Roll your own vulnerabilities

An introduction to fault-injection for exploiting bug-free code in embedded systems.

@barsteward,
BSides London 2024

What we’ll cover

- Who am I?
- What is fault injection?
- Types of fault injection attacks
- Why/where are fault injection attacks used?
- How can fault injection compromise security goals?
- Voltage FI Demo / How you can try this yourself
- Mitigation techniques & standardisation
- Other attacks

What we'll cover

- Who am I?
- What is fault injection?
- Types of fault injection attacks
- Why/where are fault injection attacks used?
- How can fault injection compromise security goals?
- Voltage FI Demo / How you can try this yourself
- Mitigation techniques & standardisation
- Other attacks

:~$ whoami

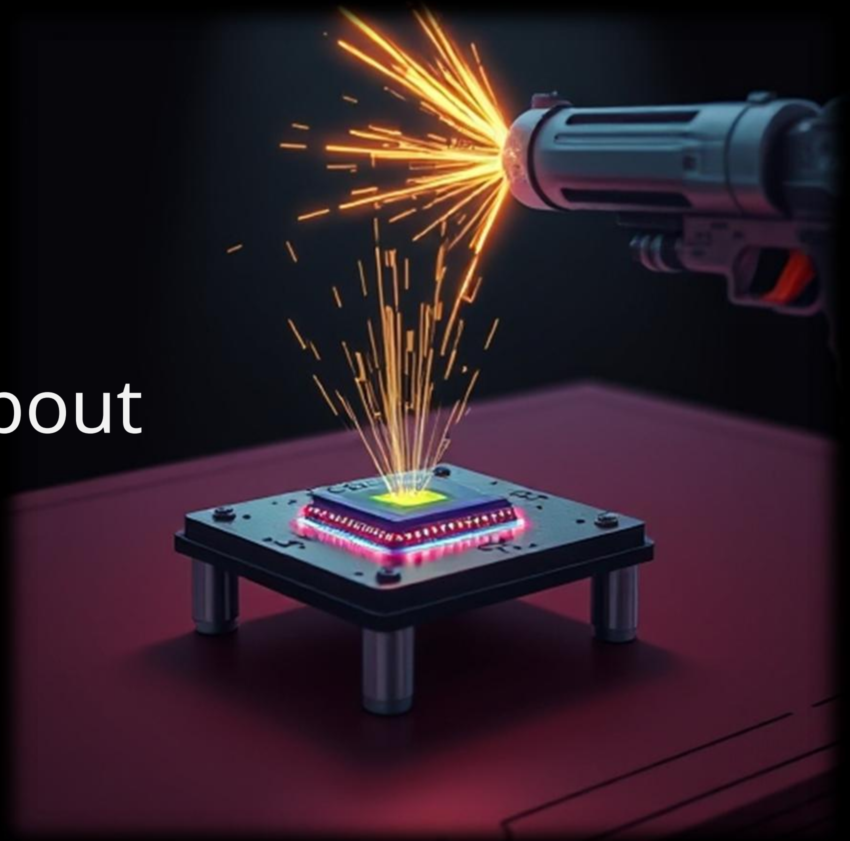
@barsteward

Bluesky: @barsteward.bsky.social
Mastodon: @barsteward@infosec.exchange
~~RIP Twitter: @barsteward~~

I'm employed to torture silicon chips
until they give up their secrets,
or agree that they work for me now.

Hardware penetration testing of claims about

- Secure Boot
- Flash Read Protection
- Debug Protection
- (and side-channel analysis resistance)



This talk does not represent the views of my employer!

What we'll cover

- Who am I?
- What is fault injection?
- Types of fault injection attacks
- Why/where are fault injection attacks used?
- How can fault injection compromise security goals?
- Voltage FI Demo / How you can try this yourself
- Mitigation techniques & standardisation
- Other attacks

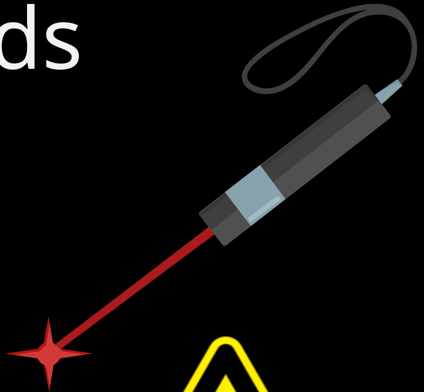
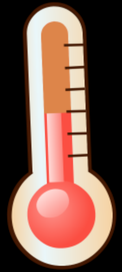
What The F (I) ?

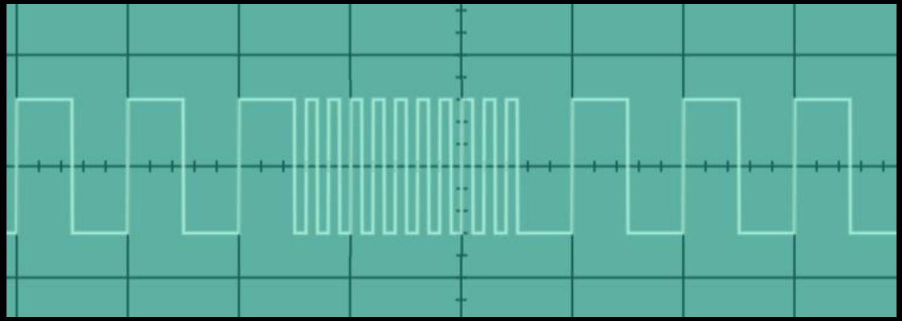
What is Fault Injection anyway?

What is Fault Injection (FI)?

Fault Injection (FI) is a class of hardware attacks in which the device is stressed in an unusual way to make it malfunction.

Extremes of
- Voltage
- Electromagnetic fields
- Clock speed
- Temperature
- Light
- Ionizing radiation…

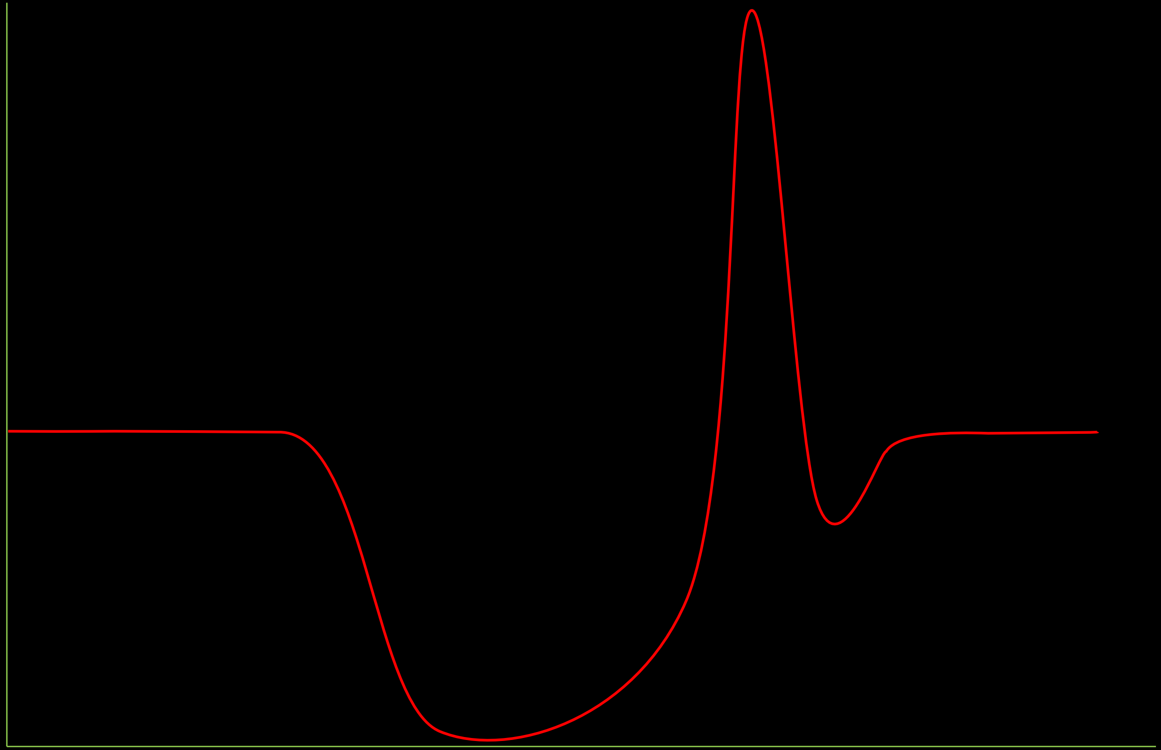


What we'll cover

- Who am I?
- What is fault injection?
- Types of fault injection attacks
- Why/where are fault injection attacks used?
- How can fault injection compromise security goals?
- Voltage FI Demo / How you can try this yourself
- Mitigation techniques & standardisation
- Other attacks

