VUSec

riscure

black hat USA 2018

# There Will Be Glitches:

## Extracting and Analyzing Automotive Firmware Efficiently

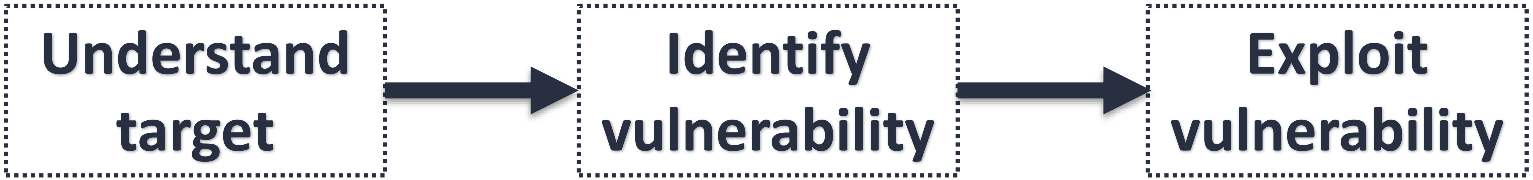
**Alyssa Milburn & Niek Timmers**

@noopwafel @tieknimmers

Santiago Cordoba,
Nils Wiersma,
Ramiro Pareja

# Today we are talking about

# Standard approach for breaking embedded systems

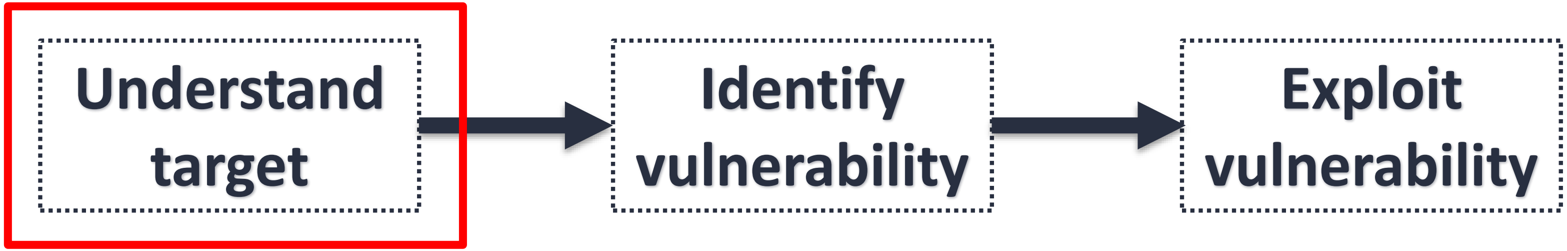


Understand target

Identify vulnerability

Exploit vulnerability

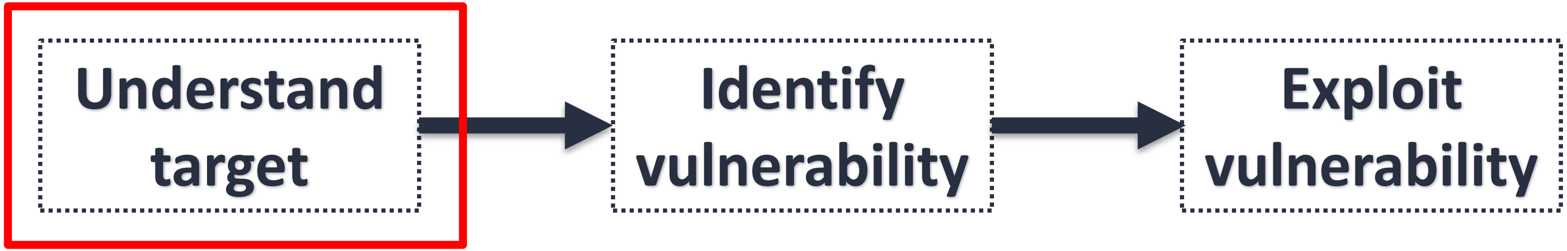
# Standard approach for breaking embedded systems



**Understand target** → **Identify vulnerability** → **Exploit vulnerability**

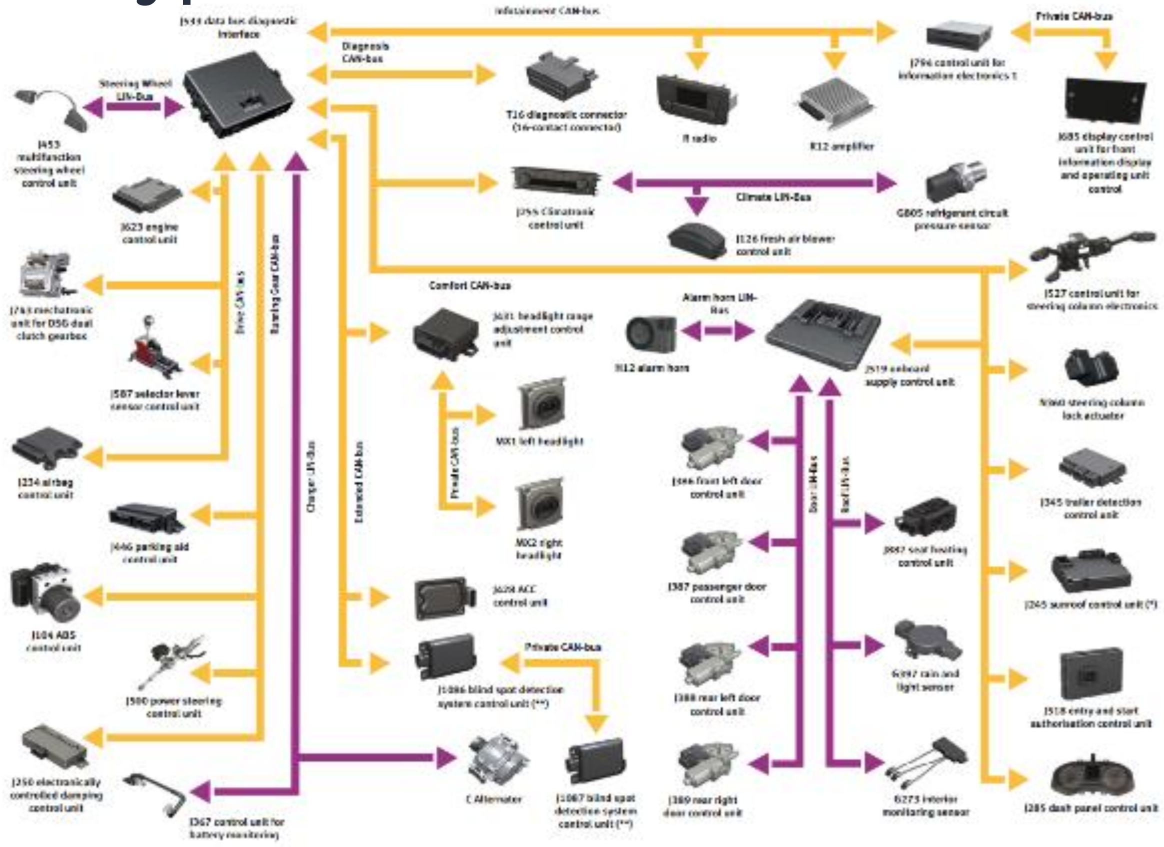
But to understand, we need the **firmware**!

# Standard approach for breaking ~~embedded systems~~ ECUs found in cars!

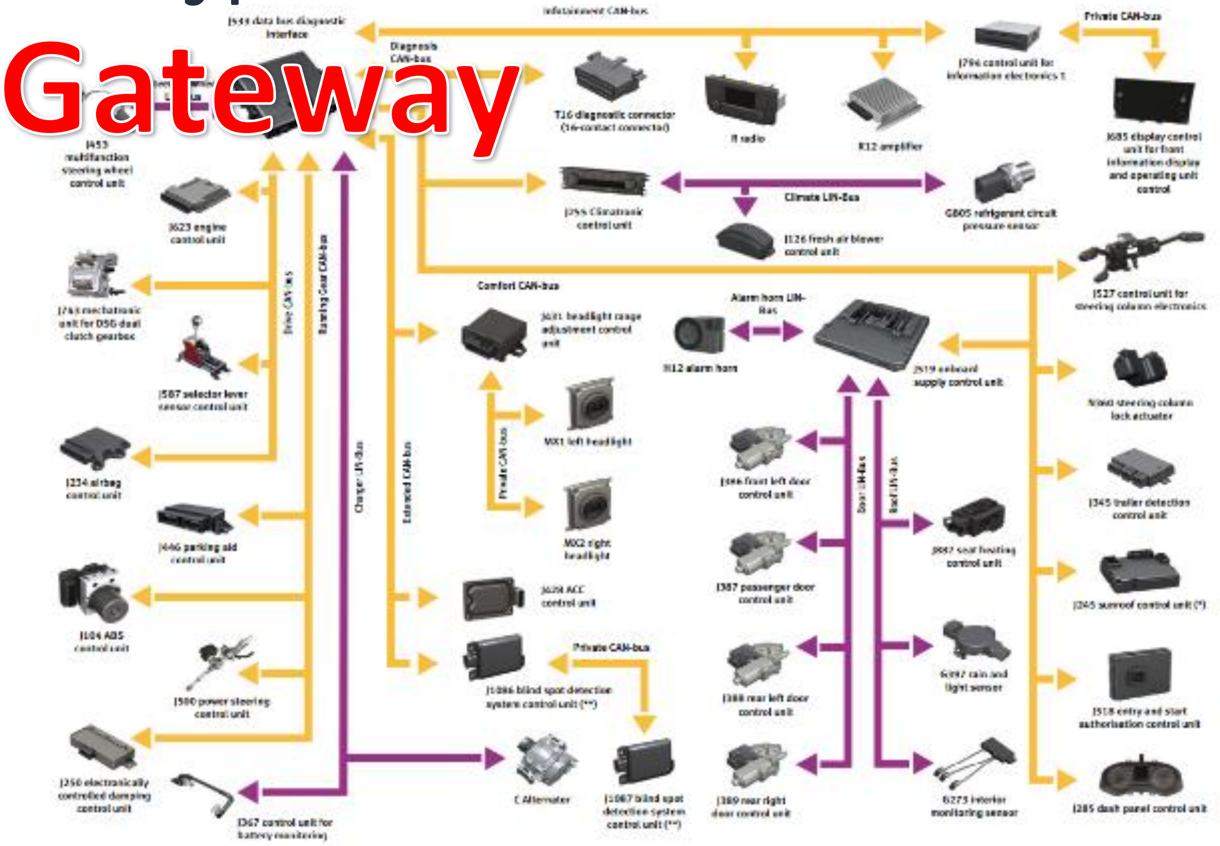


But to understand, we need the **firmware**!

# Typical ECUs found in a car...


J533 data bus diagnostic interface
Infotainment CAN-bus
Diagnosis CAN-bus
Steering Wheel LIN-Bus
J453 multifunction steering wheel control unit
T16 diagnostic connector (16-contact connector)
R radio
R12 amplifier
J794 control unit for information electronics 1
Private CAN-bus
J685 display control unit for front information display and operating unit control
J623 engine control unit
J255 Climatronic control unit
Climate LIN-Bus
G805 refrigerant circuit pressure sensor
J126 fresh air blower control unit
J743 mechatronic unit for DSG dual clutch gearbox
Drive CAN-bus
Running Gear CAN-bus
J527 control unit for steering column electronics
Comfort CAN-bus
J431 headlight range adjustment control unit
Alarm horn LIN-Bus
H12 alarm horn
J519 onboard supply control unit
J587 selector lever sensor control unit
N360 steering column lock actuator
MX1 left headlight
Private CAN-bus
J234 airbag control unit
Charger LIN-bus
Extended CAN-bus
J386 front left door control unit
Door LIN-Bus
Roof LIN-Bus
J345 trailer detection control unit
MX2 right headlight
J446 parking aid control unit
J882 seat heating control unit
J428 ACC control unit
J387 passenger door control unit
J245 sunroof control unit (*)
J104 ABS control unit
Private CAN-bus
G397 rain and light sensor
J500 power steering control unit
J1086 blind spot detection system control unit (**)
J388 rear left door control unit
J518 entry and start authorisation control unit
J250 electronically controlled damping control unit
C Alternator
J1087 blind spot detection system control unit (**)
J389 rear right door control unit
G273 interior monitoring sensor
J285 dash panel control unit
J367 control unit for battery monitoring

