



DE LA RECHERCHE À L'INDUSTRIE

Méthode combinée d'Injection de faute et d'analyse Side-Channel temps réel pour contourner un Secure-Boot d'Android

Combined Fault Injection and Real-Time Side-Channel Analysis for Android Secure-Boot Bypassing

Clément Fanjas ⁽¹⁾, Clément Gaine ⁽¹⁾ ⁽²⁾,
Driss Aboukassimi ⁽¹⁾, **Simon Pontié** ⁽¹⁾, Olivier Potin ⁽²⁾

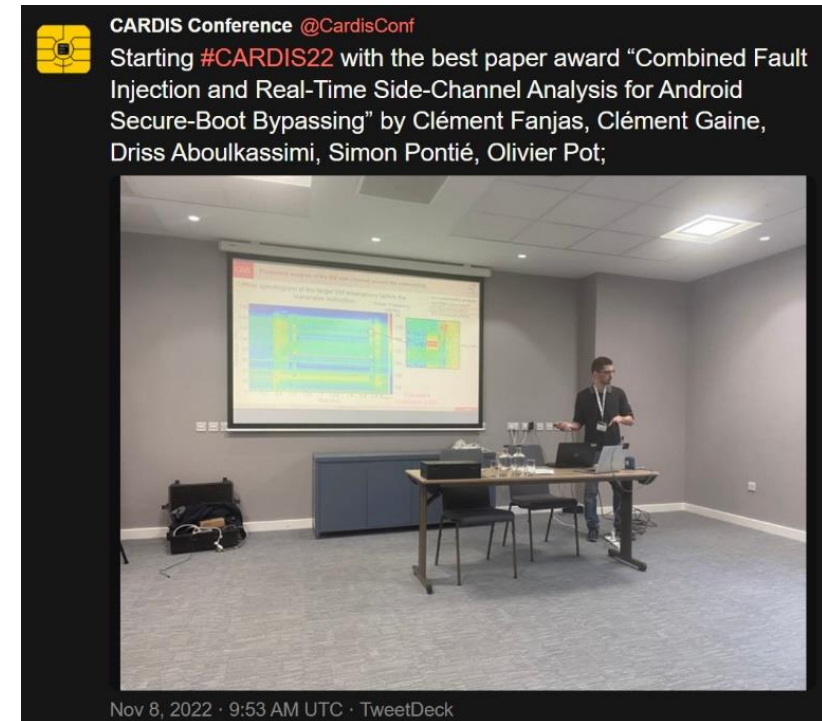
⁽¹⁾ CEA-Leti, SAS team, Gardanne France

⁽²⁾ EMSE, SAS team, Gardanne France

Authors: Clément Fanjas⁽¹⁾, Clément Gaine^{(1) (2)}, Driss Aboukassimi⁽¹⁾, Simon Pontié⁽¹⁾, Olivier Potin⁽²⁾

(1) CEA-Leti, SAS team, Gardanne France

(2) EMSE, SAS team, Gardanne France



The experiments were done in the context of the EXFILES project.



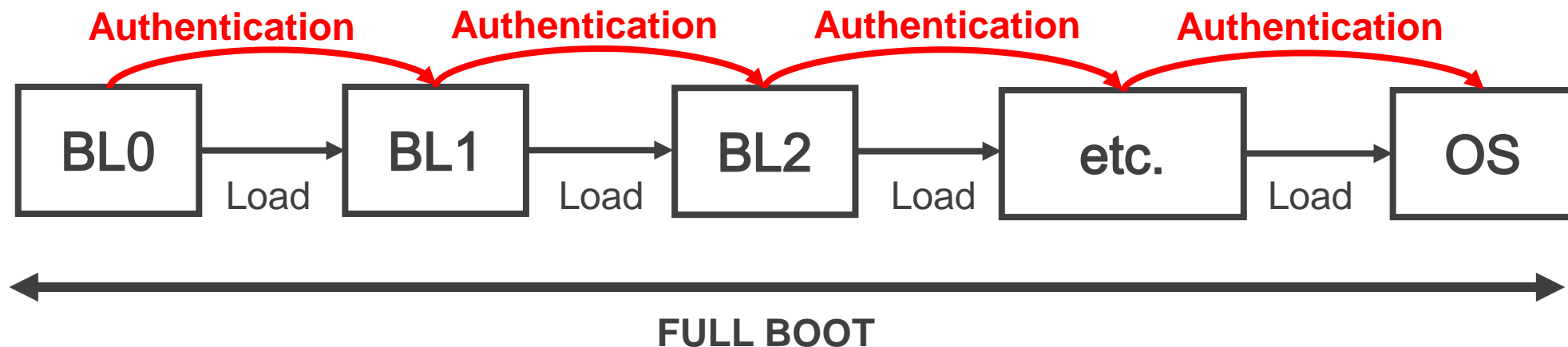
- 1. Secure-Boot and Fault Injection**
- 2. Target and methodology**
- 3. Synchronization of hardware attacks**
- 4. Combined Fault Injection and Real-Time SCA**

What is a Secure Boot ?

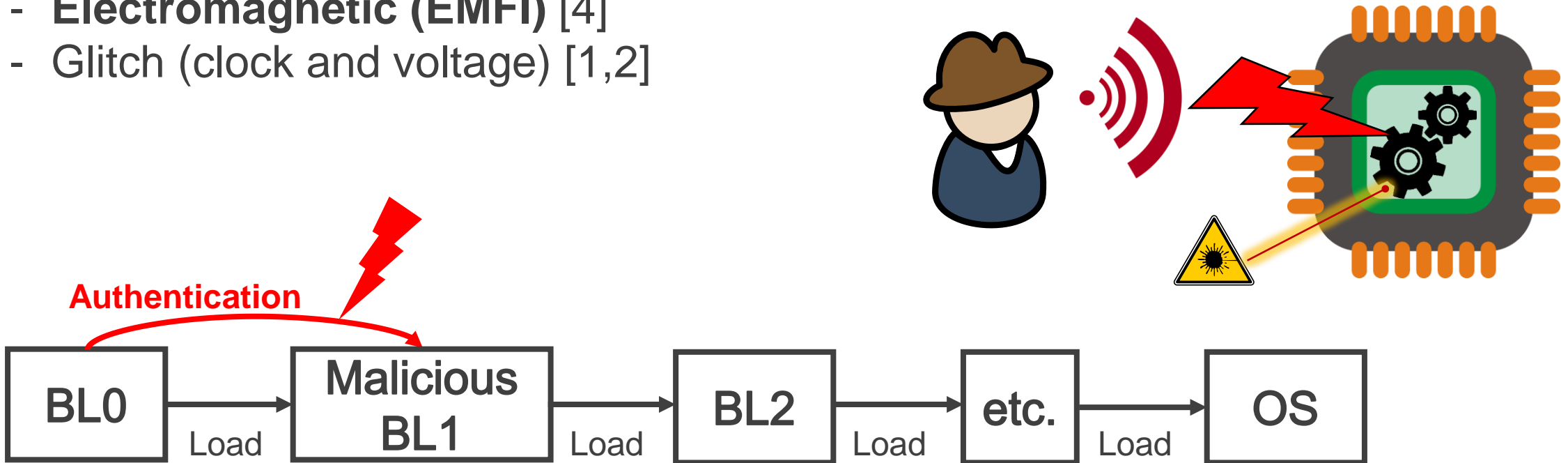
→ A security feature : it is a chain of trust where each high privilege program is authenticated before being executed.

Why it is important ?

→ It avoids running malicious program with high privilege.



- Fault injection aims at disrupting the target behaviour.
- Fault injection methods already used against Secure Boot:
 - Optical (laser) [3]
 - **Electromagnetic (EMFI)** [4]
 - Glitch (clock and voltage) [1,2]



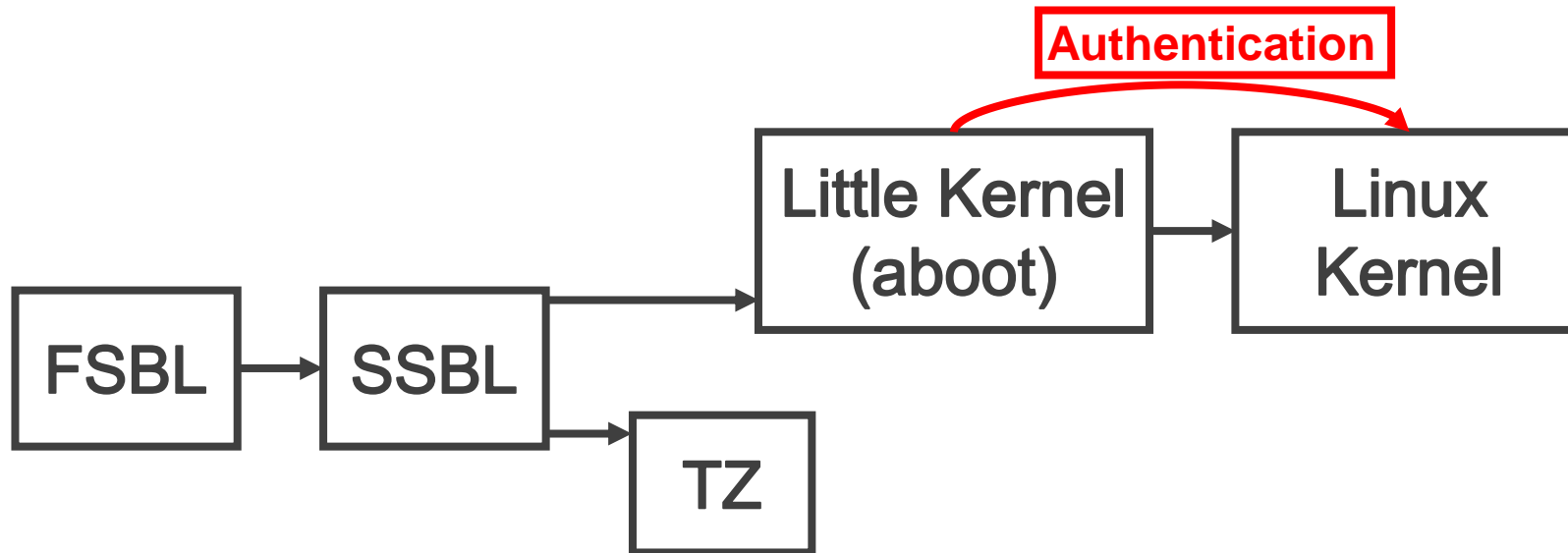
Bypassing the Secure Boot leads to a privilege escalation.

Hardware target :

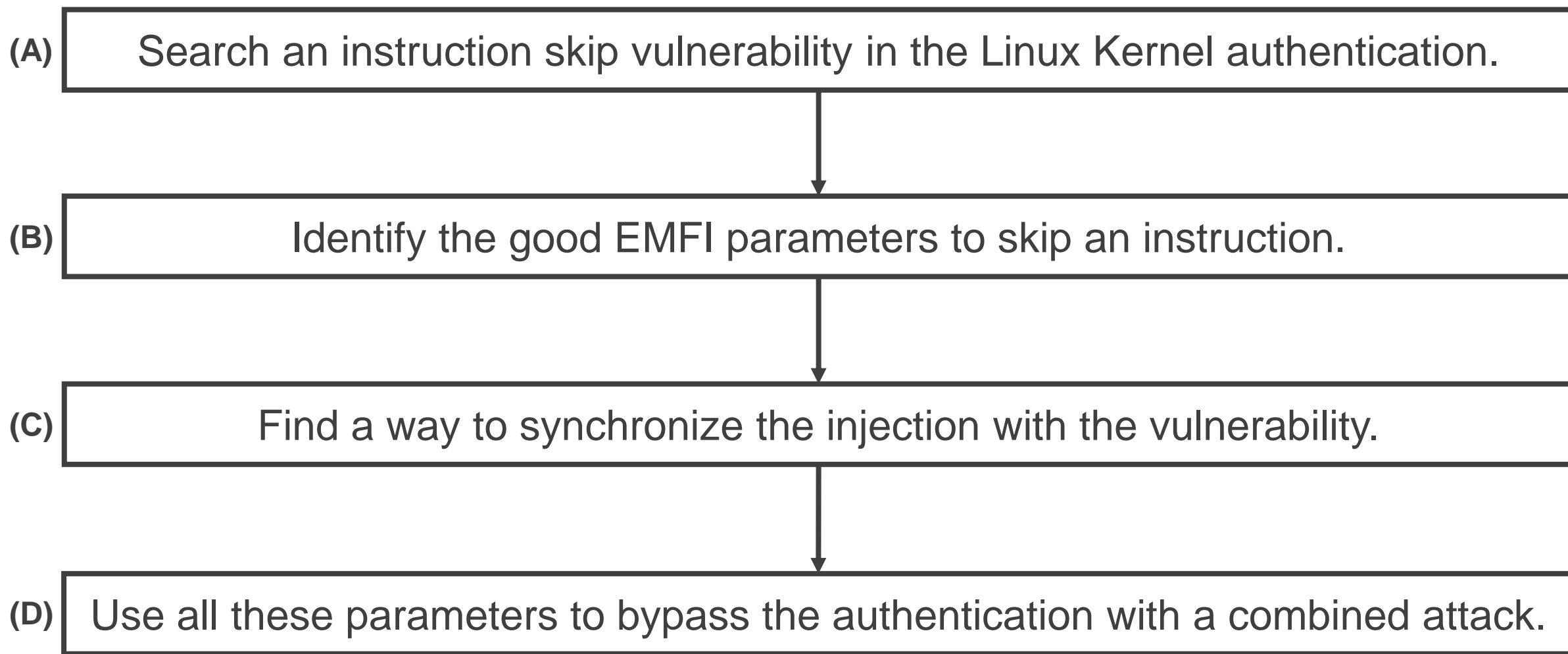
- Smartphone System-on-Chip on dev-board:
 - CPU: quad-core ARM Cortex A53
 - Maximum frequency: 1.2GHz
 - Running frequency during the boot: 800MHz
- Previous work (**Gaine et al. 2020**):
 - Using EMFI to skip an instruction is possible
- Previous work (**Tasso et al. 2021**):
 - Using EMFI to recover SIKE private key is possible