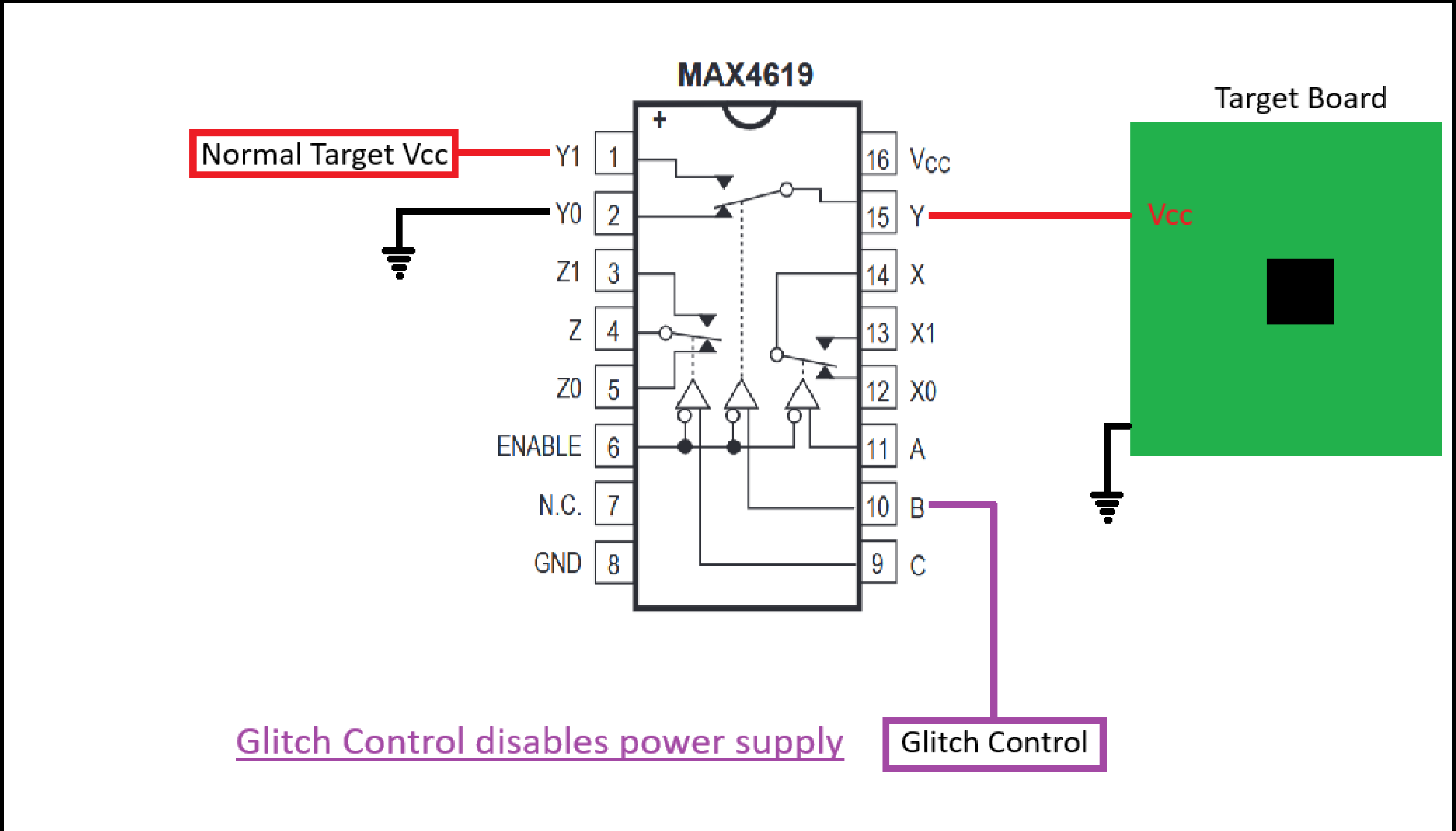
Voltage Fault Injection (VFI)

Glitches are introduced to the power rails, by briefly changing the supplied voltage, or by driving one or more pins to an incorrect voltage.



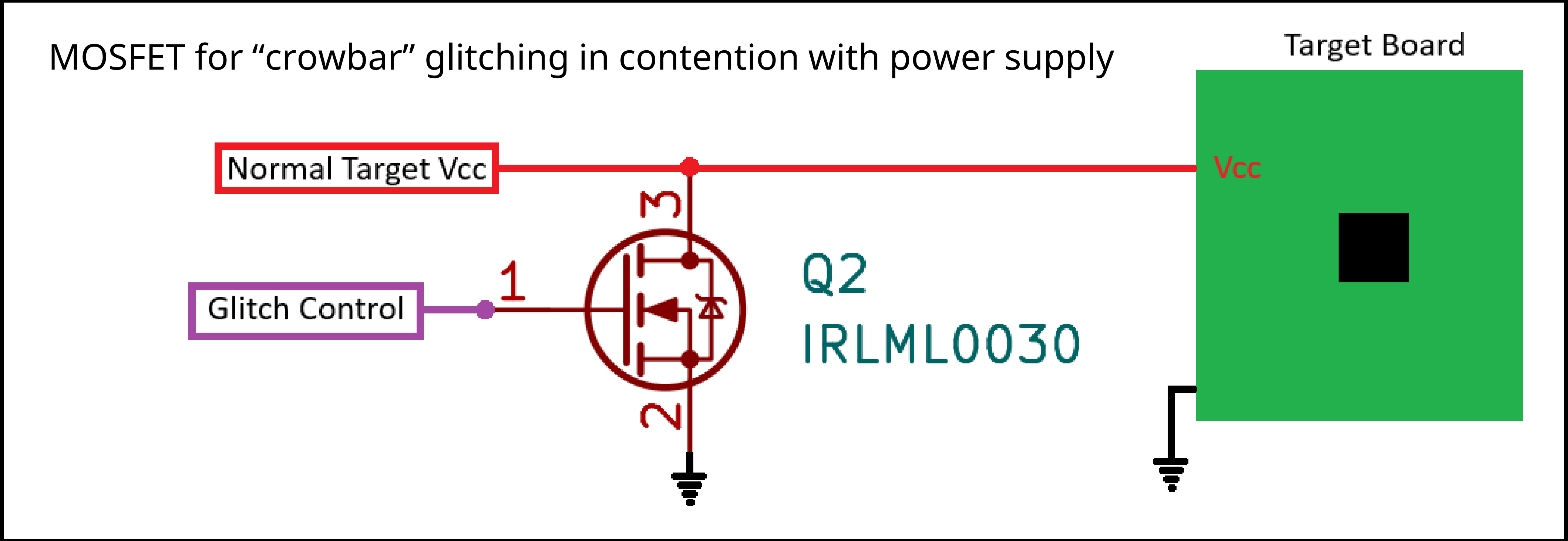
Voltage Fault Injection (VFI)

Glitches are introduced to the power rails, by briefly changing the supplied voltage, or by driving one or more pins to an incorrect voltage.



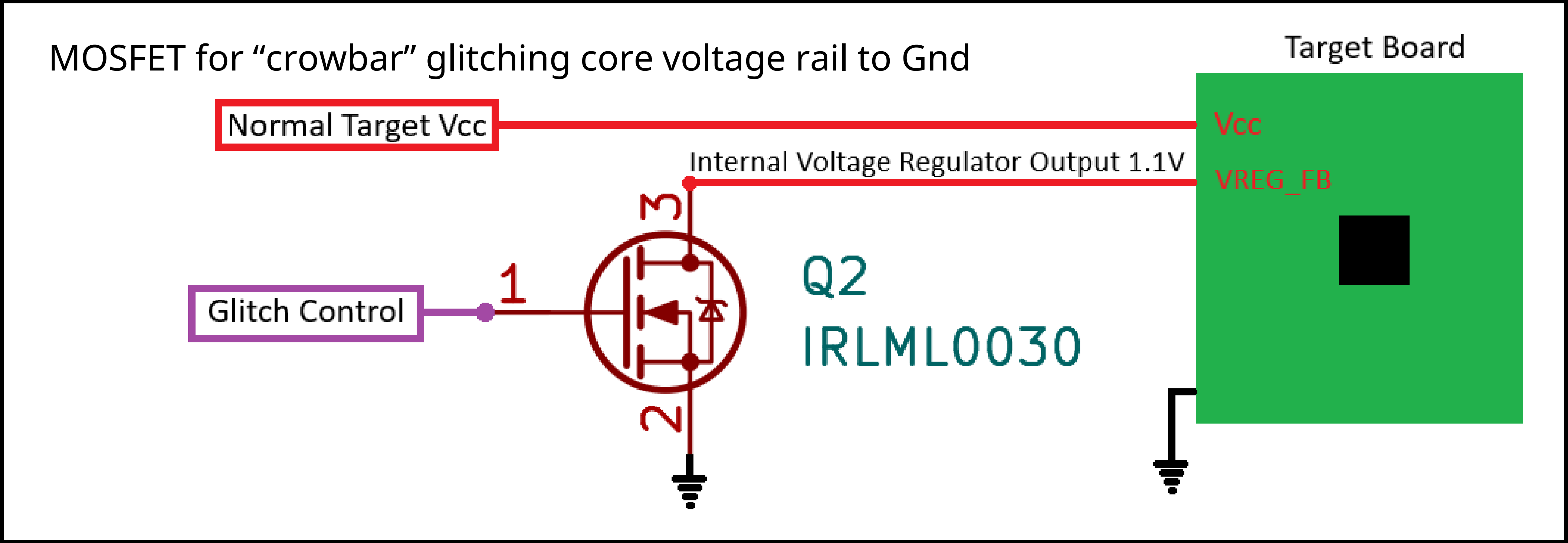
Voltage Fault Injection (VFI)

Glitches are introduced to the power rails, by briefly changing the supplied voltage, or by driving one or more pins to an incorrect voltage.