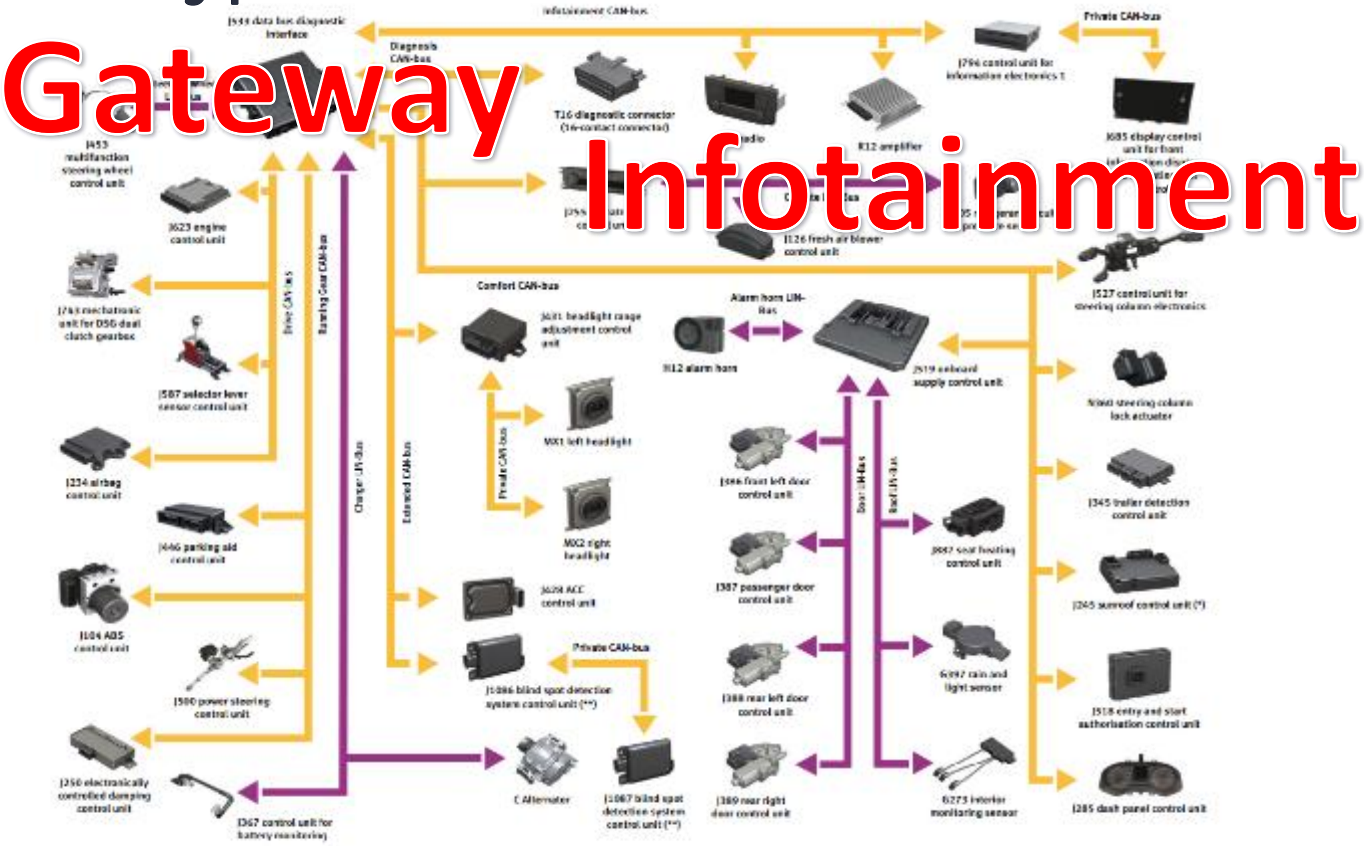
# Typical ECUs found in a car…



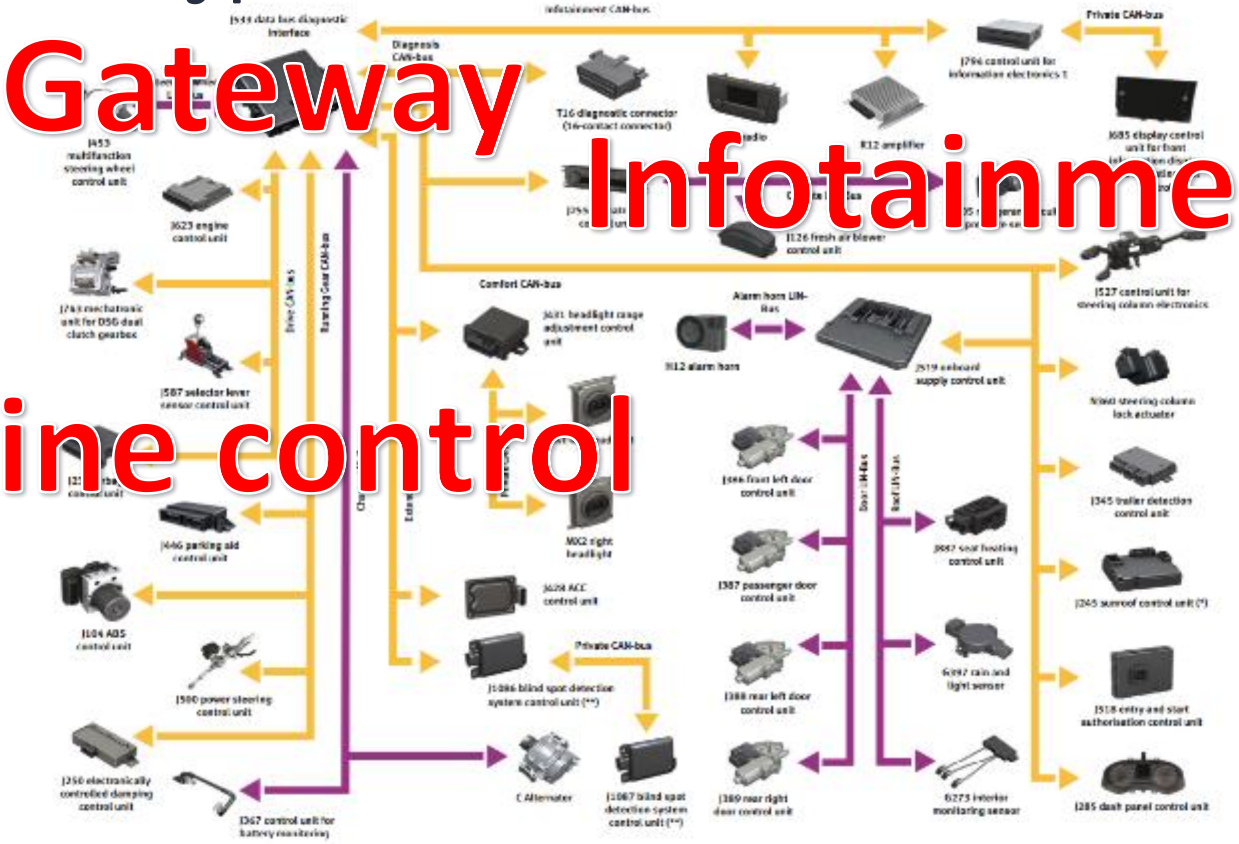
Gateway

J533 data bus diagnostic interface
Infotainment CAN-bus
Diagnosis CAN-bus
Private CAN-bus
J794 control unit for information electronics 1
T16 diagnostic connector (16-contact connector)
R radio
R12 amplifier
J685 display control unit for front information display and operating unit control
J453 multifunction steering wheel control unit
J255 Climatronic control unit
Climate LIN-Bus
G805 refrigerant circuit pressure sensor
J126 fresh air blower control unit
J623 engine control unit
J743 mechatronic unit for DSG dual clutch gearbox
J587 selector lever sensor control unit
Drive CAN-bus
Running Gear CAN-bus
J527 control unit for steering column electronics
Comfort CAN-bus
J431 headlight range adjustment control unit
Alarm horn LIN-Bus
H12 alarm horn
J519 onboard supply control unit
N360 steering column lock actuator
MX1 left headlight
MX2 right headlight
Private CAN-bus
J234 airbag control unit
J446 parking aid control unit
J104 ABS control unit
J500 power steering control unit
J250 electronically controlled damping control unit
J367 control unit for battery monitoring
Charger LIN-bus
Extended CAN-bus
J428 ACC control unit
J1086 blind spot detection system control unit (**)
C Alternator
J1087 blind spot detection system control unit (**)
J386 front left door control unit
J387 passenger door control unit
J388 rear left door control unit
J389 rear right door control unit
Door LIN-Bus
Roof LIN-Bus
J882 seat heating control unit
G397 rain and light sensor
G273 interior monitoring sensor
J345 trailer detection control unit
J245 sunroof control unit (*)
J518 entry and start authorisation control unit
J285 dash panel control unit

# Typical ECUs found in a car…


Gateway
Infotainment
J533 data bus diagnostic interface
Infotainment CAN-bus
Diagnosis CAN-bus
T16 diagnostic connector (16-contact connector)
Radio
R12 amplifier
J794 control unit for information electronics 1
Private CAN-bus
J685 display control unit for front information display
J453 multifunction steering wheel control unit
J623 engine control unit
J126 fresh air blower control unit
J527 control unit for steering column electronics
J743 mechatronic unit for DSG dual clutch gearbox
J587 selector lever sensor control unit
Drive CAN-bus
Running Gear CAN-bus
Comfort CAN-bus
J431 headlight range adjustment control unit
Alarm horn LIN-Bus
H12 alarm horn
J519 onboard supply control unit
N360 steering column lock actuator
MX1 left headlight
Private CAN-bus
J234 airbag control unit
Charger LIN-bus
Extended CAN-bus
J386 front left door control unit
Door LIN-Bus
Roof LIN-bus
J345 trailer detection control unit
J446 parking aid control unit
MX2 right headlight
J882 seat heating control unit
J387 passenger door control unit
J245 sunroof control unit (*)
J104 ABS control unit
J428 ACC control unit
Private CAN-bus
J1086 blind spot detection system control unit (**)
G397 rain and light sensor
J388 rear left door control unit
J518 entry and start authorisation control unit
J500 power steering control unit
J250 electronically controlled damping control unit
J367 control unit for battery monitoring
C Alternator
J1087 blind spot detection system control unit (**)
J389 rear right door control unit
G273 interior monitoring sensor
J285 dash panel control unit

# Typical ECUs found in a car…

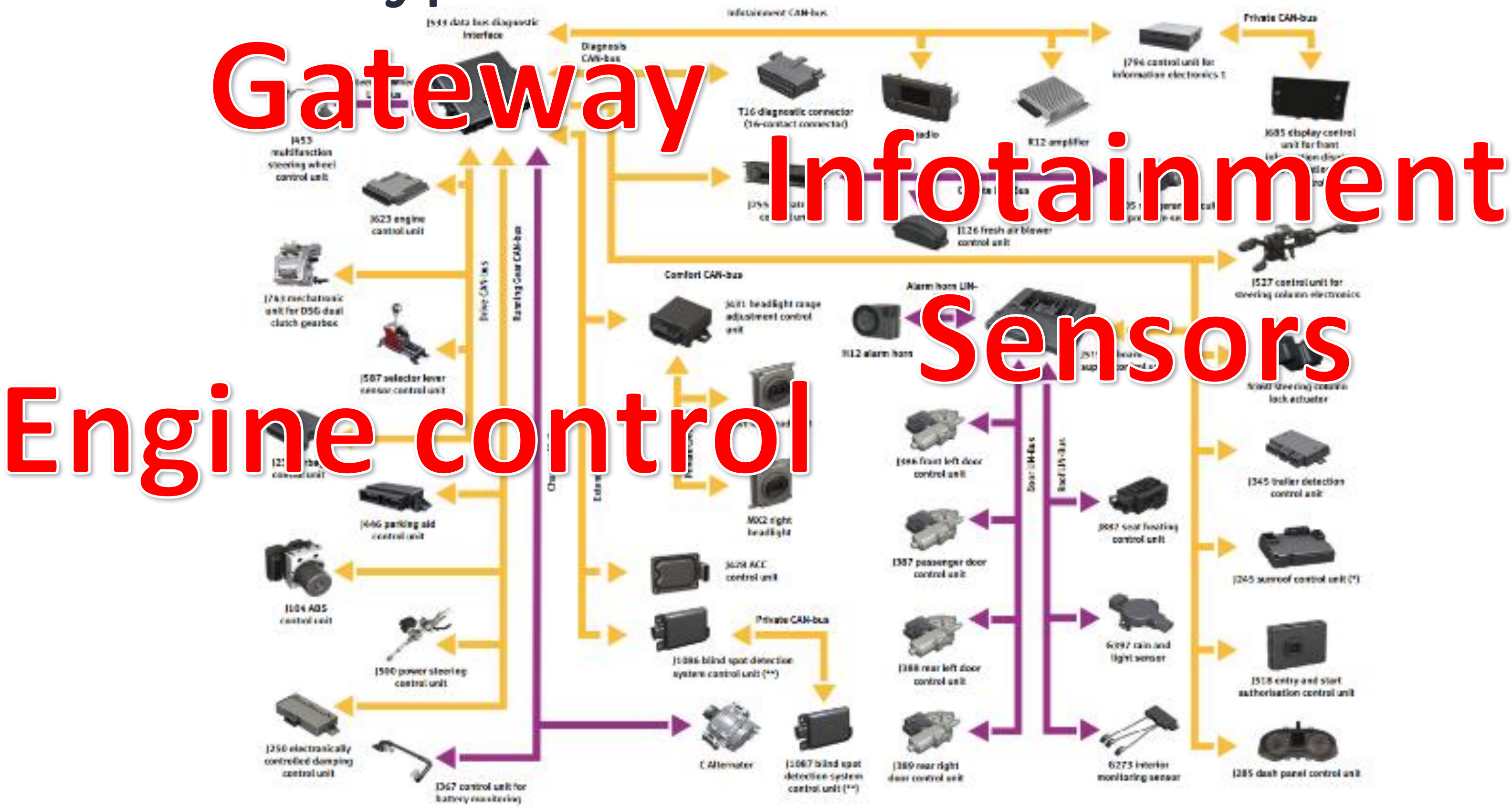

J533 data bus diagnostic interface
Infotainment CAN-bus
Private CAN-bus
Diagnosis CAN-bus
T16 diagnostic connector (16-contact connector)
radio
R12 amplifier
J794 control unit for information electronics 1
J685 display control unit for front information display
J453 multifunction steering wheel control unit
J623 engine control unit
J126 fresh air blower control unit
J527 control unit for steering column electronics
J743 mechatronic unit for DSG dual clutch gearbox
Drive CAN-bus
Running Gear CAN-bus
Comfort CAN-bus
J431 headlight range adjustment control unit
Alarm horn LIN-Bus
H12 alarm horn
J519 onboard supply control unit
J587 selector lever sensor control unit
N360 steering column lock actuator
J386 front left door control unit
Door LIN-Bus
Roof LIN-Bus
J345 trailer detection control unit
J446 parking aid control unit
MX2 right headlight
J387 passenger door control unit
J882 seat heating control unit
J245 sunroof control unit (*)
J104 ABS control unit
J428 ACC control unit
Private CAN-bus
J1086 blind spot detection system control unit (**)
J388 rear left door control unit
G397 rain and light sensor
J518 entry and start authorisation control unit
J500 power steering control unit
J250 electronically controlled damping control unit
J367 control unit for battery monitoring
C Alternator
J1087 blind spot detection system control unit (**)
J389 rear right door control unit
G273 interior monitoring sensor
J285 dash panel control unit

Gateway

Infotainment

Engine control

# Typical ECUs found in a car…



Gateway

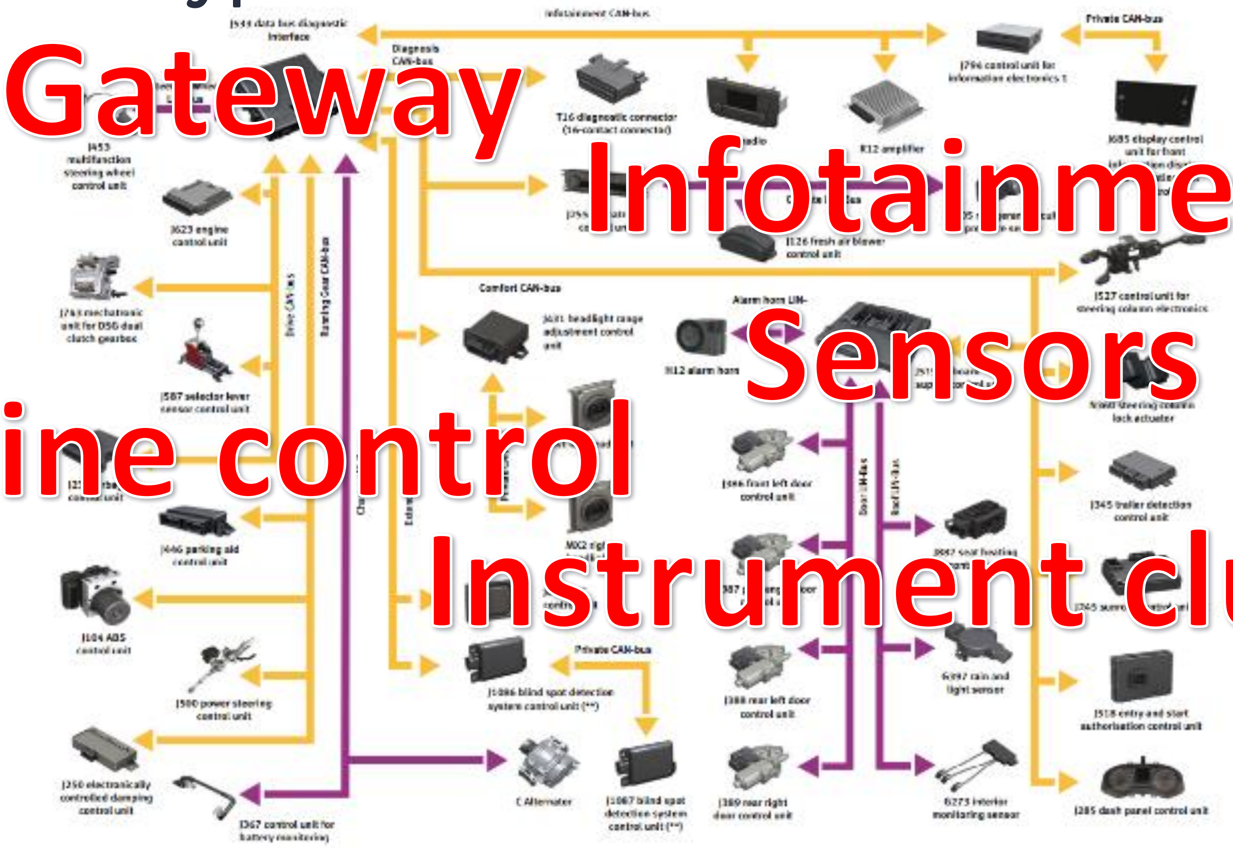
Infotainment

Sensors

Engine control

J533 data bus diagnostic interface
Infotainment CAN-bus
Diagnosis CAN-bus
T16 diagnostic connector (16-contact connector)
Radio
R12 amplifier
J794 control unit for information electronics 1
Private CAN-bus
J685 display control unit for front information display
J453 multifunction steering wheel control unit
J623 engine control unit
J126 fresh air blower control unit
J527 control unit for steering column electronics
J743 mechatronic unit for DSG dual clutch gearbox
Drive CAN-bus
Running Gear CAN-bus
Comfort CAN-bus
J431 headlight range adjustment control unit
Alarm horn LIN-
H12 alarm horn
J587 selector lever sensor control unit
N360 steering column lock actuator
MX2 right headlight
J446 parking aid control unit
J104 ABS control unit
J428 ACC control unit
J1086 blind spot detection system control unit (**)
Private CAN-bus
J500 power steering control unit
J250 electronically controlled damping control unit
J367 control unit for battery monitoring
C Alternator
J1087 blind spot detection system control unit (**)
J386 front left door control unit
J387 passenger door control unit
J388 rear left door control unit
J389 rear right door control unit
Door LIN-bus
Roof LIN-bus
J887 seat heating control unit
G397 rain and light sensor
G273 interior monitoring sensor
J345 trailer detection control unit
J245 sunroof control unit (*)
J518 entry and start authorisation control unit
J285 dash panel control unit

# Typical ECUs found in a car…


J533 data bus diagnostic interface
Infotainment CAN-bus
Diagnosis CAN-bus
Private CAN-bus
T16 diagnostic connector (16-contact connector)
J794 control unit for information electronics 1
R12 amplifier
J685 display control unit for front
J453 multifunction steering wheel control unit
J623 engine control unit
J126 fresh air blower control unit
J527 control unit for steering column electronics
Drive CAN-bus
Running Gear CAN-bus
Comfort CAN-bus
J431 headlight range adjustment control unit
Alarm horn LIN-
H12 alarm horn
J743 mechatronic unit for DSG dual clutch gearbox
J587 selector lever sensor control unit
N360 steering column lock actuator
J386 front left door control unit
J345 trailer detection control unit
J446 parking aid control unit
MX2 right
J104 ABS control unit
J1086 blind spot detection system control unit (**)
J388 rear left door control unit
G397 rain and light sensor
J518 entry and start authorisation control unit
J500 power steering control unit
J250 electronically controlled damping control unit
J367 control unit for battery monitoring
C Alternator
J1087 blind spot detection system control unit (**)
J389 rear right door control unit
G273 interior monitoring sensor
J285 dash panel control unit

