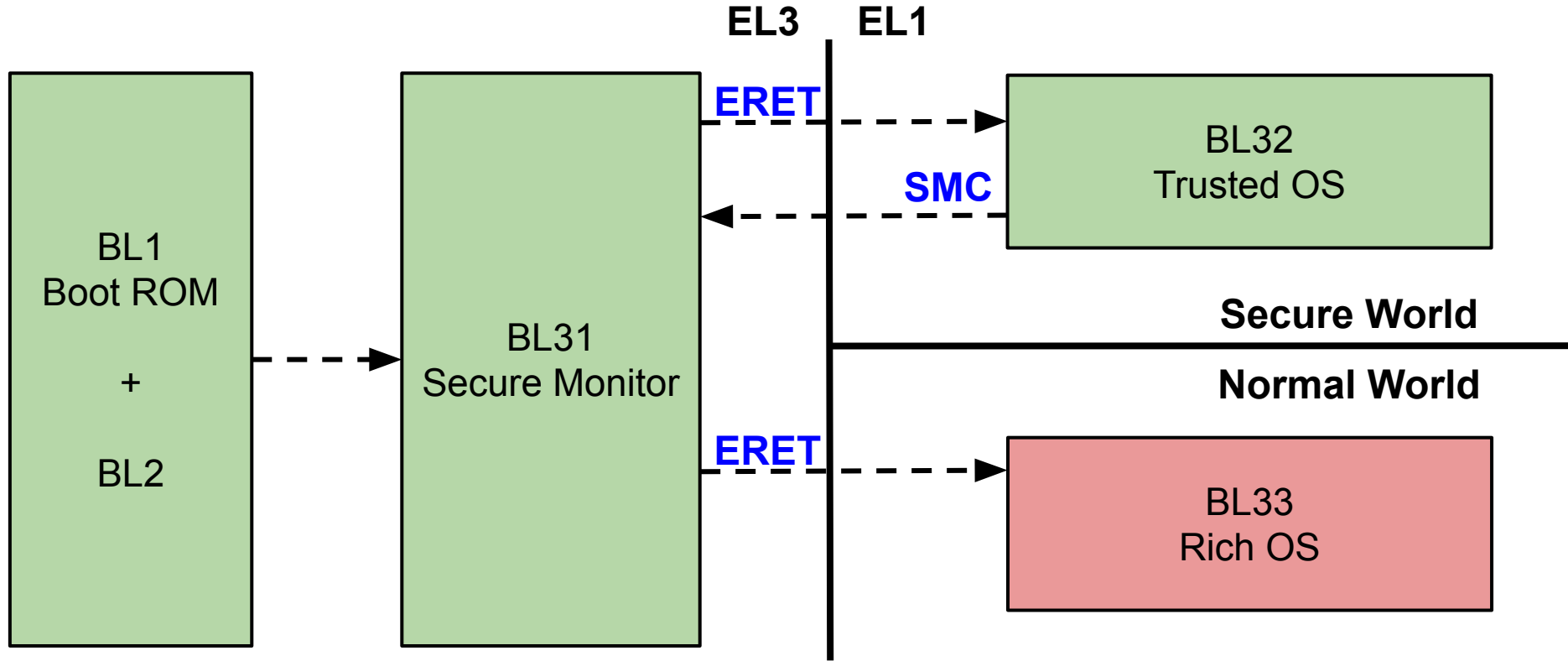
# Partial Rehosting of Secure Monitors



- 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
```

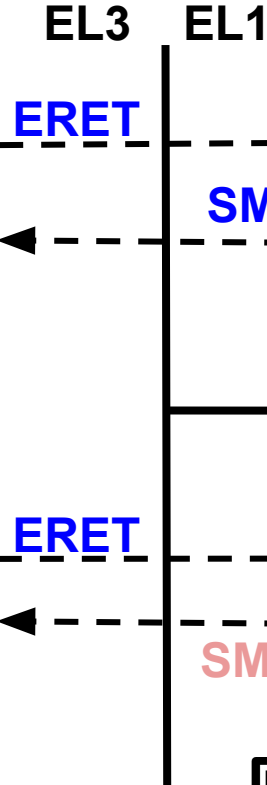
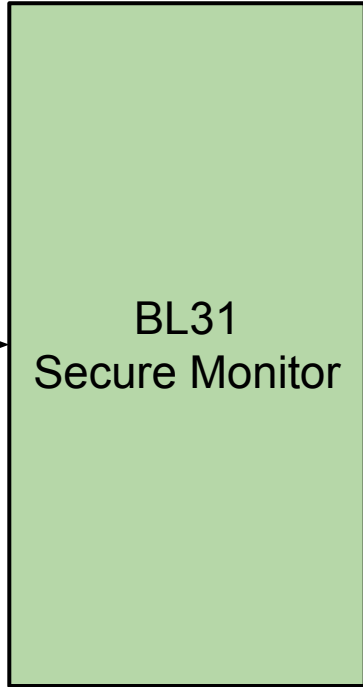
```
typedef struct entry_point_info {
    param_header_t h;
    uintptr_t pc;
    uint32_t spsr;
    aapcs64 _params_t args;
} entry_point_info_t;
```

prepare entry addr

configure CPU state

# Software Dependencies