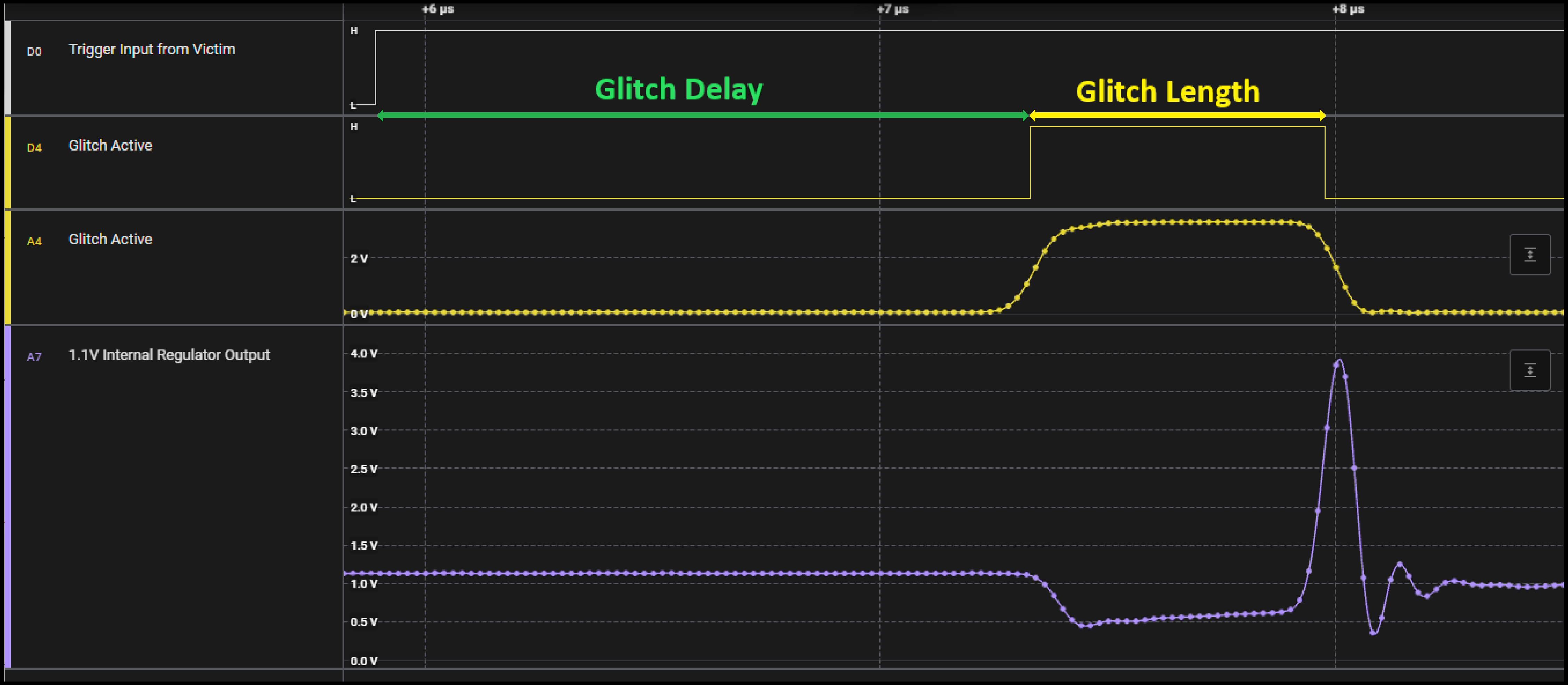


Voltage Fault Injection (VFI)

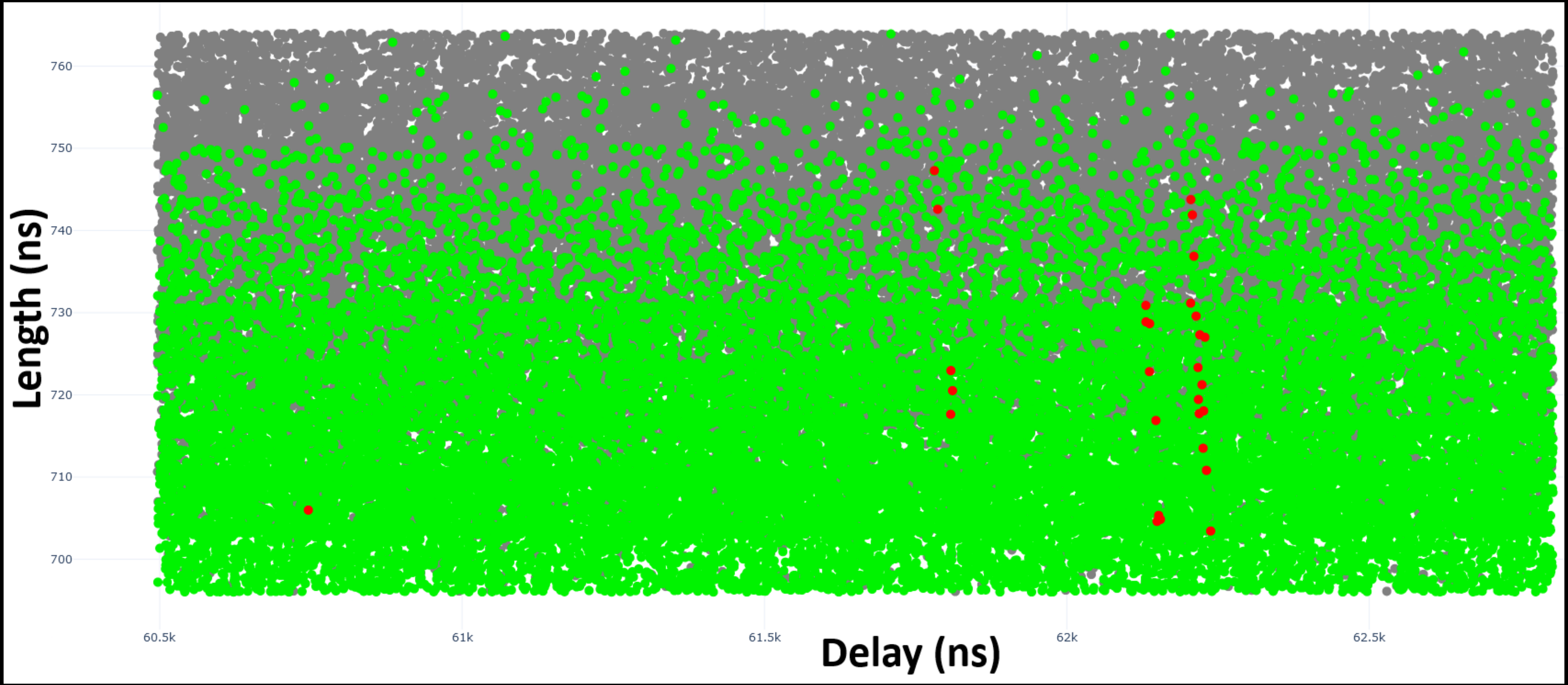
Glitches are introduced to the power rails, by briefly changing the supplied voltage, or by driving one or more pins to an incorrect voltage.



VFI: Glitch Parameters

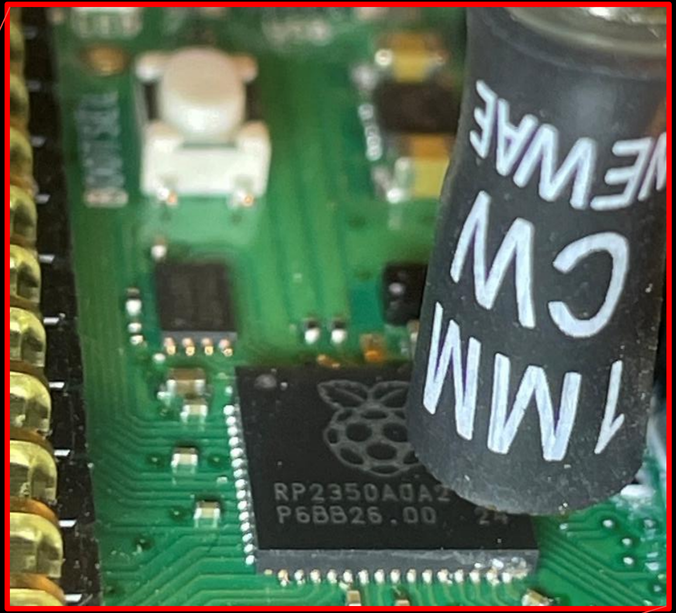


VFI: Glitch Parameter Narrowing



Electro-Magnetic Fault Injection (EMFI)

An EM pulse is delivered from a coil close to the chip



NewAE ChipSHOUTER (CW520)
~£3.5K

Riscure EMFI Transient Probe
much more expensive

Homemade circuitry/coil
<£100 but high voltages
involved, so not advised!

Generates wide range of glitch effects in target device; a real life "magic wand":
Corruption of reads/write values, program flow alteration, influencing of compare operations...
occasional release of magic smoke and chip self-destruct!

What we'll cover

- Who am I?
- What is fault injection?
- Types of fault injection attacks
- Why/where are fault injection attacks used?
- How can fault injection compromise security goals?
- Voltage FI Demo / How you can try this yourself
- Mitigation techniques & standardisation
- Other attacks

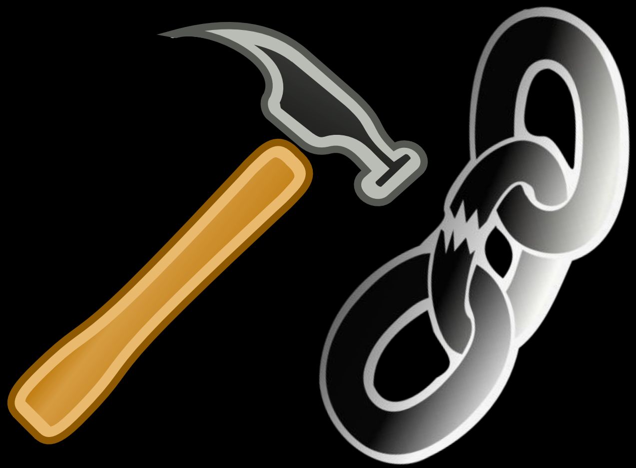
Why The F (I) ?