Gateway

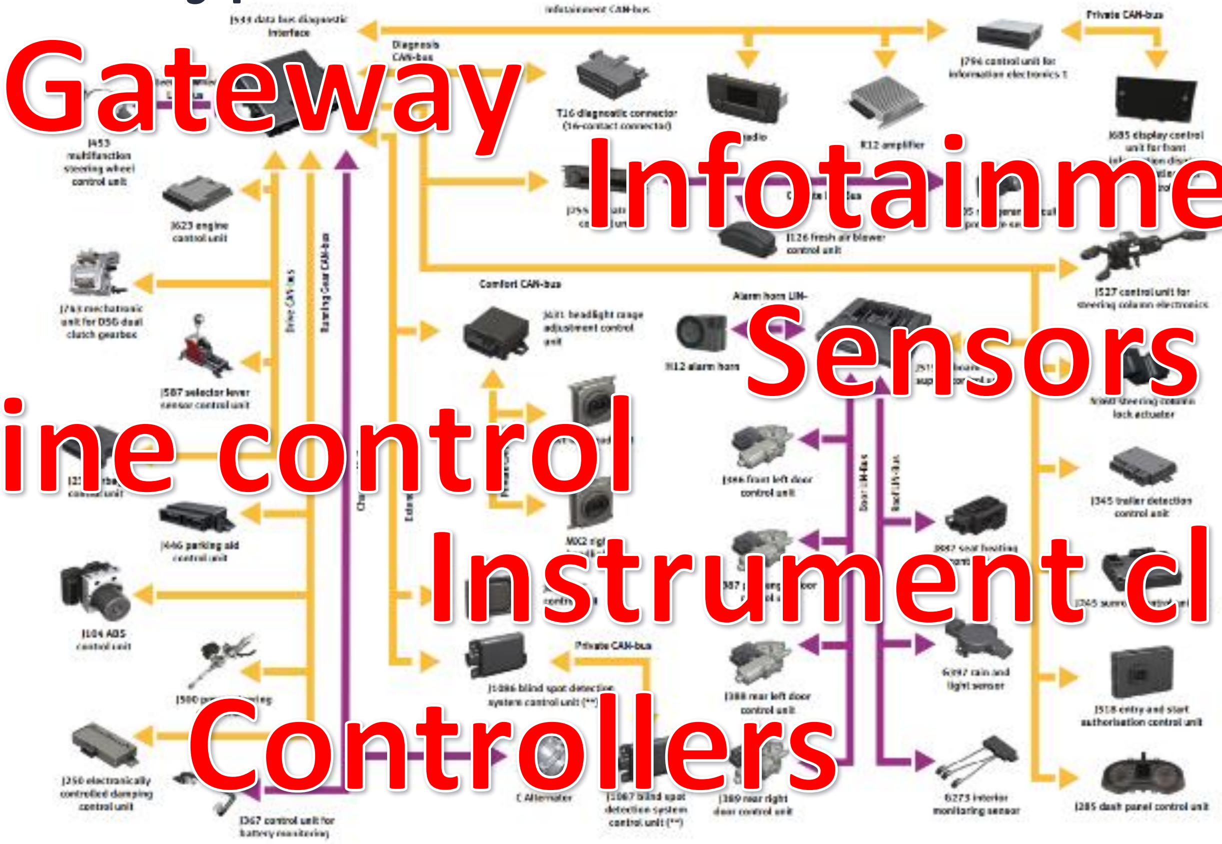
Infotainment

Sensors

Engine control

Instrument cluster

# Typical ECUs found in a car...


Infotainment CAN-bus
J533 data bus diagnostic interface
Diagnosis CAN-bus
T16 diagnostic connector (16-contact connector)
radio
R12 amplifier
J794 control unit for information electronics 1
Private CAN-bus
J685 display control unit for front information display
J453 multifunction steering wheel control unit
J623 engine control unit
J126 fresh air blower control unit
J527 control unit for steering column electronics
J743 mechatronic unit for DSG dual clutch gearbox
Drive CAN-bus
Running Gear CAN-bus
Comfort CAN-bus
J431 headlight range adjustment control unit
Alarm horn LIN-
H12 alarm horn
N360 steering column lock actuator
J587 selector lever sensor control unit
J386 front left door control unit
Door LIN-bus
J345 trailer detection control unit
J446 parking aid control unit
MX2 right
J104 ABS control unit
G397 rain and light sensor
J500 power steering
Private CAN-bus
J1086 blind spot detection system control unit (**)
J388 rear left door control unit
J518 entry and start authorisation control unit
J250 electronically controlled damping control unit
J367 control unit for battery monitoring
C Alternator
J1087 blind spot detection system control unit (**)
J389 rear right door control unit
G273 interior monitoring sensor
J285 dash panel control unit

Gateway

Infotainment

Sensors

Engine control

Instrument cluster

Controllers

*Just like embedded systems,*

*these ECUs come in all* ***forms****,* ***shapes*** *and* ***sizes****!*

# Lots of them are stuck in cars worldwide…

… and you can buy them cheaply!

ebay

# Today we target an **Instrument Cluster**

# Today we target an **Instrument Cluster**

*Why?*

*It has blinky lights!*

*We want to **understand** our target…*

*So we need to its **firmware**!*

# Let’s get our target’s firmware!

# Let’s get our target’s firmware!

Firmware upgrade

Leaks

# Let’s get our target’s firmware!

~~Firmware~~ upgrade

~~Leaks~~

# Let’s get our target’s firmware!

~~Firmware upgrade~~

~~Leaks~~

Interfaces

Chips

Software

# Let’s get our target’s firmware!

~~Firmware upgrade~~

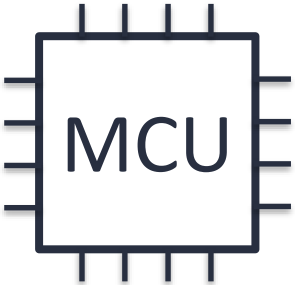
~~Leaks~~

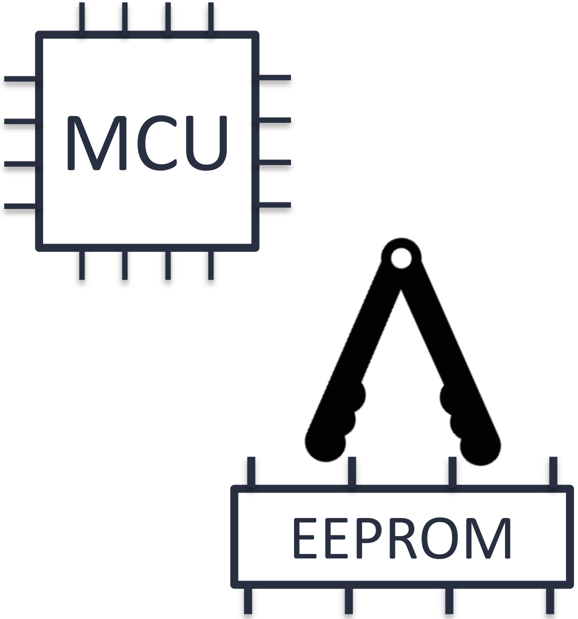
Interfaces

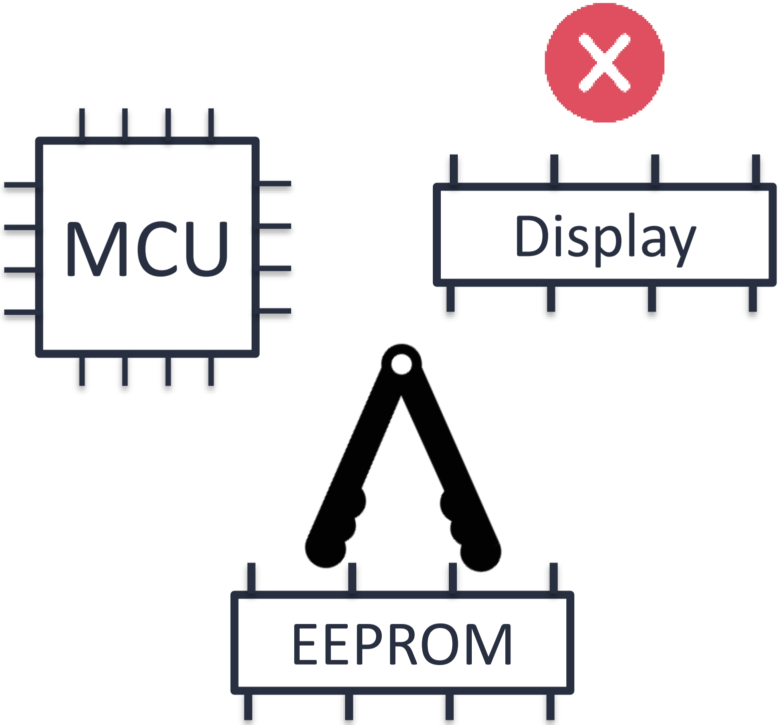
Chips

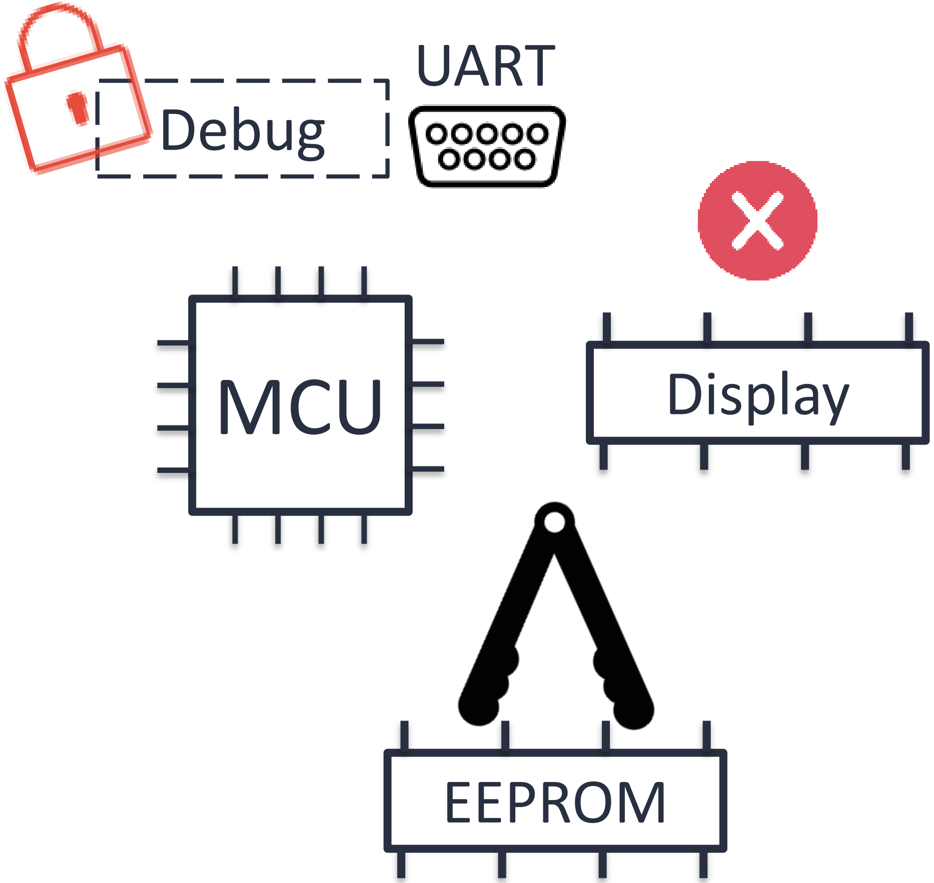
Software

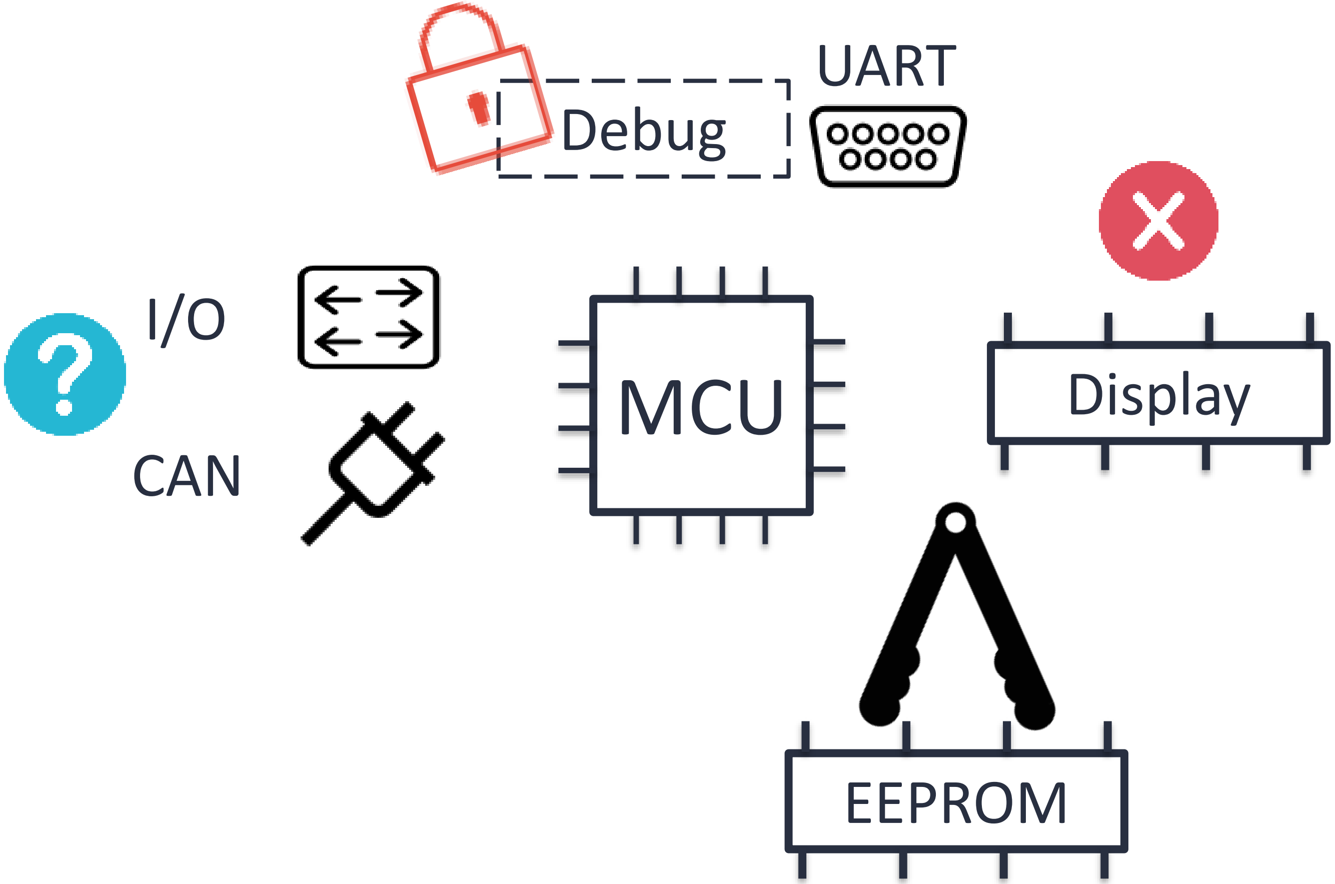
Let’s **open up** our **Instrument Cluster!**


MCU


MCU
EEPROM


MCU
Display
EEPROM


Debug
UART
MCU
Display
EEPROM


Debug
UART
I/O
CAN
MCU
Display
EEPROM

# Most ECUs speak UDS over CAN!

Most ECUs speak UDS over CAN!

Yay, UDS! *Wait… what?*

# Unified Diagnostic Services (UDS)

ISO14229

- Diagnostics
- Data Transmission
  - Read and write memory
- Security Access check
- And loads of more stuff...