```
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
```



```
mov r0, #0xb2000000  
mov r1, #0x36000000  
mov r2, #0x200000  
mov r3, #0x4000  
smc #0
```

**Secure World**

**Normal World**

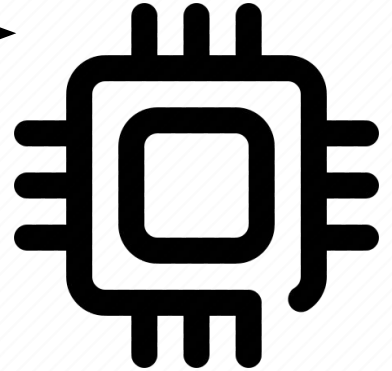


**Snapshot!**

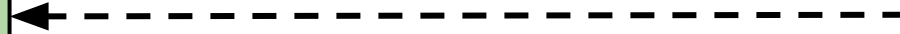
# MMIO Interactions for Configuration



`mmio_write(addr, config_value)`



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





# Handling Hardware Dependencies



to make snapshot

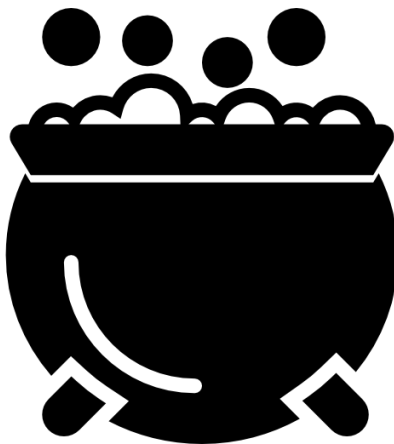
