MINimum Failure

Stealing Bitcoins with Electromagnetic Fault Injection

Colin O'Flynn

C.T.O., NewAE Technology Inc.

Assistant Professor, Dalhousie University

My Dual Life

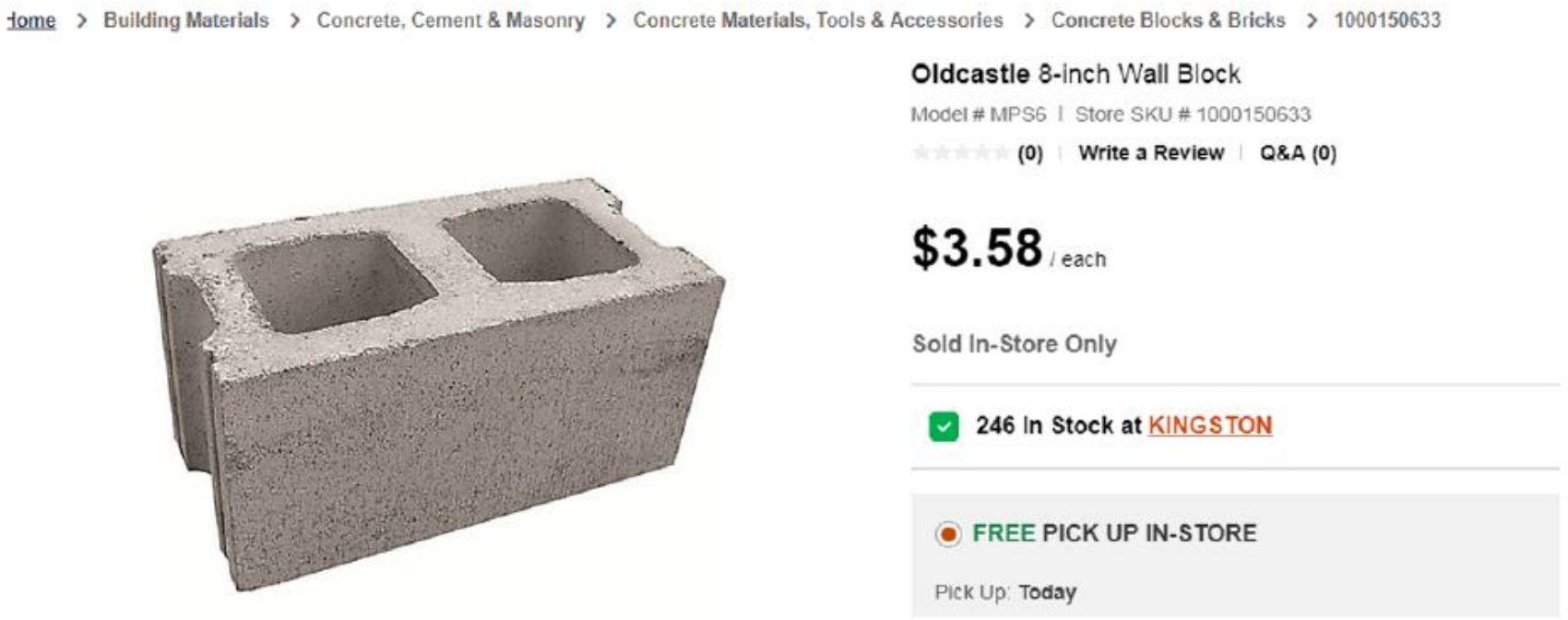
- C.T.O @ NewAE Technology Inc.
  - NewAE produces embedded tooling for hardware security validation
    - 700+ customers in 44 countries.
  - Started the open-source *ChipWhisperer* project.

- Assistant Professor @ Dalhousie University
  - Part of Electrical & Computer Engineering Dept.
  - Working on cybersecurity research + with local embedded startups.

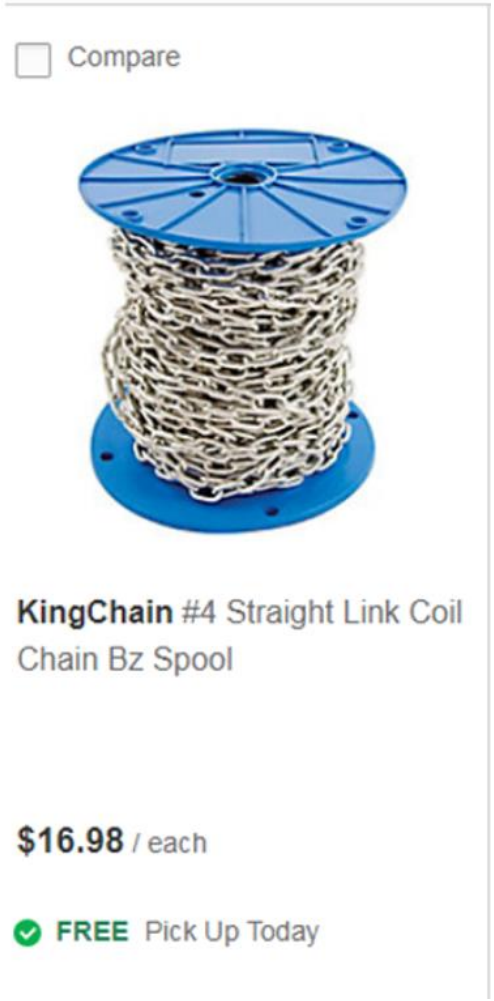
The Talk With it All!! In no particular order:

- A true history of block chain!
- Me gambling with 0.3 BTC!
- Physical give-aways!
- Attacking bitcoin wallets & stealing funds!
- Stealing authentication credentials!
- Crass commercialization!
- Open-Source tool release!
- Fixing your crappy code!

Blockchain Background (1/2)



Blockchain Background (2/2)



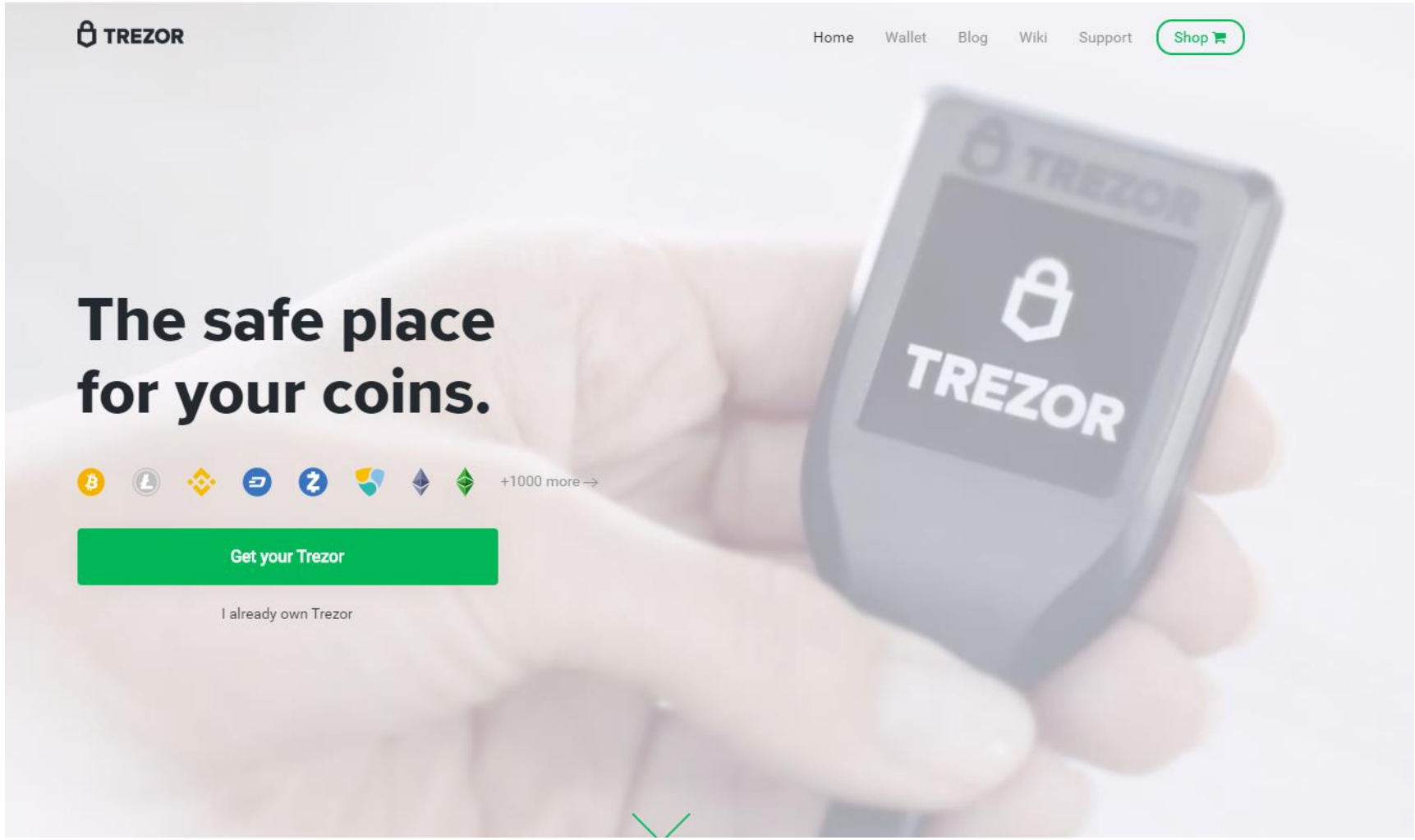
Example: Blockchain Secures Voting Booth