# Quick analysis of our dashboard

# Quick analysis of our dashboard

- Read/write memory functions
  - Protected

# Quick analysis of our dashboard

- Read/write memory functions
    - Protected

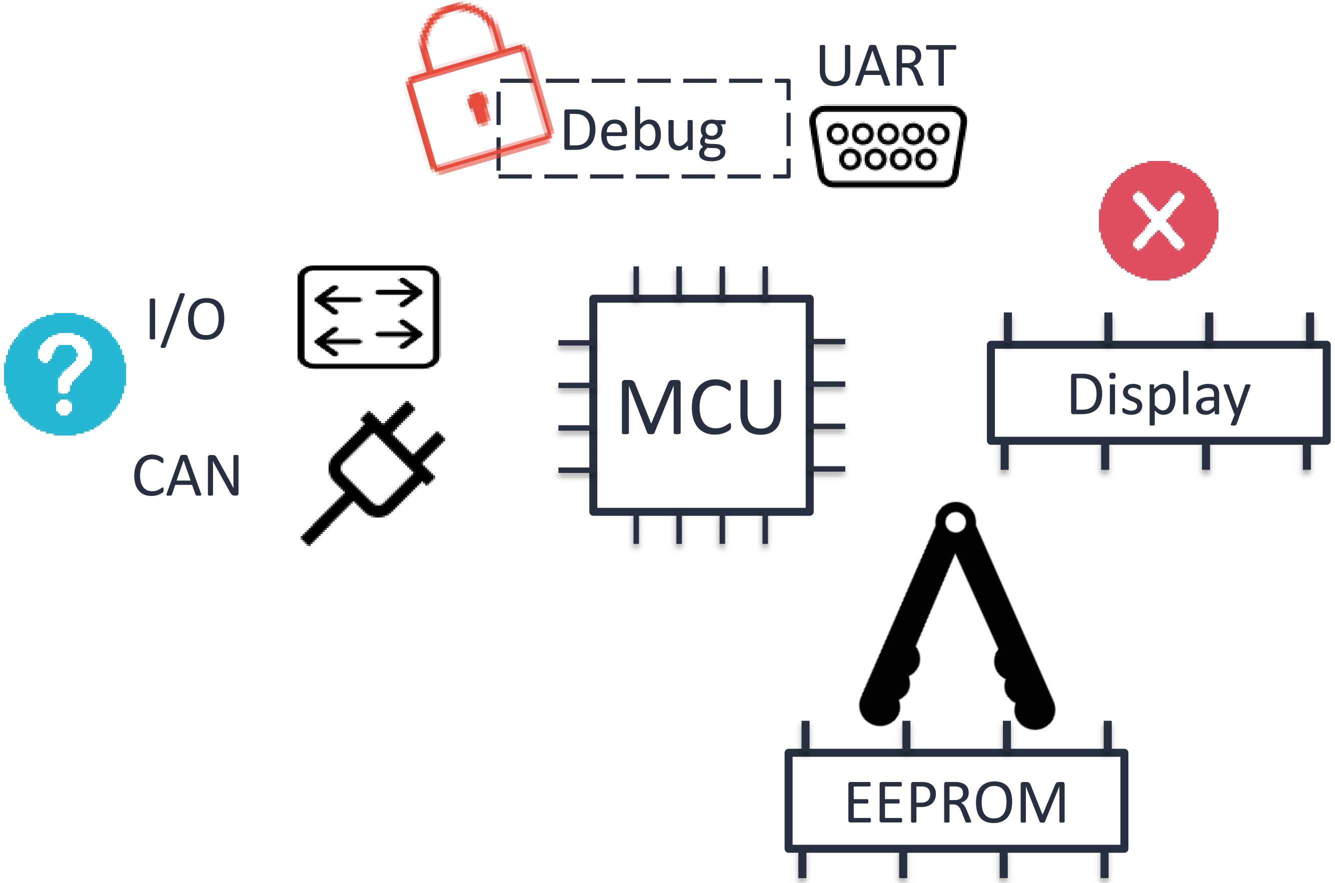
- Black-box vulnerability discovery
    - Possible; but too difficult