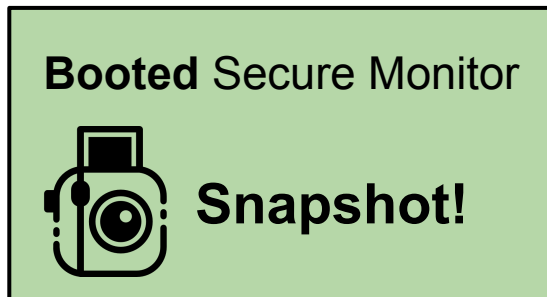
```
0x480000: "making after SecMon is finished  
booting -> first address in NW",  
}
```

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

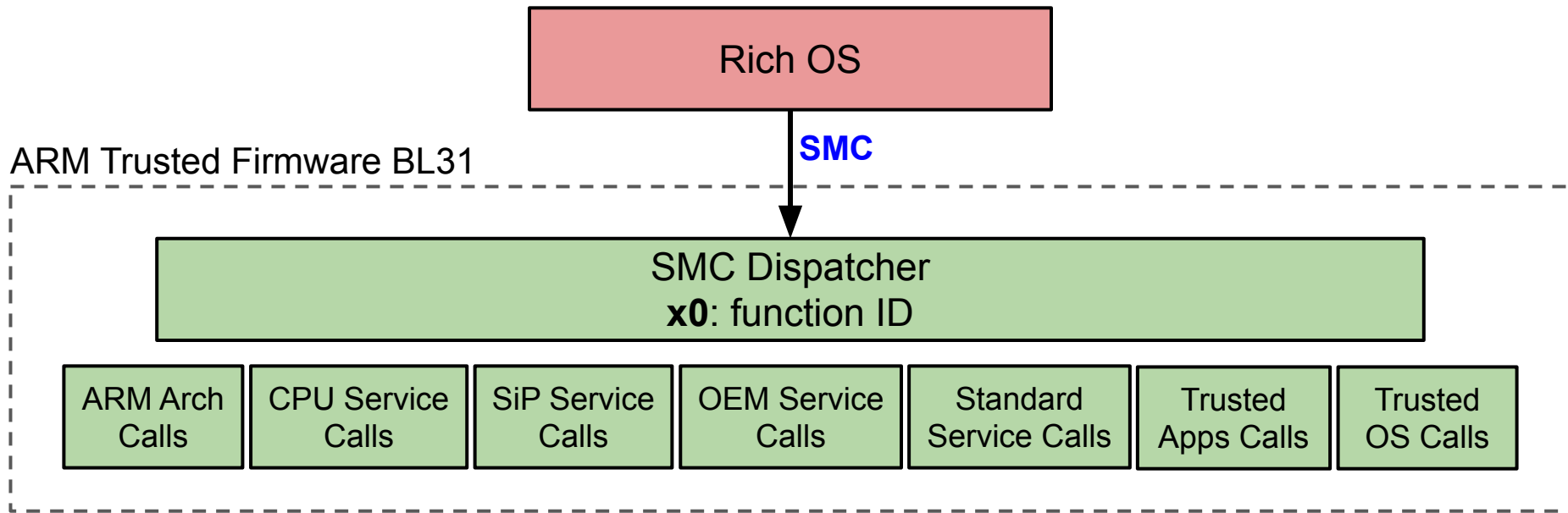
```
write_register("x0", 0x1)
```

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

# Let's brew a Secure Monitor Fuzzer



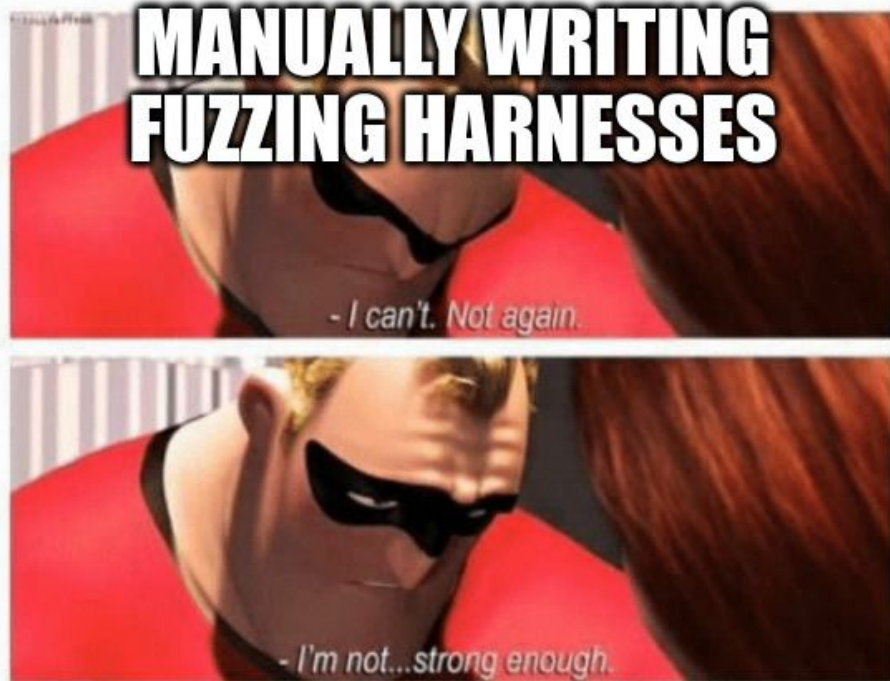
# Fuzzing Secure Monitors - SMC Interface



# 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);
```