History

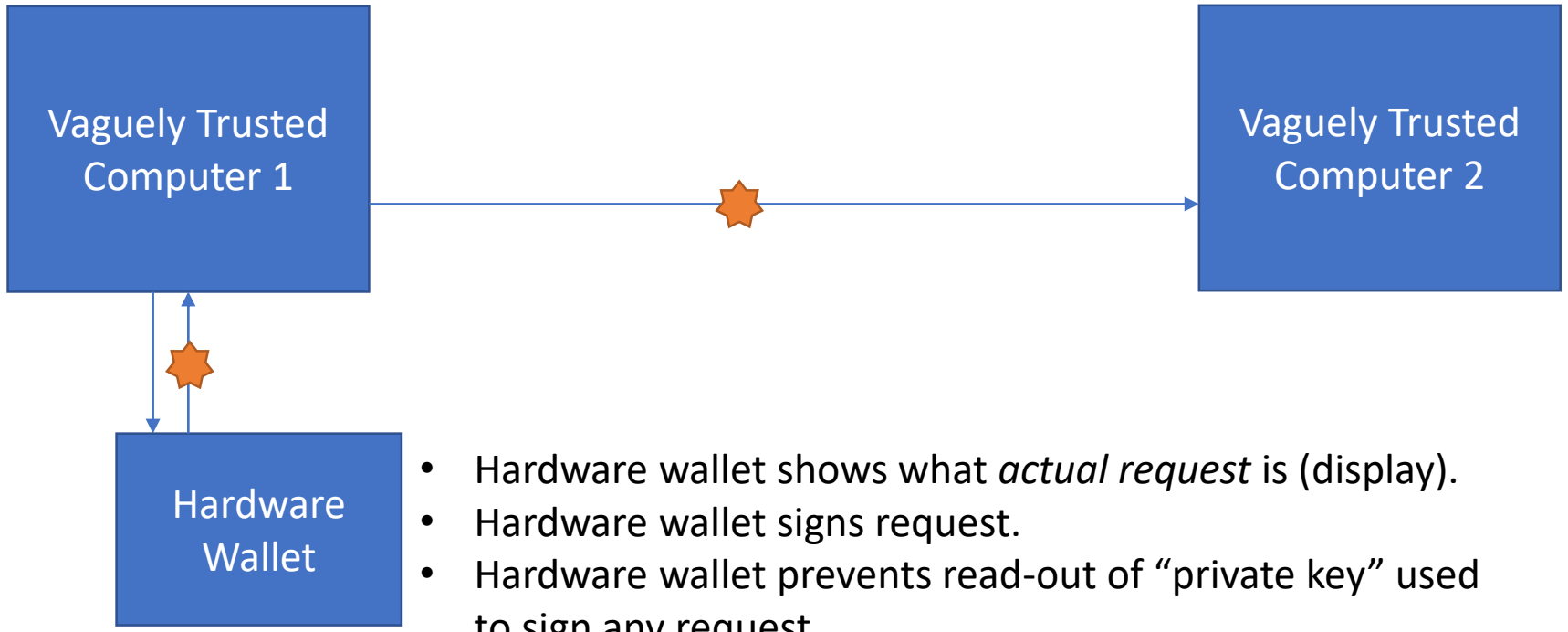
- May 11, 2008: First concrete implementation.
  (H. Simpson)
- Oct 31, 2008: More well-known computer implementation.
  (S. Nakamoto)

- ~~**A true history of block chain!**~~
- Me gambling with 0.3 BTC!
- Physical give-aways!
- Attacking bitcoin wallets & stealing funds!
- Stealing authentication credentials!
- Crass commercialization!
- Open-Source Tool Release!
- Fixing your crappy code!

Let’s Embed Those Blockchains



Why a Hardware Wallet?



- Hardware wallet shows what *actual request* is (display).
- Hardware wallet signs request.
- Hardware wallet prevents read-out of “private key” used to sign any request.

Trezor Features

Security measures of Trezor.

Firmware verification.

The bootloader verifies the firmware signature. The device only runs if the firmware is correctly signed by SatoshiLabs. Otherwise, the Trezor warns you.

Protected key operations.

Operations with private and public keys are only allowed after user authentication via PIN.

Additional passphrase support.

Trezor supports BIP39 passphrases, which are never stored or remembered on the device.

Reliable backup & recovery.

Your recovery seed protects you against theft, loss or destruction of your device. Simply restore the recovery seed, and your wallet is back.

Ultrasound hardware seal.

Trezor hardware case is ultrasonically welded, making it difficult to be restored after breakage.

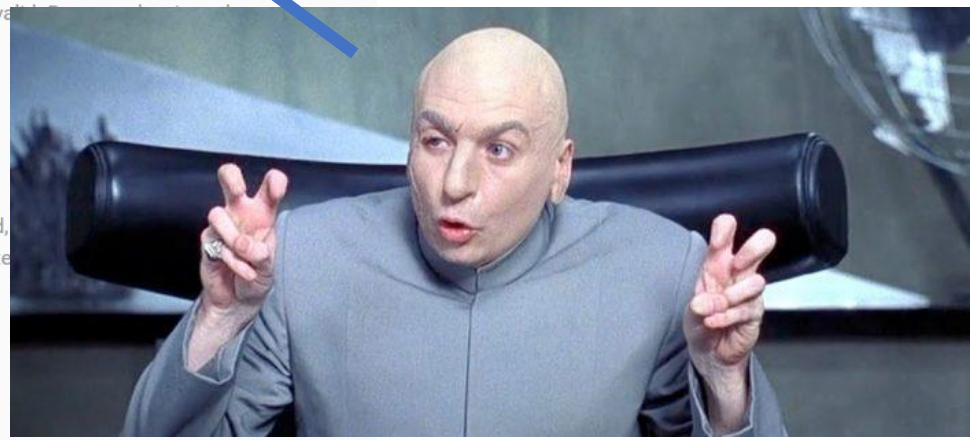
Secure update procedure.

The bootloader erases memory on firmware update and res es it only if the firmware signature is va
memory.

Write-protected bootloader.

The bootloader is write protected,
safeguarded by the Memory Prote

Learn more about security



Example: Ultrasonic Seal

<Hopefully my camera works>

Why Pick on Trezor?

- Trezor is Open Source
  - Anyone can validate the source code, modify it, etc.
  - Lessons learned on Trezor can be applied outside of bitcoin wallets.
    - Embedded systems, IoT, automotive, etc.

- **This problem I'm disclosing has been fixed with issued firmware patch.**

About Bitcoin Recovery Seeds

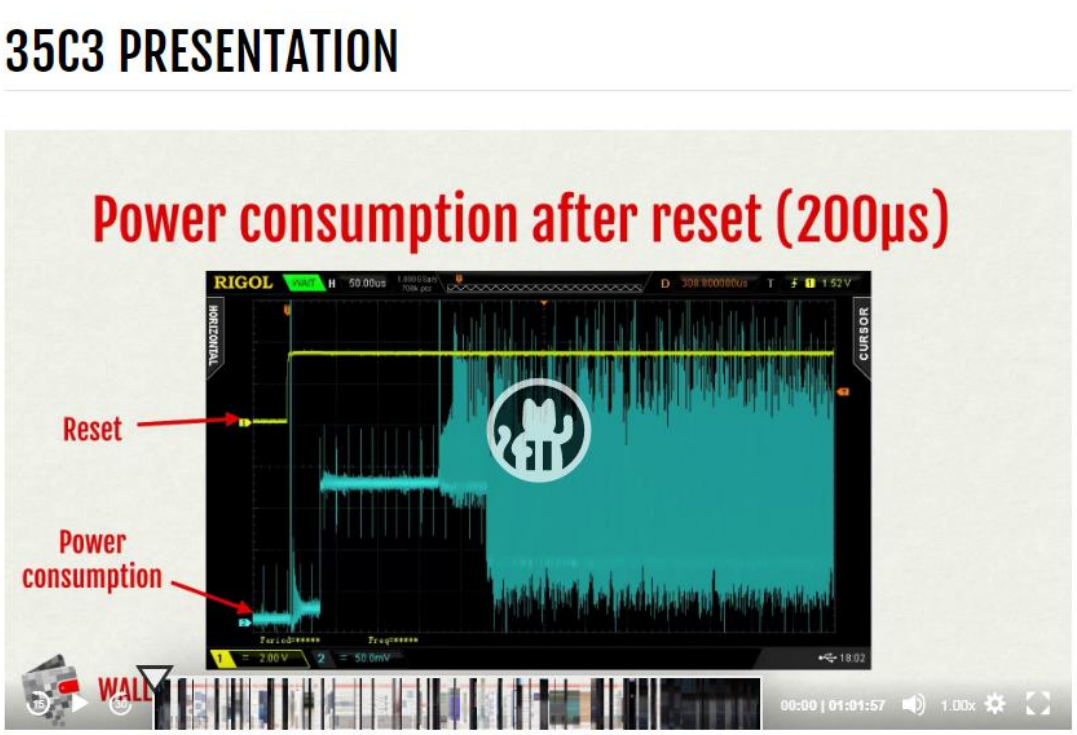
From Trezor documentation:

> **Understanding the recovery seed.**
>
> The recovery seed is a crucial element for the security of your Trezor hardware wallet. If your device is lost, damaged or stolen, you can use your **recovery seed to restore access to your entire wallet, passwords and other data** associated with it. The process is simple; you only have to enter the words of your seed into your new Trezor device. *(You may also use any other wallets or applications that use the same standard as the Trezor.)*

Important Caveat: You can also password-protect this seed, but it's not done by default. If password protected, the attacks I'm going to describe don't work!

What’s inside the Trezor?

How to Get Recovery Seeds?