Software target :

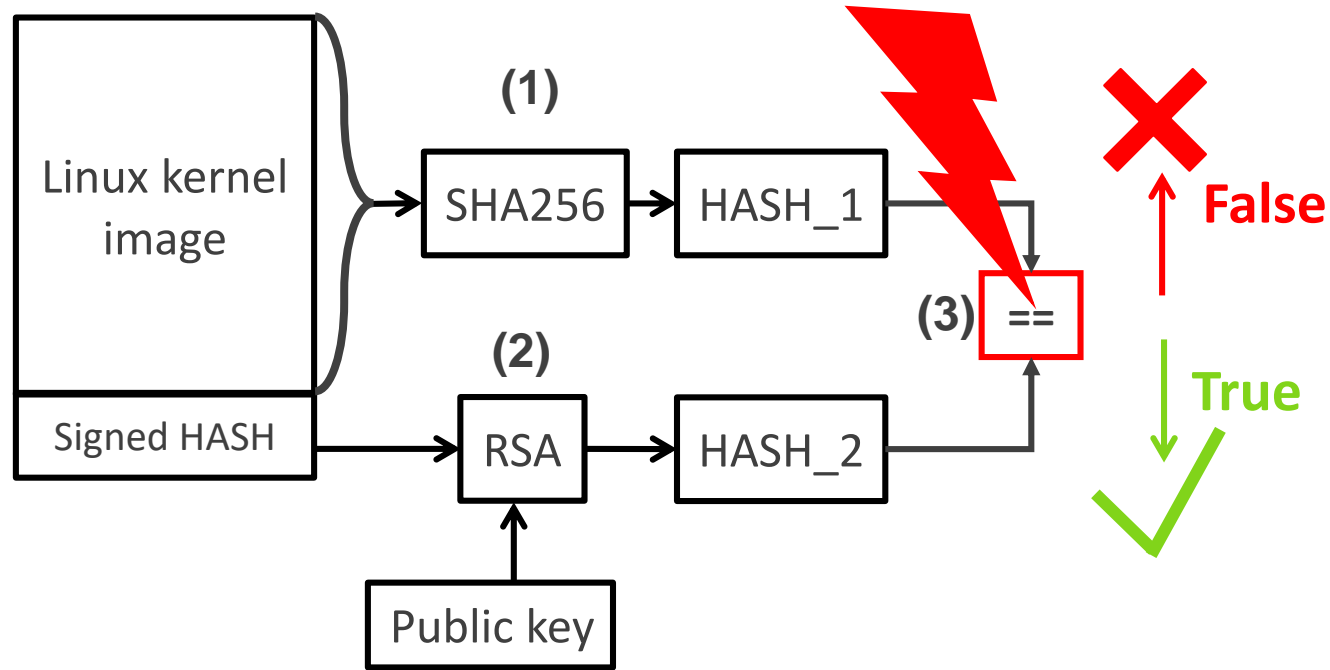
- Android Secure-Boot
 - ➔ Linux kernel authentication



Our goal is to **bypass this authentication using EMFI** and **load a malicious Linux Kernel** on our target.

- 
- ```
graph TD; A["(A) Search an instruction skip vulnerability in the Linux Kernel authentication."] --> B["(B) Identify the good EMFI parameters to skip an instruction."]; B --> C["(C) Find a way to synchronize the injection with the vulnerability."]; C --> D["(D) Use all these parameters to bypass the authentication with a combined attack."];
```
- (A) Search an instruction skip vulnerability in the Linux Kernel authentication.
- (B) Identify the good EMFI parameters to skip an instruction.
- (C) Find a way to synchronize the injection with the vulnerability.
- (D) Use all these parameters to bypass the authentication with a combined attack.





- (A) Vulnerability analysis
- (B) EMFI parameters
- (C) Synchronization
- (D) Combined attack

HASH Comparison in Little Kernel (C code)

```
ret = memcmp(HASH_1, HASH_2);
if(ret == 0)
 auth = 1;
```

HASH Comparison in Little Kernel (ASM code)

```
bl <memcmp>
clz r6, r0
lsr r6, r6, #5
```

Skipping this LSR allows to bypass the authentication

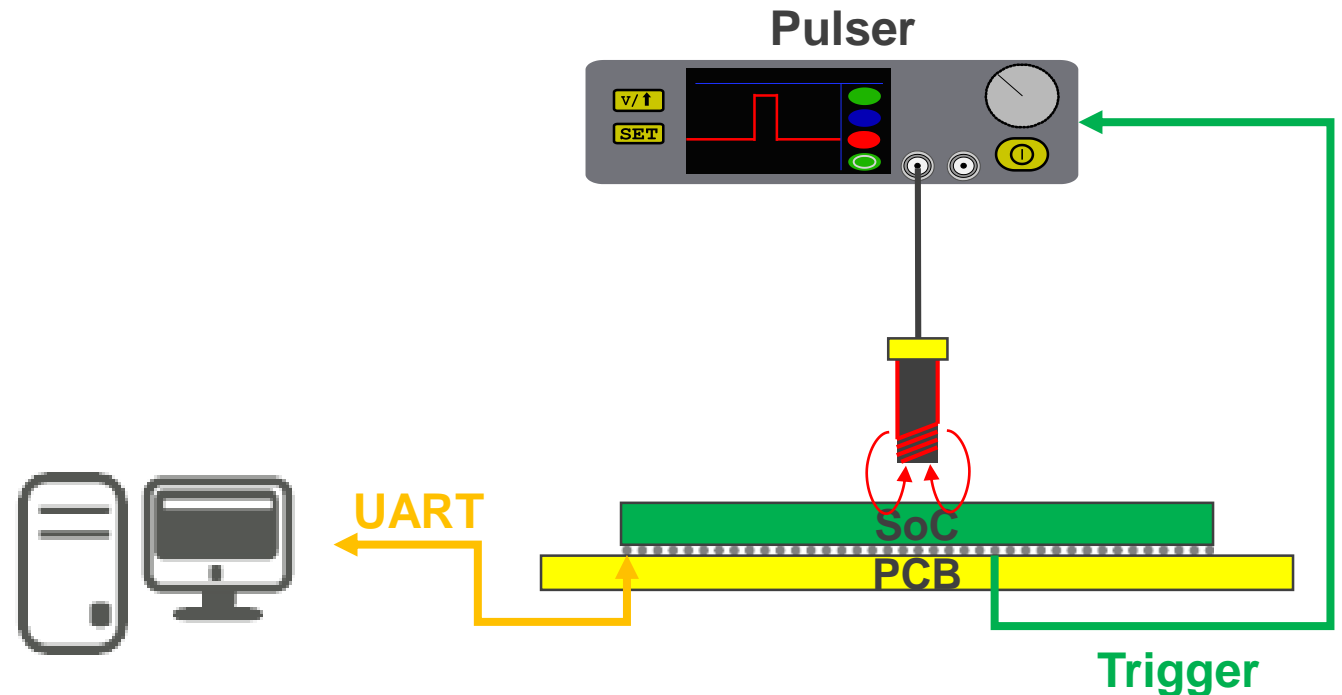
How do we inject fault using EMFI:

- Sending a voltage pulse into an active probe located over the targeted chip.
- Depending on the probe position over the chip, an EM coupling is created between the target and the probe.
- This coupling induces a transient voltage inside the chip which can corrupt the normal operation.

- ✓ (A) Vulnerability analysis
- (B) EMFI parameters
- (C) Synchronization
- (D) Combined attack

We need to find the good parameters:

- Probe position
- Pulse parameters: voltage and pulse width



To find the good parameters we use a target without Secure-Boot.

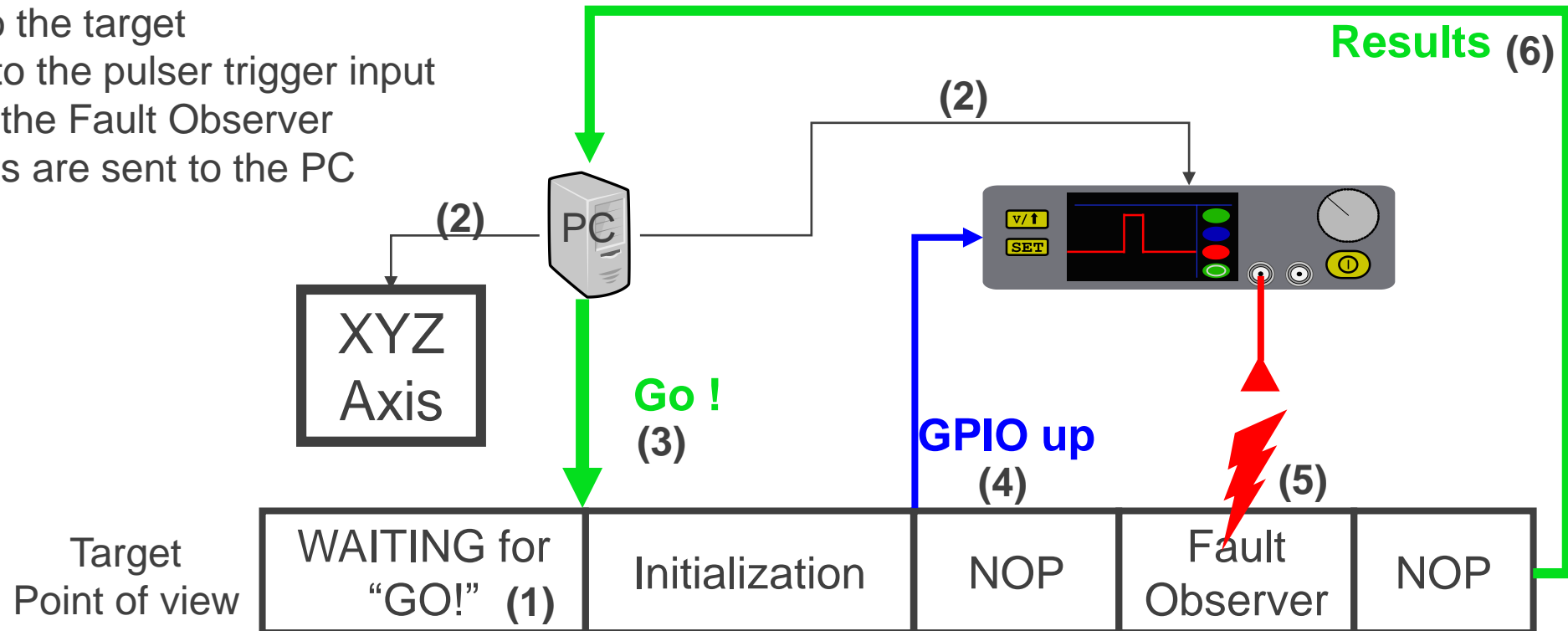
We run a program with a known behaviour: we call it a "Fault Observer".

We observe the output of this program while injecting with different parameters.