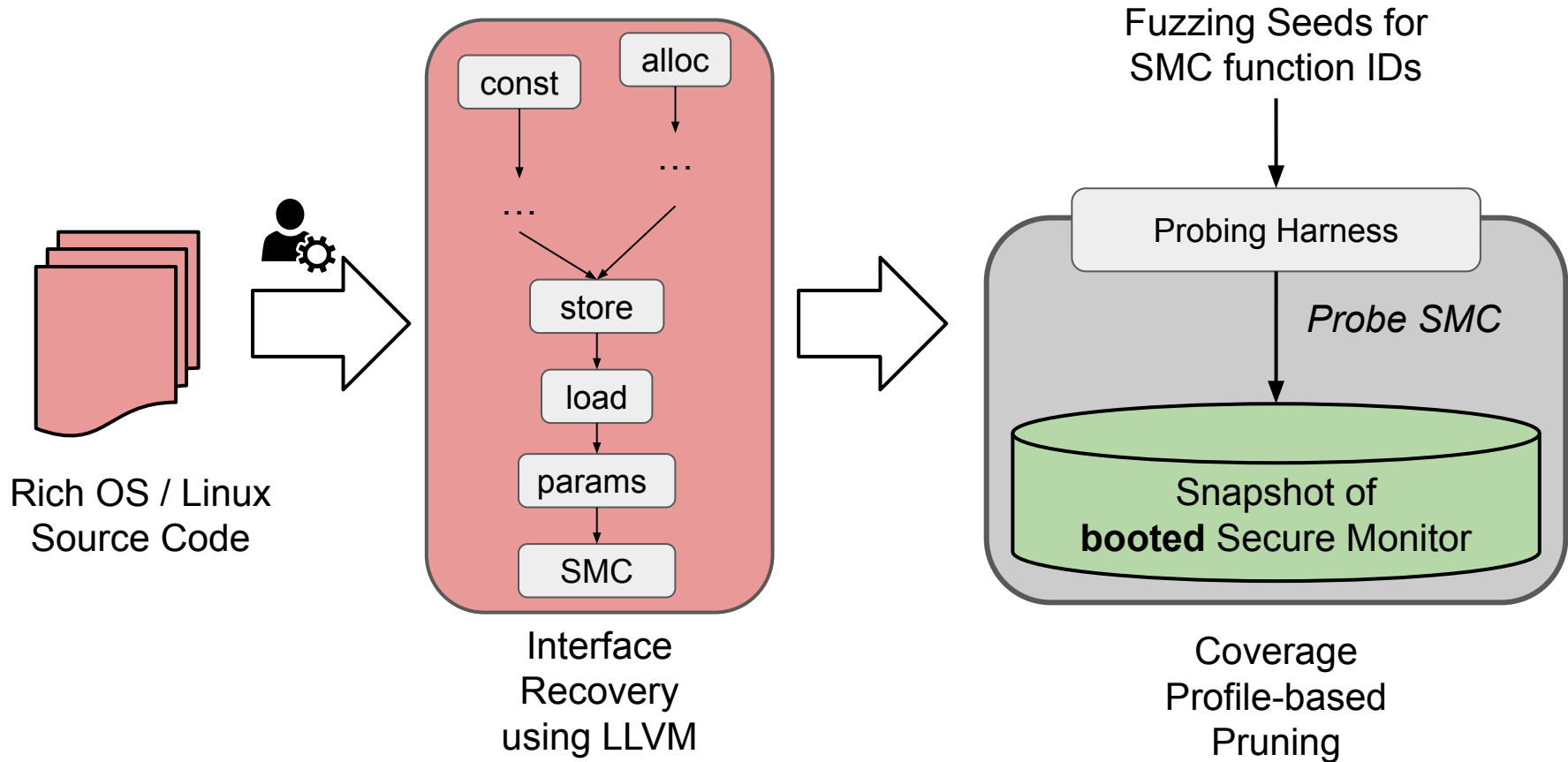
Several tens of runtime services with unique APIs...