Dmitry Nedospasov

Dmitry Nedospasov is a hardware design and security engineer, security researcher, trainer, speaker and reverse-engineerer. In 2014 Dmitry received his PhD (Dr-Ing.) in IC Security at TU Berlin.

Josh Datko

Josh Datko is an embedded systems engineer, security researcher and former submarine officer. Josh is best known for his 2017 presentation on insecurities in cryptocurrency hardware wallets.

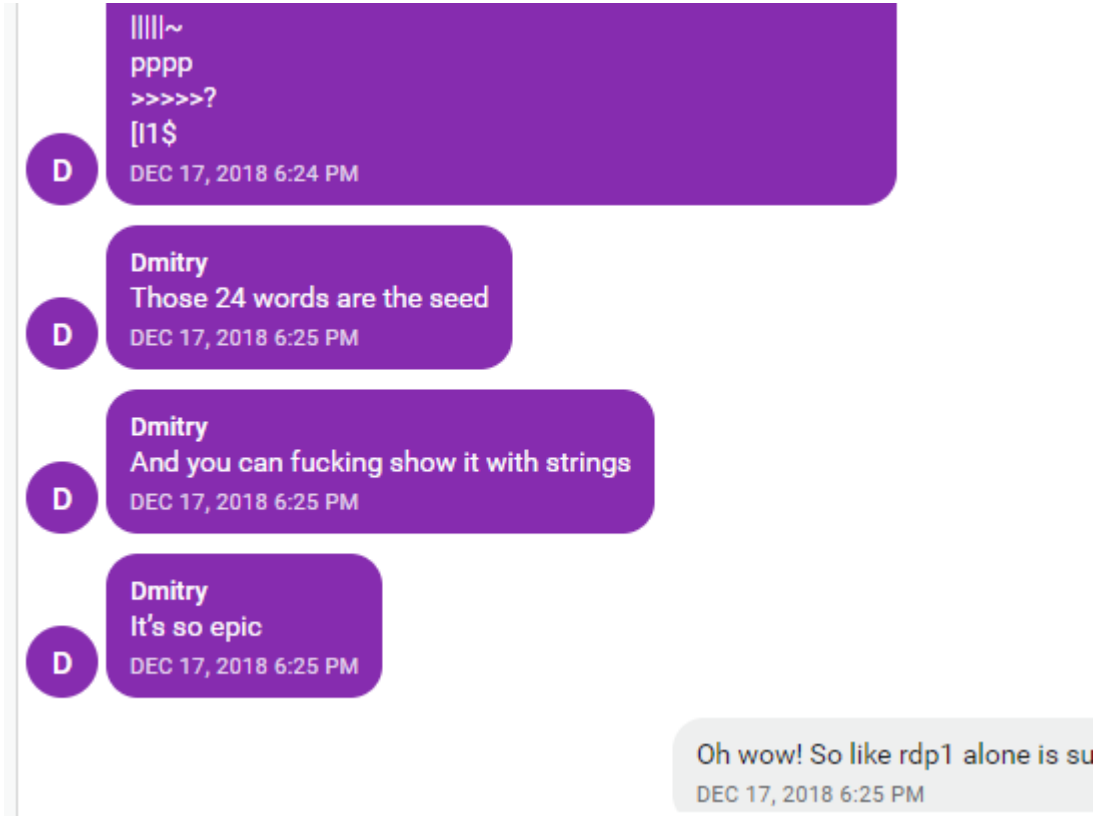
Thomas Roth

Thomas Roth was named as one of the 30 under 30 in Technology by the Forbes Magazine. His main focus is on mobile and embedded systems with published research on topics like TrustZone, payment terminals, and embedded security.

Wallet.Fail

- Vulnerabilities on several Bitcoin Wallets.
- For Trezor specifically:
  - How to copy recovering seed out of a backup stored in SRAM.
  - Required physical access to PCB (open enclosure), but can be performed with high reliability.

My Inspiration – Wallet.Fail Talk



How else to get recovery seed?

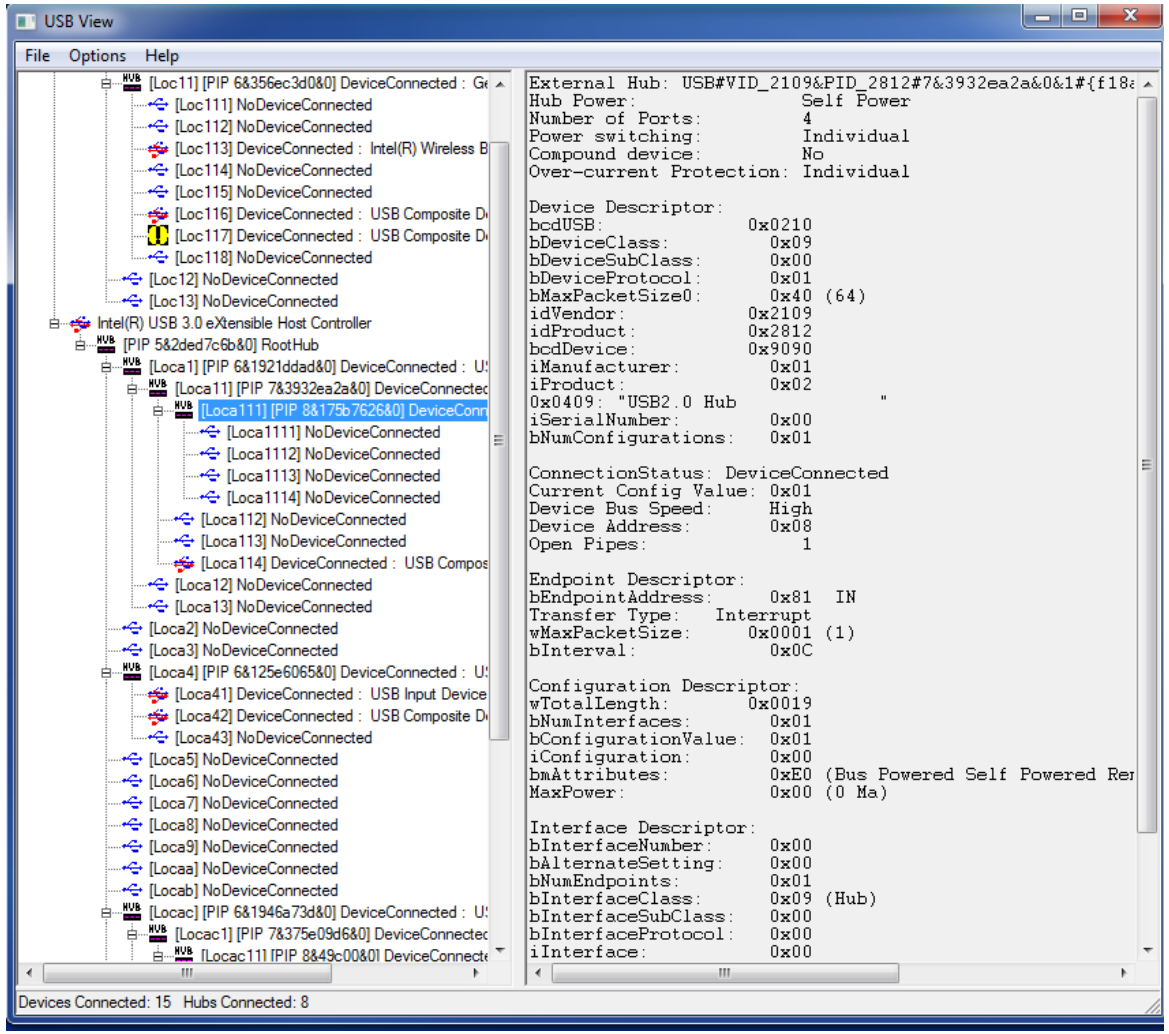
```
flash memory layout:

   name    |          range           |  size   |     function
-----------+--------------------------+---------+------------------
Sector  0  | 0x08000000 - 0x08003FFF  |  16 KiB | bootloader code
Sector  1  | 0x08004000 - 0x08007FFF  |  16 KiB | bootloader code
-----------+--------------------------+---------+------------------
Sector  2  | 0x08008000 - 0x0800BFFF  |  16 KiB | metadata area
Sector  3  | 0x0800C000 - 0x0800FFFF  |  16 KiB | metadata area
-----------+--------------------------+---------+------------------
Sector  4  | 0x08010000 - 0x0801FFFF  |  64 KiB | application code
Sector  5  | 0x08020000 - 0x0803FFFF  | 128 KiB | application code
Sector  6  | 0x08040000 - 0x0805FFFF  | 128 KiB | application code
Sector  7  | 0x08060000 - 0x0807FFFF  | 128 KiB | application code
===========+==========================+==========================
Sector  8  | 0x08080000 - 0x0809FFFF  | 128 KiB | application code
Sector  9  | 0x080A0000 - 0x080BFFFF  | 128 KiB | application code
Sector 10  | 0x080C0000 - 0x080DFFFF  | 128 KiB | application code
Sector 11  | 0x080E0000 - 0x080FFFFF  | 128 KiB | application code
```

What's going on up in here?

Recovery seed, device PIN saved here!