- ✓ (A) Vulnerability analysis
- (B) EMFI parameters
- (C) Synchronization
- (D) Combined attack

Methodology:

- (1) The target waits for an order
- (2) The PC sets the pulse parameters and moves the XYZ axis
- (3) The PC sends an order to the target
- (4) The target rise a GPIO into the pulser trigger input
- (5) A pulse is injected during the Fault Observer
- (6) The Fault Observer results are sent to the PC



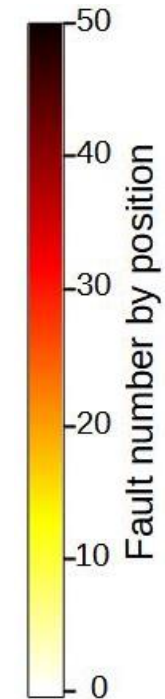
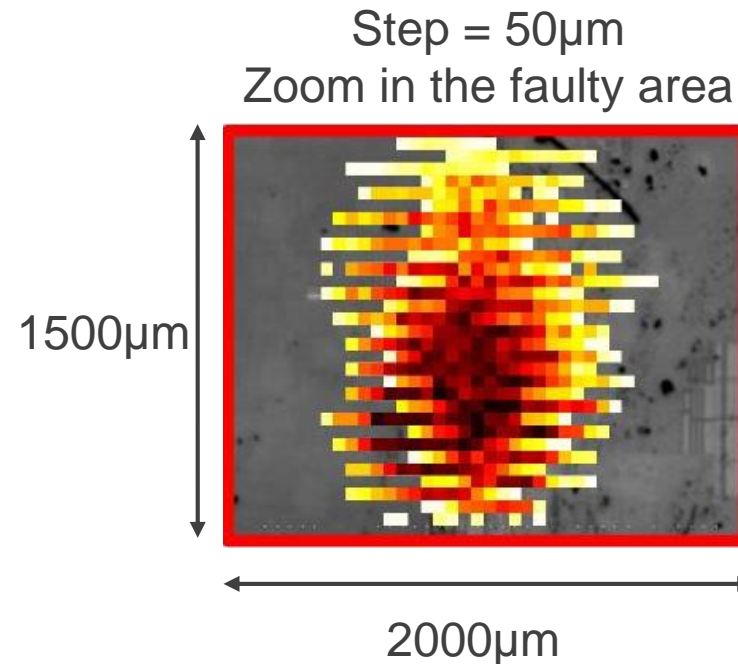
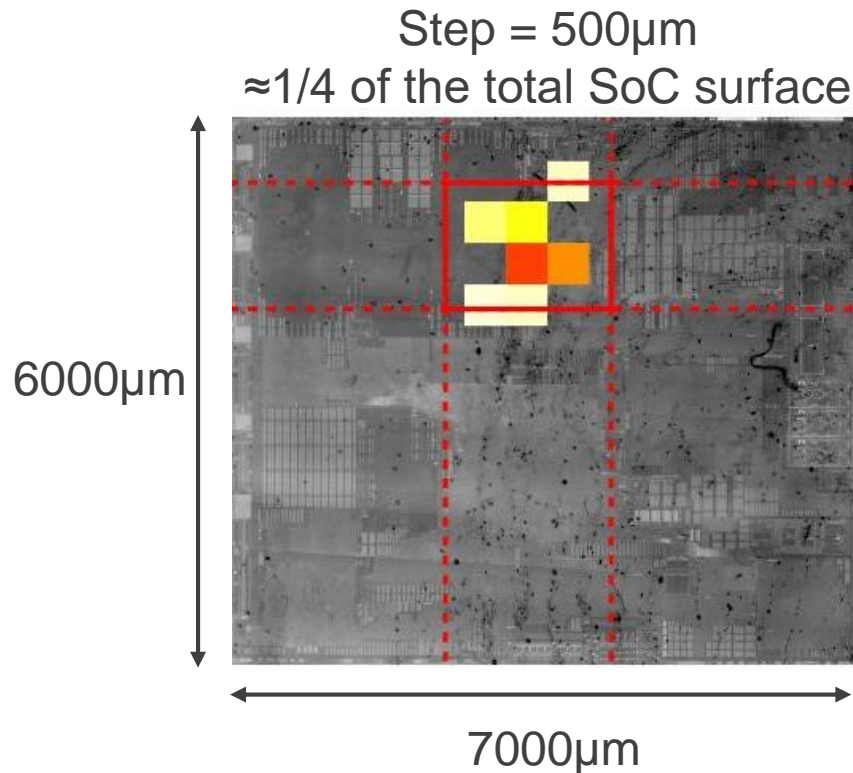
## Fault model: Instruction skip

Faulty area with SoC IR imaging as background.

Pulse voltage = 400V

Pulse Width = 10ns

- ✓ (A) Vulnerability analysis
- (B) EMFI parameters
- (C) Synchronization
- (D) Combined attack



A triggering event is used as temporal reference to synchronize the injection.

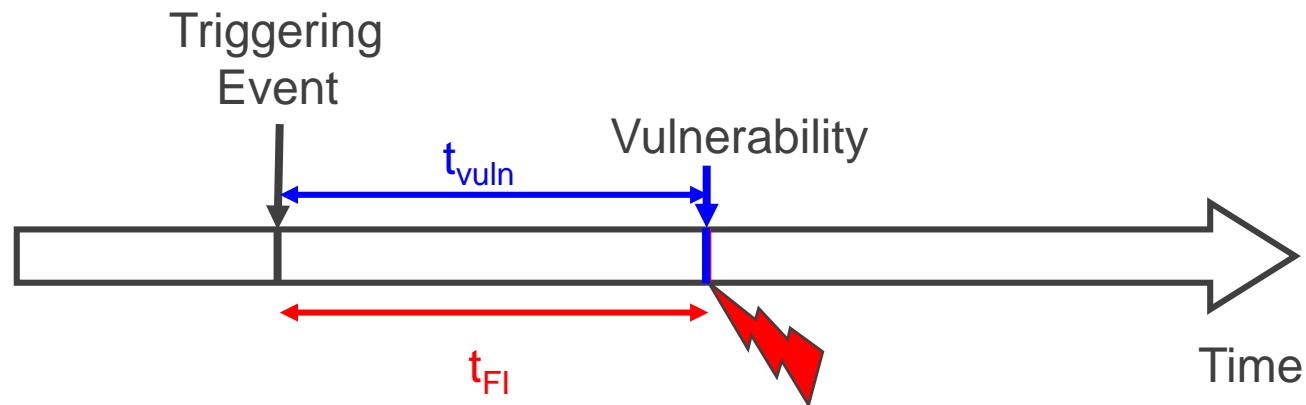
$t_{vuln}$  = delay between the triggering event and the vulnerability.

$t_{FI}$  = delay between the triggering event and the attack.

The attack is successful when  $t_{vuln} = t_{FI}$

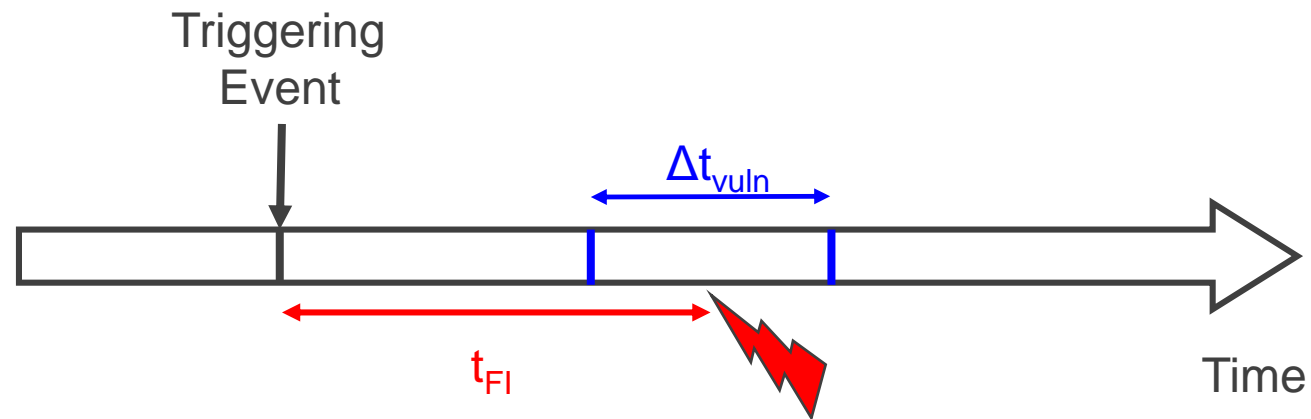
→ the injection and the vulnerability happen at the same time

- ✓ (A) Vulnerability analysis
- ✓ (B) EMFI parameters
- (C) Synchronization**
- (D) Combined attack



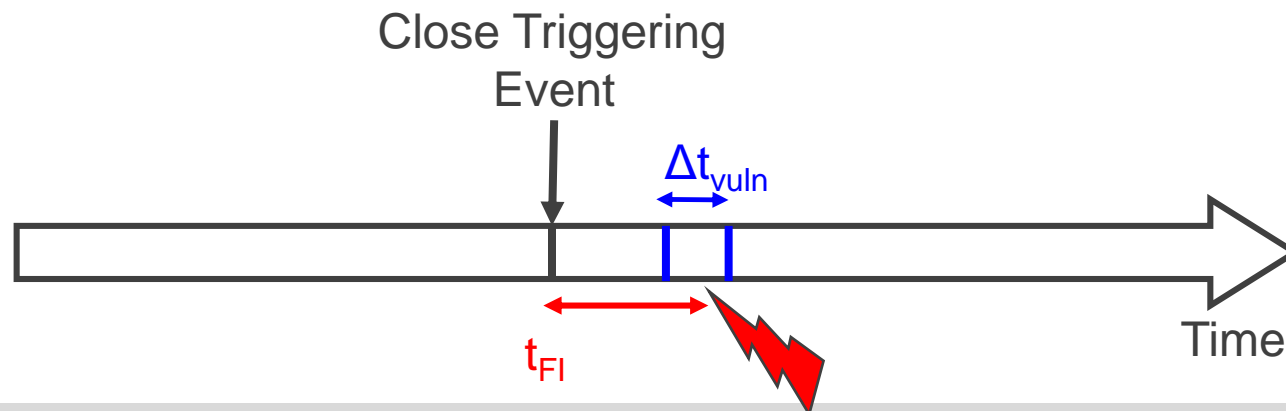
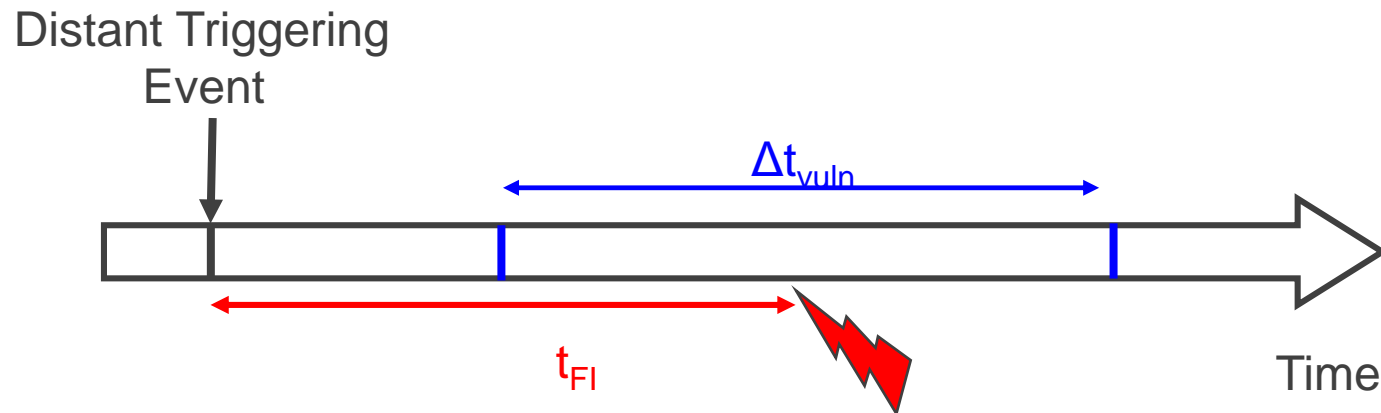
We set  $t_{FI}$ , but we have no influence on  $t_{vuln}$ .  
 $t_{vuln}$  is confined in a temporal window  $\Delta t_{vuln}$  also called jitter.