# The Rich OS uses the SMC Interface



# EL3XIR's Fuzzing Harness Synthesis Pipeline



# Ok all set? Let's fuzz!

```
american fuzzy lop ++3.15a (default) [fast] {-1}
```

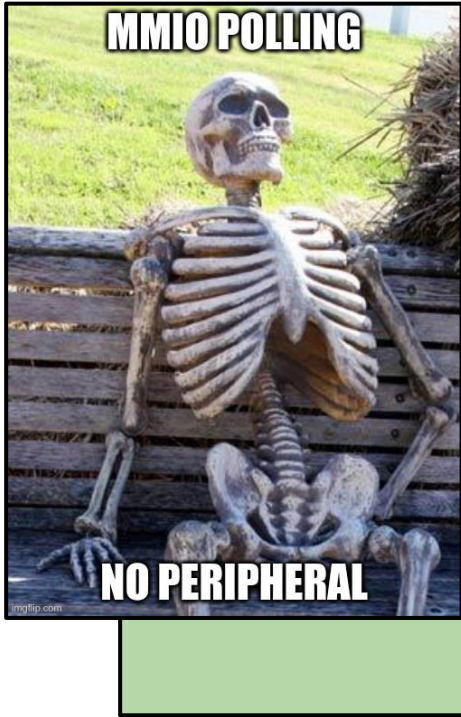
```
process timing |-----| overall results |-----|
  run time   : 0 days, 0 hrs, 5 min, 0 sec      cycles done : 0
  last new path : 0 days, 0 hrs, 0 min, 50 sec  total paths : 17
  last uniq crash : none seen yet              uniq crashes : 0
  last uniq hang : 0 days, 0 hrs, 4 min, 53 sec  uniq hangs  : 4
cycle progress |-----| map coverage |-----|
now processing : 0.1 (0.0%)                    map density : 0.00% / 0.01%
paths timed out : 0 (0.00%)                   count coverage : 1.66 bits/byte
stage progress |-----| findings in depth |-----|
now trying : havoc                             favored paths : 1 (5.88%)
stage execs : 6389/32.8k (19.50%)              new edges on  : 16 (94.12%)
total execs : 6529                             total crashes : 0 (0 unique)