USB Descriptors



wLength → Host Provided Max Request Size

9.4.3 Get Descriptor

- This request returns the current device configuration value.

	bmRequest Type	bRequest	wValue		wIndex		wLength	
bytes	0	1	2	3	4	5	6	7

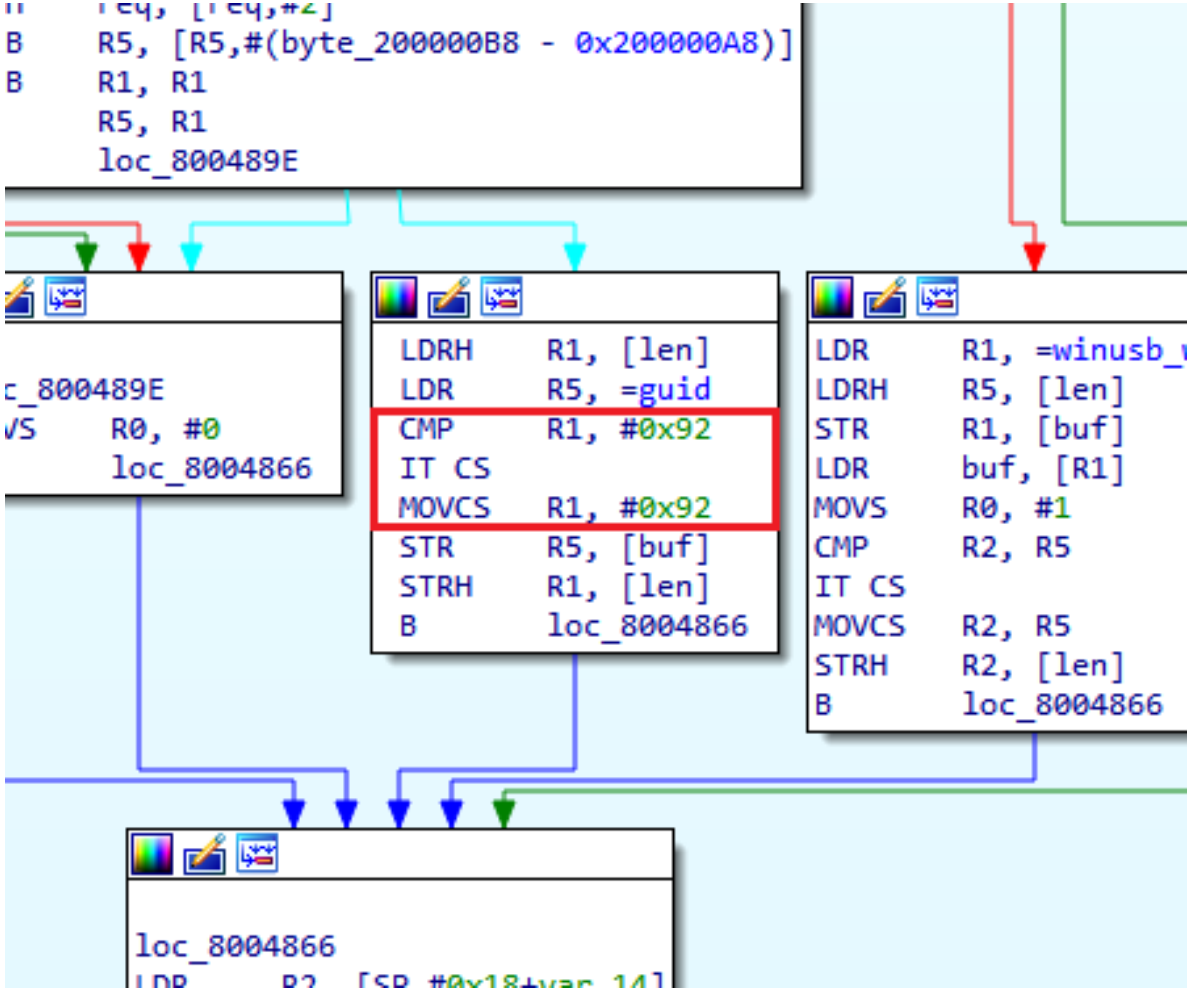
Direction 0x80	GET DESCRIPTOR	Descriptor Type (HI) and Descriptor Index (LO)	Zero or Language ID (9.6.7)	Descriptor Length
1: D-to-H 0: Device	6	High byte: Descriptor Types 1: DEVICE 2: CONFIGURATION 3: STRING 4: INTERFACE 5: ENDPOINT 6: DEVICE_QUALIFIER 7: OTHER_SPEED_CONFIG 8: INTERFACE_POWER Low Byte: Descriptor Index	String Descriptors: Language ID Others: Zero	The number of bytes to return.
Data Descriptor				

Send MIN() of wLength & Struct Length

```
} else if (((req->bmRequestType & USB_REQ_TYPE_RECIPIENT) == USB_REQ_TYPE_INTERFACE) &&
        (req->wIndex == WINUSB_REQ_GET_EXTENDED_PROPERTIES_OS_FEATURE_DESCRIPTOR) &&
        (usb_descriptor_index(req->wValue) == winusb_wcid.functions[0].bInterfaceNumber)) {

        *buf = (uint8_t*)(&guid);
        *len = MIN(*len, guid.header.dwLength);
        status = USBD_REQ_HANDLED;
```

Checking Implementation Details



Open Source FTW :)

- I can be lazy since firmware is fully known & I can modify it even.
- We can ‘simulate’ the glitch to ensure things will work as we expect.

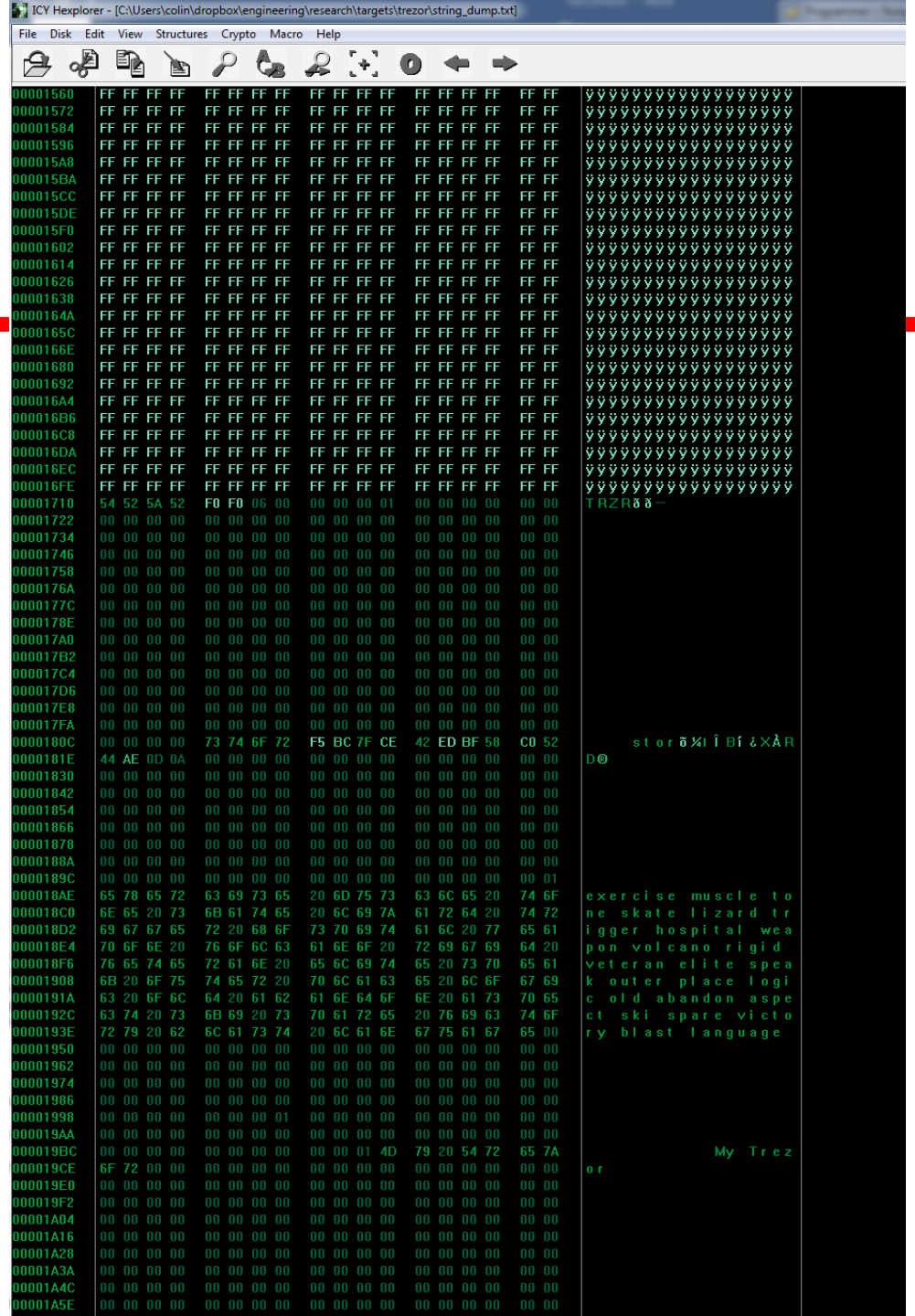
Validating This Will Work (1/2)

Index	m:s.ms.us.ns	Len	Err	Dev	Ep	Record	Summary
0	0:00.000.000.000					Capture started (Aggregate)	[02/06/19 00:45:55]
1	0:00.000.000.000					<Host connected>	
2	0:00.000.633.500					<Full-speed>	
3	0:23.658.183.950	146 B		22	00	Control Transfer	92 00 00 00 00 01 05 00 01 00 8
24	0:06.791.576.583	146 B		22	00	Control Transfer	92 00 00 00 00 01 05 00 01 00 8
45	0:03.879.450.166	146 B		22	00	Control Transfer	92 00 00 00 00 01 05 00 01 00 8
66	1:58.972.722.583	65535 B		22	00	Control Transfer	92 00 00 00 00 01 05 00 01 00 8
4171	0:11.333.695.616					Capture stopped	[02/06/19 00:48:40]