- ✓ (A) Vulnerability analysis
- ✓ (B) EMFI parameters
- (C) Synchronization**
- (D) Combined attack



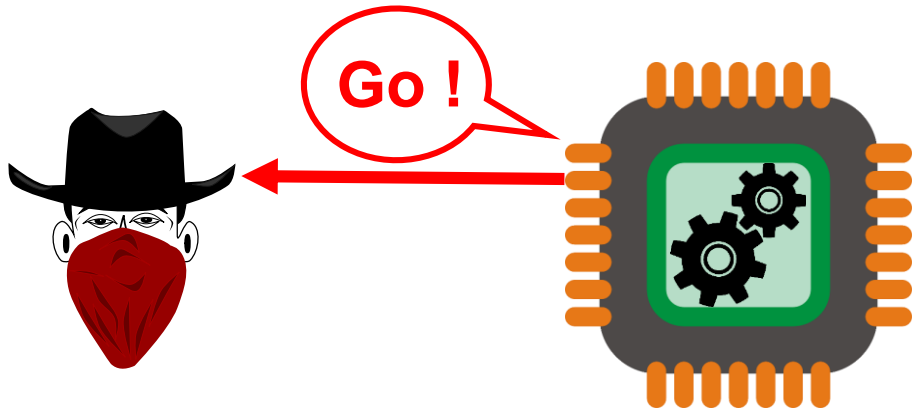
To maximize the attack success rate we need to reduce  $\Delta t_{\text{vuln}}$ .  
The best way to do it is to get the triggering event as close as possible to the vulnerability.

- ✓ (A) Vulnerability analysis
- ✓ (B) EMFI parameters
- (C) Synchronization**
- (D) Combined attack

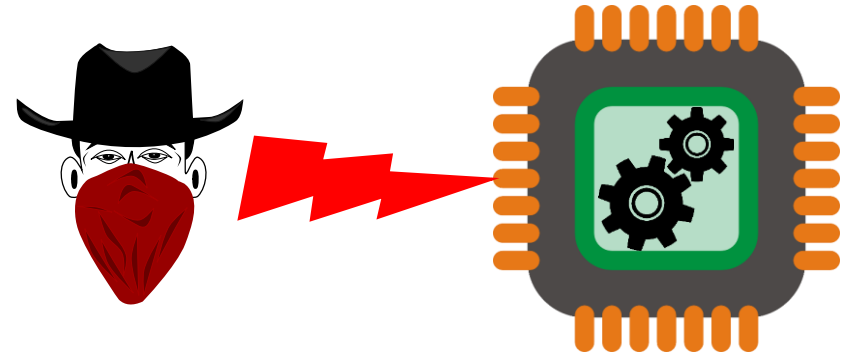


## Solution 1: Trigger with fully controlled output such as GPIO

### Step 1:



### Step 2:



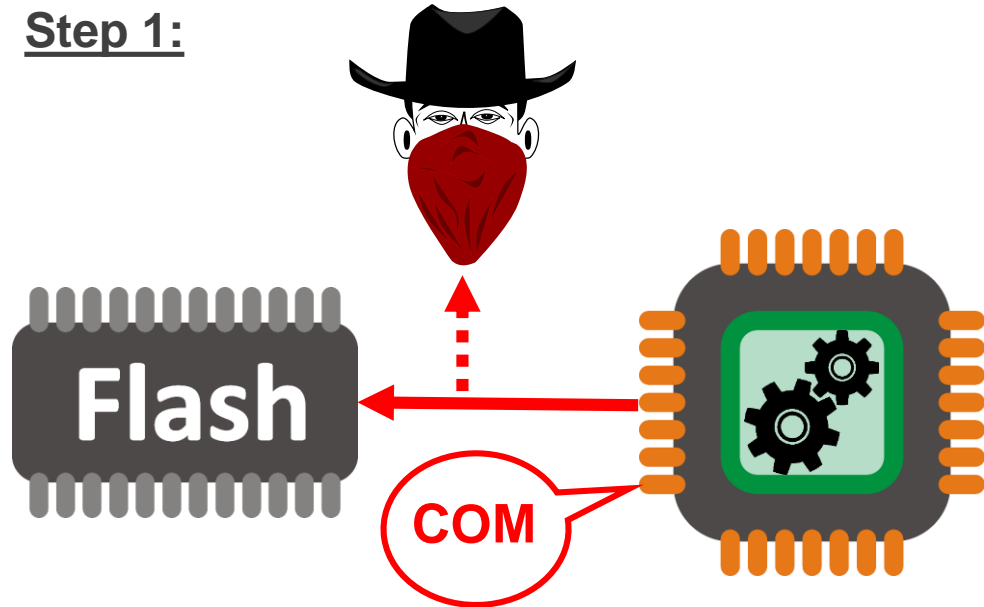
- ✓ (A) Vulnerability analysis
- ✓ (B) EMFI parameters
- (C) Synchronization**
- (D) Combined attack

The attack is voluntarily triggered by the target.  
The synchronization is optimal but it needs a high level of control over the target.

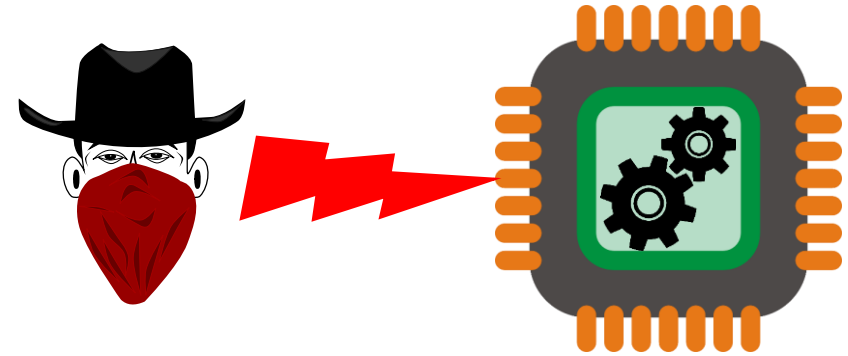


## Solution 2: Trigger on uncontrolled I/O

### Step 1:



### Step 2:

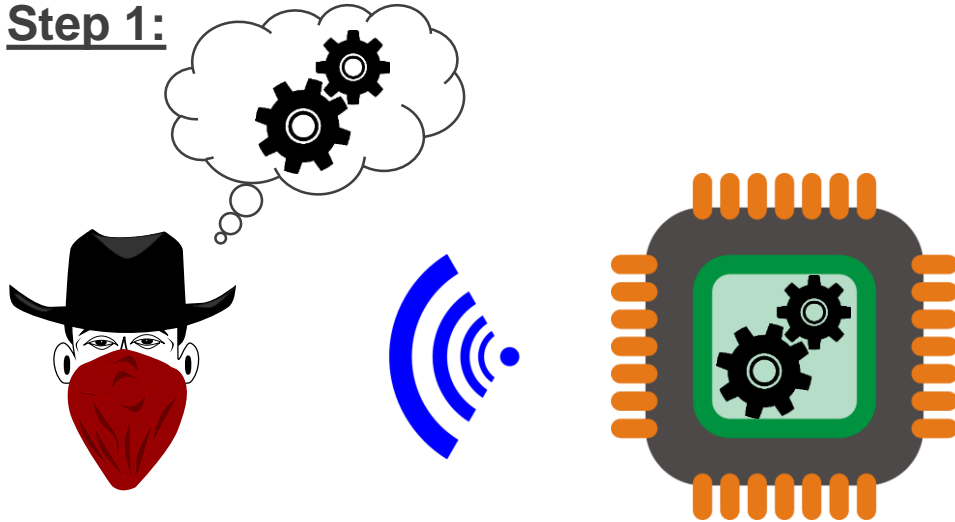


- ✓ (A) Vulnerability analysis
- ✓ (B) EMFI parameters
- (C) Synchronization**
- (D) Combined attack

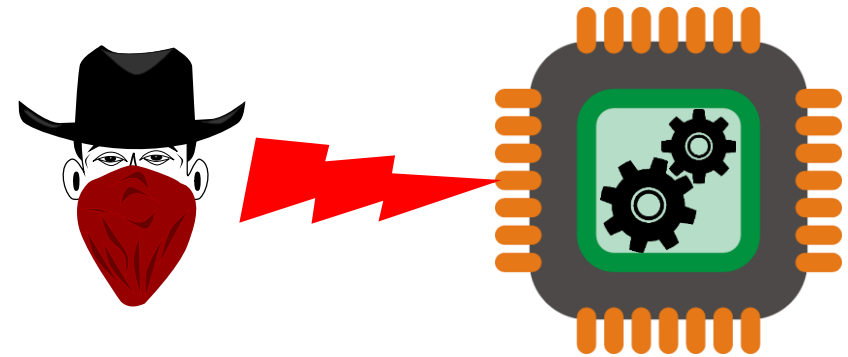
This kind of triggering event is not always accurate but there is no need to control the target.

## Solution 3: Triggering on a Side-Channel event

Step 1:



Step 2:



- ✓ (A) Vulnerability analysis
- ✓ (B) EMFI parameters
- (C) Synchronization
- (D) Combined attack

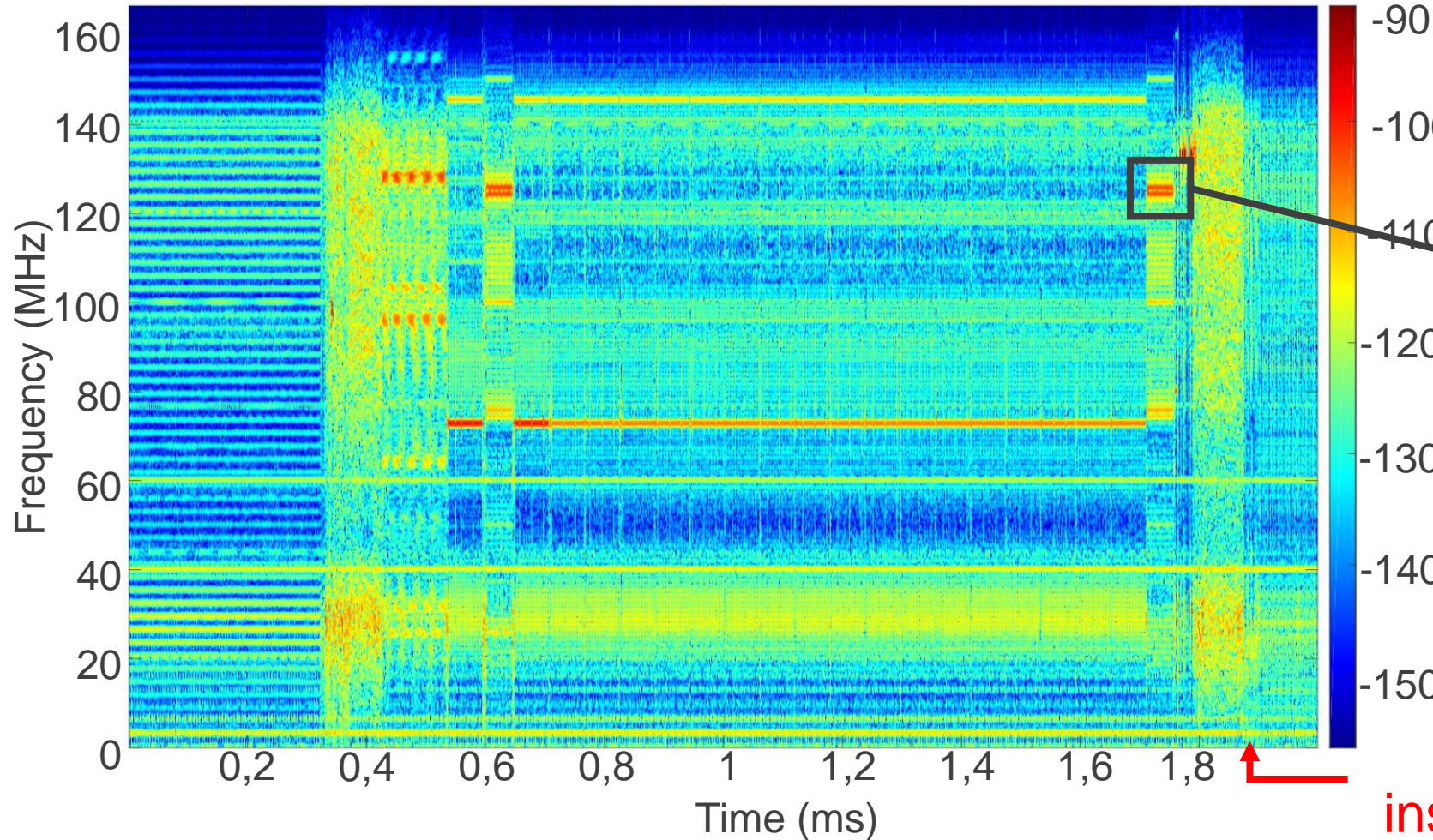
The attacker has a great degree of freedom in the event choice.