exec speed  : 30.71/sec (slow!)                total tmouts  : 1194 (4 unique)
fuzzing strategy yields |-----| path geometry |-----|
bit flips   : disabled (default, enable with -D)  levels      : 2
byte flips  : disabled (default, enable with -D)  pending     : 17
arithmetics : disabled (default, enable with -D)  pend fav    : 1
known ints  : disabled (default, enable with -D)  own finds   : 16
dictionary  : n/a                               imported    : 0
havoc/splice : 0/0, 0/0                         stability   : 100.00%
py/custom/rq : unused, unused, unused, unused
trim/eff    : n/a, disabled
[CPU: 62%]
```

slow!

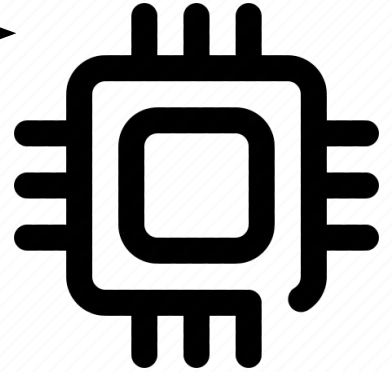
A lot of hangs and timeouts...

^C

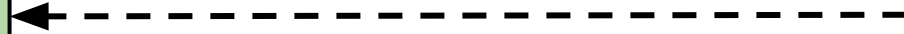
# MMIO (obviously) happens after boot... a lot



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

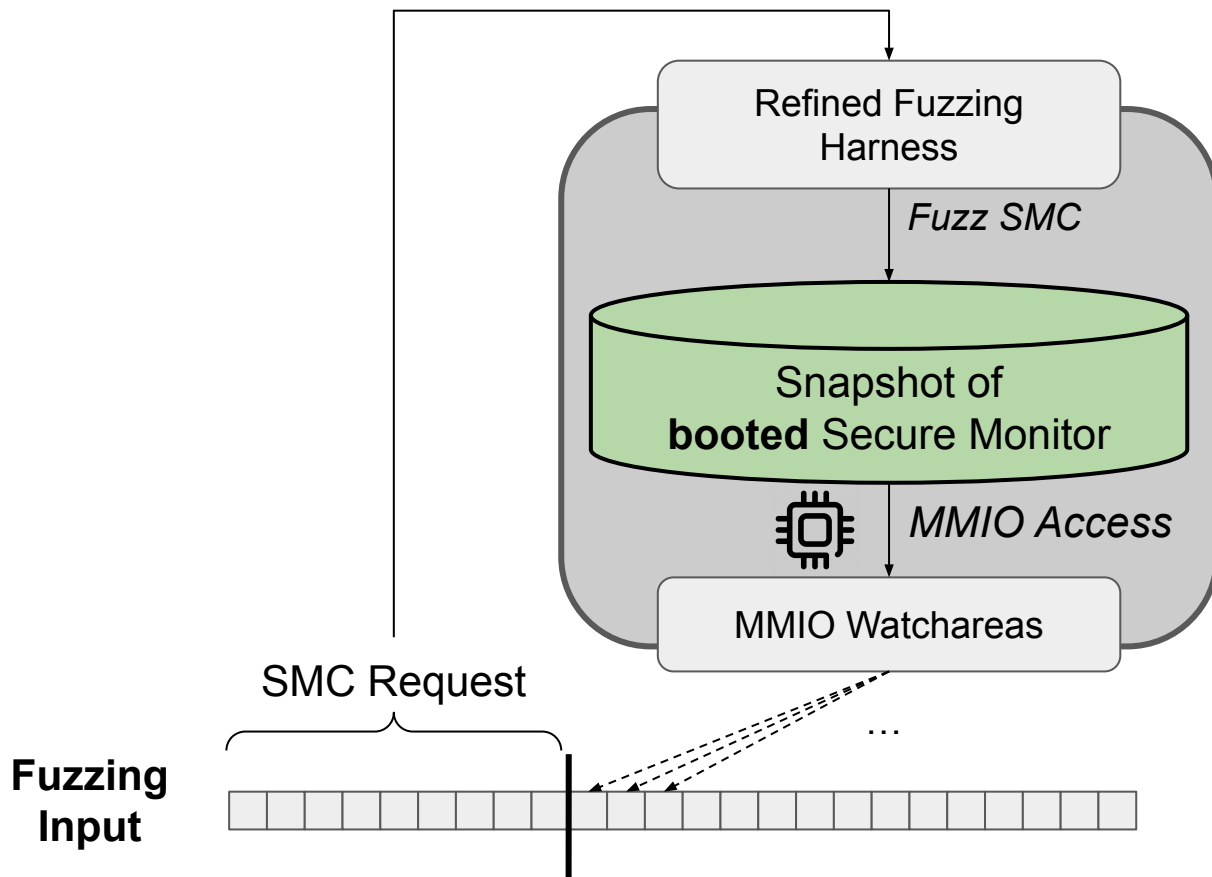


```
while(true)  
    if(mmio_read(addr) == expected)  
        break;
```