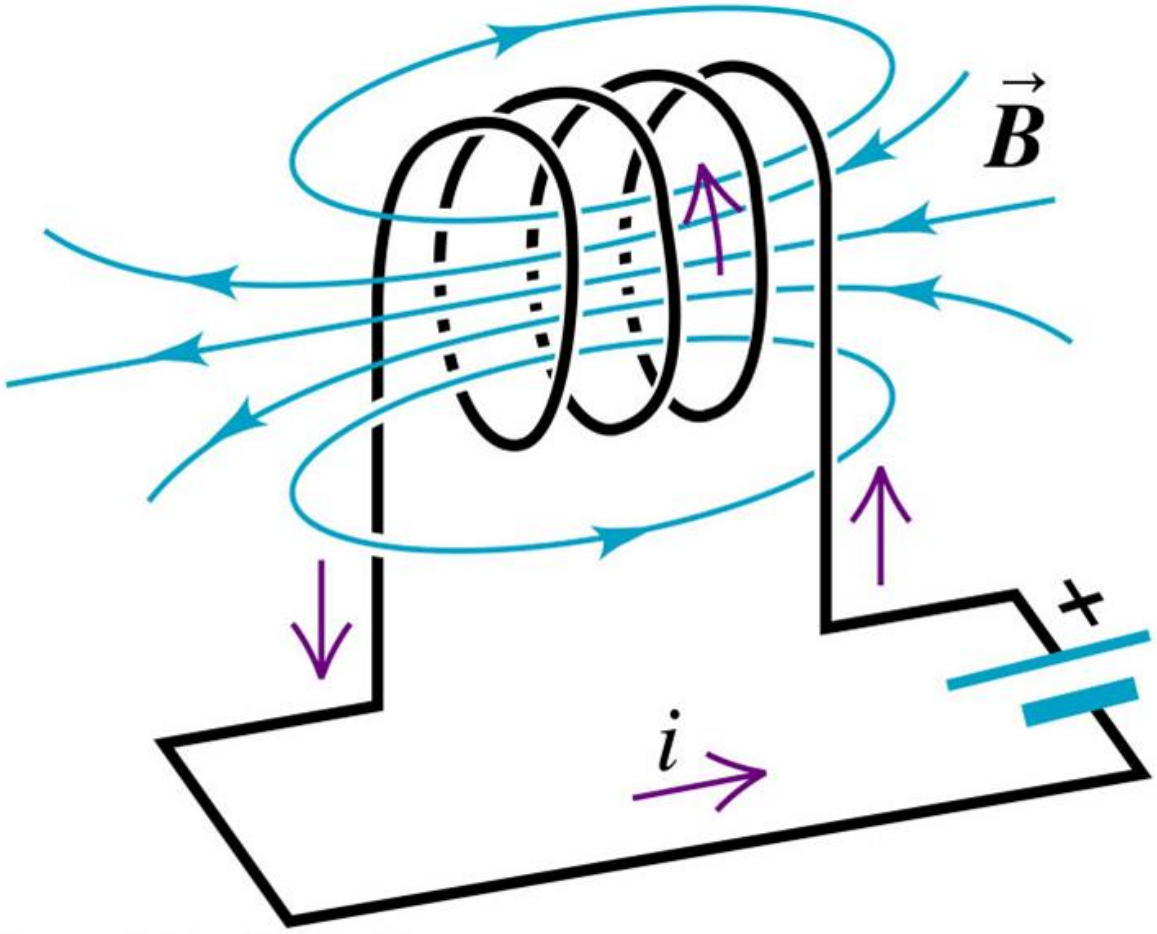
Expected response (146 bytes)

Use debugger to skip MIN() check.

Validating This Will Work (2/2)