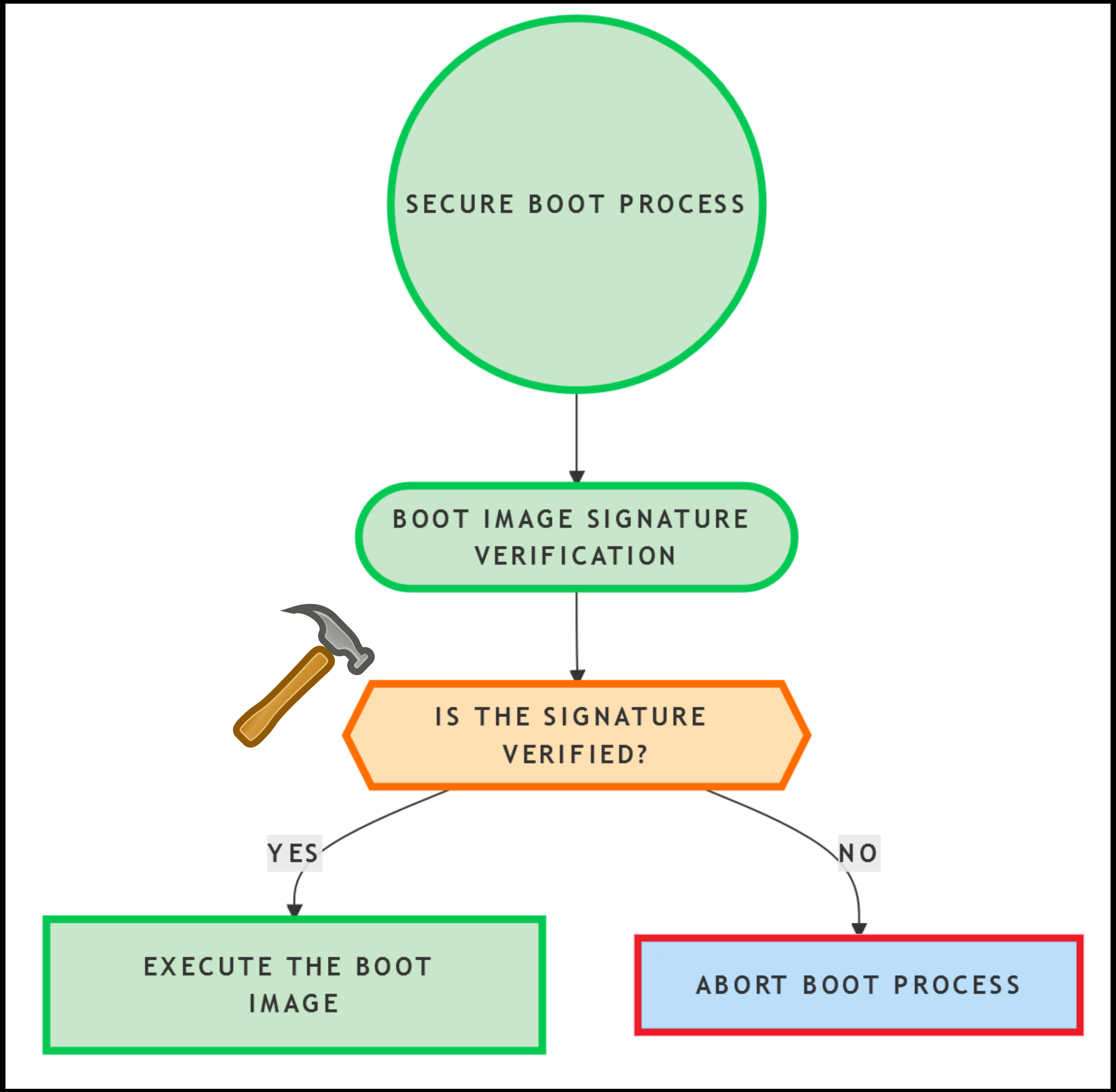
Why use Fault Injection anyway?

Why an attacker may use fault injection

<u>Secure Boot Bypass
for arbitrary code execution</u>

```
result =
VerifySignature(&image);
if (result == true)
     run(&image);
else
     Error("Abort Boot");
```

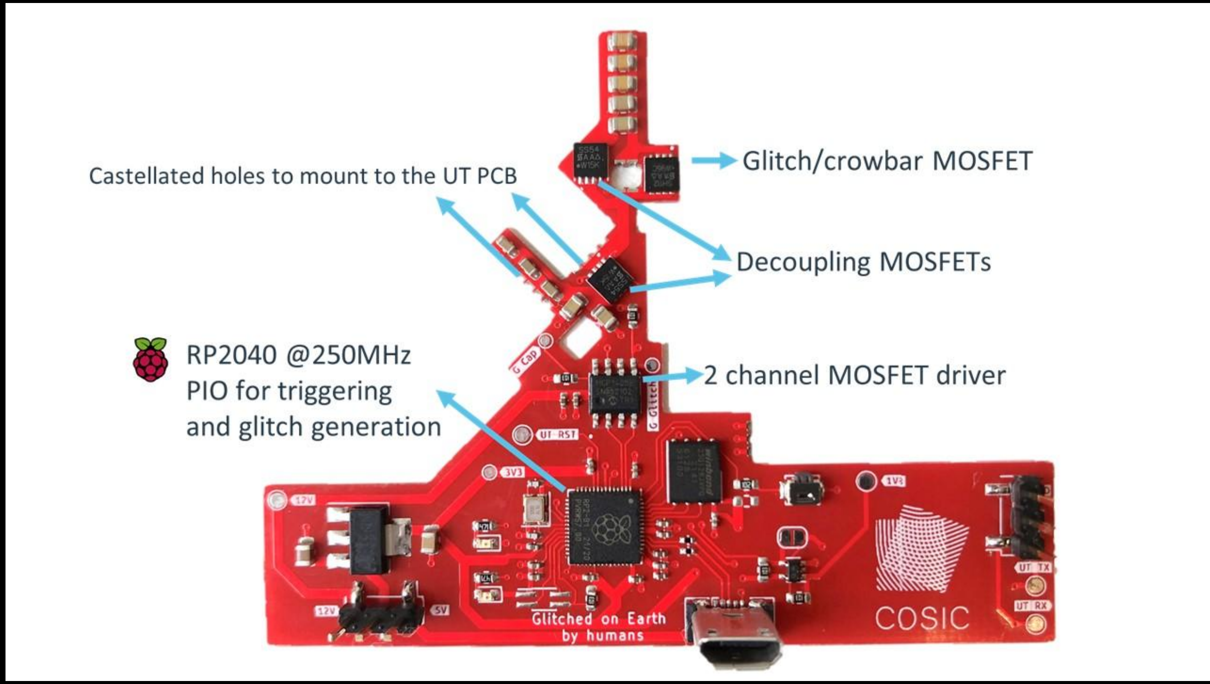




Why an attacker may use fault injection

Secure Boot Bypass

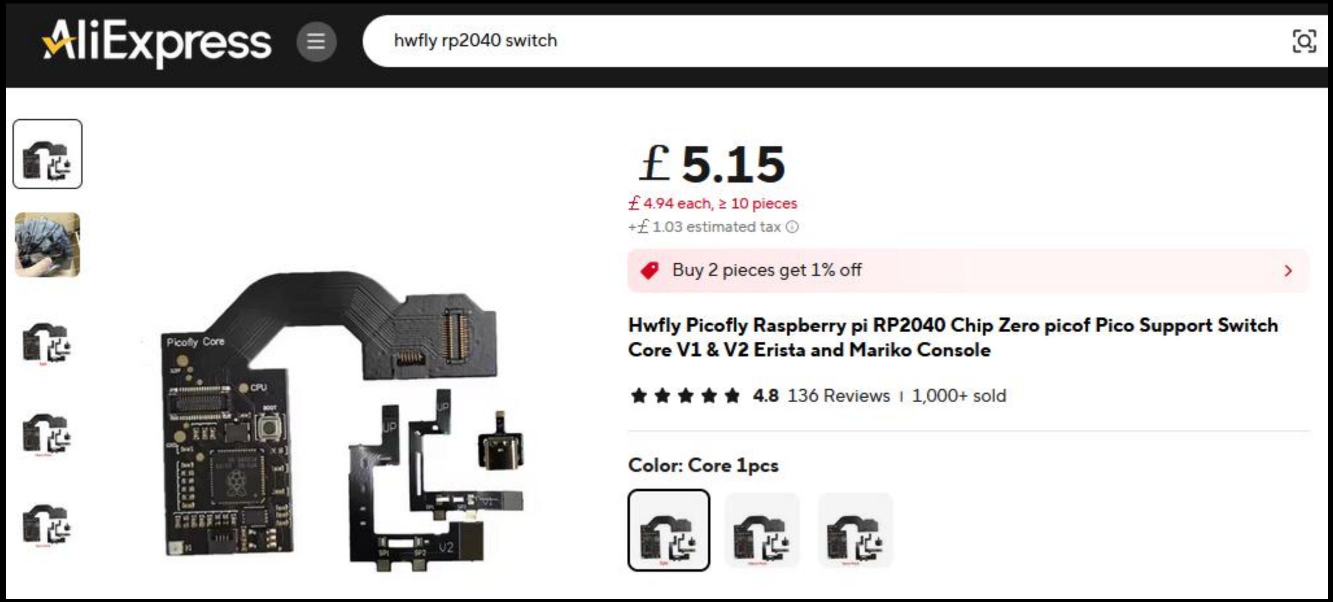
Lennert Wouters: Glitched on Earth by Humans: A Black-Box Security Evaluation of the SpaceX Starlink User Terminal
https://github.com/KULeuven-COSIC/Starlink-FI
https://www.youtube.com/watch?v=NXqLMmGwJm0