# Quick analysis of our dashboard

- Read/write memory functions
  - **Protected**

- Black-box vulnerability discovery
  - **Possible; but too difficult**

- We want something easy…



Debug

UART

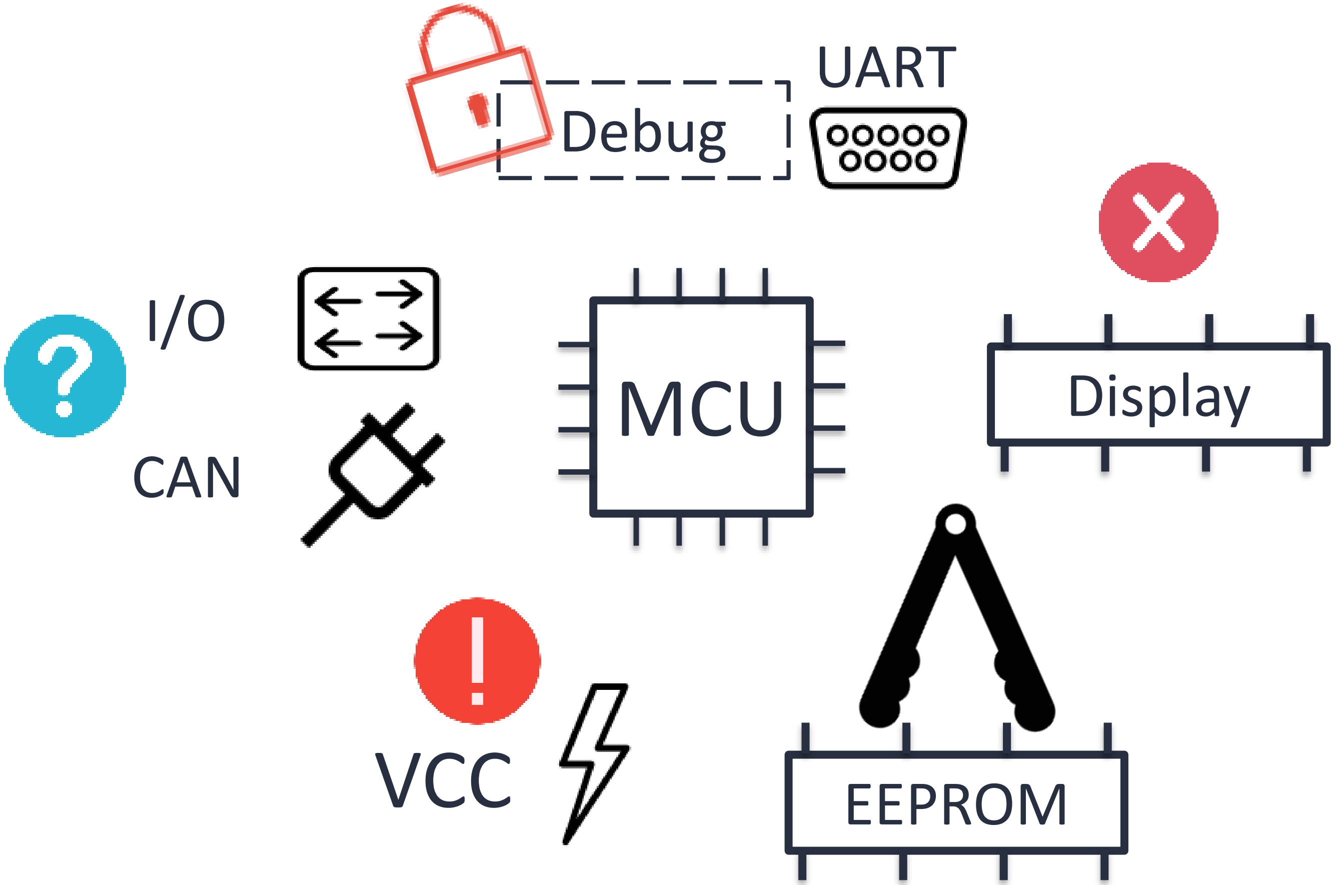
I/O

CAN

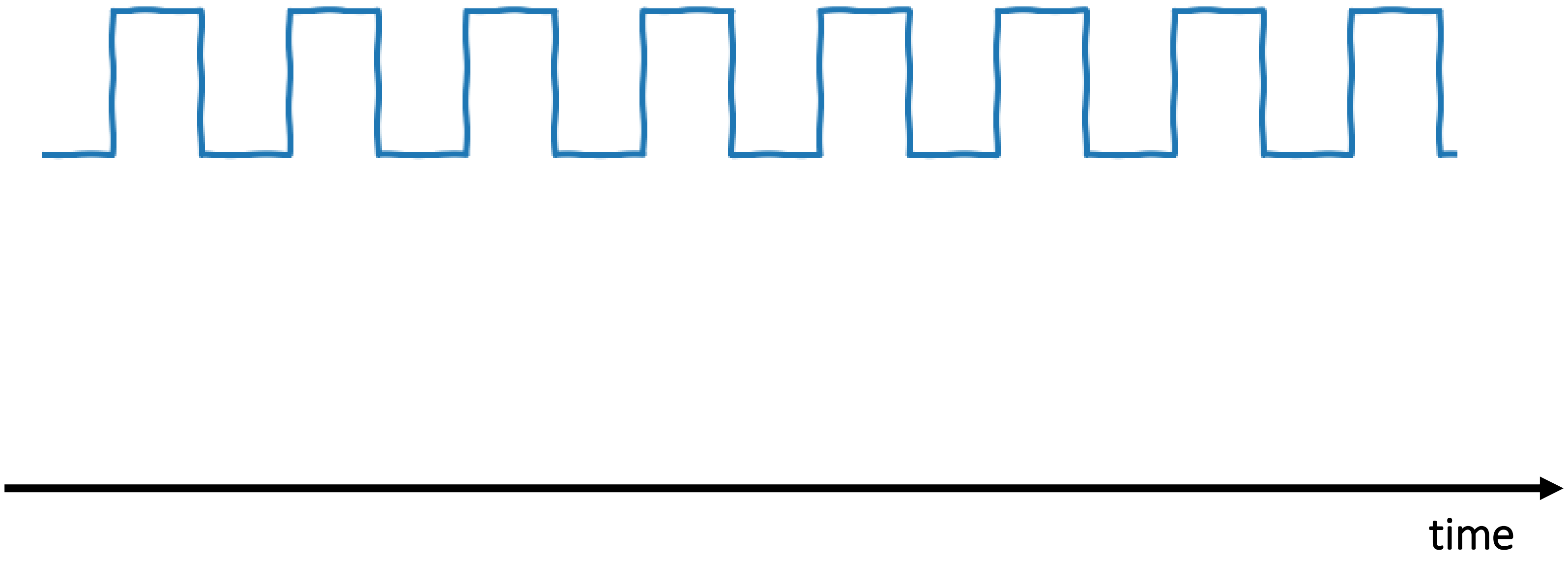
MCU

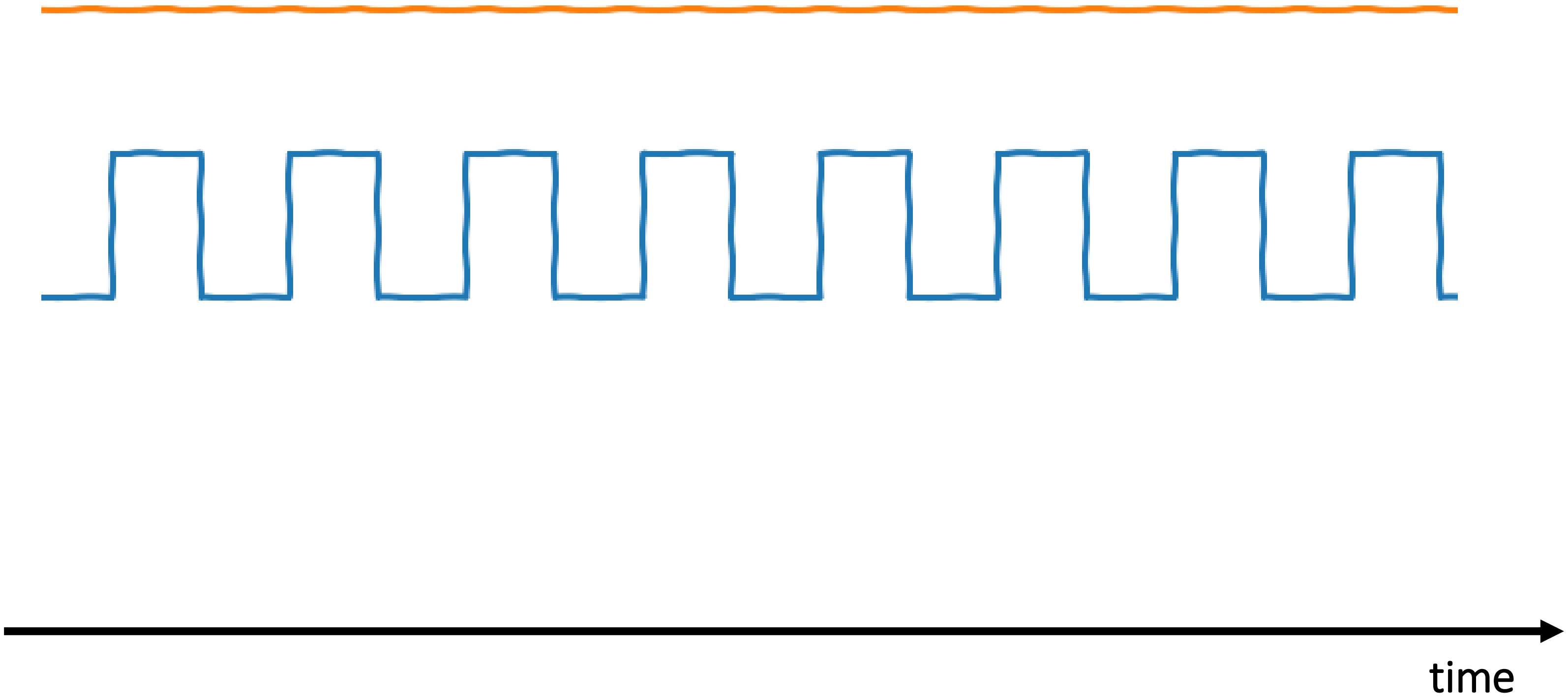
Display

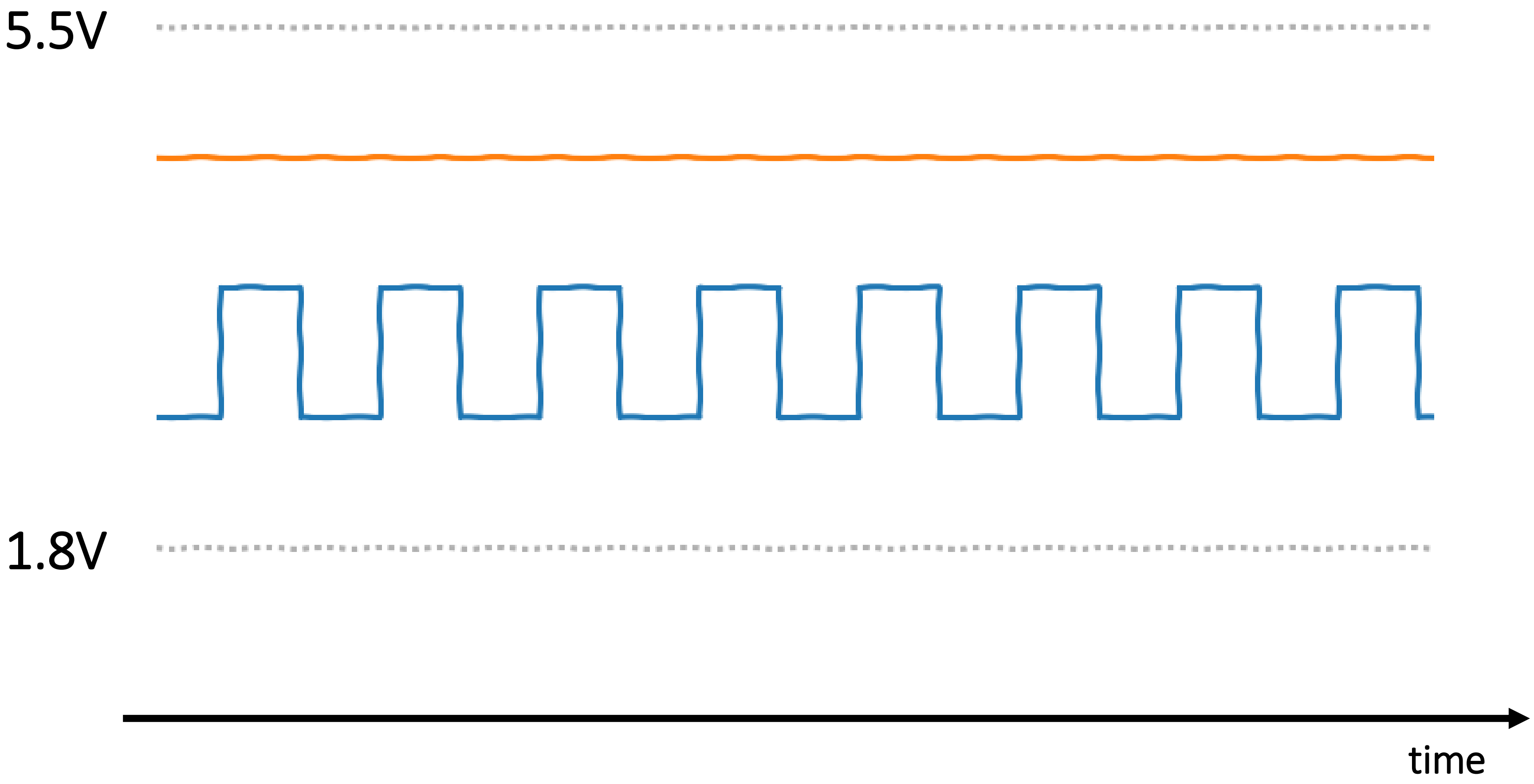
EEPROM

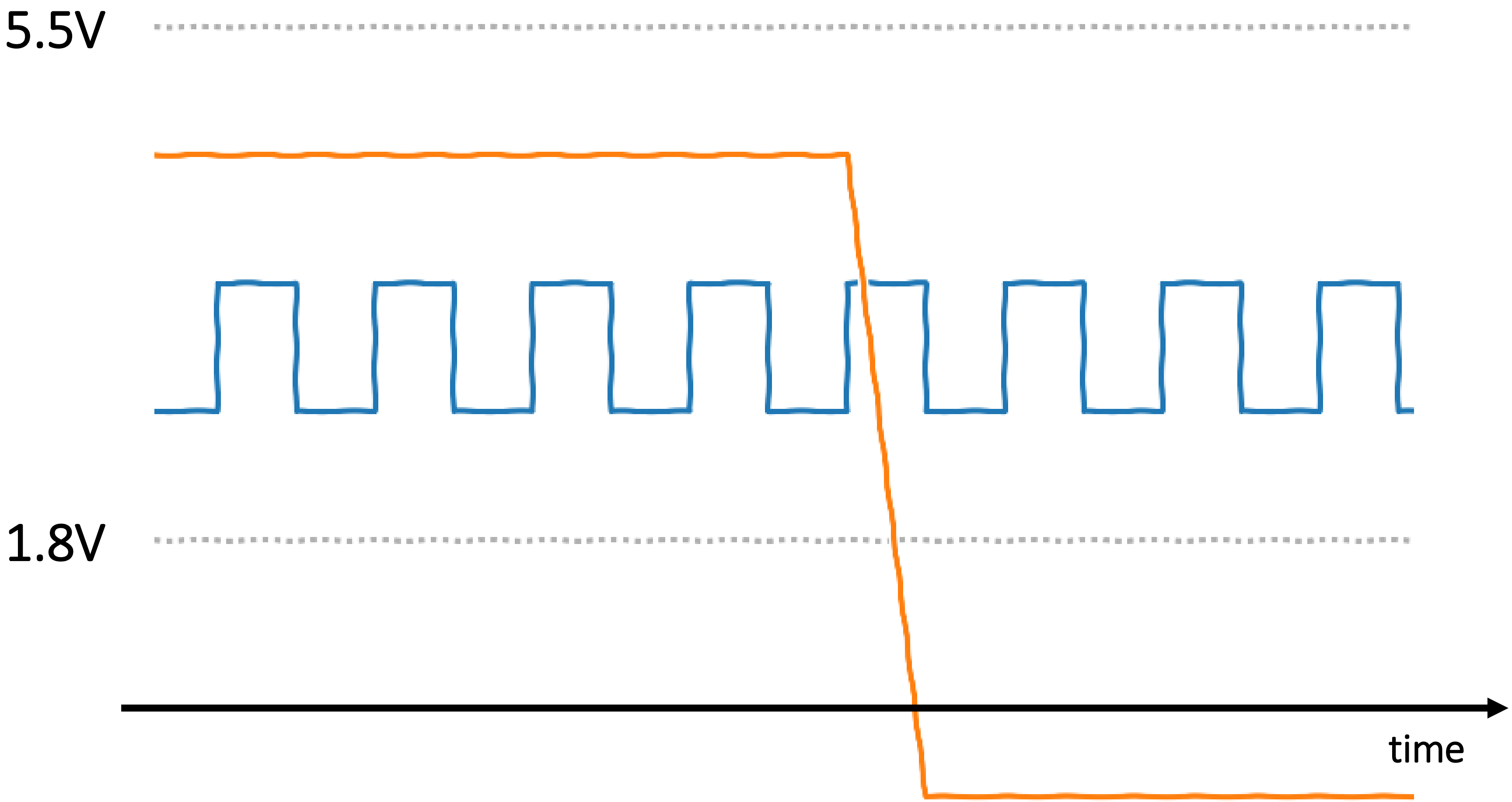

Debug
UART
I/O
CAN
MCU
Display
VCC
EEPROM

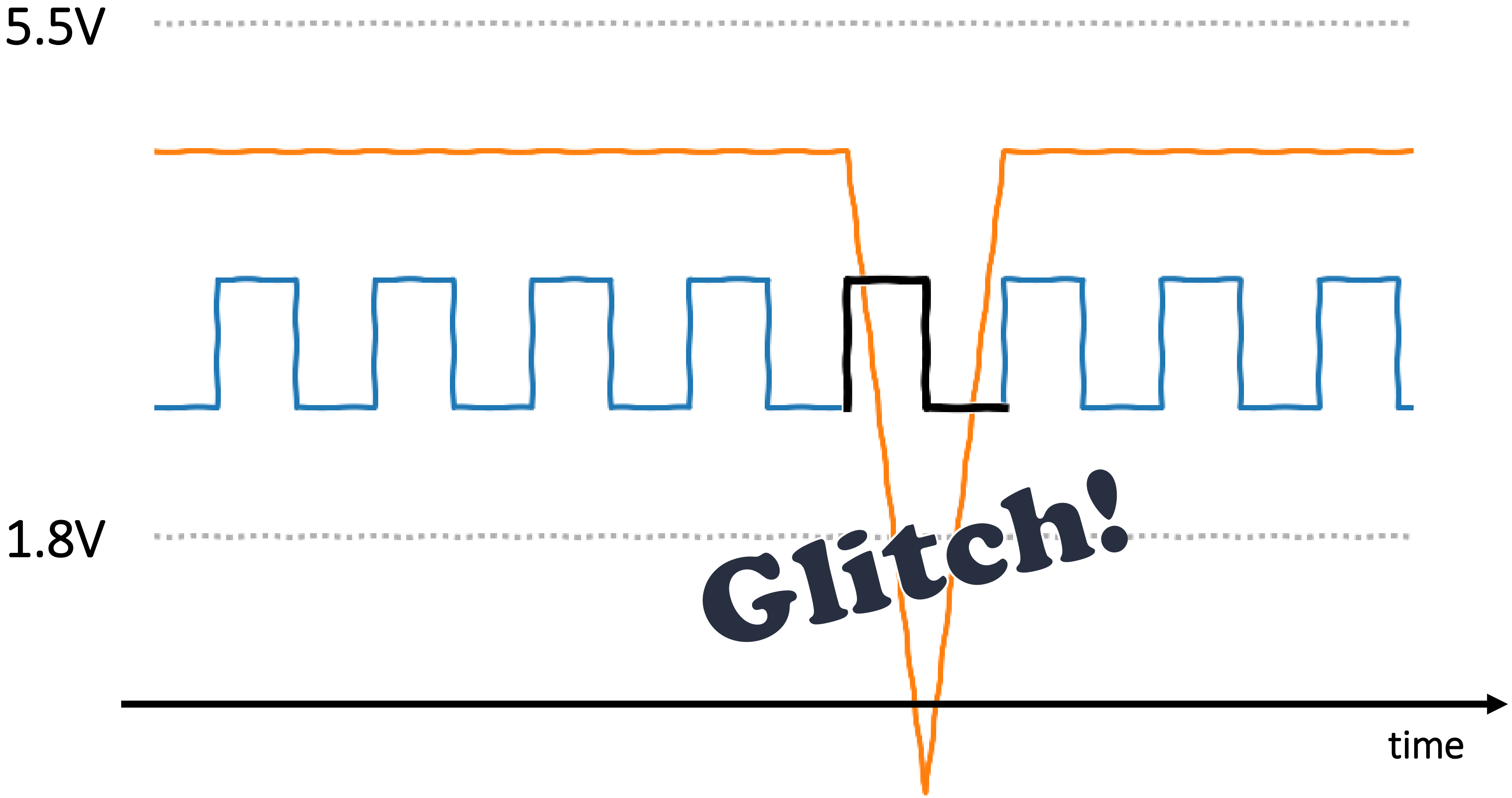
# *Voltage Fault Injection !!!*


time


time

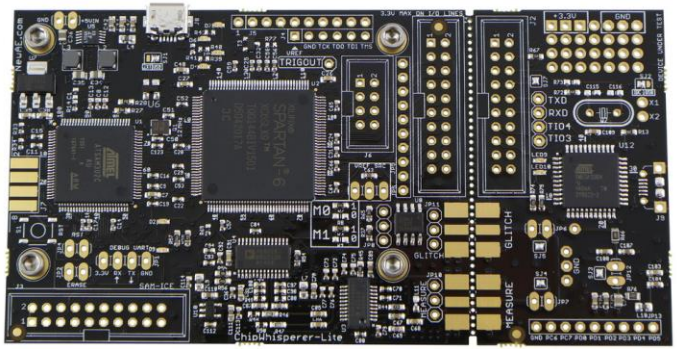

5.5V
1.8V
time


5.5V
1.8V
time


5.5V
1.8V
Glitch!
time

# Fault Injection – Tooling

*Open source*



**ChipWhisperer®**

*Commercial*

**Inspector FI**

***Fault Injection tooling is available to the masses!***

# What happens when we glitch?

## Things go wrong!

# Fault Injection breaks things!

- Memory contents
- Register contents
- Executed instructions

*You cannot trust anything anymore…*

*We can modify instructions and data!*

*We can modify instructions and data!*

*Yes, this also means we can skip instructions!*

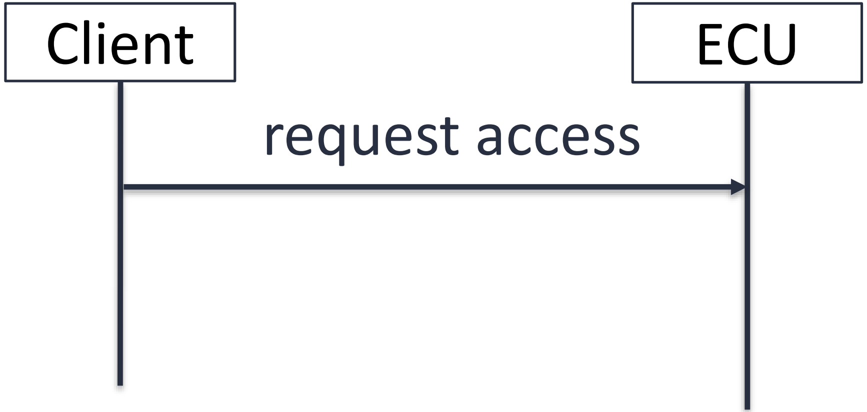
*We can modify instructions and data!*

*Yes, this also means we can <u>skip instructions</u>!*

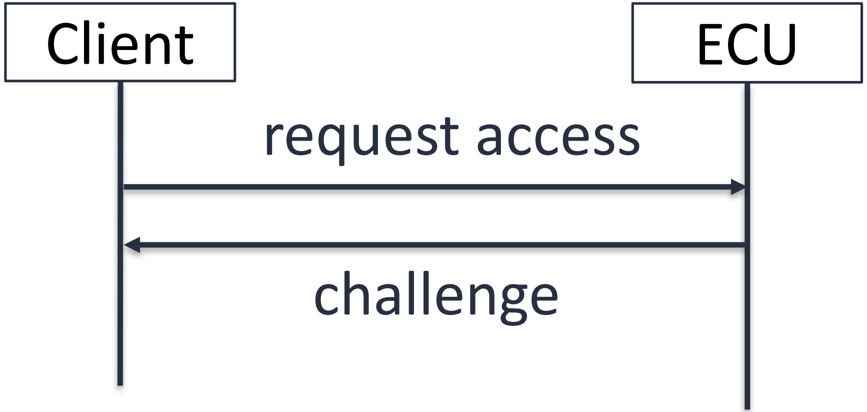
*This works on all standard architectures:*
***ARM, MIPS, PowerPC, SH4, V850, Intel**, etc.*

*Let’s glitch something…*

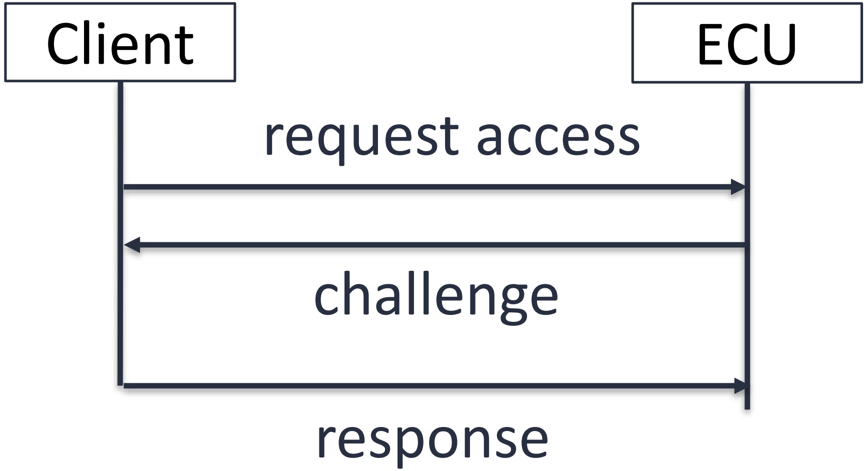
# Glitching the Security Access Check



# Glitching the Security Access Check



# Glitching the Security Access Check



Client

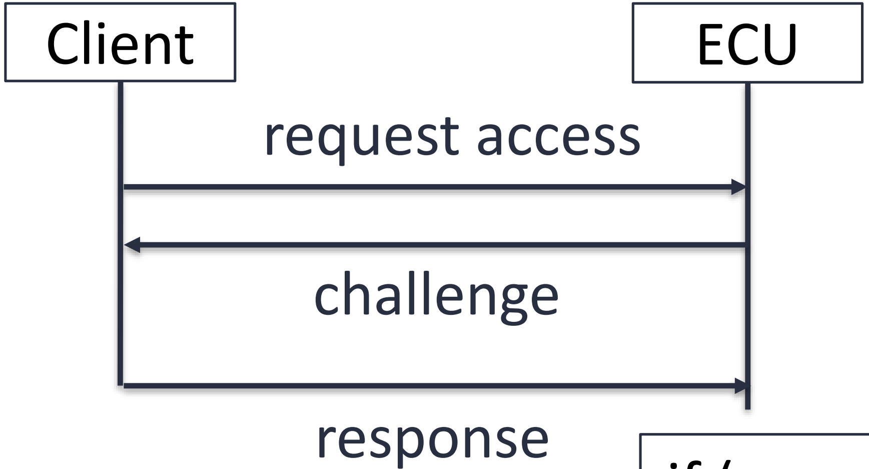
ECU

request access

challenge

response

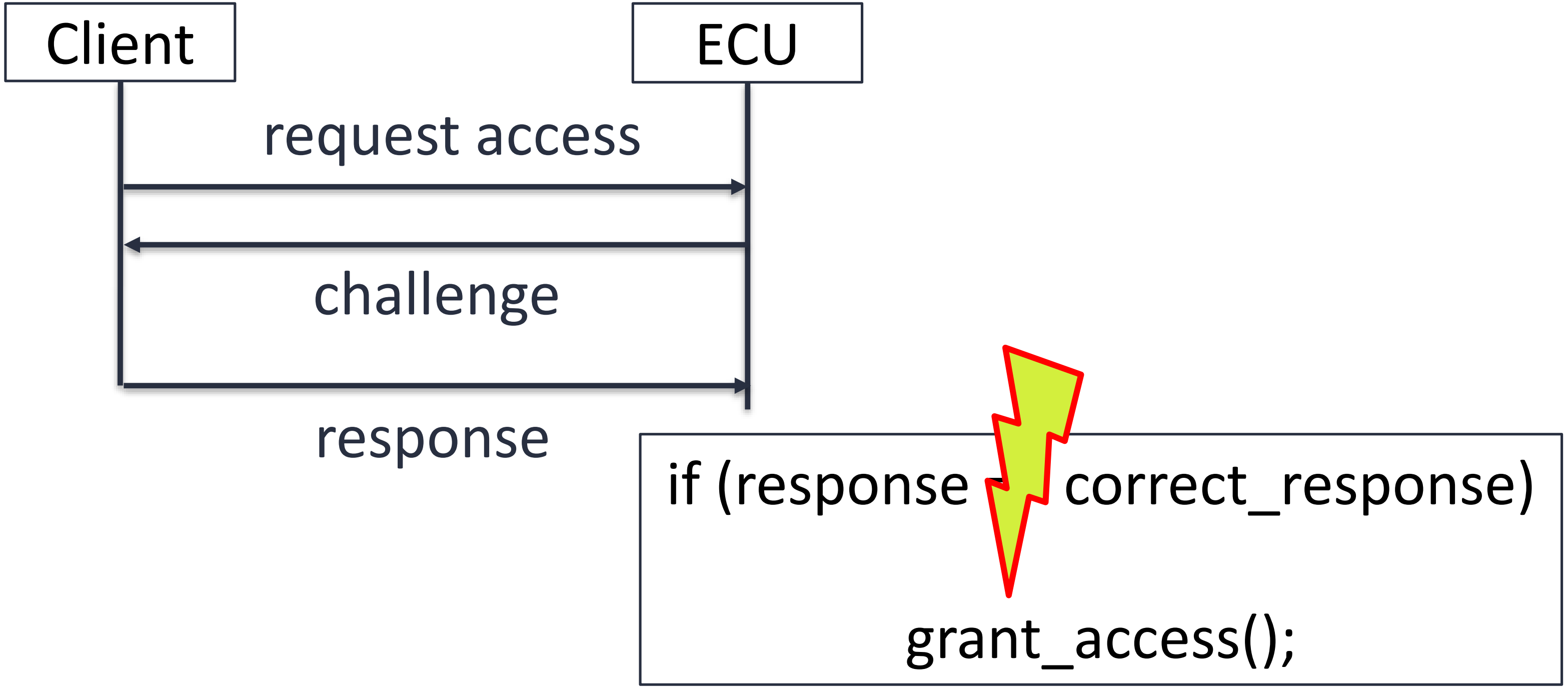
# Glitching the Security Access Check



```
if (response == correct_response)

        grant_access();
```

# Glitching the Security Access Check



```
if (response == correct_response)
        grant_access();
```

# Glitching the Security Access Check Results

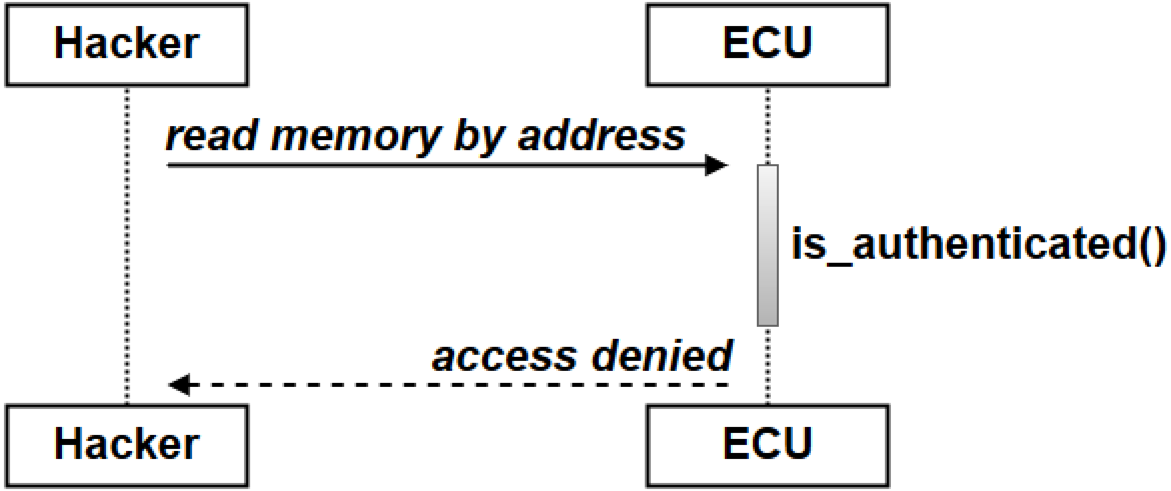
- Not successful :’(
- There’s a 10 minute timeout after 3 failed attempts
- Simply not practical for us (or an attacker)