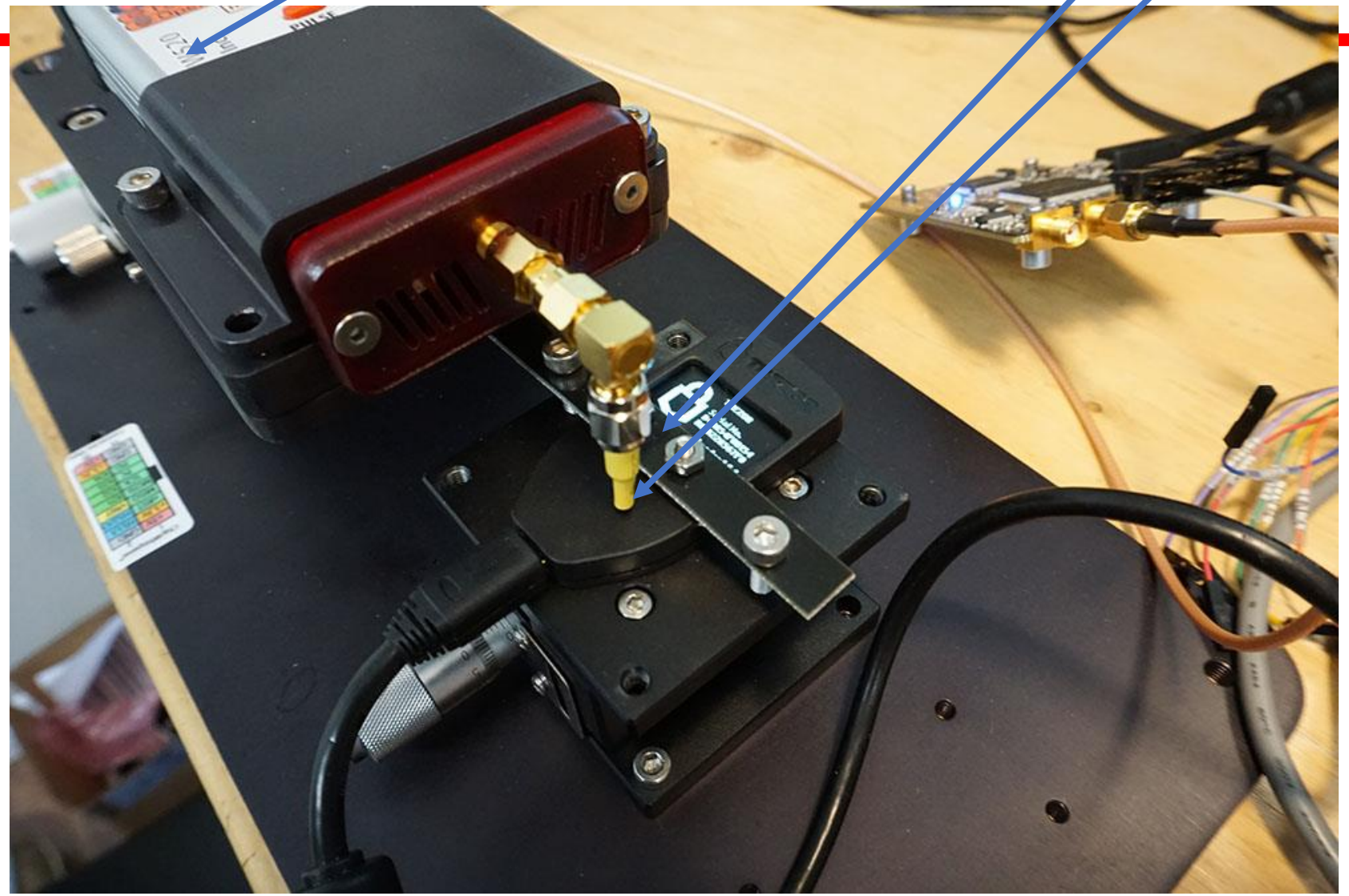


Generated and Induced Magnetic Field



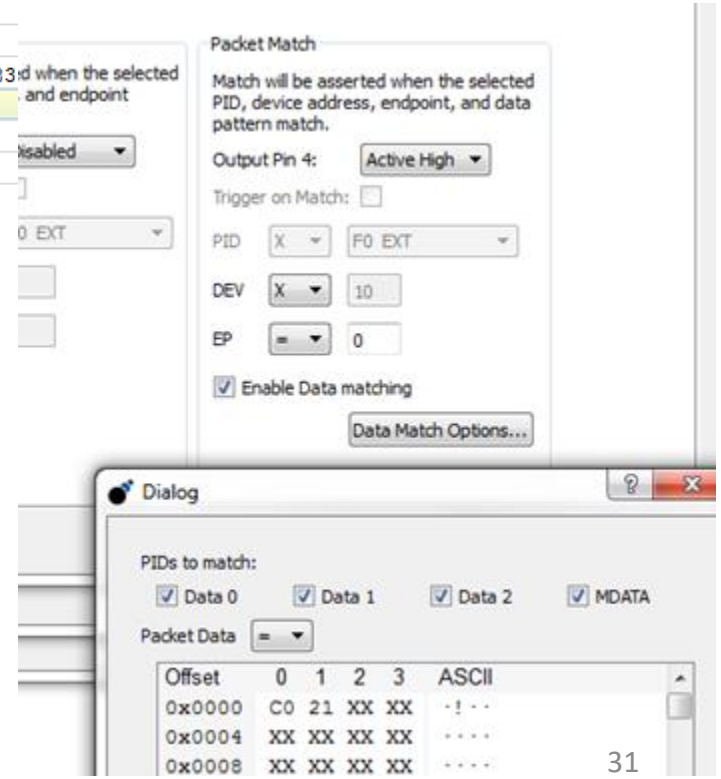
Example of Coils

EMFI Setup

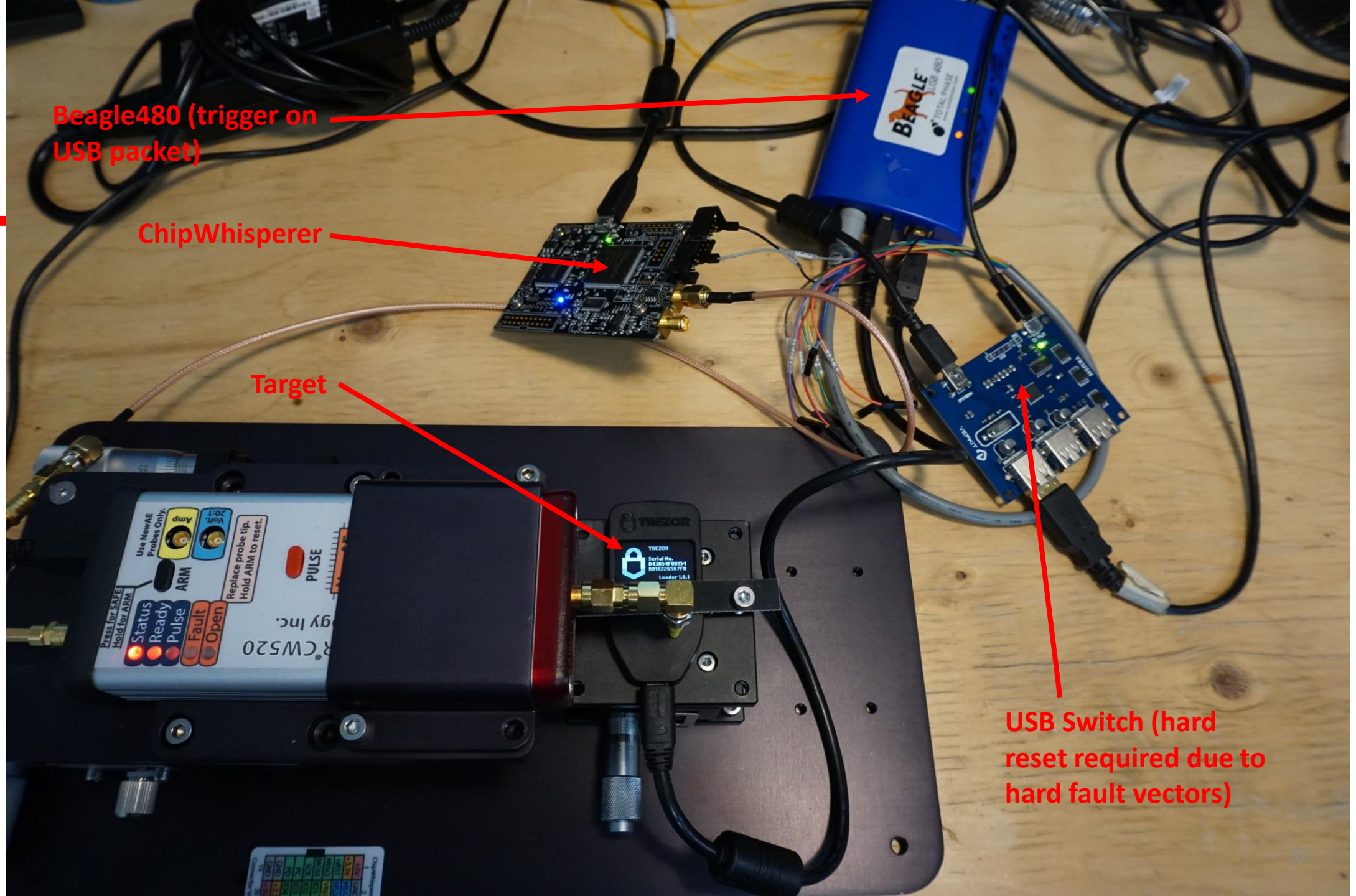


Triggering EMFI

FS	6546642	0:00.056.668.166	146 B		28	00	Control Transfer	92 00 00 00 00 01 05 00 01 00 88 00 00 00 07 00 00 00 2A 00 44 00 65 00…
FS	6546643	0:00.000.000.000	8 B		28	00	SETUP txn	C1 21 00 00 05 00 FF 1A
FS	6546647	0:00.000.025.333	64 B		28	00	IN txn	92 00 00 00 00 01 05 00 01 00 88 00 00 00 07 00 00 00 2A 00 44 00 65 00…
FS	6546651	0:00.000.070.750	64 B		28	00	IN txn	00 00 7B 00 30 00 32 00 36 00 33 00 62 00 35 00 31 00 32 00 2D 00 38 00…
FS	6546655	0:00.000.071.083	18 B		28	00	IN txn	39 00 64 00 38 00 65 00 66 00 35 00 7D 00 00 00 00 00
FS	6546659	0:00.000.026.333	0 B		28	00	OUT txn	
FS	6546663	0:00.025.202.500	8 B	T	28	00	SETUP txn	C1 21 00 00 05 00 FF 1A
FS	6546664	0:00.000.000.000	3 B		28	00	SETUP packet	2D 1C B8
FS	6546665	0:00.000.003.416	11 B		28	00	DATA0 packet	C3 C1 21 00 00 05 00 FF 1A 83
FS	6546666	0:00.000.008.666	1 B		28	00	ACK packet	D2
FS	6546667	0:00.000.013.166	1.99 s		28	00	[41215 IN-NAK]	[Periodic Timeout]
FS	6546668	0:02.000.005.333	1.99 s		28	00	[41201 IN-NAK]	[Periodic Timeout]

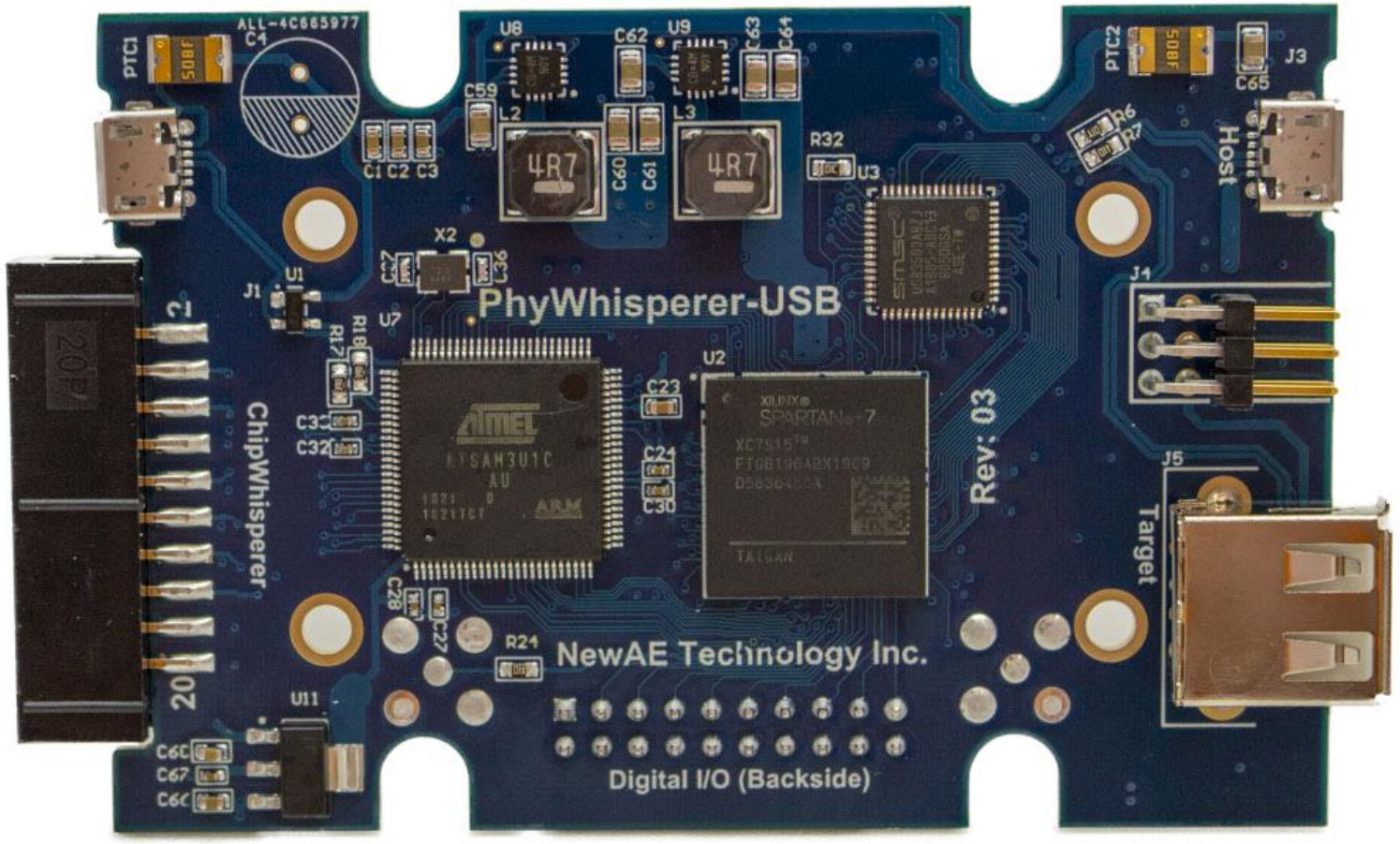


- TotalPhase Beagle 480
- Hardware trigger on USB physical-layer packets!