Why an attacker may use fault injection

Secure Boot Bypass

Nintendo Switch modchip for running custom firmware
https://www.retrosix.wiki/picofly-hwfly-rp2040-nintendo-switch
https://www.youtube.com/watch?v=NXqLMmGwJmo



Why an attacker may use fault injection

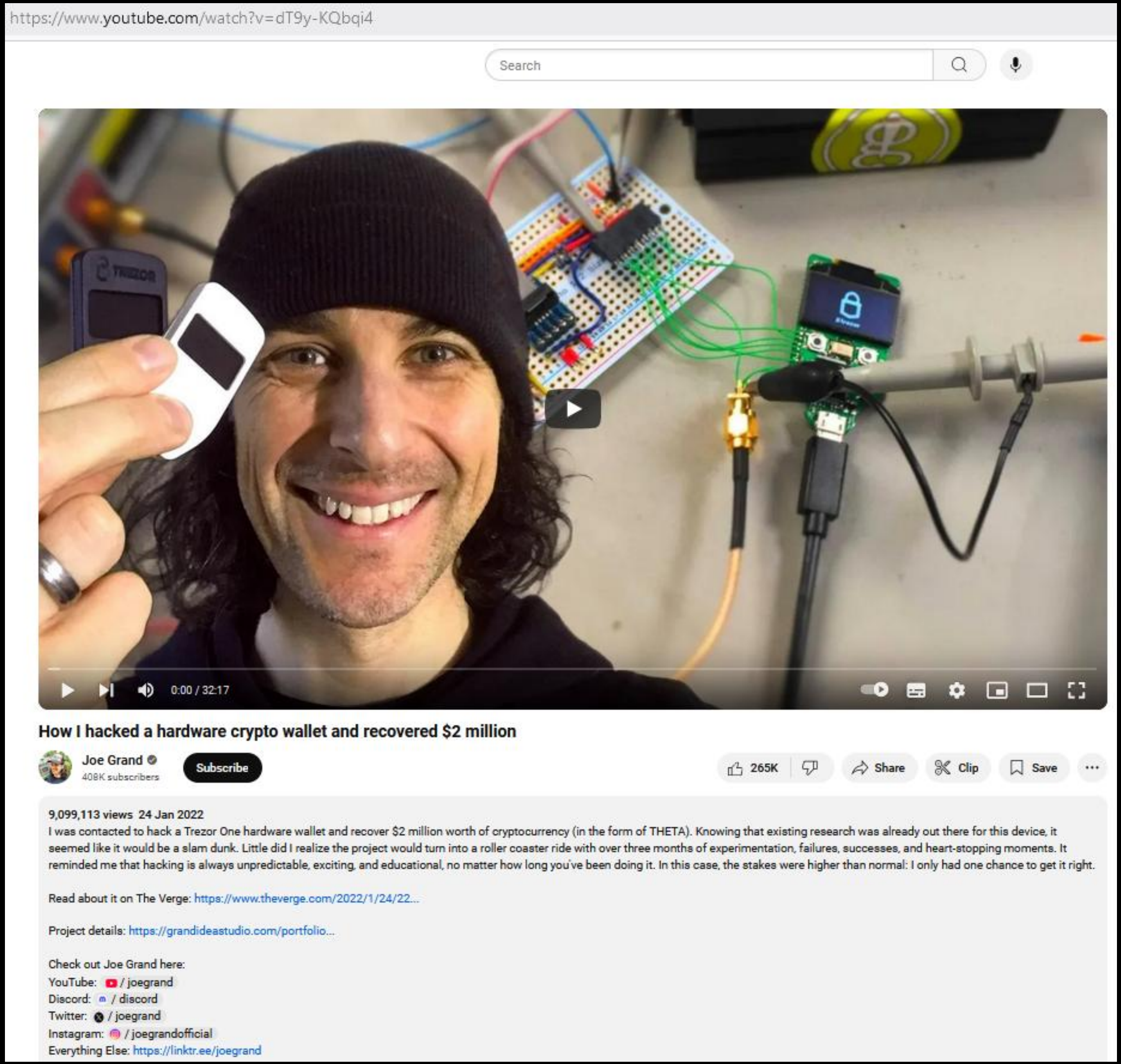
<u>Read Protection Bypass</u>
Yifan Lu was successful in dumping (reading) the ROM of the Playstation Vita using the NewAE ChipWhisperer to perform voltage fault injection.

Injecting Software Vulnerabilities with Voltage Glitching, Yifan Lu
https://arxiv.org/abs/1903.08102

Why an attacker may use fault injection

<u>Read Protection Bypass</u>

Joe Grand recovered $2m of THETA cryptocurrency from a Trezor One hardware wallet, using voltage FI
https://www.youtube.com/watch?v=dT9y-KQbqi4



What we'll cover

- Who am I?
- What is fault injection?
- Types of fault injection attacks
- Why/where are fault injection attacks used?
- How can fault injection compromise security goals?
- Voltage FI Demo / How you can try this yourself
- Mitigation techniques & standardisation
- Other attacks

How The F (I) ?

How does Fault Injection cause security violations?

How FI affects a device

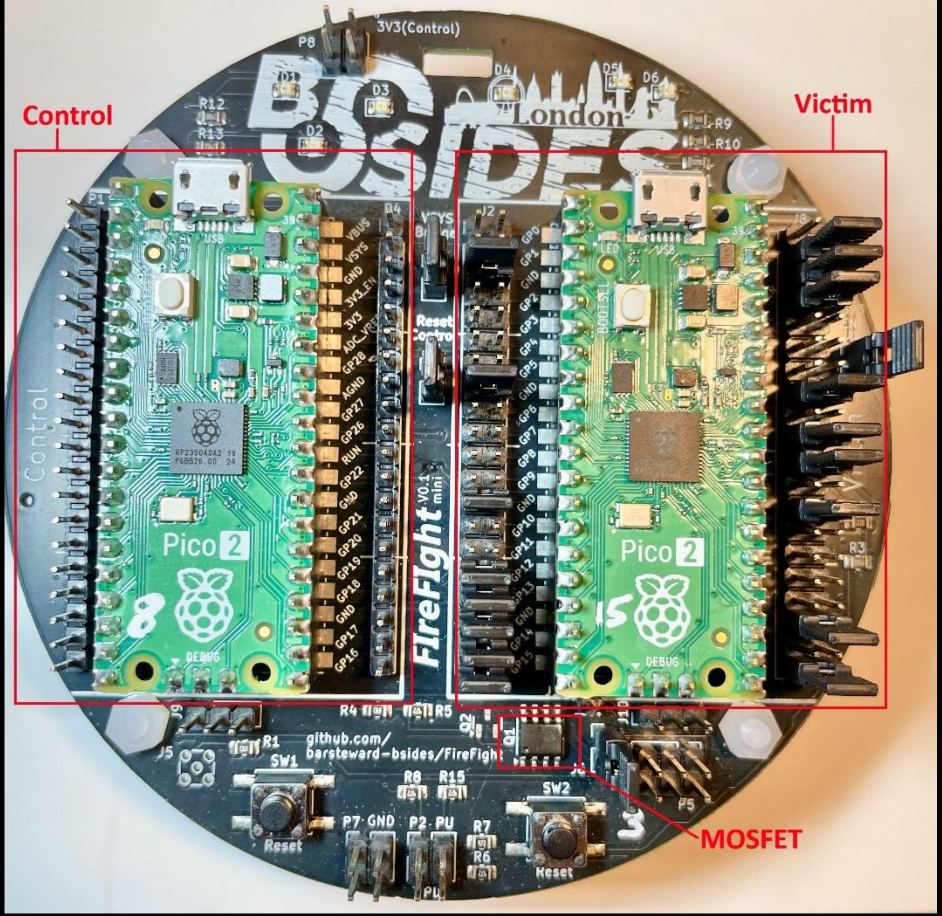
Affects program flow, security settings and internal variables. Mainly a transient effect, but can lead to permanent data changes too.

- **Instruction skipping…**
  - Most prevalent effect
- **Data in flight corruption…**
  - Misread of stored value, address/data bus corruption
- **Out of order operation…**
  - Read operation may complete early, before data fetch is complete
- **Op Code Corruption…**
  - Use of incorrect register
  - The “Jungle Jump” – program counter gets set to incorrect address and execution continues from there!