# Glitching the Security Access Check Results

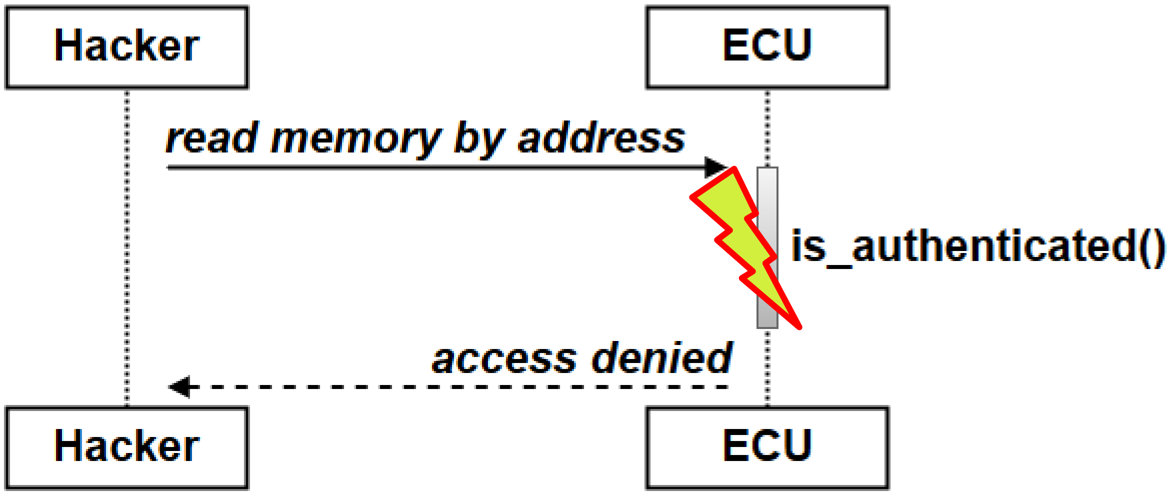
- Not successful :’(
- There’s a 10 minute timeout after 3 failed attempts
- Simply not practical for us (or an attacker)

***You win some, you lose some!***

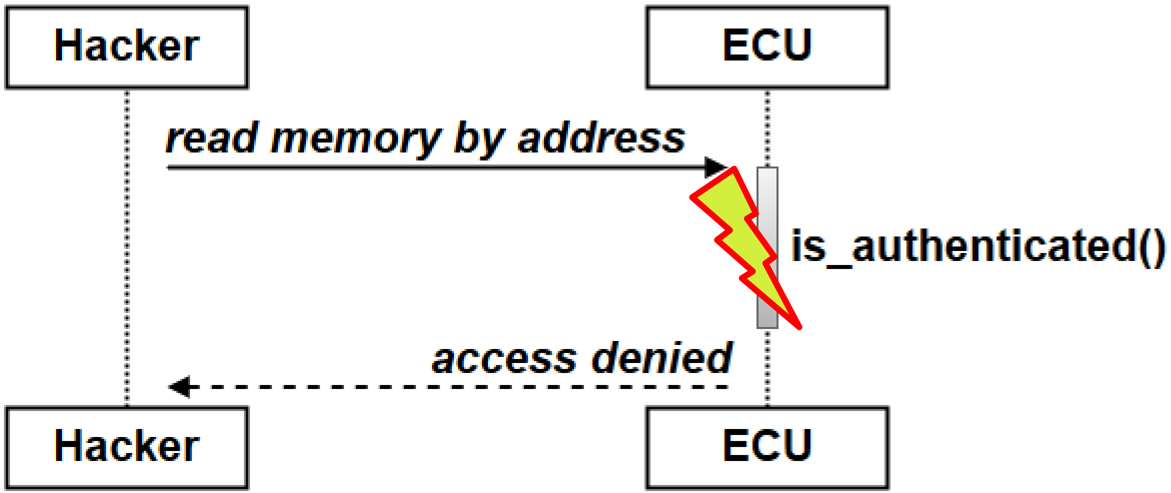
# Glitching ReadMemoryByAddress



# Glitching ReadMemoryByAddress



# Glitching ReadMemoryByAddress



*No restrictions on failed attempts!*

# Glitching ReadMemoryByAddress Results

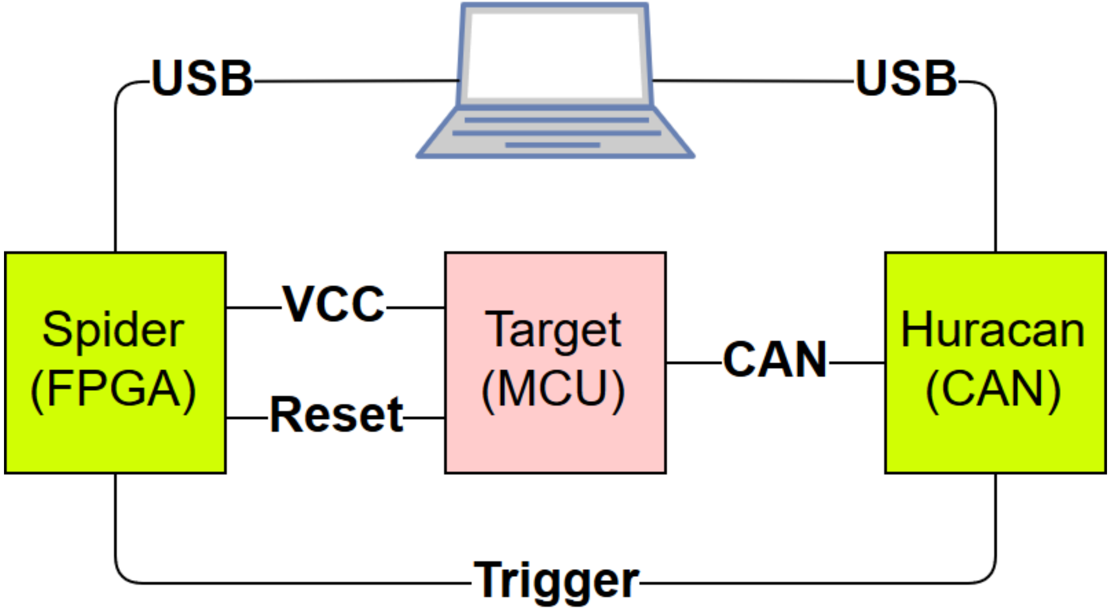
- Successful on several different ECUs
  - Which are designed around different MCUs

- Depending on the target…
  - Allows reading out **N** bytes from an **arbitrary** address

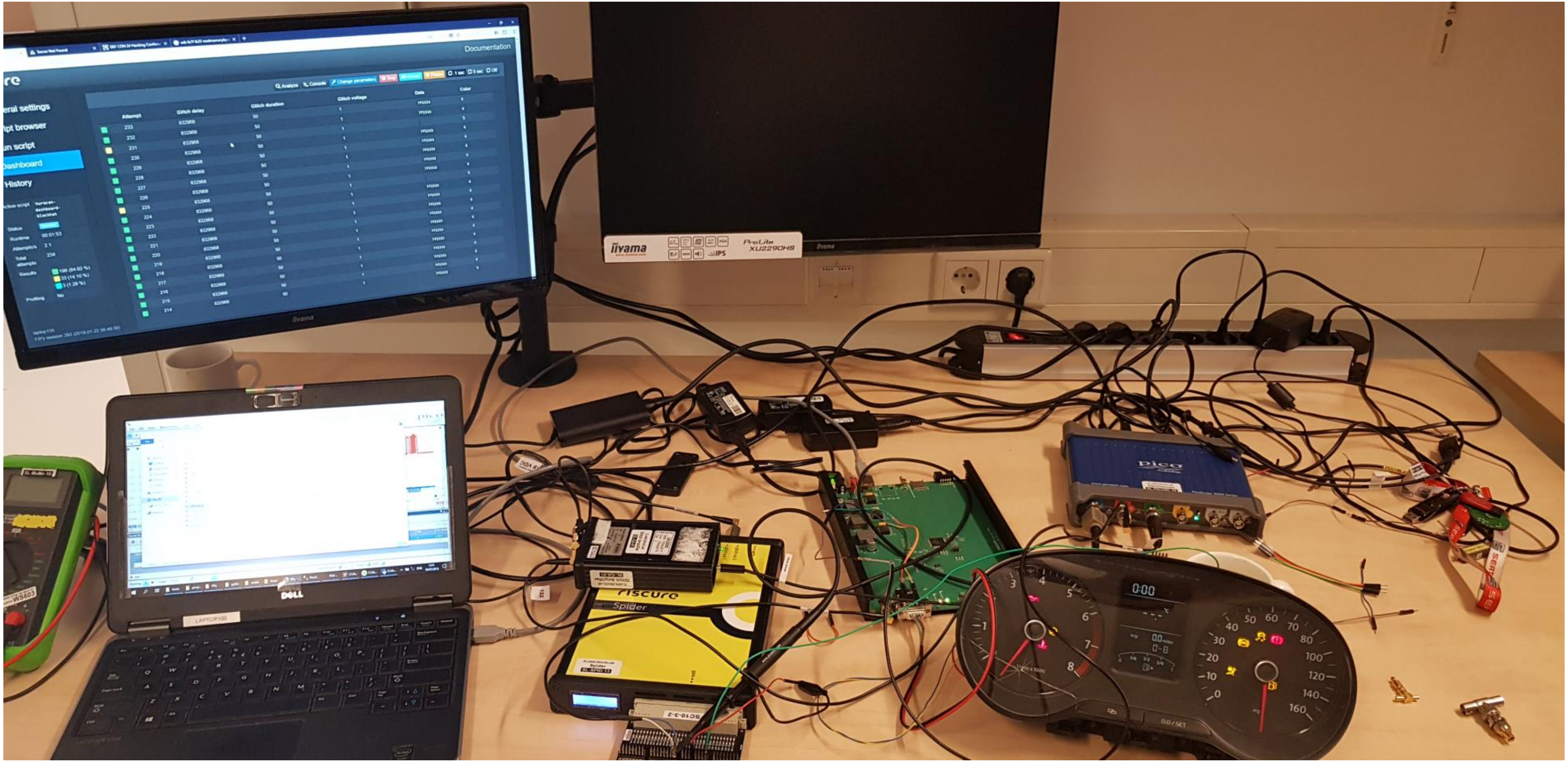
- Complete firmware extracted in the order of days
  - Depended on the size of the internal memory

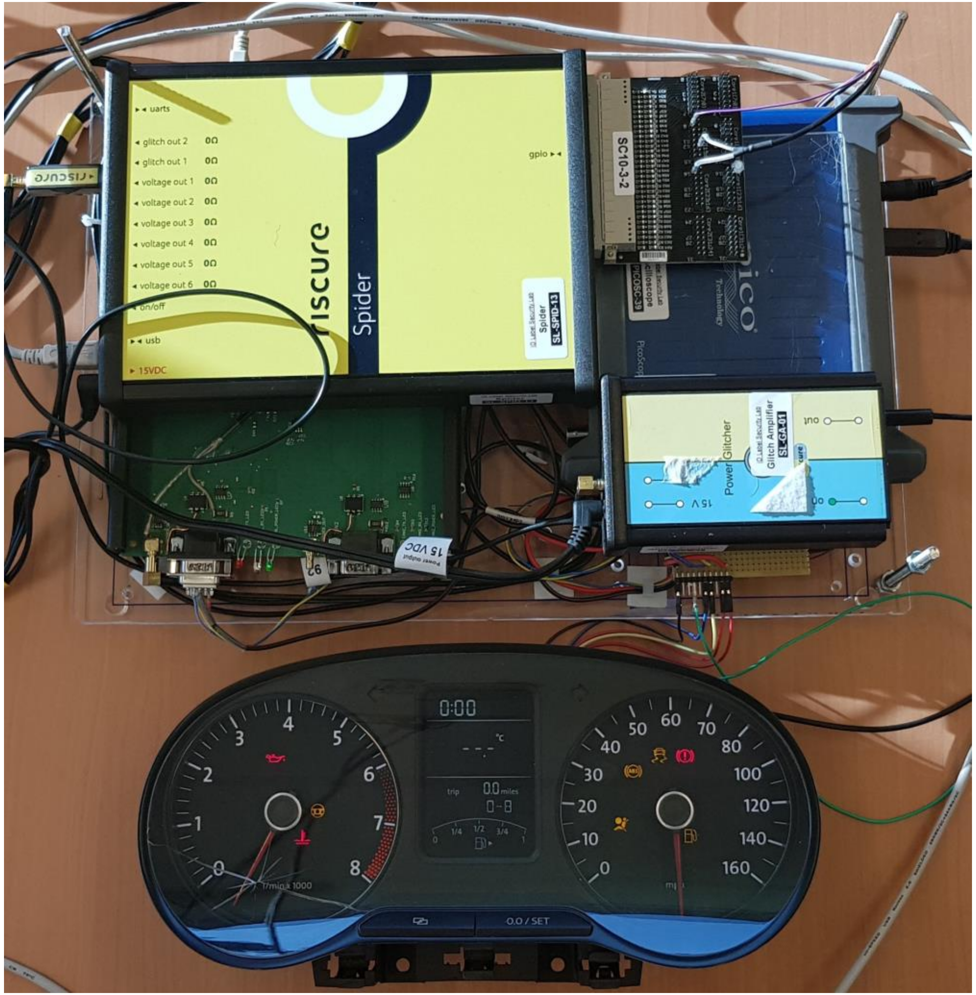
# *DEMO TIME !!!*

# Fault Injection demo setup

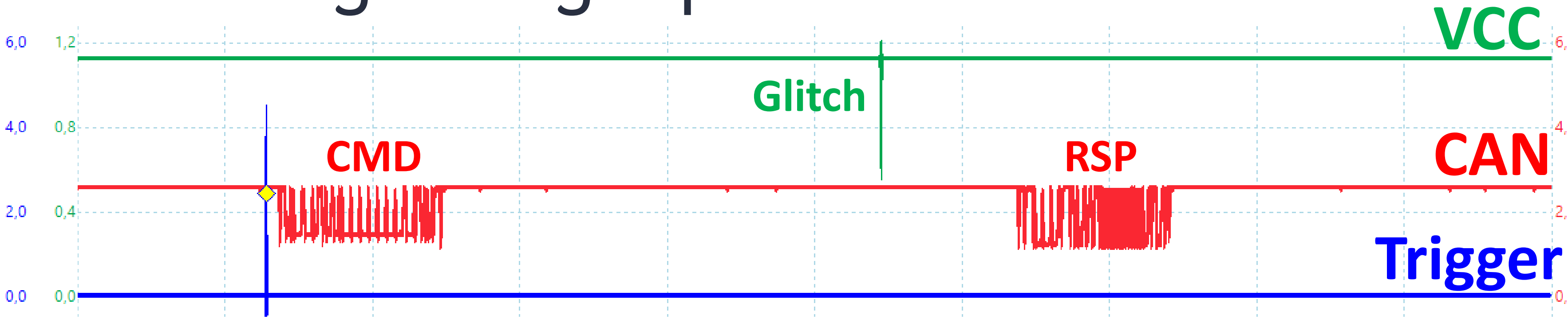


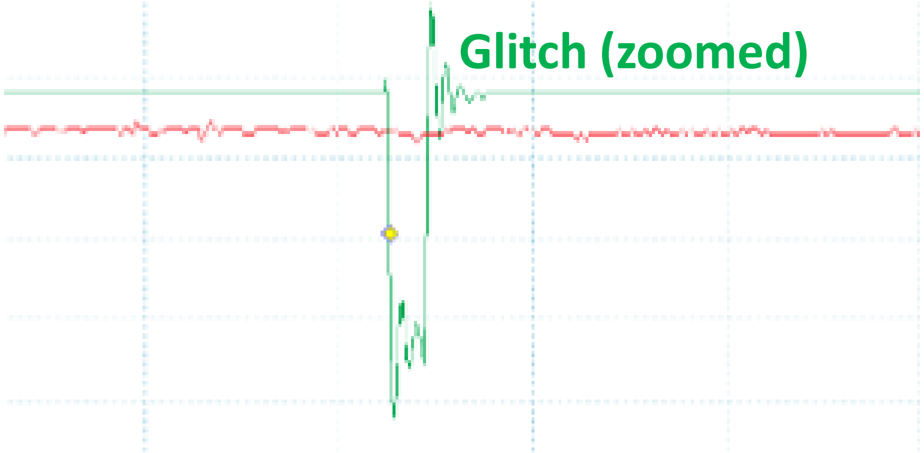
CAN UDS CMD: ReadMemoryByAddress(0x00000000, 0x40)