Beagle480 (trigger on USB packet)
ChipWhisperer
Target
USB Switch (hard reset required due to hard fault vectors)

PhyWhisperer-USB

- Cheapish ($250 USD) triggering device for USB physical-layer packets.
- Works as a sniffer too.
- Open-source HW/SW.
- Needs separate fault injection driver - ChipSHOUTER (EMFI), ChipWhisperer (Voltage glitching), or Mux (Voltage glitching)



https://github.com/newaetech/phywhispererusb

PhyWhisperer-USB

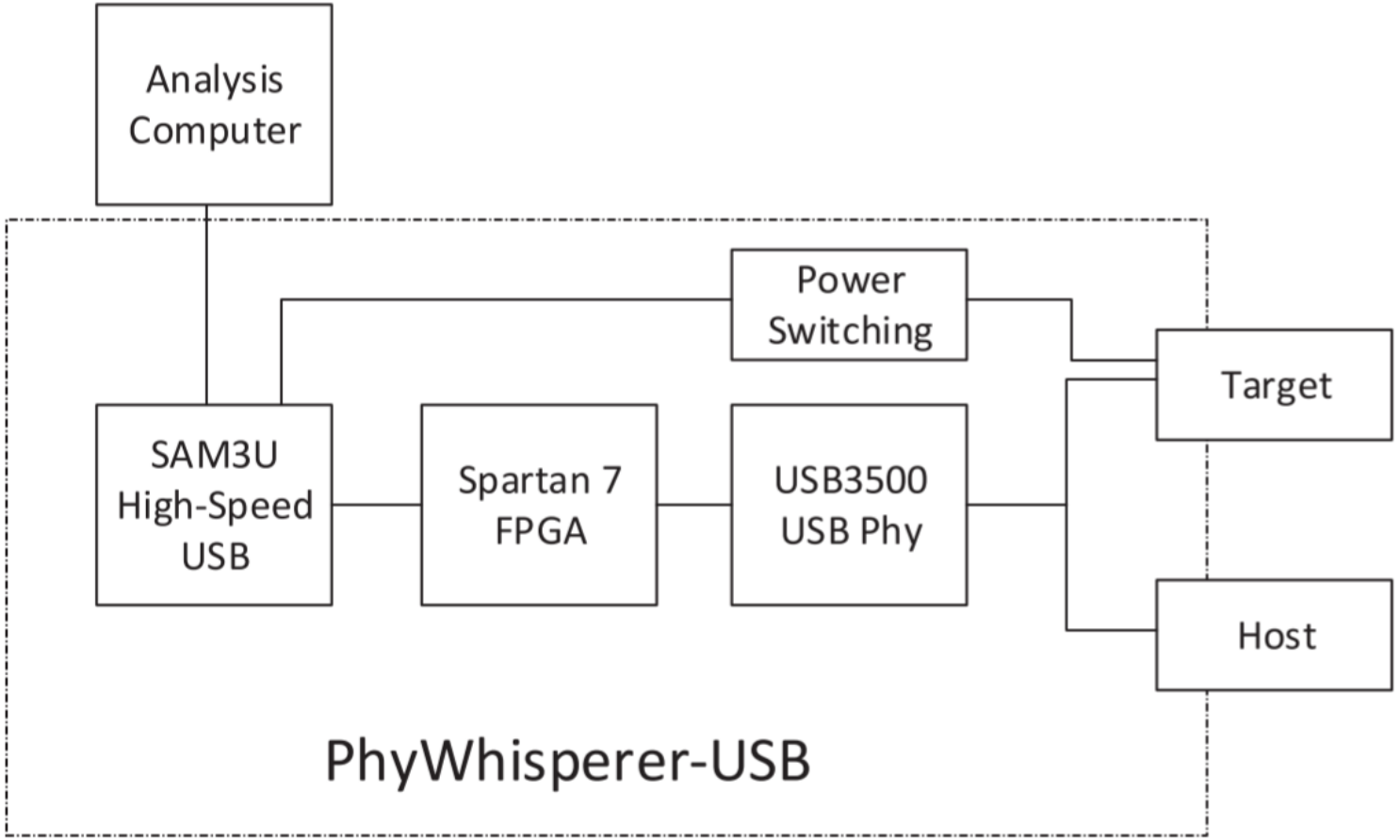
- USB 2.0 LS/FS/HS Phy
- Can switch power on/off to target (critical for glitching).
- Interpose with a real (external) USB host.

Stuff that works also:

- Sniffing USB packets (Wireshark as front-end).

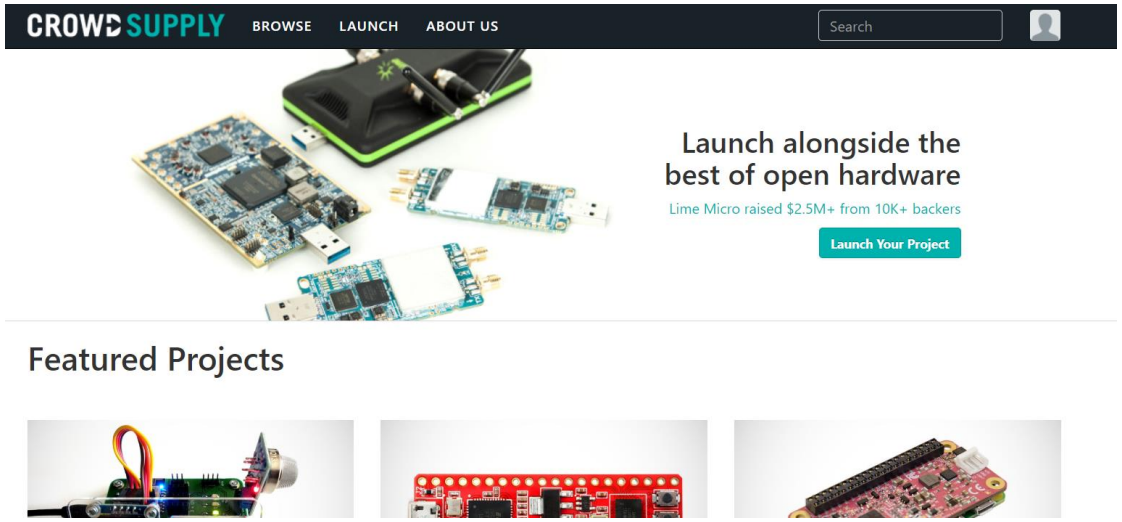
Stuff you *could* do (but I'm way too lazy for):

- Generate USB packets.



- ~~A true history of block chain!~~
- Me gambling with 0.3 BTC!
- Physical give-aways!
- ~~Attacking bitcoin wallets & stealing funds!~~
- Stealing authentication credentials!
- Crass commercialization!
- ~~**Open-Source Tool Release!**~~
- Fixing your crappy code!

PhyWhisperer on CrowdSupply



- Everyone loves OSHW.
- Nobody wants to actually build it.
- Solution: Search “PhyWhisperer” to see crowd funding campaign!

- ~~A true history of block chain!~~
- Me gambling with 0.3 BTC!
- Physical give-aways!
- ~~Attacking bitcoin wallets & stealing funds!~~
- Stealing authentication credentials!
- ~~**Crass commercialization!**~~
- ~~Open-Source Tool Release!~~
- Fixing your crappy code!

EMFI Demo

<Hopefully my camera works>

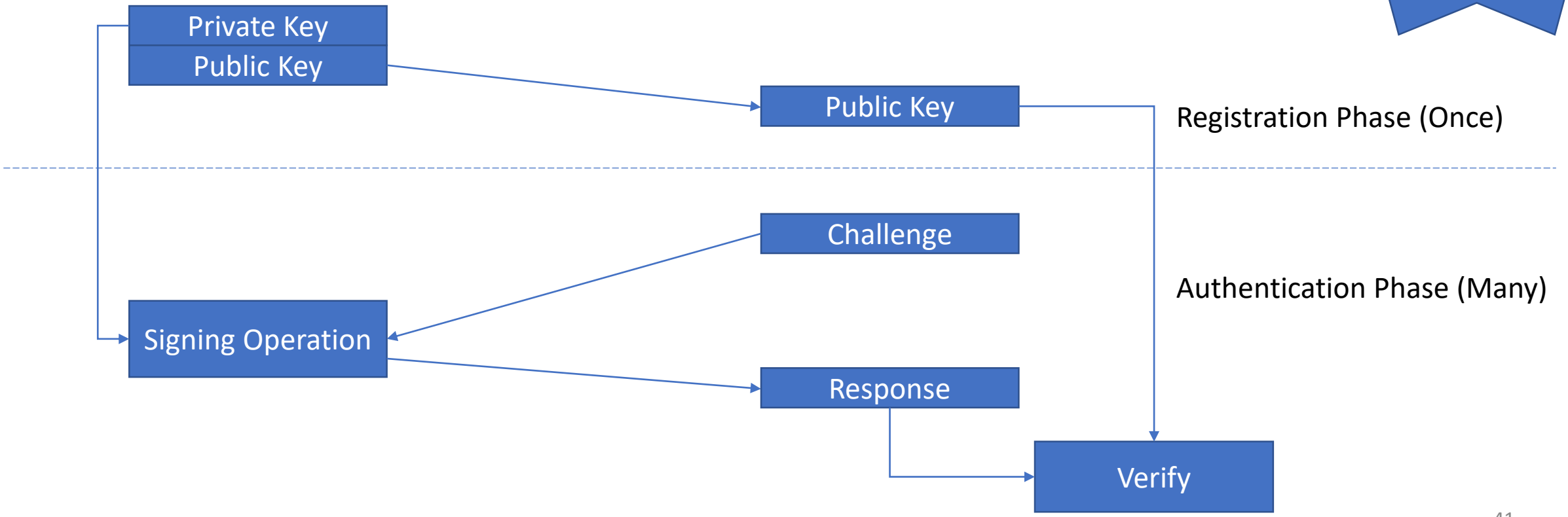
- ~~A true history of block chain!~~
- ~~**Me gambling with 0.3 BTC!**~~
- Physical give-aways!
- ~~Attacking bitcoin wallets & stealing funds!~~
- Stealing authentication credentials!
- ~~Crass commercialization!~~
- ~~Open-Source Tool Release!~~
- Fixing your crappy code!