What we'll cover

- Who am I?
- What is fault injection?
- Types of fault injection attacks
- Why/where are fault injection attacks used?
- How can fault injection compromise security goals?
- Voltage FI Demo / How you can try this yourself
- Mitigation techniques & standardisation
- Other attacks

Voltage FI Demo: FIreFIght

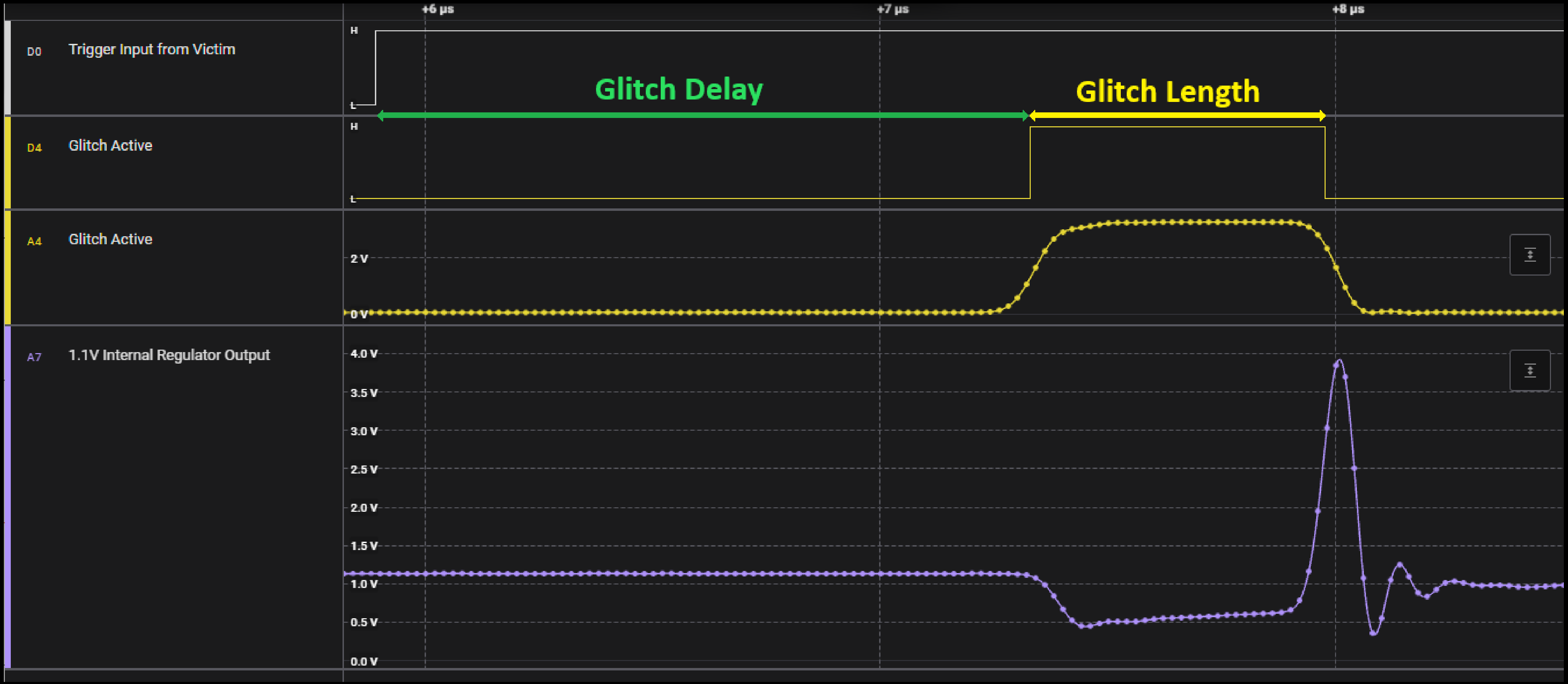


This demo will (hopefully!) show a Pi Pico 2 performing voltage glitching on another Pi Pico 2, with the aid of a MOSFET (crowbar glitching the 1.1V internal regulator output)

On the left side we have the “Control” board, and on the right, the “Victim” board.

All PCB gerber/production files and source code for this are available on github: https://github.com/barsteward-bsides/FireFight
(Thanks to AsFaBw for the PCB design work)

Voltage FI Demo: FIreFIght



https://github.com/barsteward-bsides/FireFight

Voltage FI Demo: FIreFIght DFA

Just causing an instruction skip, or a misread didn't seem to have enough jeopardy for a live conference demo, so instead let's try something a bit more tricky…

<u>Differential Fault Analysis (DFA)</u> to try to recover the AES key by analysing the faulty ciphertext outputs

Round 8	Start of round				After SubBytes				After ShiftRows				After MixColumns				Round Key Value			
80	5A	19	A3	7A	BE	D4	0A	DA	BE	D4	0A	DA	00	B1	54	FA	EA	B5	31	7F
80	41	49	E0	8C	83	3B	E1	64	3B	E1	64	83	51	C8	76	1B	D2	8D	2B	8D
80	42	DC	19	04	2C	86	D4	F2	D4	F2	2C	86	2F	89	6D	99	73	BA	F5	29
80	B1	1F	65	0C	C8	C0	4D	FE	FE	C8	C0	4D	D1	FF	CD	EA	21	D2	60	2F

EAD27321B58DBAD2312BF5607F8D292F

Round 9	Start of round				After SubBytes				After ShiftRows				After MixColumns				Round Key Value			
1B	EA	04	65	85	87	F2	4D	97	87	F2	4D	97	47	40	A3	4C	AC	19	28	57
1B	83	45	5D	96	EC	6E	4C	90	6E	4C	90	EC	37	D4	70	9F	77	FA	D1	5C
1B	5C	33	98	B0	4A	C3	46	E7	46	E7	4A	C3	94	E4	3A	42	66	DC	29	00
1B	F0	2D	AD	C5	8C	D8	95	A6	A6	8C	D8	95	ED	A5	A6	BC	F3	21	41	6E

AC7766F319FADC2128D12941575C006E

Round 10	Start of round				After SubBytes				After ShiftRows				~~After MixColumns~~				Round Key Value			
36	EB	59	8B	1B	E9	CB	3D	AF	E9	CB	3D	AF	E9	CB	3D	AF	D0	C9	E1	B6
36	40	2E	A1	C3	09	31	32	2E	31	32	2E	09	31	32	2E	09	14	EE	3F	63
36	F2	38	13	42	89	07	7D	2C	7D	2C	89	07	7D	2C	89	07	F9	25	0C	0C
36	1E	84	E7	D2	72	5F	94	B5	B5	72	5F	94	B5	72	5F	94	A8	89	C8	A6

D014F9A8C9EE2589E13F0CC8B6630CA6

Output Ciphertext			
39	02	DC	19
25	DC	11	6A
84	09	85	0B
1D	FB	97	32

3925841D02DC09FBDC118597196A0B32

Voltage FI Demo: FIreFIght DFA

Just causing an instruction skip, or a misread didn't seem to have enough jeopardy for a live conference demo, so instead let's try something a bit more tricky…

Differential Fault Analysis (DFA) to try to recover the AES key by analysing the faulty ciphertext outputs

Round 8	Start of round				After SubBytes				After ShiftRows				After MixColumns				Round Key Value				EAD27321B58DBAD2312BF5607F8D292F
80	5A	19	A3	7A	BE	D4	0A	DA	BE	D4	0A	DA	00	B1	54	FA	EA	B5	31	7F	
80	41	49	E0	8C	83	3B	E1	64	3B	E1	64	83	50	C8	76	1B	D2	8D	2B	8D	
80	42	DC	19	04	2C	86	D4	F2	D4	F2	2C	86	2F	89	6D	99	73	BA	F5	29	
80	B1	1F	65	0C	C8	C0	4D	FE	FE	C8	C0	4D	D1	FF	CD	EA	21	D2	60	2F	

Round 9	Start of round				After SubBytes				After ShiftRows				After MixColumns				Round Key Value				AC7766F319FADC2128D12941575C006E
1B	EA	04	65	85	87	F2	4D	97	87	F2	4D	97	47	40	A3	56	AC	19	28	57	
1B	82	45	5D	96	13	6E	4C	90	6E	4C	90	13	37	D4	70	7A	77	FA	D1	5C	
1B	5C	33	98	B0	4A	C3	46	E7	46	E7	4A	C3	94	E4	3A	BD	66	DC	29	00	
1B	F0	2D	AD	C5	8C	D8	95	A6	A6	8C	D8	95	ED	A5	A6	43	F3	21	41	6E	

Round 10	Start of round				After SubBytes				After ShiftRows				~~After MixColumns~~				Round Key Value				D014F9A8C9EE2589E13F0CC8B6630CA6
36	EB	59	8B	01	E9	CB	3D	7C	E9	CB	3D	7C	E9	CB	3D	7C	D0	C9	E1	B6	
36	40	2E	A1	26	09	31	32	F7	31	32	F7	09	31	32	F7	09	14	EE	3F	63	
36	F2	38	13	BD	89	07	7D	7A	7D	7A	89	07	7D	7A	89	07	F9	25	0C	0C	
36	1E	84	E7	2D	72	5F	94	D8	D8	72	5F	94	D8	72	5F	94	A8	89	C8	A6	

Output Ciphertext

39	02	DC	CA
25	DC	C8	6A
84	5F	85	0B
70	FB	97	32

3925847002DC5FFBDCC88597CA6A0B32

Voltage FI Demo: FIreFIght

Live
Demo

This is not a new attack and it heavily relies upon open source libraries such as @Doegox's PhoenixAES

What we'll cover

- Who am I?
- What is fault injection?
- Types of fault injection attacks
- Why/where are fault injection attacks used?
- How can fault injection compromise security goals?
- Voltage FI Demo / How you can try this yourself
- Mitigation techniques & standardisation
- Other attacks