Documentation
eral settings
ipt browser
un script
Dashboard
History
Attempt
Glitch delay
Glitch duration
Glitch voltage
Data
Color
iiyama
ProLite XU2290HS
riscure
Spider
pico
LAPTOP100
DELL


uarts
glitch out 2
glitch out 1
voltage out 1
voltage out 2
voltage out 3
voltage out 4
voltage out 5
voltage out 6
on/off
usb
15VDC
gpio
riscure
Spider
Spider
SL-SPID-13
SC10-3-2
PicoScope
pico Technology
Power Glitcher
Glitch Amplifier
SL_GA-01
out
15 V
15 VDC
0:00
trip
0.0 miles
0.0 / SET

# Finding the right parameters: Randomize

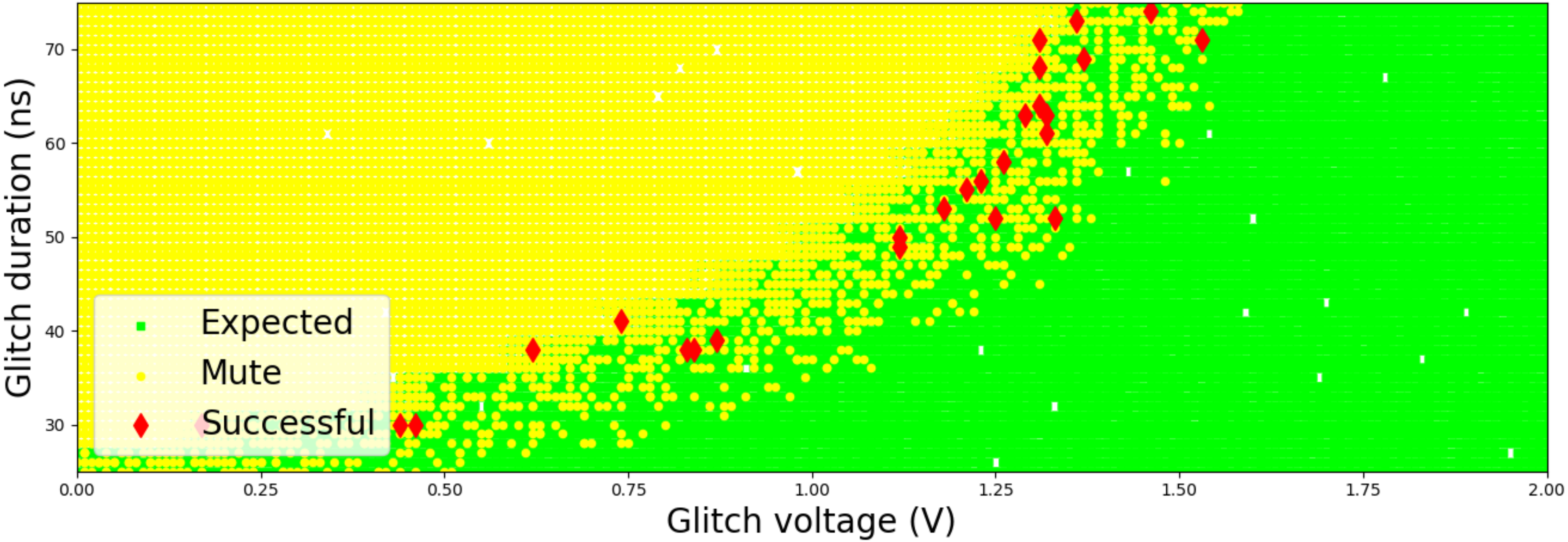




**Glitch Parameters**

- Glitch Delay
- Glitch Duration
- Glitch Voltage

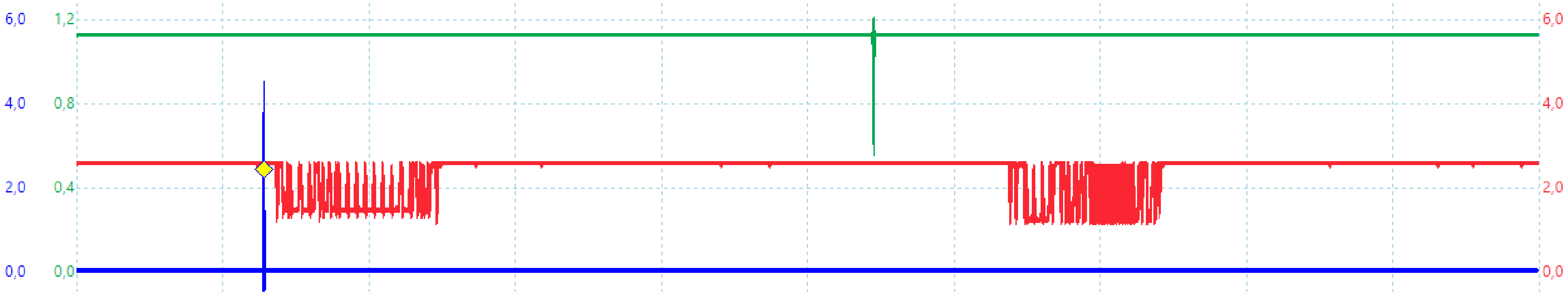
# There is a relationship!



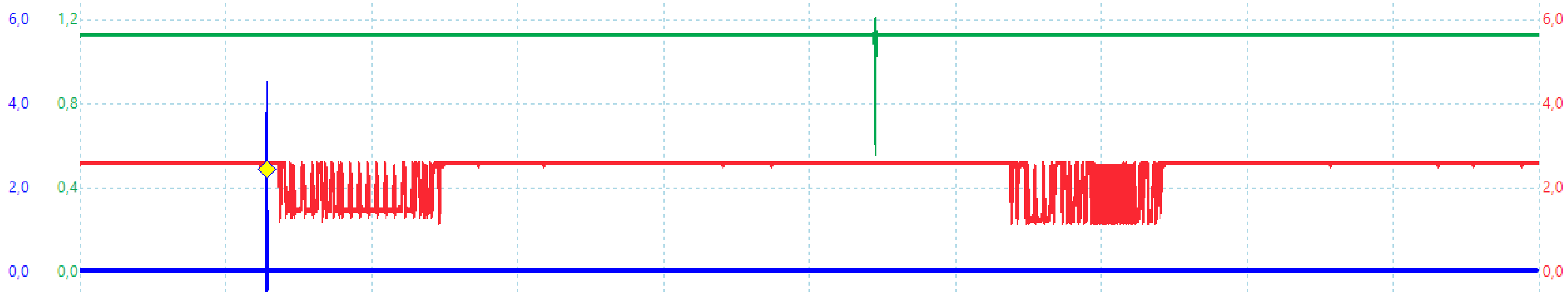
This relationship allows minimizing the parameter search space!