Wait – What About Authentication Tokens?

- Since submitting & preparing this I realized several other interesting targets, and rather than a second talk I give you…

How do these things work?

BONUS CONTENT



Who’s using these tokens?

BONUS CONTENT

Microsoft

Password-less protection

Reduce your risk exposure with password alternatives

Example: Solo Keys Authentication Token

BONUS CONTENT

Address	Variable Name	Description
0x200000bc	USBD_HID_Desc	HID Descriptor.
0x20001b0c	_signing_key	Pointer to key.
0x20001b10	master_secret	HMAC secret.
0x20001b50	privkey.8369	ECC Private Key.
0x20001b70	sha256_ctx	SHA256 context.

```
else if( (req->wValue >> 8) ==
            HID_DESCRIPTOR_TYPE){
    pbuf = USBD_HID_Desc;
    len = MIN(USB_HID_DESC_SIZ, req->wLength);
}
```

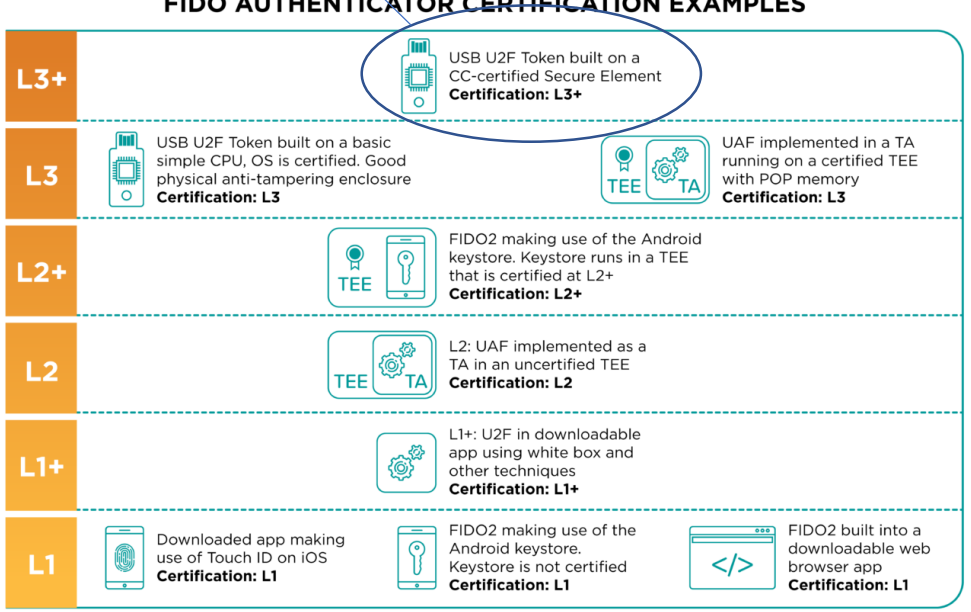
TIP: *You can find ECC private keys in memory if you have public key to compare with. If memory layout is unknown, we can register a new service (to get public key from device), perform attack, and figure out if any private keys in our memory dump.*

What about other FIDO2 Keys?

BONUS CONTENT

e.g., Yubikey 5 *should* reach this

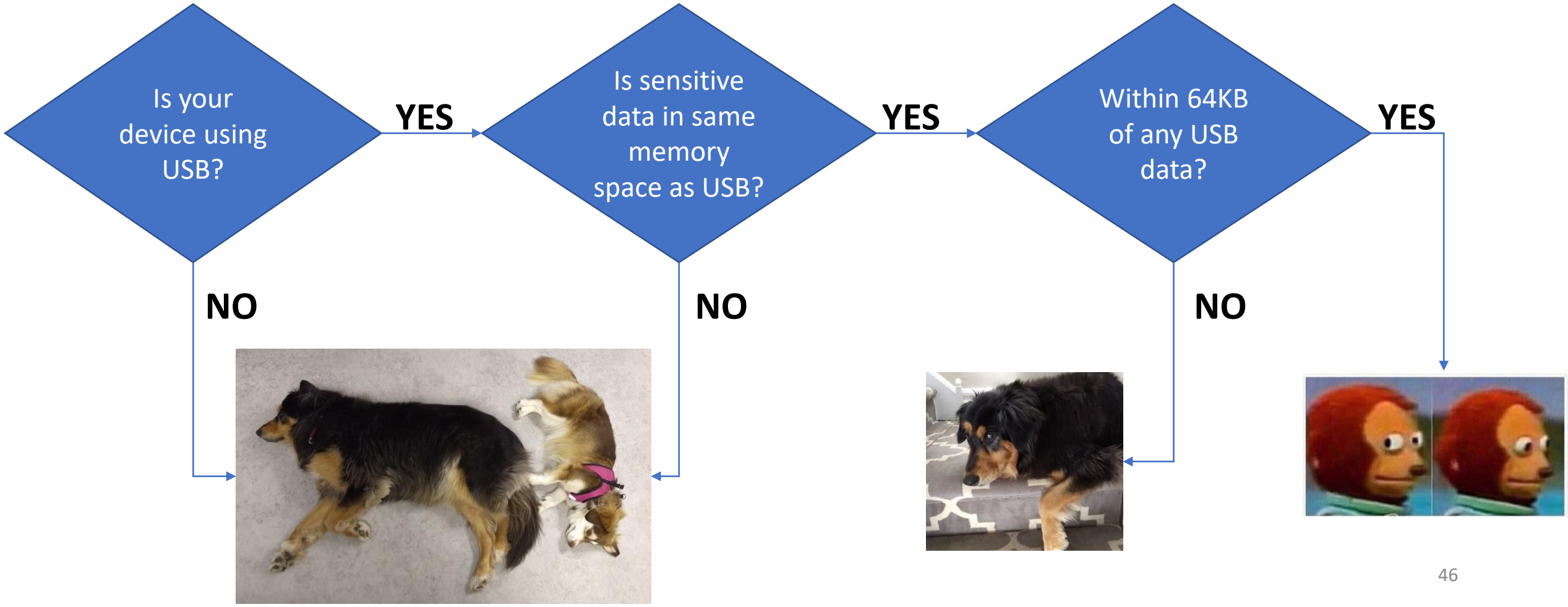
SAMPLE DEVICE HARDWARE & SOFTWARE REQUIREMENTS		DEFENDS AGAINST
Protection against chip fault injection and invasive attacks	L3+	Chip level attacks on captured devices
Circuit board potting, package on package memory, encrypted RAM...	L3	Circuit board attacks on captured devices
Device must support allowed Restricted Operating Environment (ROE) (e.g., TEE, Secure Element...), or intrinsically be an ROE (e.g., a USB token or Smart Card...)	L2+	Device OS compromise
	L2	
Any device HW or SW	L1+	White Box Cryptography to defend against OS compromise
	L1	Phishing, server credential breaches and MiTM attacks (better than passwords)



https://fidoalliance.org/certification/authenticator-certification-levels/

- ~~A true history of block chain!~~
- ~~Me gambling with 0.3 BTC!~~
- Physical give-aways!
- ~~Attacking bitcoin wallets & stealing funds!~~
- ~~**Stealing authentication credentials!**~~
- ~~Crass commercialization!~~
- ~~Open-Source Tool Release!~~
- Fixing your crappy code!

How Concerned Should You Be?



How to Fix It?

- Why can you send back 64K of memory? No descriptors are that big!
- Devices have memory protection.
  - We can armour the sensitive data with invalid memory segments, or “disable” memory segments when read-out shouldn’t be needed.
- Move memory layout around so we can’t read into sensitive data.
  - Less useful as another fault might let us corrupt a pointer instead.
- Encrypt data in-place.
  - Don’t allow a “dumb” dump to figure out this critical data!

Trezor Fixes

- **The disclosed problem has been fixed in latest firmware patch.**

- **The disclosed problem did not affect people using passphrases.**

https://blog.trezor.io/details-of-security-updates-for-trezor-one-firmware-1-8-0-and-trezor-model-t-firmware-2-1-0-408e59dc012

Solo Key Fixes

- **The disclosed problem has been fixed in GIT!**

- ~~A true history of block chain!~~
- ~~Me gambling with 0.3 BTC!~~
- Physical give-aways!
- ~~Attacking bitcoin wallets & stealing funds!~~
- ~~Stealing authentication credentials!~~
- ~~Crass commercialization!~~
- ~~Open-Source Tool Release!~~
- ~~**Fixing your crappy code!**~~

Takeaways

- Tamper-resistant enclosures aren't enough when discussing "near-physical" attacks. Commercially available EMFI tools exist (see - ChipSHOUTER) and can be purchased sometimes...
- If implementing a USB device, validate your response size makes sense to avoid many attacks!
- Testing against EMFI is useful to understand vulnerability – but you can do some testing with simulation/emulation and through code review.

Let's Do It!

- I've got PhyWhisperer-USB PCBs here for you ☺ Join the CrowdSupply and make the full thing happen!

- See my WOOT'19 paper (linked from **oflynn.com**) & White Paper!

Blog	oflynn.com
Twitter	@colinoflynn
Email	colin@oflynn.com
Dal Email	coflynn@dal.ca
Company	newae.com

I'm terrible on response times right now, please don't take it personally if it takes weeks++ :/

For More, See:

- WOOT 2019 Paper: https://www.usenix.org/conference/woot19/presentation/oflynn
- My blog post: http://colinoflynn.com/2019/03/glitching-trezor-using-emfi-through-the-enclosure/
- PhyWhisperer-USB repo: https://github.com/newaetech/phywhispererusb