For Fault Injection it needs a real time analysis:

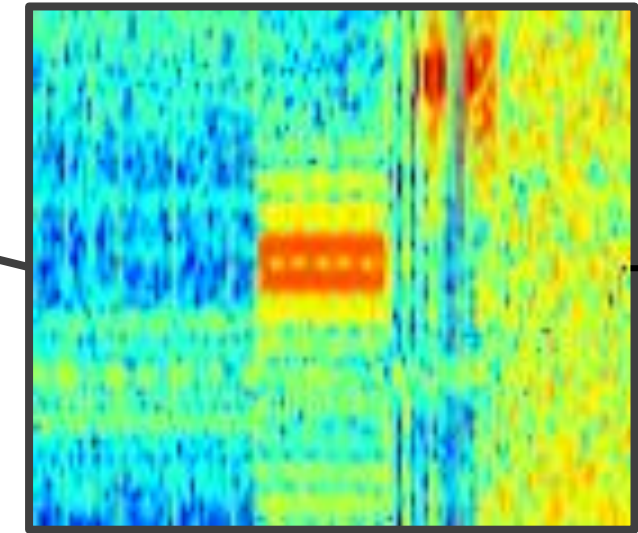
**Analysis of high frequency events may be difficult**

Offline spectrogram of the target EM emanations before the vulnerable instruction.

Power Frequency  
(dB/Hz)

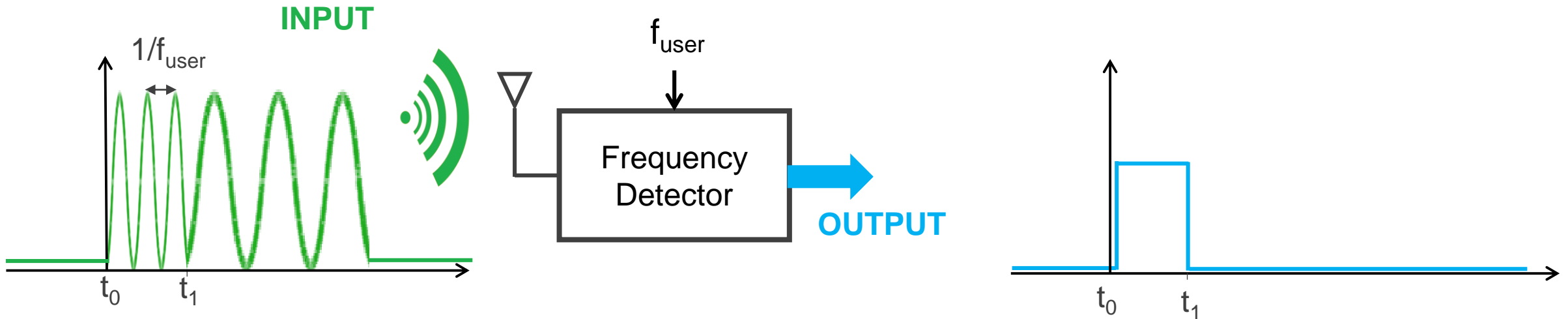


- ✓ (A) Vulnerability analysis
- ✓ (B) EMFI parameters
- (C) Synchronization**
- (D) Combined attack

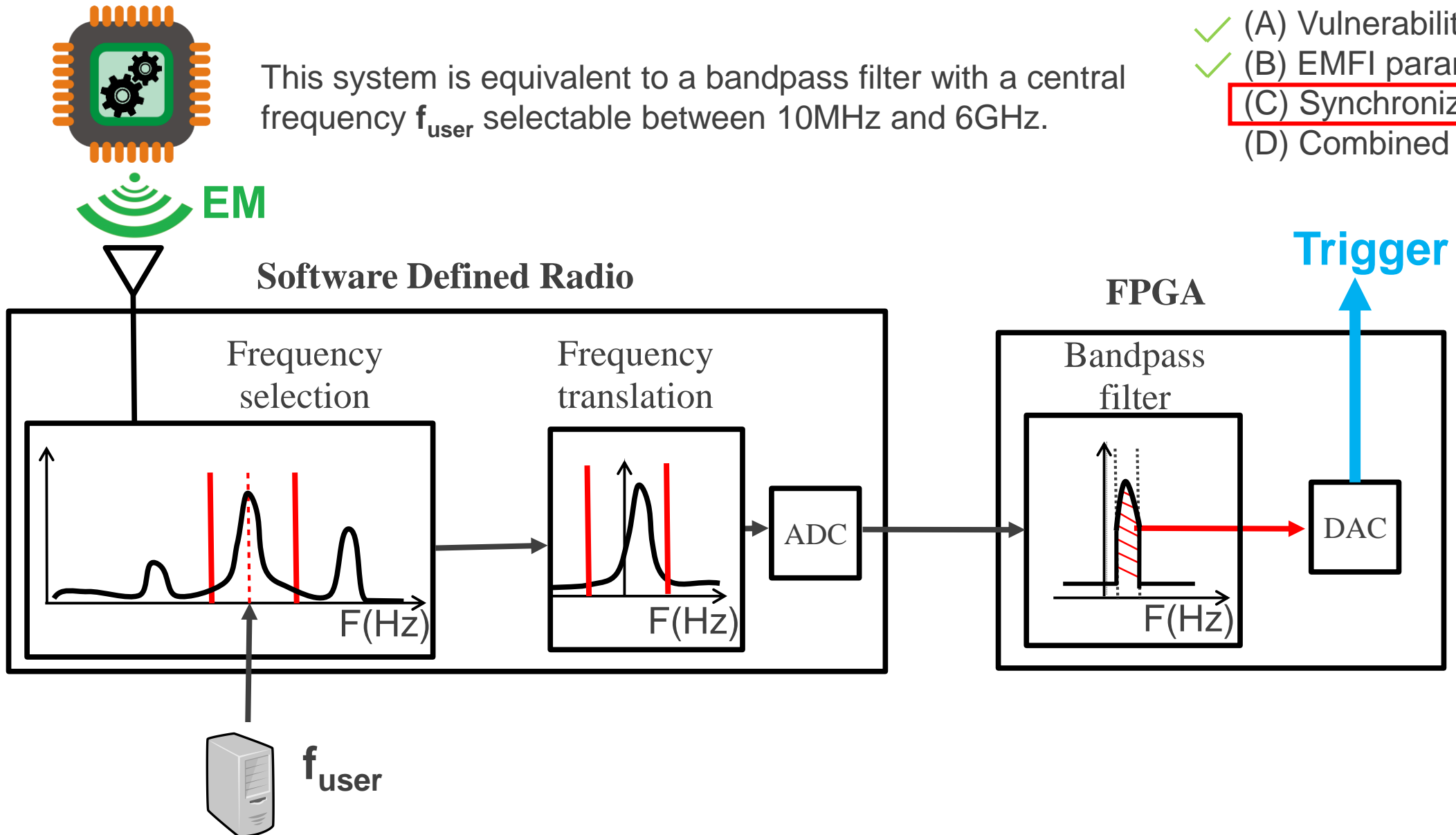


Vulnerable  
instruction (LSR)

- ✓ (A) Vulnerability analysis
- ✓ (B) EMFI parameters
- (C) Synchronization**
- (D) Combined attack

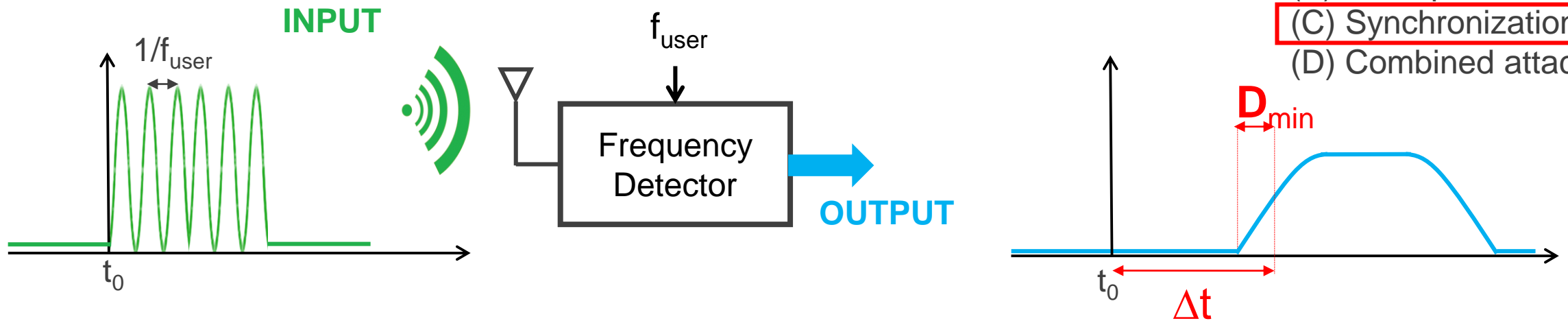


We need a device that can generate a trigger signal when a chosen frequency is active.



- ✓ (A) Vulnerability analysis
- ✓ (B) EMFI parameters
- (C) Synchronization
- (D) Combined attack

## Frequency detector performances



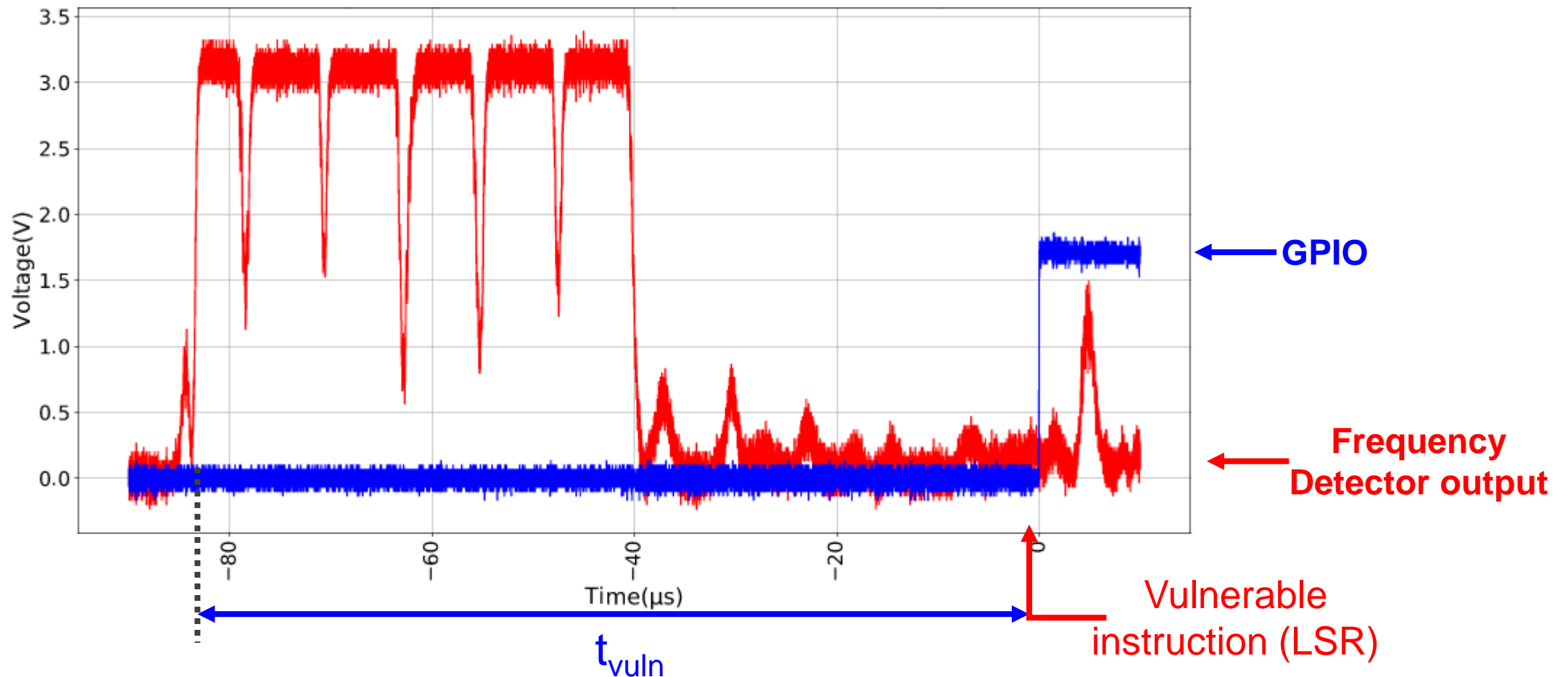
- ✓ (A) Vulnerability analysis
- ✓ (B) EMFI parameters
- (C) Synchronization
- (D) Combined attack