Mitigation Techniques

To protect the security goals of a system, numerous mitigation techniques can be employed:

- Software hardening techniques
  - Fail-safe default initialised values
  - Avoid trivial values for constants such as 0 and 1; maximise hamming distance
  - Repeated checks for comparisons
  - Checks that loops completed the correct number of iterations
  - Randomised timing delays, to make repeatability and attack parameter narrowing harder
  - Control flow integrity checks
  - … See Riscure's "Fault Mitigation Patterns" whitepaper for more details: https://www.riscure.com/publication/fault-mitigation-patterns/

It's really tricky to write code that will fail safe during a hardware attack

Mitigation Techniques

To protect the security goals of a system, numerous mitigation techniques can be employed:

- Hardware techniques
  - Glitch resistant internal power circuitry
  - Glitch detectors
    - Voltage monitoring circuitry
    - Oscillator disturbance detection
    - Honeypot logic paths
  - Memory Protection Units to prevent code execution in restricted areas
  - Shielding
  - Control flow integrity mechanisms

Mitigation Techniques: RP2350 (Raspberry Pi Pico 2)

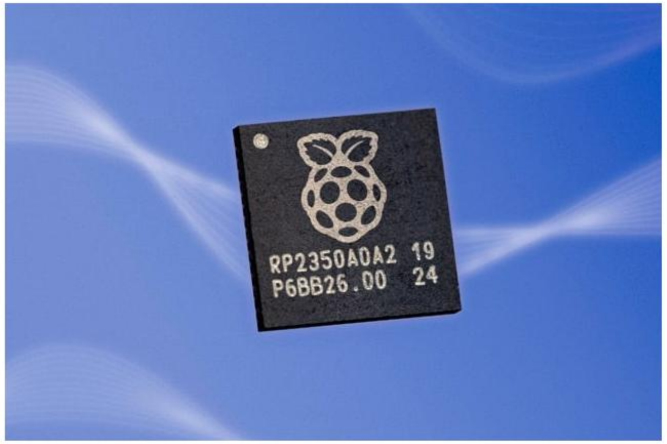
The RP2350 chip used for the Pi Pico 2 includes a configurable glitch detector, and there's a ~~$10,000~~ $20,000 bug bounty for bypassing the chip security features and recovering a secret stored in the OTP flash memory

https://www.raspberrypi.com/news/30000-badges-and-still-no-hack/

Can you hack our new chip?

16th Aug 2024 Chris Boross 2 comments

We think RP2350, our new high-performance, secure microcontroller, is pretty safe and sturdy. Care to test that theory? (We fully admit that everything is hackable given enough time and resources.)



Challenge Accepted!

Before we launched RP2350 and Raspberry Pi Pico 2, we wanted to do some testing on the security features of the chip and software, so we worked with some of the best names in the security testing game: Thomas Roth and Colin O'Flynn.

https://github.com/raspberrypi/rp2350_hacking_challenge

10.9. Glitch Detector

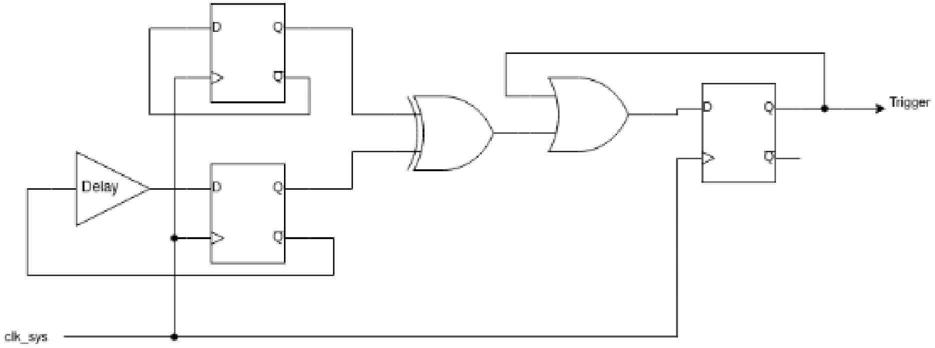
The glitch detector detects loss of setup and hold margin in the system clock domain, which may be caused by deliberate external manipulation of the system clock or core supply voltage. When it detects loss, the glitch detector triggers a system reset rather than allowing software to continue to execute in a possibly undefined state. It responds within one system clock cycle, unlike the brownout detector, which has much more limited analog bandwidth.

The glitch detector is disabled by default, and can be armed by setting the `GLITCH_DETECTOR_ENABLE` flag in OTP. For debugging purposes, you can also enable the glitch detector via the ARM register. This is not recommended in security-sensitive applications, as the system is vulnerable until the point that software can enable the detectors.

10.9.1. Theory of Operation

The glitch detector is comprised of four identical detector circuits, based on a pair of D flip-flops. These detector circuits are each placed in different, physically distant locations within the core voltage domain.

Figure 42. Glitch detector trigger circuit. Two flops each toggle on every system clock cycle. One has a programmable delay line in its feedback path, the other does not. Loss of setup or hold margin causes one of the flops to fail to toggle, so the flops values differ, setting the trigger output.



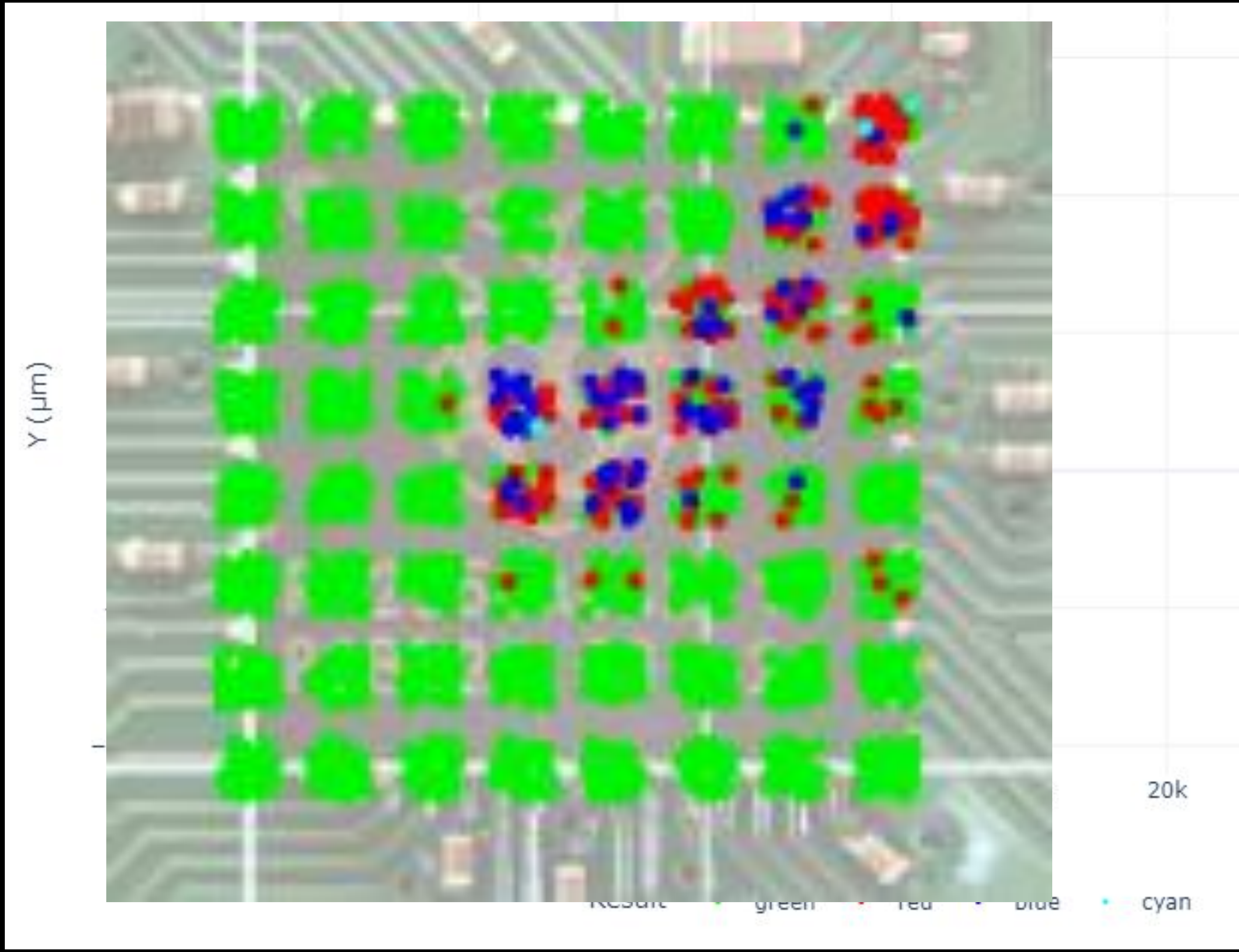
https://datasheets.raspberrypi.com/rp2350/rp2350-datasheet.pdf

Mitigation Techniques: RP2350 (Raspberry Pi Pico 2)

So how effective is the glitch detector?