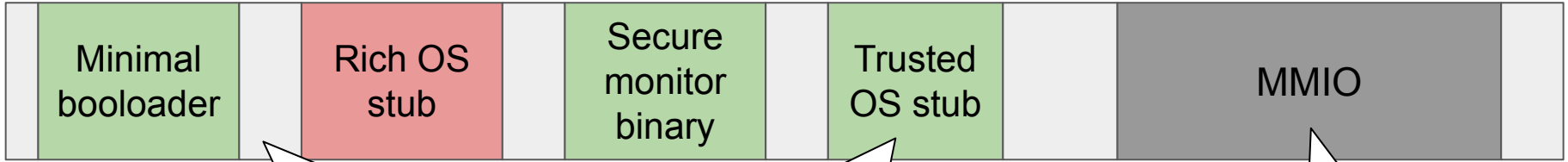




# EL3XIR's Reflected Peripheral Modeling



# Physical Memory Layout



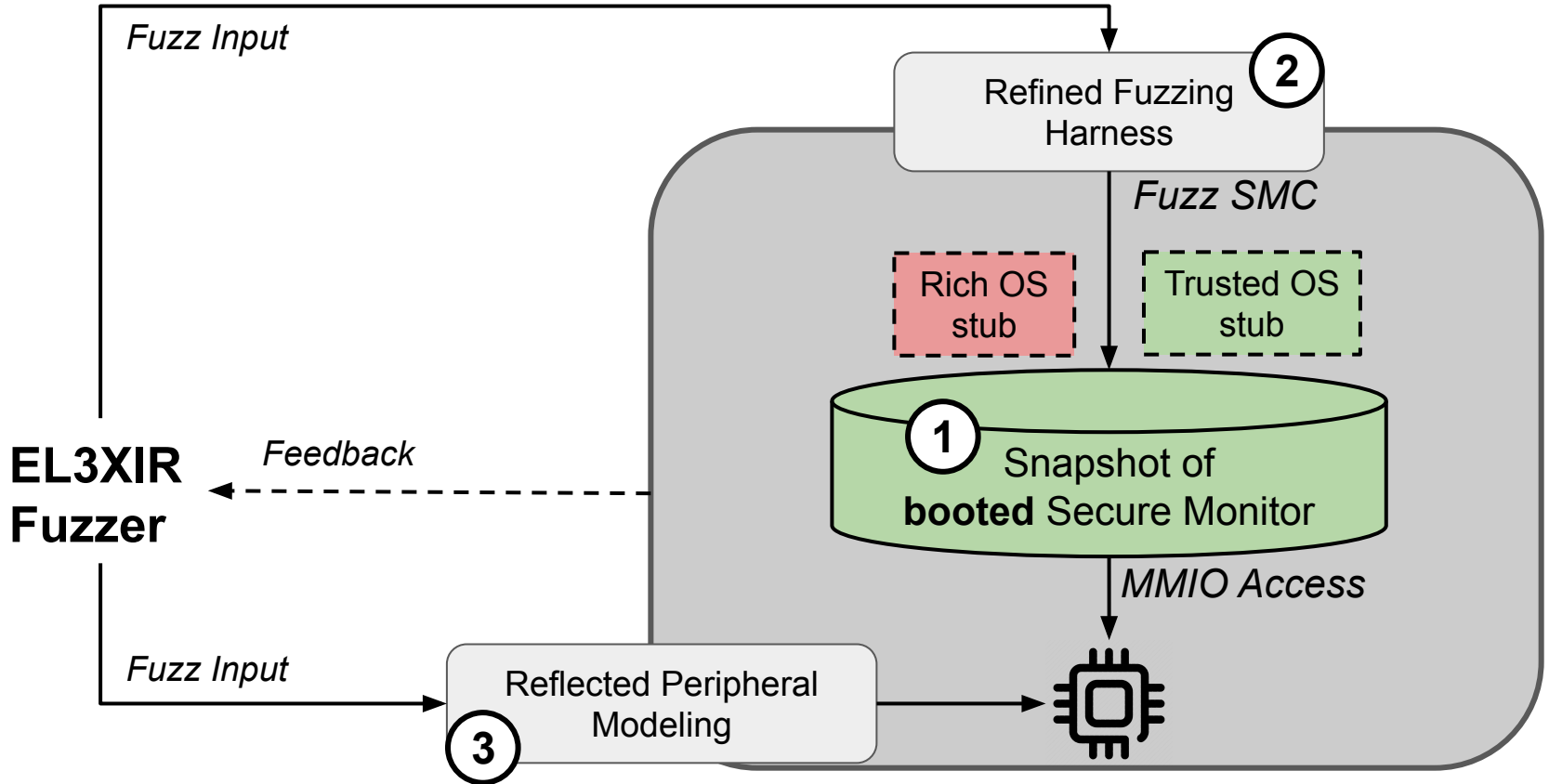
add\_memory\_range()

add\_secure\_stub()

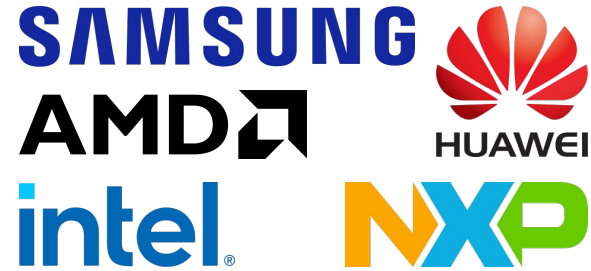
add\_in\_memory\_buffer\_peripheral()  
...  
request\_afl\_data(value, size)  
...