- $\Delta t$  is the delay between the activation of  $f_{user}$  and the detection of  $f_{user}$ . This delay follows a normal distribution.
- In order to be detected,  $f_{user}$  needs to stay active during at least  $D_{min}$ .

|                  | Results     |
|------------------|-------------|
| $\Delta t_{avg}$ | 2.5 $\mu s$ |
| $\Delta t_{std}$ | 60 ns       |
| $D_{min}$        | 450 ns      |

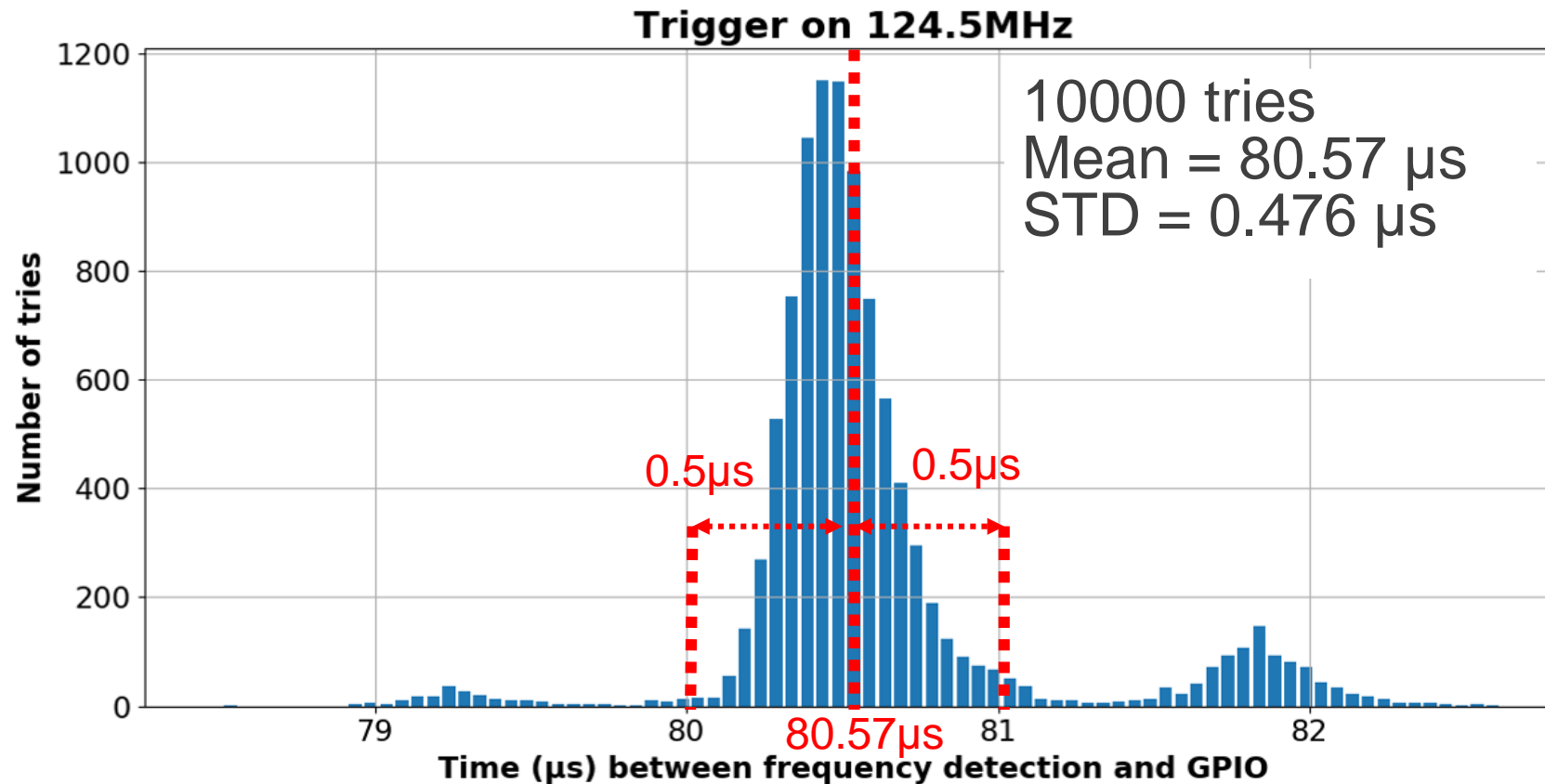
We use a modified Little Kernel which rises a GPIO just **after** the vulnerability. We set the frequency detector to trigger on the 124,5MHz frequency we identified before.

- ✓ (A) Vulnerability analysis
- ✓ (B) EMFI parameters
- (C) Synchronization
- (D) Combined attack



This measure is performed 10000 times to identify the mean delay and the jitter.

- ✓ (A) Vulnerability analysis
- ✓ (B) EMFI parameters
- (C) Synchronization**
- (D) Combined attack

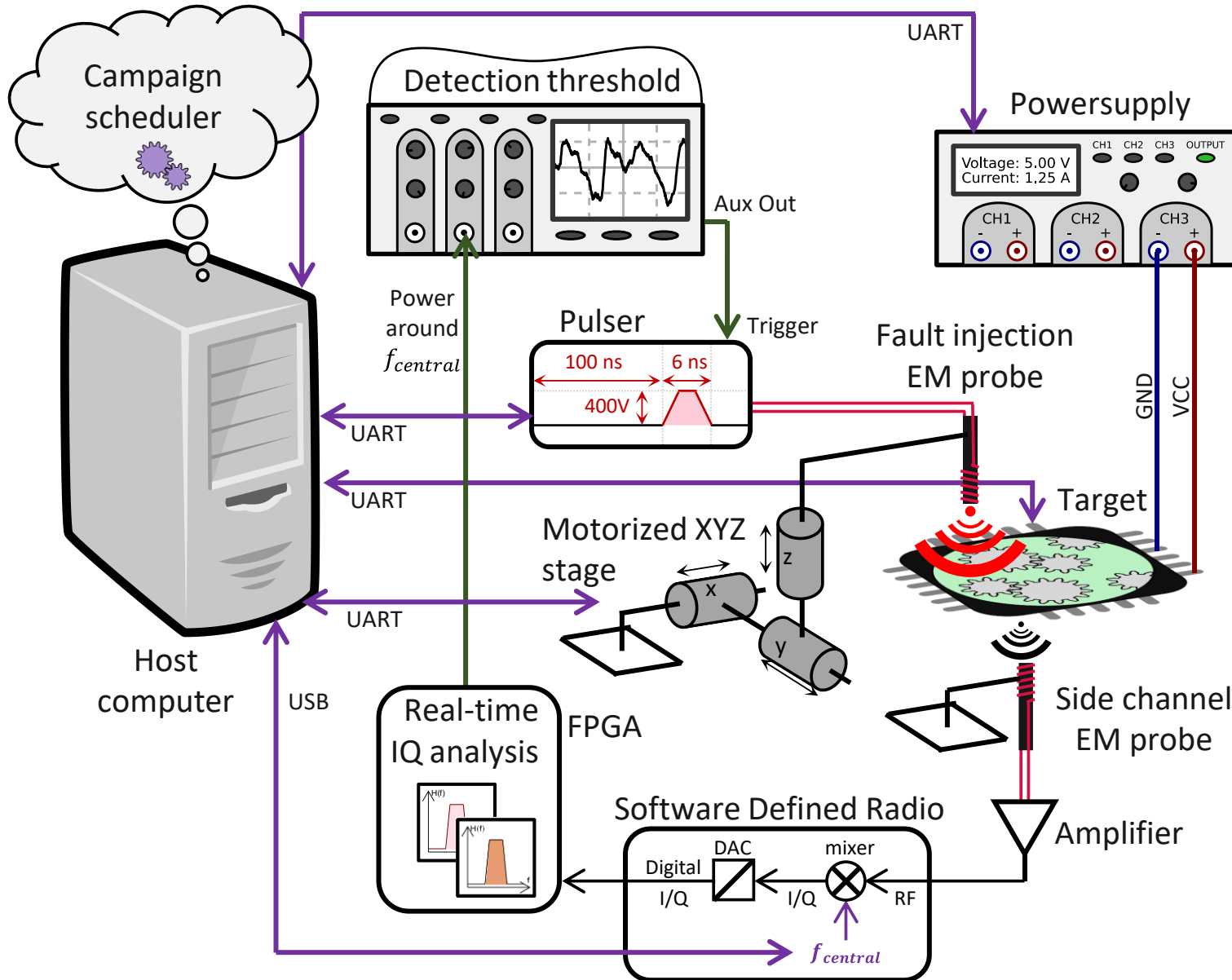




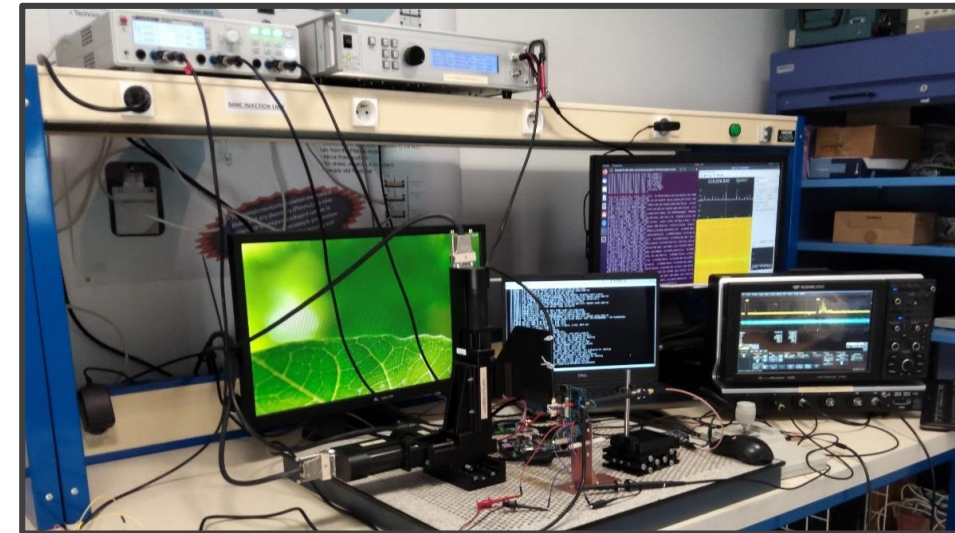
- ✓ (A) Vulnerability analysis
- ✓ (B) EMFI parameters
- (C) Synchronization**
- (D) Combined attack

Comparison with other triggering events:

| Triggering event          | Mean value of vulnerability delay | STD value of vulnerability delay | Equivalent instruction number at 800MHz |
|---------------------------|-----------------------------------|----------------------------------|-----------------------------------------|
| Frequency detector        | 80,57 $\mu$ s                     | 476ns                            | $\approx$ 400 instructions              |
| Trigger on UART character | 113,8ms                           | 2,06 $\mu$ s                     | $\approx$ 1600 instructions             |
| Board power-on            | 1,248s                            | 5ms                              | $\approx$ 4000000 instructions          |



- ✓ (A) Vulnerability analysis
- ✓ (B) EMFI parameters
- ✓ (C) Synchronization
- ✓ (D) Combined attack



15000 attacks in 18h00:

- ✓ (A) Vulnerability analysis
- ✓ (B) EMFI parameters
- ✓ (C) Synchronization
- (D) Combined attack