This is an EMFI scan I did before enabling the glitch detector, on a simple nested for loop counter…

Red = Successful glitch
Green = Expected response
Blue = Device reboot
Cyan = Corrupted response



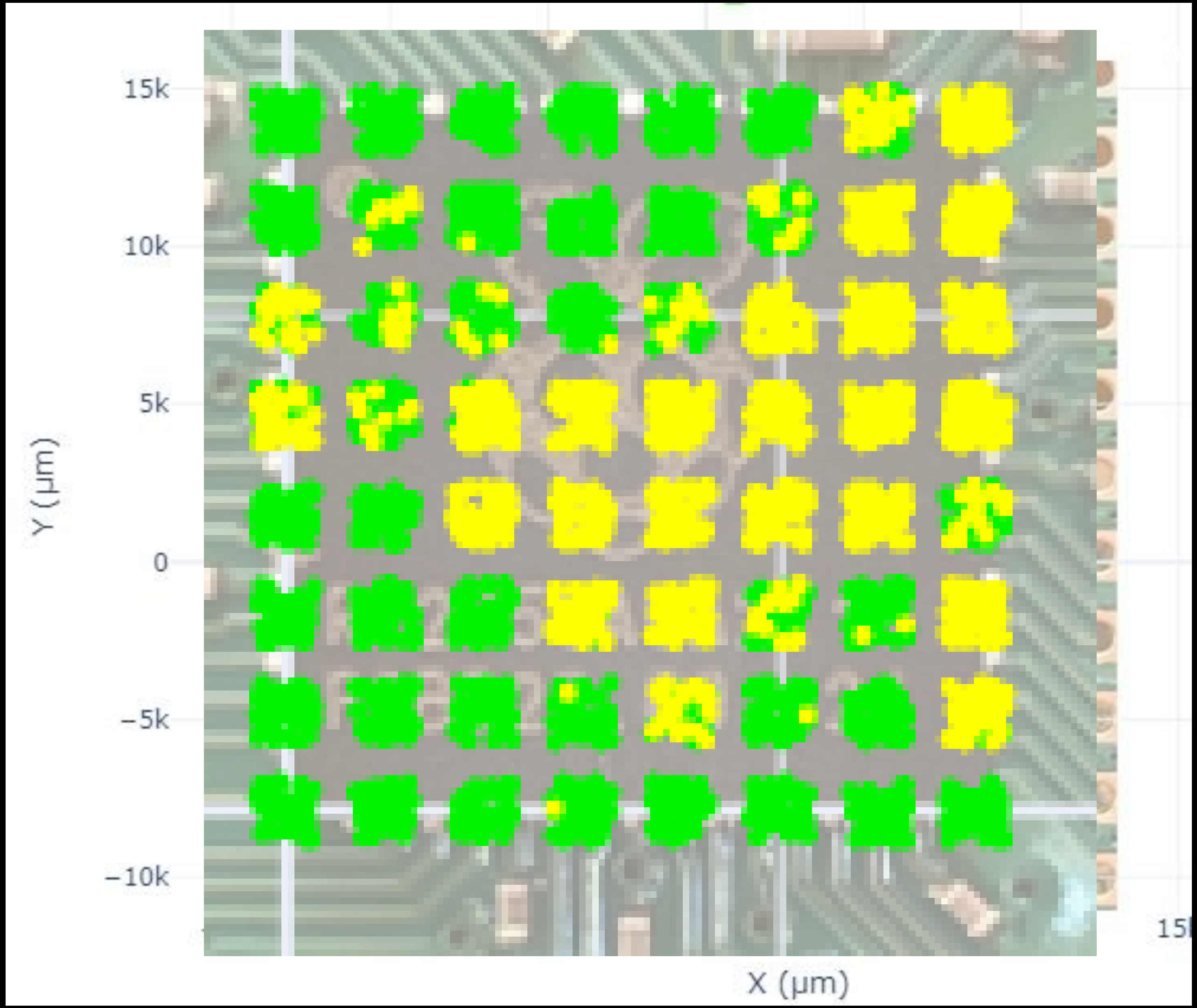
Mitigation Techniques: RP2350 (Raspberry Pi Pico 2)

So how effective is the glitch detector?

And with the glitch detector enabled…

Yellow = Glitch Detected
Green = Expected response
Red = Successful glitch

(1mm Coil, lower power, plus a week of trying)



Mitigation Techniques: RP2350 (Raspberry Pi Pico 2)

However… on Dec 27th at 38C3, there's a talk claiming to have defeated the challenge

Aedan Cullen

Most of what I do is related to embedded systems, robotics, or efficient computing. Other fields, like security research, are just a byproduct of always learning how things work :)

Session

12-27
23:00
60min

Hacking the RP2350

Aedan Cullen

Raspberry Pi's RP2350 microcontroller introduced a multitude of new hardware security features over the RP2040, and included a Hacking Challenge which began at DEF CON to encourage researchers to find bugs. The challenge has been defeated and the chip is indeed vulnerable (in at least one way). This talk will cover the process of discovering this vulnerability, the method of exploiting it, and avenues for deducing more about the relevant low-level hardware behavior.

Security

Saal ZIGZAG

https://fahrplan.events.ccc.de/congress/2024/fahrplan/talk/39HFD9/

Standards and certifications

Standards and certification schemes are influencing things:

- Common Criteria (ISO/IEC 15408:2022) is the certification standard for Smart Cards and high security devices, but this is very expensive to comply with.
- NIST FIPS 140-3
- Automotive standard ISO/SAE 21434:2021 has forced automotive manufacturers to consider these types of attacks
- Certification schemes such as ARM PSA and SESIP have a number of levels, some of which require resistance to FI attacks
- The EU Cyber Resilience Act will enforce strict incident reporting rules, which may also influence product security decisions.

What we'll cover

- Who am I?
- What is fault injection?
- Types of fault injection attacks
- Why/where are fault injection attacks used?
- How can fault injection compromise security goals?
- Voltage FI Demo / How you can try this yourself
- Mitigation techniques & standardisation
- Other attacks

Invasive attacks(if time!)

- Decapsulation and optical read of ROM
- Micro-probing to connect to internal signals or to connect/disconnect internal lines
- Body Bias Injection: voltage glitch to the ground plane inside the chip – this raises the Gnd voltage and can cause localised data misreads due to the reduced potential difference between Gnd and core voltage.

Side Channel Analysis (if time!)

- Detection of tiny data dependant fluctuations in timing, power or electromagnetic emissions.
- It can be possible to fully recover a cryptographic key that's in use, by capturing and analysing the EM emmissions from the chip, by placing a near-field microprobe close to the chip surface.

https://www.langer-emv.de/en/product/near-field-microprobe-sets-icr-hh-h-field/26/icr-hh100-27-set-near-field-microprobe-1-5-mhz-6-ghz/768/icr-hh100-27-near-field-microprobe-1-5-mhz-to-6-ghz/101

Conclusions

- It’s hard (and costly) to protect against physical attacks on hardware if people can get access to the chip.
- These attacks are becoming more widely known/exploited and the tools are getting cheaper.
- Glitch detectors (and other mitigations) can make a huge difference to the difficulty and repeatability of a fault injection attack, but they’re not perfect.
- There is more effort going into hardware and software protection mechanisms now too.
- System design that avoids storage of secrets is a great defence, but not always practical

Code Credit

FIreFIght control interface, including PIO glitch control: @barsteward

https://github.com/barsteward-bsides/FireFight

DFA Key recovery library phoenixAES: Philippe Teuwen @doegox

https://github.com/SideChannelMarvels/JeanGrey/tree/master/phoenixAES

AES key schedule library aeskeyschedule: Marcel Nageler @fanoster

https://github.com/fanosta/aeskeyschedule

Other Credits

PCB Design: AsFaBw https://github.com/AsFaBw

ChipSHOUTER EMFI probe: Colin O'Flynn @oflynn.com (NewAE)
https://www.newae.com/ (Check out the ChipWhisperer too)

Incredible patience: My wife
Fortunately, no link!

Image Credits

- Morph image used under licence: https://creativecommons.org/licenses/by/4.0/
  Science Museum Group. Model of 'Morph'. 1999-5162 Science Museum Group Collection Online.
  https://collection.sciencemuseumgroup.org.uk/objects/co8180635/model-of-morph
- AI image of chip torture https://deepai.org/
- Homer Simpson light switch https://giphy.com/gifs/season-3-the-simpsons-3x24-xT5LMWOExnRmXt2vFS
- MAX4619 image under fair usage from https://www.analog.com/media/en/technical-documentation/data-sheets/MAX4617-MAX4619.pdf
- Starlink Dish image https://commons.wikimedia.org/wiki/File:A_Bright_New_Day_for_Broadband_%E2%80%94_Starlink_(51016637753).jpg Steve Jurvetson from Los Altos, USA, Creative Commons Attribution 2.0
- Starlink Modchip image used with permission from Lennert Wouters modchip.jpg: https://github.com/KULeuven-COSIC/Starlink-FI
- Joe Grand screenshot from YouTube used under fair usage https://www.youtube.com/watch?v=dT9y-KQbqi4
- Nintendo Switch image by PantheraLeo1359531: https://upload.wikimedia.org/wikipedia/commons/7/70/Nintendo_Switch_OLED-Modell_%28BeatEmUps%29_20211001_08.png
  Licensed under the Creative Commons Attribution 3.0 Unported license
- All other images and clipart unrestricted or included under fair usage

These slides, along with the FIreFIght PCB design files, and the embedded FIreFIght code for the DFA demo are available at:

https://github.com/barsteward-bsides/FireFight
(Definitely not a Rick-Roll)

- Bluesky: @barsteward.bsky.social
- Mastodon: @barsteward@infosec.exchange
- ~~RIP Twitter: @barsteward~~