# EL3XIR



# Found EL3 Bugs and CVEs

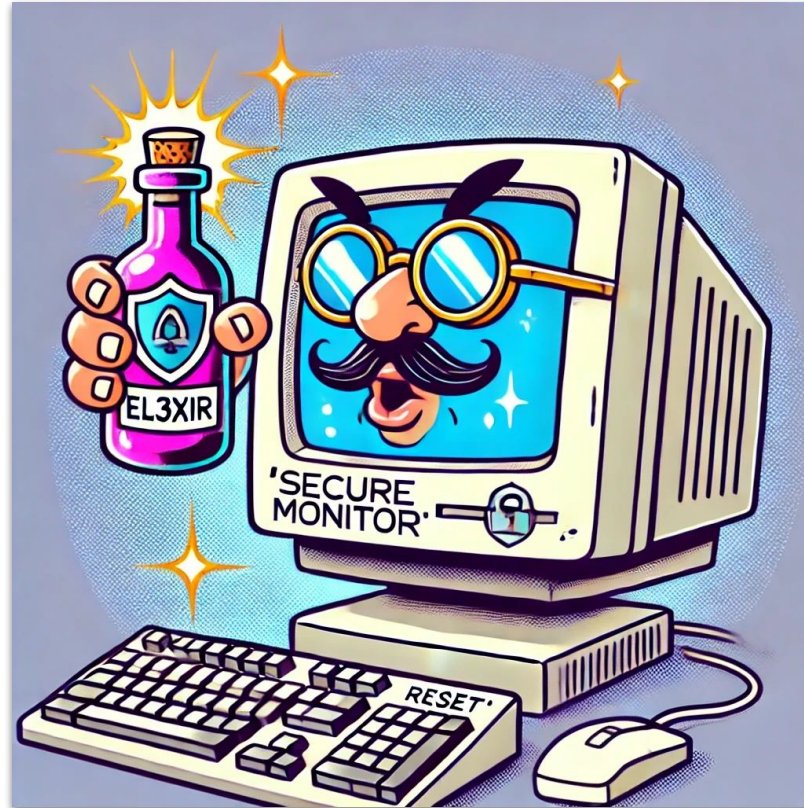


- 7 targets from 6 different vendors
  - 4 open-source, 3 closed-source
- EL3XIR triggered 34 bugs (**17** security relevant) in 5 targets
- Responsible disclosure resulted in 6 CVEs plus 11 confirmed bugs

**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**



# 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
```

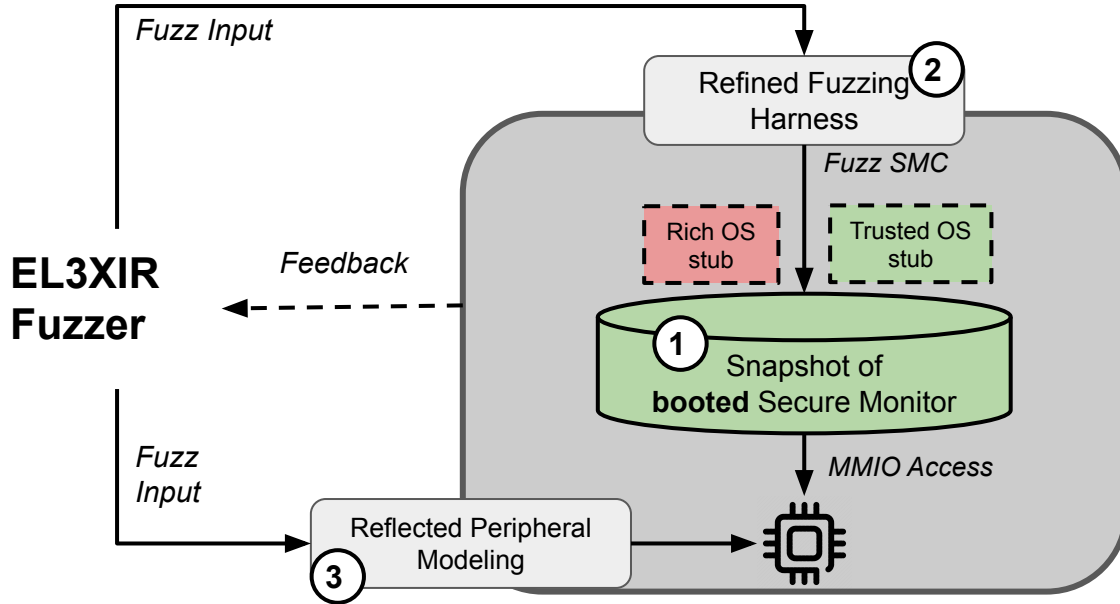
```
ADRP X1, #0x35C20E98@PAGE           ⇒ X19 = *(0x35c20E98)
LDR X19, [X1, #0x35C20E98@PAGEOFF]  ⇒ X20 = X19 + 0x6000
ADD X20, X19, #6, LSL#12             ⇒ if !X0 return
CBZ X0, loc_35C08C38
LDR X1, [X20, #0xA28]                ⇒ X1 = *(X20 + 0xa28)
BLR X1                               ⇒ X1()
```



Sploit available @ [github.com/HexHive/EL3XIR](https://github.com/HexHive/EL3XIR)

THX  
Moritz!

# EL3XIR: Fuzzing COTS Secure Monitors



[github.com/HexHive/EL3XIR](https://github.com/HexHive/EL3XIR)



[christian.lindenmeier@fau.de](mailto:christian.lindenmeier@fau.de)

X@\_chli\_

[marcel.busch@epfl.ch](mailto: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/itm8f2672e74aaca>
- [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/BUAZB4YMHDZ3oIII.large>