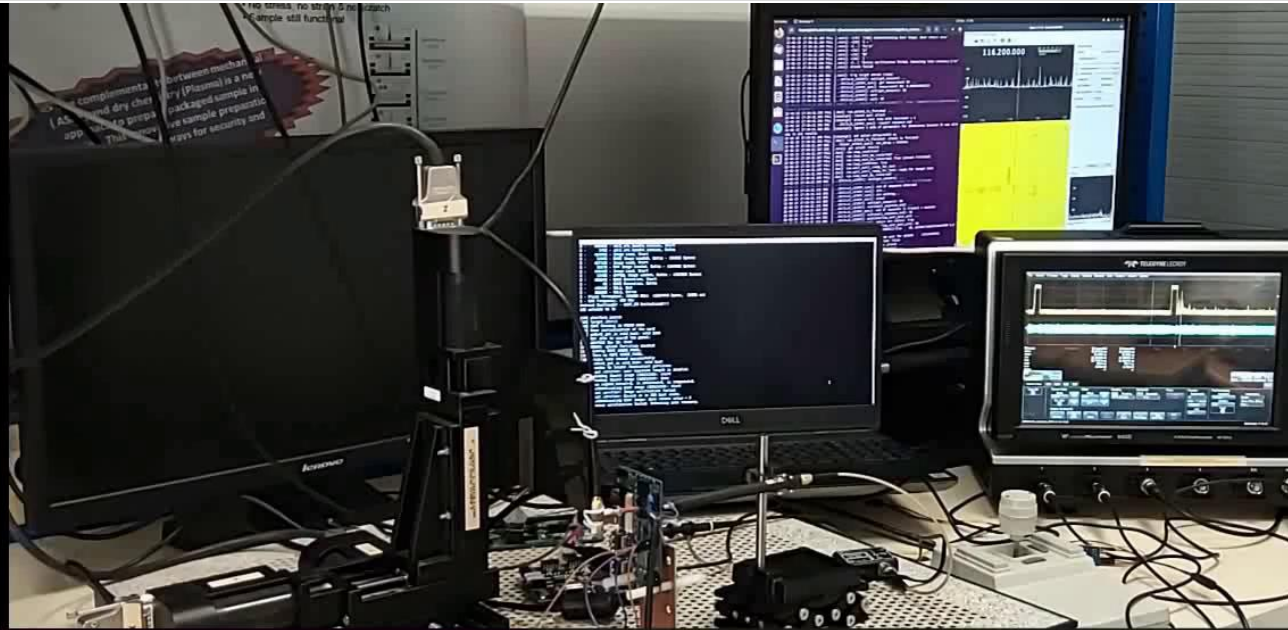
| Scenarios             | Results        |
|-----------------------|----------------|
| Crash                 | 7005 (46,777%) |
| No effect             | 7912 (52,75%)  |
| Authentication bypass | 83 (0,53%)     |

≈ 1 bypass every 15 minutes.

```

[10] welcome to tk
[10] platform_init()
[10] target_init()
[50] SDHC Running in HS200 mode
[60] Done initialization of the card
[70] pm8x41_get_is_cold_boot: cold boot
[70] Not able to search the panel:
[70] ADV7533 Rev ID: 0x14
[80] ERROR: splash Partition invalid
[180] Config MIPI VIDEO PANEL.
[190] Turn on MIPI_VIDEO_PANEL.
[200] Video lane tested successfully
[210] pm8x41_get_is_cold_boot: cold boot
[210] Unable to locate /bootselect partition
[220] boot_verifier: user keystore length is invalid.
[220] Loading (boot) image (16521216): start
[340] Loading (boot) image (16521216): done
[350] use_signed_kernel=1, is_unlocked=0, is_tampered=0.
[360] Authenticating boot image (16521216): start
[470] boot_verifier: Image verification failed.
[480] boot_verifier: Device is in RED boot state.
[480] Authenticating boot image: done return value = 0
[80] Device verification failed. Rebooting into recovery.

```

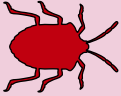


```

22-02-23 17:30:32.308 FULL [.Run]: run loop must be restarted? ...
22-02-23 17:30:32.331 INFO [...OSCILLO_LECROY_sync]: 1 acquired traces in oscilloscope
22-02-23 17:30:32.331 DBG [...OSCILLO_LECROY_sync]: end of sequence detected
22-02-23 17:30:32.332 DBG [...OSCILLO_LECROY_sync]: arm...
22-02-23 17:30:32.332 DBG [...OSCILLO_LECROY_sync]: arm waiting...
22-02-23 17:30:32.332 DBG [...OSCILLO_LECROY_sync]: armed
22-02-23 17:30:32.332 DBG [...OSCILLO_LECROY_sync]: (get_answers): OK
22-02-23 17:30:32.332 DBG [...OSCILLO_LECROY_sync]: (sequence_end) ...
22-02-23 17:30:32.332 DBG [...OSCILLO_LECROY_sync]: of sequence (1 traces) = success
22-02-23 17:30:32.332 DBG [...OSCILLO_LECROY_sync]: (sequence_end): OK
22-02-23 17:30:32.332 DBG [...OSCILLO_LECROY_sync]: (lap_wait_post_arm) ...
22-02-23 17:30:32.332 DBG [...OSCILLO_LECROY_sync]: (arm status): True
22-02-23 17:30:32.332 DBG [...OSCILLO_LECROY_sync]: (overlap_wait_post_arm): OK
22-02-23 17:30:32.332 DBG [...OSCILLO_LECROY_sync]: (172/4199921):try= 10, pulser(amplitude=400 V,
delay=34575)
22-02-23 17:30:32.332 DBG [...OSCILLO_LECROY_sync]: (results from user for attack 172/4199921
22-02-23 17:30:32.332 DBG [...OSCILLO_LECROY_sync]: (loop must be restarted): False
22-02-23 17:30:32.334 FULL [.Run]: run.wait_until_ready_for_attack ...
22-02-23 17:30:32.334 DBG [...Pulser_AVTECH]: wait(ready to trig) ...

```

Detected  
malicious  
Linux kernel



- **53%: detected malicious Linux kernel**
- 45%: timeout ou crash
- 1.67%: execution of malicious Linux kernel followed by a crash
- 0.53%: success, execution of malicious Linux kernel and Android OS.

Un contournement de Secure Boot toutes les 15 minutes

```

[220] Loading (boot) Image (16521216): start
[340] Loading (boot) Image (16521216): done
[350] use signed kernel? 0 tampered=0.
[360] Authenticating boot image (16521216): start
[470] undefined abort, halting
[470] r0 0x8f6a00b1 r1 0x0000005e r2 0x00000001 r3 0x00000001
[470] r4 0x0000199d r5 0x8f690b04 r6 0x8f690b90 r7 0x8f69fa78
[470] r8 0x8f6a00ac r9 0x8f63ceb4 r10 0x8f69ff58 r11 0x8f69fcb0
[470] r12 0x0000005e usp 0x00000000 ulr 0x00000000 pc 0x8f6314c0
[470] spsr 0xa0000153
[470] fiq r13 0x8f680400 r14 0x864001b8
[470] irq r13 0x8f685300 r14 0x8f610964
[470] *svc r13 0x8f6953b0 r14 0x8f690b90
[470] und r13 0x8f68034c r14 0x8f61d798
[470] sys r13 0x00000000 r14 0x00000000
[470] bottom of stack at 0x8f6953b0:
0x8f6953b0: 8f69fa78 00000001 00000033 8f690b90 |x.i.....i.|
0x8f6953c0: 00000100 8f6452e4 00000100 8f6a02c4 |.....Rd.....j.|
0x8f6953d0: 00000000 00000001 8f69fdc0 8f69ff6c |.....i.l.i.|
0x8f6953e0: bb67ae85 00000001 a54ff53a 8f690b90 |..g.....0'
0x8f6953f0: 8f69fdc0 8f69fcb0 8f69bdb8 90' |...i.|
0x8f695400: 00fc180f 8f69bd4c 00000000 8f61af5c |....L.i.....a.|
0x8f695410: 00000001 90000000 8f69543b 00000001 |...
0x8f695420: 8f690b90 8f60dc20 0d303130 86600906 |.....Ti.....|
[470] HALT: re

```

SoC  
crash

```

22-02-23 17:30:36.013 FULL [.main]: <(49) ' r7 0x8f69fa78 r8 0x8f63ceb4 r9 0x8f63c'
22-02-23 17:30:36.014 FULL [.main]: <(49) 'eb4 r10 0x8f69ff58 r11 0x8f69fcb0\r\n[470] r12 0x00'
22-02-23 17:30:36.014 FULL [.main]: <(49) '00005e usp 0x00000000 ulr 0x00000000 pc 0x8f6314'
22-02-23 17:30:36.015 FULL [.main]: <(49) 'c0\r\n[470] spsr 0xa0000153\r\n[470] fiq r13 0x8f680'
22-02-23 17:30:36.015 FULL [.main]: <(49) '400 r14 0x864001b8\r\n[470] irq r13 0x8f685300 r14'
22-02-23 17:30:36.015 FULL [.main]: <(49) '0x8f610964\r\n[470] *svc r13 0x8f6953b0 r14 0x8f69'
22-02-23 17:30:36.016 FULL [.main]: <(49) '0b90\r\n[470] und r13 0x8f68034c r14 0x8f61d798\r\n[4'
22-02-23 17:30:36.055 FULL [.main]: <(49) '470] sys r13 0x00000000 r14 0x00000000\r\n[470] bo'
22-02-23 17:30:36.056 FULL [.main]: <(49) 'ttom of stack at 0x8f6953b0:\r\n0x8f6953b0: 8f69fa7'
22-02-23 17:30:36.056 FULL [.main]: <(49) '8 00000001 00000033 8f690b90 |x.i.....i.|'
22-02-23 17:30:36.057 FULL [.main]: <(49) '0x8f6953c0: 00000100 8f6452e4 00000100 8f6a02c4 |'
22-02-23 17:30:36.057 FULL [.main]: <(49) '.....Rd.....j.|'
22-02-23 17:30:36.057 FULL [.main]: <(49) '8f69fdc0 8f69ff6c |.....i.l.i.|'
22-02-23 17:30:36.058 FULL [.main]: <(49) 'bb67ae85 00000001 a54ff53a 8f690b90 |..g.....0'
22-02-23 17:30:36.058 FULL [.main]: <(49) '...i.|'
22-02-23 17:30:36.059 FULL [.main]: <(49) '000000 |..i...i...i...|\r\n0x8f695400: 00fc180f 8'
22-02-23 17:30:36.059 FULL [.main]: <(49) 'f69bd4c 00000000 8f61af5c |....L.i.....a.|'
22-02-23 17:30:36.070 FULL [.main]: <(49) 'f695410: 00000001 90000000 8f69543b 00000001 |...'
22-02-23 17:30:36.070 FULL [.main]: <(49) '.....Ti.....|\r\n0x8f695420: 8f690b90 8f60dc20 0d3'
22-02-23 17:30:36.071 FULL [.main]: <(49) '03130 86600906 |..i.....|\r\n[470] HALT: re'

```

- 53%: detected malicious Linux kernel
- **45%: timeout ou crash**
- 1.67%: execution of malicious Linux kernel followed by a crash
- 0.53%: success, execution of malicious Linux kernel and Android OS.

Un contournement de Secure Boot toutes les 15 minutes

```

S - Flash Throughput, 105000 KB/s (1187440 Bytes, 11285 us)
S - DDR Frequency, 400 MHz
Android Bootloader - UART_DM Initialized!!!
[0] welcome to lk

[10] platform_init()
[10] target_init()
[50] SDHC Running in HS200 mode
[60] Done initialization of the card
[70] pm8x41_get_is_cold_boot: cold boot
[70] Not able to search the panel:
[70] ADV7533 Rev ID: 0x14
[80] ERROR: splash Partition invalid
[180] Config MIPI_VIDEO_PANEL.
[190] Turn on MIPI_VIDEO_PANEL.
[210] Video lane tested successfully
[210] pm8x41_get_is_cold_boot: cold boot
[210] Unable to locate /bootselect partition
[220] boot verifier: user keystore length is invalid.
[220] Loading (boot) image (16521216): start
[340] Loading (boot) image (16521216): done
[350] use signed_kernel=1, is_unlocked=0, is_tampered=0.
[360] Authenticating boot image (16521216): start

```



```

22-02-23 17:30:40.966 FULL [..Run]: run.loop_must_be_restarted? ...
22-02-23 17:30:40.966 DBG [...OSCILLO_LECROY]: wait(get_answers) ...
22-02-23 17:30:40.982 INFO [...OSCILLO_LECROY_sync]: 1 acquired traces in oscilloscope
22-02-23 17:30:40.983 DBG [...OSCILLO_LECROY_sync]: end of sequence detected
22-02-23 17:30:40.983 DBG [...OSCILLO_LECROY_sync]: arm...
22-02-23 17:30:40.983 DBG [...OSCILLO_LECROY_sync]: arm waiting...
22-02-23 17:30:40.984 DBG [...OSCILLO_LECROY_sync]: armed
22-02-23 17:30:40.984 DBG [...OSCILLO_LECROY]: wait(get_answers): OK
22-02-23 17:30:40.984 DBG [...OSCILLO_LECROY]: wait(wait_sequence_end) ...
22-02-23 17:30:40.984 DBG [...OSCILLO_LECROY_sync]: end of sequence (1 traces) = success
22-02-23 17:30:40.984 DBG [...OSCILLO_LECROY]: wait(wait_sequence_end): OK
22-02-23 17:30:40.984 DBG [...OSCILLO_LECROY]: wait(overlap_wait_post_arm) ...
22-02-23 17:30:40.984 DBG [...OSCILLO_LECROY_sync]: return arm status: True
22-02-23 17:30:40.984 DBG [...OSCILLO_LECROY]: wait(overlap_wait_post_arm): OK
22-02-23 17:30:40.985 DBG [Campaign]: 0.00%=(174/4199921):try= 10, pulser(amplitude=400 V,
elay=34500 ns) 0.15 step/s
22-02-23 17:30:40.985 FULL [Campaign]: received results from user for attack 174/4199921
22-02-23 17:30:40.985 FULL [..Run]: run.loop_must_be_restarted: False
22-02-23 17:30:40.985 FULL [..Run]: run.wait_until_ready_for_attack ...
22-02-23 17:30:40.985 DBG [...Pulser_AVTECH]: wait(ready to trig) ...

```

- 53%: detected malicious Linux kernel
- **45%: timeout ou crash**
- 1.67%: execution of malicious Linux kernel followed by a crash
- 0.53%: success, execution of malicious Linux kernel and Android OS.

Un contournement de Secure Boot toutes les 15 minutes

## Execution of the malicious Linux kernel

```

Android bootloader > BAKT_ON initialized!
[0] welcome to lk

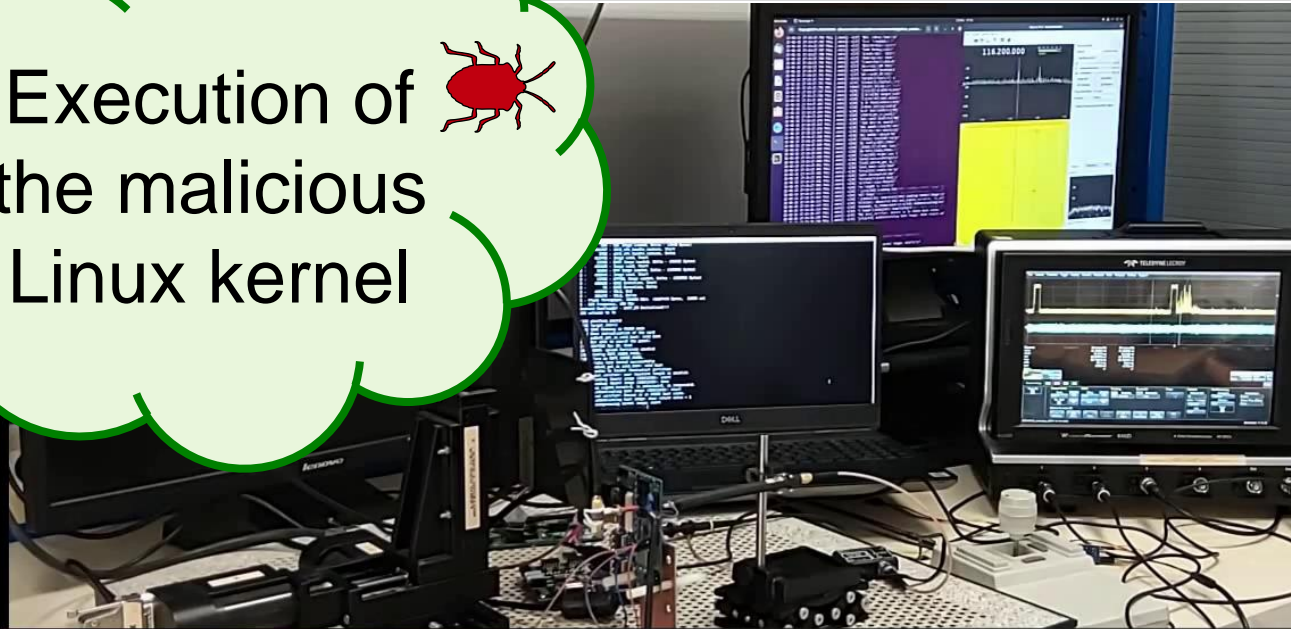
[10] platform_init()
[10] target_init()
[50] SDHC Running in HS200 mode
[60] Done initialization of the card
[70] pm8x41 get is cold boot: cold boot
[70] Not able to search the panel:
[70] ADV7533 Rev ID: 0x14
[80] ERROR: splash Partition invalid
[180] Config MIPI_VIDEO_PANEL.
[180] Turn on MIPI_VIDEO_PANEL.
[200] Video lane tested successfully
[210] pm8x41 get is cold boot: cold boot
[210] Unable to locate /bootselect partition
[220] boot_verifier: user keystore length is invalid.
[220] Loading (boot) image (16521216): start
[340] Loading (boot) image (16521216): done
[340] use signed kernel=1, is unlocked=0, is_tampered=0.
[350] Authenticating boot image (16521216): start
[470] boot_verifier: Device is in GREEN boot state.
[470] Authenticating boot image: done return value = 1
[480] decompressing kernel image: start

```

```

22-02-23 17:33:27.983 FULL [.main]: <(20) 'verifier: user keysto'
22-02-23 17:33:27.983 FULL [.main]: <(20) 're length is invalid'
22-02-23 17:33:27.983 FULL [.main]: <(20) 'n[220] Loading (bo'
22-02-23 17:33:27.983 FULL [.main]: <(29) 'ot) image'
22-02-23 17:33:28.113 FULL [.main]: <(49) ' (16521216): start\r\n[340] Loading (boot) image (1'
22-02-23 17:33:28.114 FULL [.main]: <(49) '6521216): done\r\n[340] use_signed_kernel=1, is_unl'
22-02-23 17:33:28.114 FULL [.main]: <(49) 'ocked=0, is_tampered=0.\r\n[350] Authenticating boo'
22-02-23 17:33:28.237 FULL [.main]: <(29) 't image (16521216): start\r\n[4'
22-02-23 17:33:28.238 FULL [.main]: <(49) '70] boot_verifier: Device is in GREEN boot state.'
22-02-23 17:33:28.238 FULL [.main]: <(49) '\r\n[470] Authenticating boot image: done return va'
22-02-23 17:33:28.238 FULL [.main]: <(06) 'lue = '
22-02-23 17:33:28.239 FULL [.main]: <(..) '1\r\n'
22-02-23 17:33:28.239 ERR [.main]: ret = 1
22-02-23 17:33:28.240 FULL [.main]: <(02) '[4'
22-02-23 17:33:28.240 FULL [.main]: <(02) '80'
22-02-23 17:33:28.240 FULL [.main]: <(02) ']'
22-02-23 17:33:28.241 FULL [.main]: <(..) 'decompressing kernel image: start\r\n'
22-02-23 17:33:28.241 ERR [.main]: attack success
22-02-23 17:33:28.637 FULL [.main]: <(16) '[880] decompress'
22-02-23 17:33:28.637 FULL [.main]: <(16) 'ing kernel image'

```



- 53%: detected malicious Linux kernel
- 45%: timeout ou crash
- 1.67%: execution of malicious Linux kernel followed by a crash
- **0.53%: success, execution of malicious Linux kernel and Android OS.**

Un contournement de Secure Boot toutes les 15 minutes

```

14.744220] platform: sound.00: Driver msm8x16_asoc_wcd requests p
14.744338] spmi msm8x16_wcd_codec-fffffc0329dc000: Driver wcd-sp
14.775369] No soundcards[14.779365] Freeing unused kernel me
14.805523] SELinux: 2048 avtab hash slots, 8186 rules.
14.828086] SELinux: 2048 avtab hash slots, 8186 rules.
14.832408] SELinux: 1 users, 2 roles, 650 types, 0 bools, 1 sens
14.839396] SELinux: 86 classes, 8186 rules
14.847784] SELinux: Completing initialization.
14.851369] SELinux: Setting up existing superblocks.
14.856515] SELinux: initialized (dev sysfs, type sysfs), uses gen
14.863813] SELinux: initialized (dev rootfs, type rootfs), uses g
14.871006] SELinux: initialized (dev bdev, type bdev), not config
14.878574] SELinux: initialized (dev proc, type proc), uses genfs
14.885521] SELinux: initialized (dev tmpfs, type tmpfs), uses tra
14.892877] SELinux: initialized (dev debugfs, type debugfs), uses
14.909778] SELinux: initialized (dev sockfs, type sockfs), uses t
14.915634] SELinux: initialized (dev pipefs, type pipefs), uses t
14.922529] SELinux: initialized (dev anon_inodefs, type anon_inod
14.931542] SELinux: initialized (dev devpts, type devpts), uses t
14.938939] SELinux: initialized (dev selinuxfs, type selinuxfs),
14.946775] SELinux: initialized (dev configfs, type configfs), no
14.955055] SELinux: initialized (dev tmpfs, type tmpfs), uses tra
14.962310] SELinux: initialized (dev sysfs, type sysfs), uses gen
15.075134] type=1403 audit(16.959:2): policy loaded auid=42949672

```



```

22-02-23 17:33:43.884 FULL [.main]: <(99) 'ialized (dev tmpfs, type tmpfs), uses transition SIDs\r\
[14.892877] SELinux: initialized (dev de
22-02-23 17:33:43.885 FULL [.main]: <(99) 'bugfs, type debugfs), uses genfs_contexts\r\n[14.90977
] SELinux: initialized (dev sockfs, type so
22-02-23 17:33:43.885 FULL [.main]: <(99) 'ckfs), uses task SIDs\r\n[14.915634] SELinux: initiali
ed (dev pipefs, type pipefs), uses task SID
22-02-23 17:33:43.927 FULL [.main]: <(99) 's\r\n[14.922529] SELinux: initialized (dev anon_inodef
, type anon_inodefs), not configured for la
22-02-23 17:33:43.927 FULL [.main]: <(99) 'beling\r\n[14.931542] SELinux: initialized (dev devpts
type devpts), uses transition SIDs\r\n[14
22-02-23 17:33:43.928 FULL [.main]: <(99) '.938939] SELinux: initialized (dev selinuxfs, type selinu
fs), uses genfs_contexts\r\n[14.946775]
22-02-23 17:33:43.928 FULL [.main]: <(99) 'SELinux: initialized (dev configfs, type configfs), not c
nfigured for labeling\r\n[14.955055] SEL
22-02-23 17:33:43.928 FULL [.main]: <(99) 'inux: initialized (dev tmpfs, type tmpfs), uses transitio
SIDs\r\n[14.962310] SELinux: initialize
22-02-23 17:33:44.038 FULL [.main]: <(99) 'd (dev sysfs, type sysfs), uses genfs_contexts\r\n[15.
75134] type=1403 audit(16.959:2): policy lo
22-02-23 17:33:44.539 FULL [.main]: <(99) 'aded auid=4294967295 ses=4294967295\r\n'
22-02-23 17:33:45.040 FULL [.main]: <(99) ''
Press Enter to continue...

```

- 53%: malicious Linux kernel
- 45%: timeout ou crash
- 1.67%: execution of malicious Linux kernel followed by a crash
- **0.53%: success, execution of malicious Linux kernel and Android OS.**

Un contournement de Secure Boot toutes les 15 minutes



- ✓ (A) Vulnerability analysis
- ✓ (B) EMFI parameters
- ✓ (C) Synchronization
- ✓ (D) Combined attack

## Conclusion

- We identified a vulnerability to fault injection in the Secure Boot of our target.
- We present a new synchronization method to trigger hardware attacks on high frequency event.
- By using this synchronization method we successfully synchronized a fault injection with the vulnerability identified, bypassing the Linux Kernel authentication step of our target Secure Boot.

## Prospect

- Use the same methodology on other targets:
  - ➔ Hardware: smartphones SoC
  - ➔ Software: previous Secure Boot steps
- Improving the hardware of our frequency detector



**Thank you for your attention.**

## Secure Boot attacks:

1. **The forgotten threat of voltage glitching: A case study on nvidia tegra x2 socs. (2021)**

Otto Bittner, Thilo Krachenfels, Andreas Galauner, and Jean-Pierre Seifert

2. **Controlling pc on arm using fault injection (2016)**

Niek Timmers, Albert Spruyt, and Marc Witteman

3. **Laser-induced fault injection on smartphone bypassing the secure boot (2017)**

Aurélien Vasselle, Hugues Thiebeauld, Quentin Maouhoub, Adèle Morisset, and Sébastien Ermeneux

4. **BADFET: Defeating Modern Secure Boot Using Second-Order Pulsed Electromagnetic Fault Injection (2017)**

Ang Cui and Rick Housley