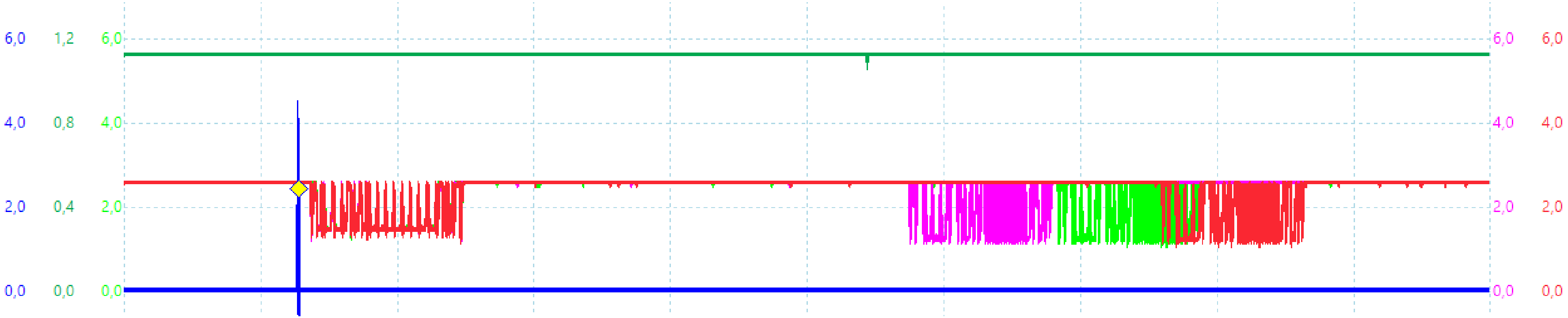
*Let’s switch to the other laptop…*

# Why not a 100% success rate? :’(

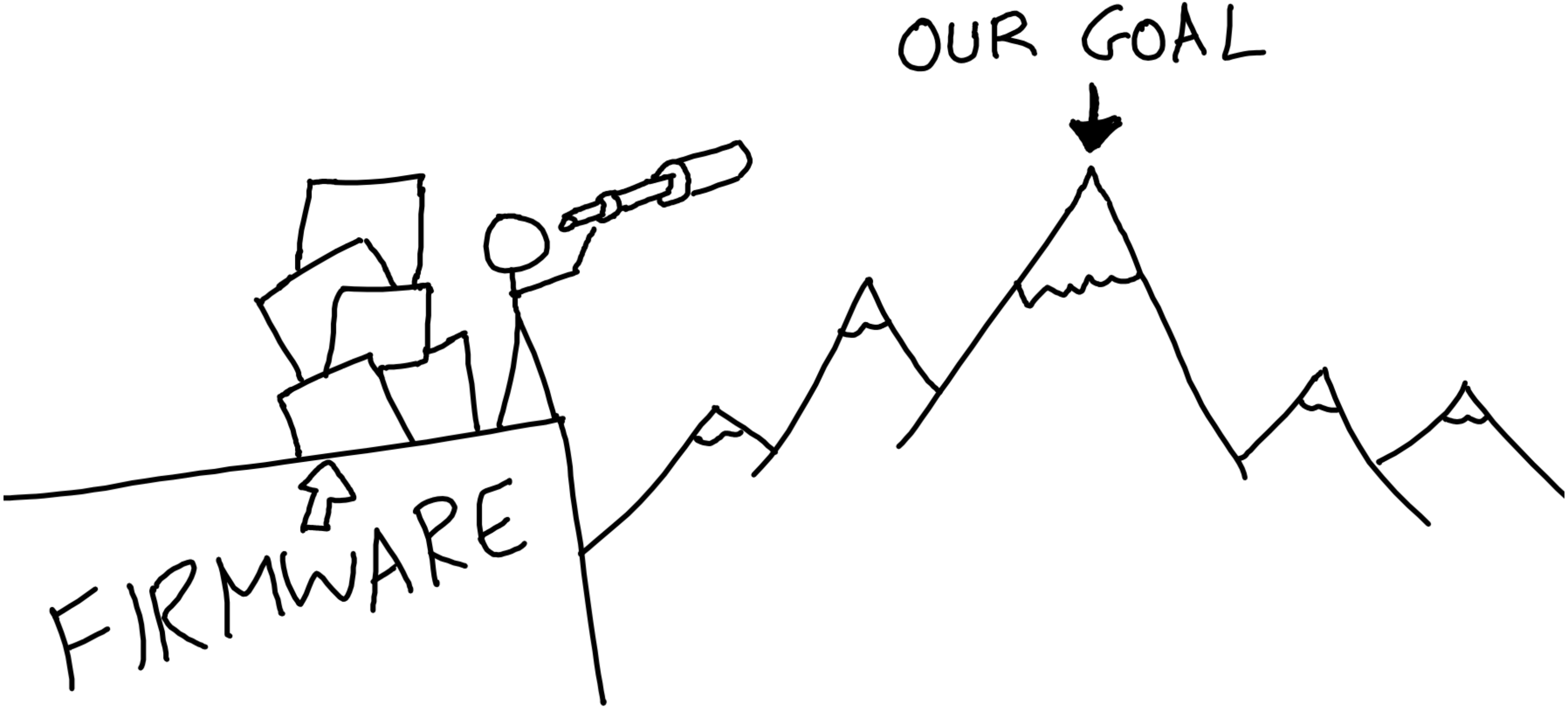


# Why not a 100% success rate? :’(





# *We have the firmware… now what?*


OUR GOAL
FIRMWARE

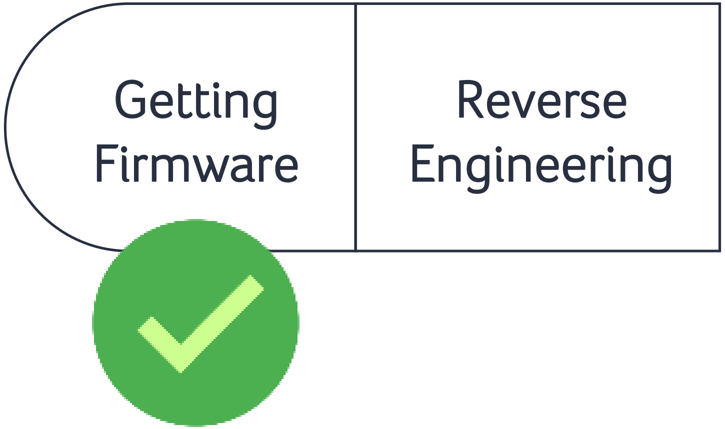
# The Plan

Getting Firmware

# The Plan

Getting Firmware

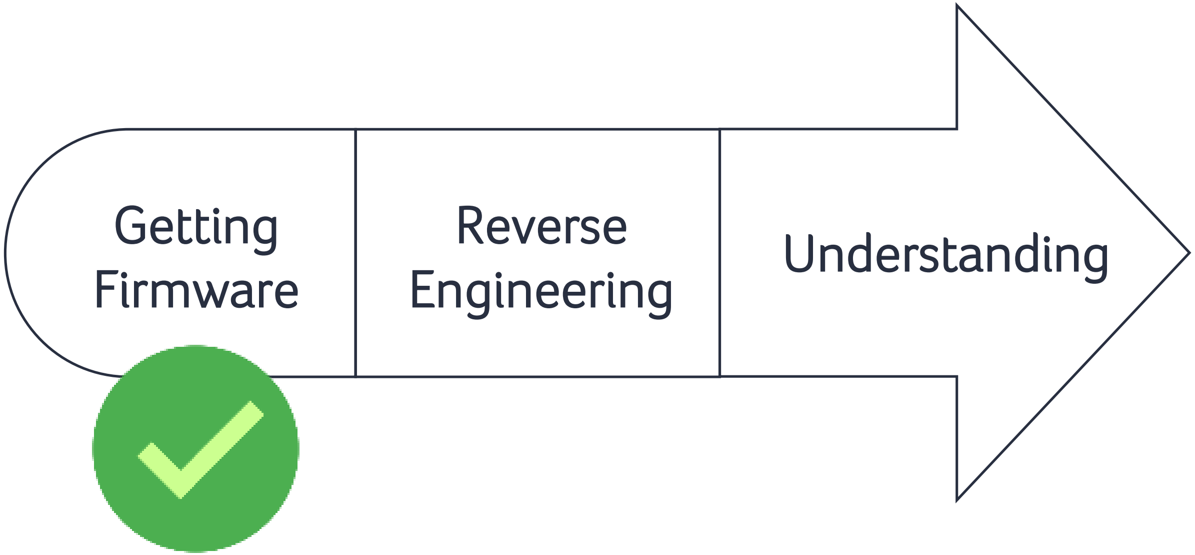
# The Plan



Getting Firmware

Reverse Engineering

# The Plan

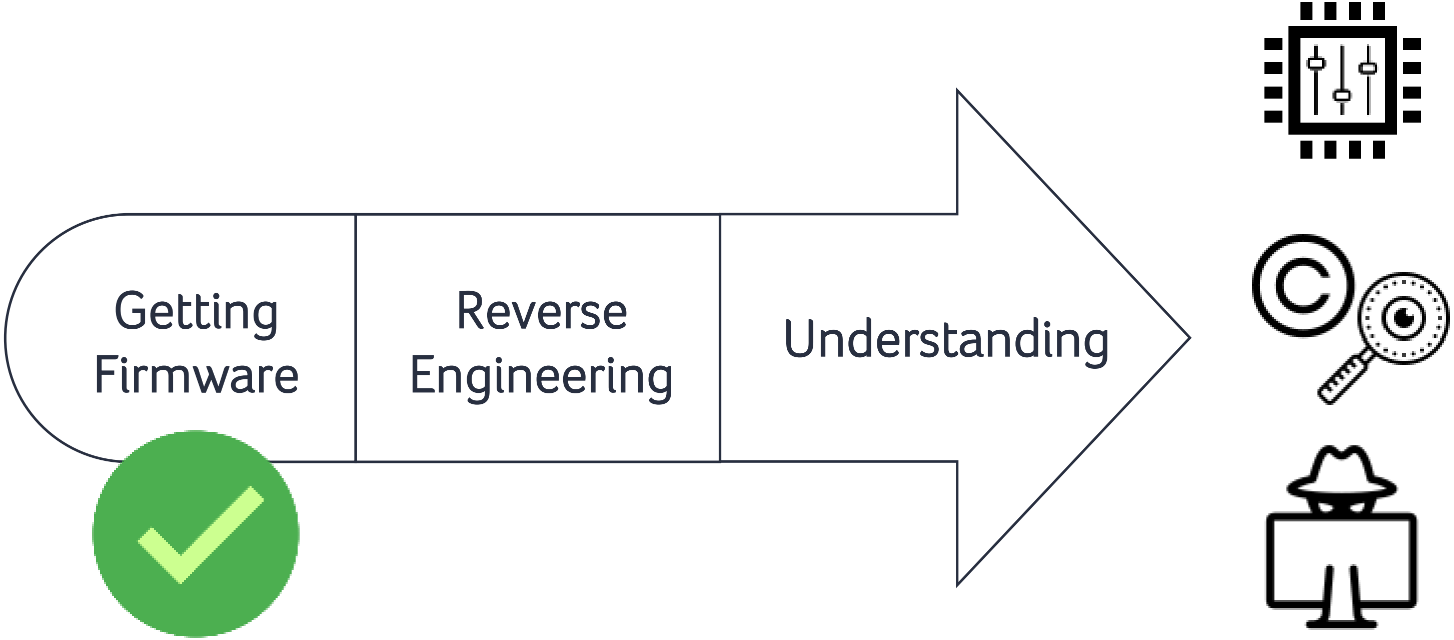


Getting Firmware

Reverse Engineering

Understanding

# The Plan



Getting Firmware ✓

Reverse Engineering

Understanding

Tuning

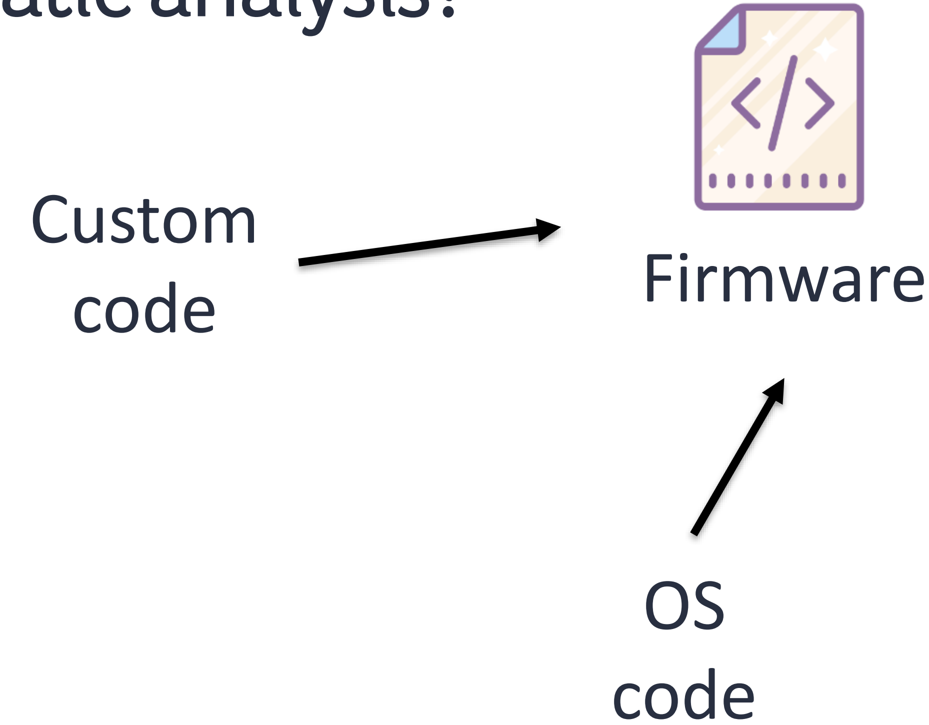
Secrets!

Hacking

# Static analysis?

Firmware

# Static analysis?

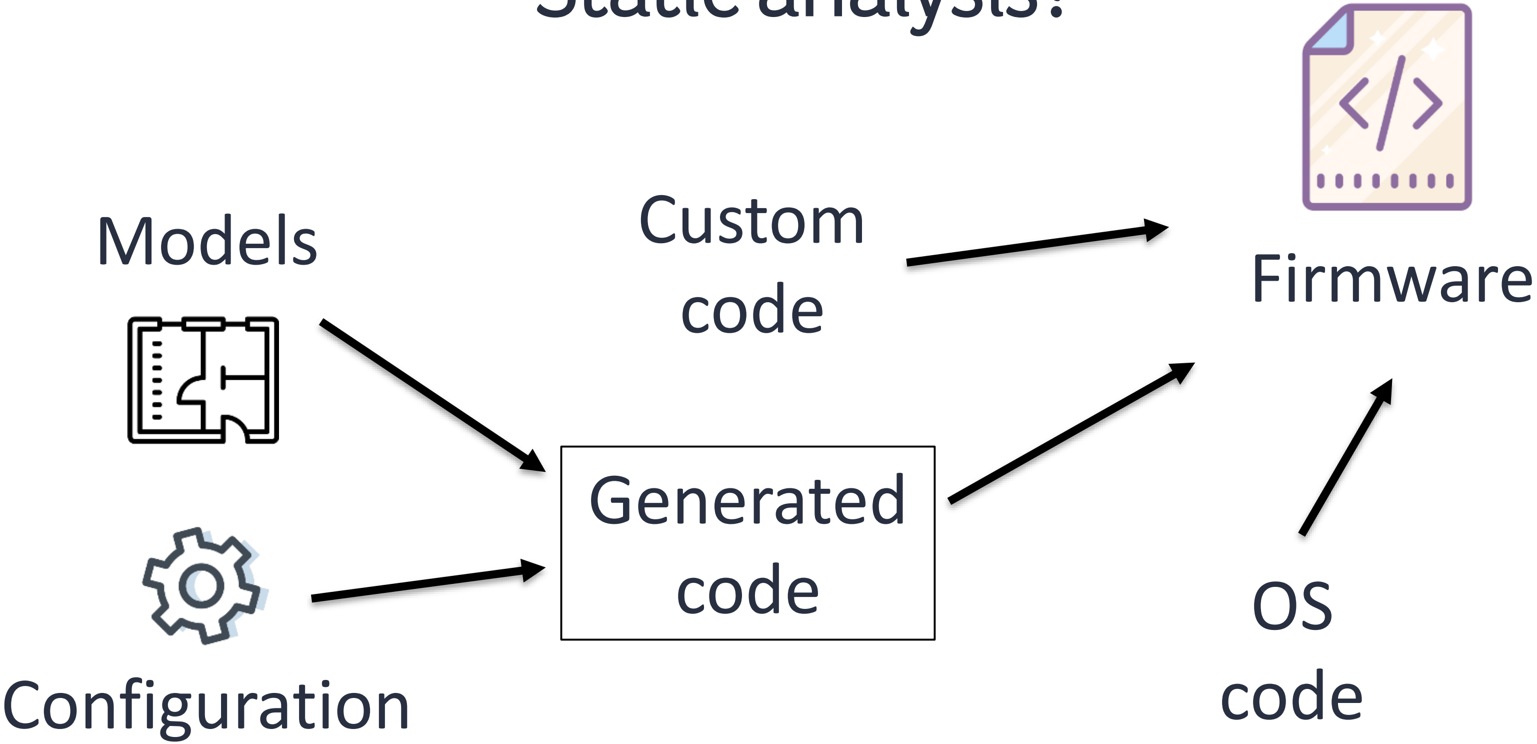


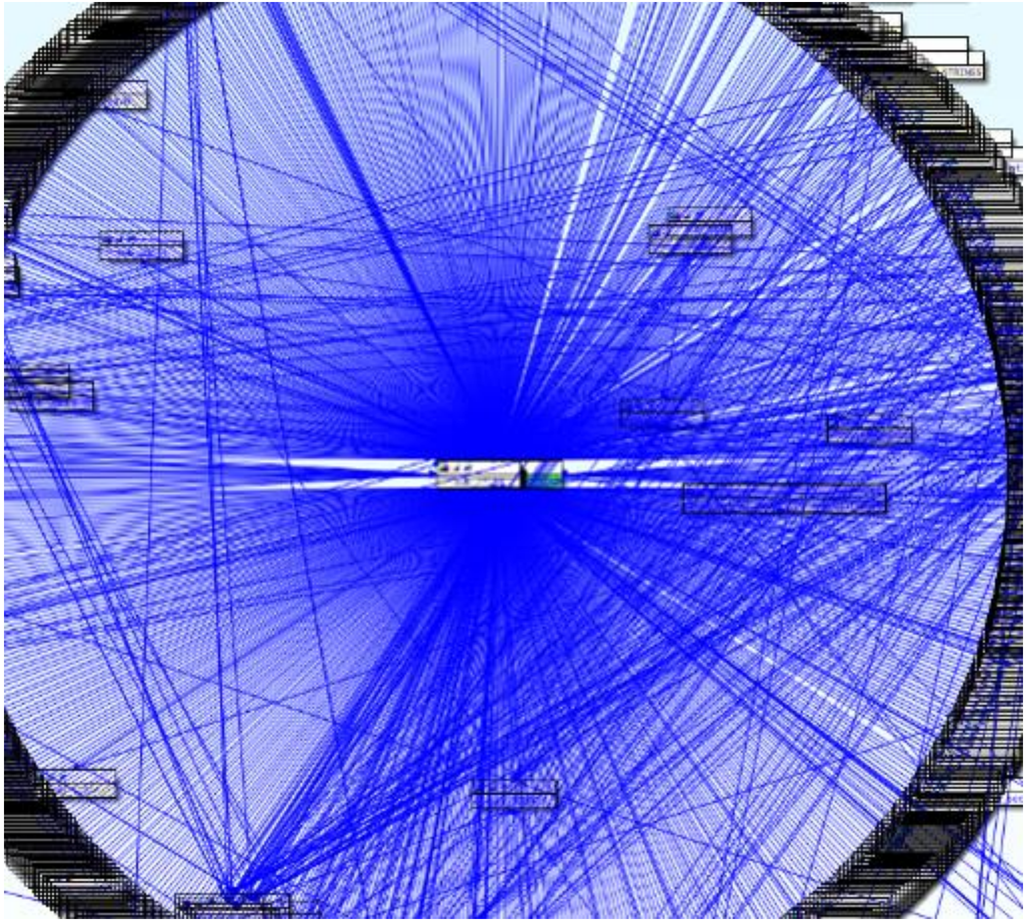
Custom code

Firmware

OS code

# Static analysis?





# Tools?

$ARCH is supported by IDA Pro **and** radare2

# Tools?

Badly ☹

^

$ARCH is supported by IDA Pro **and** radare2

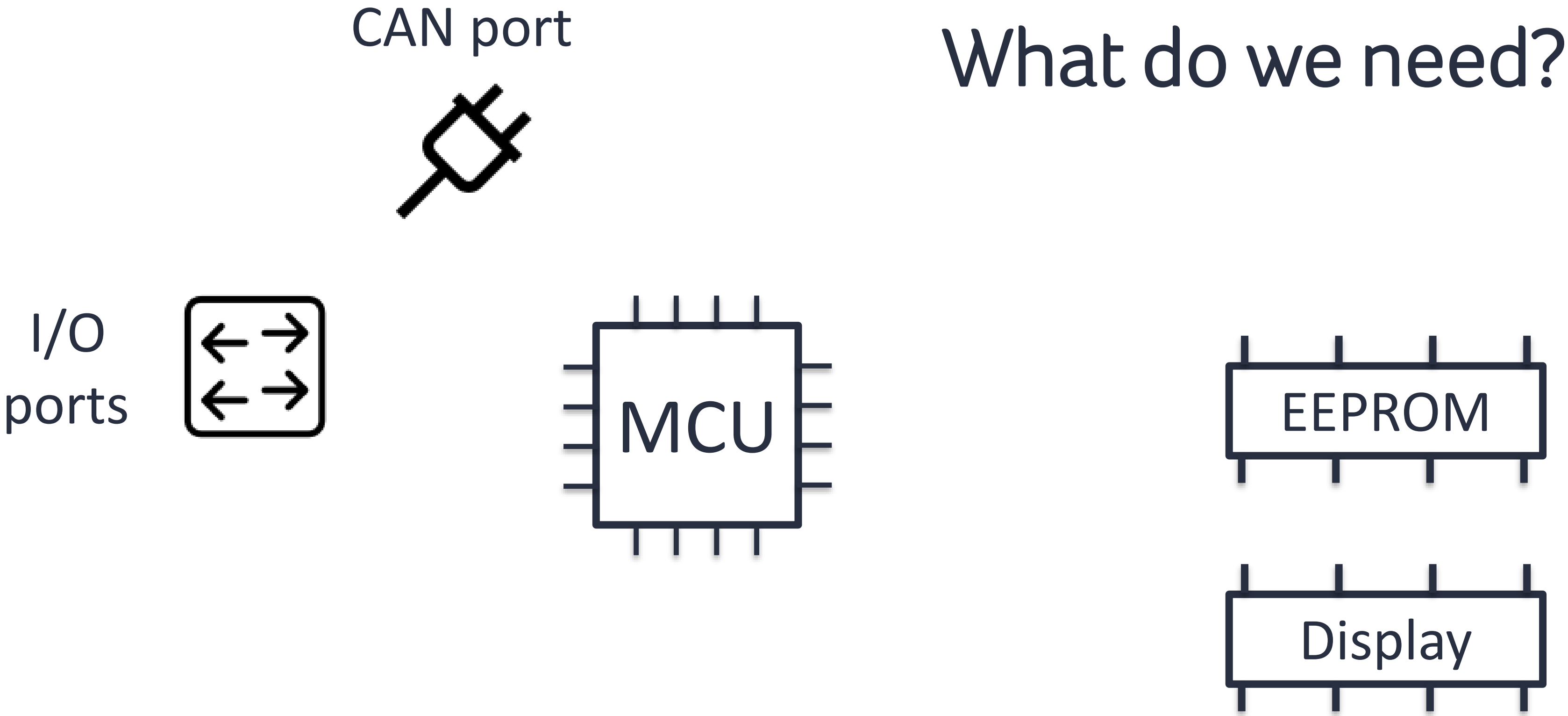
# Tools?

Badly ☹
^
$ARCH is supported by IDA Pro **and** radare2

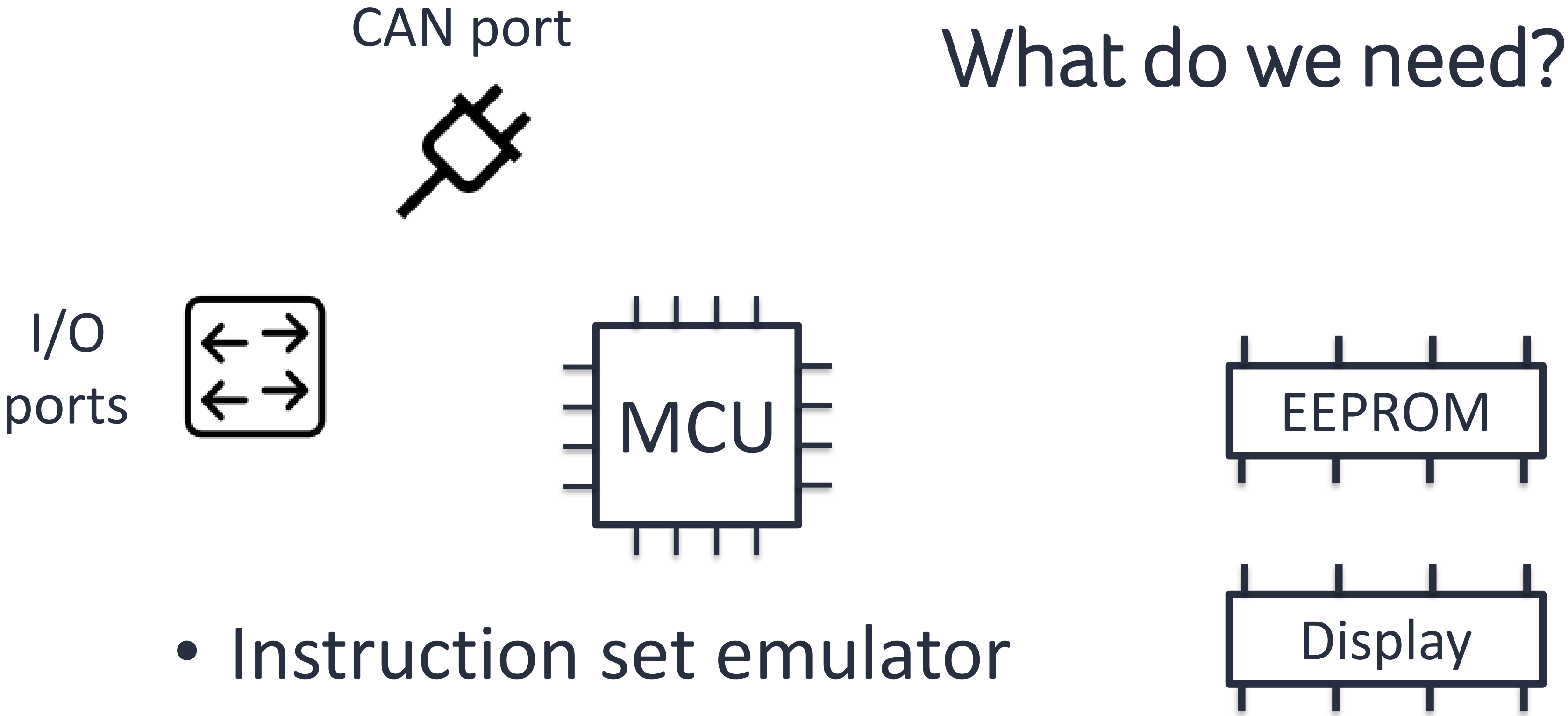
$ARCH is **not** supported by qemu

# *No tools? Let’s make some tools!*

# What do we need?

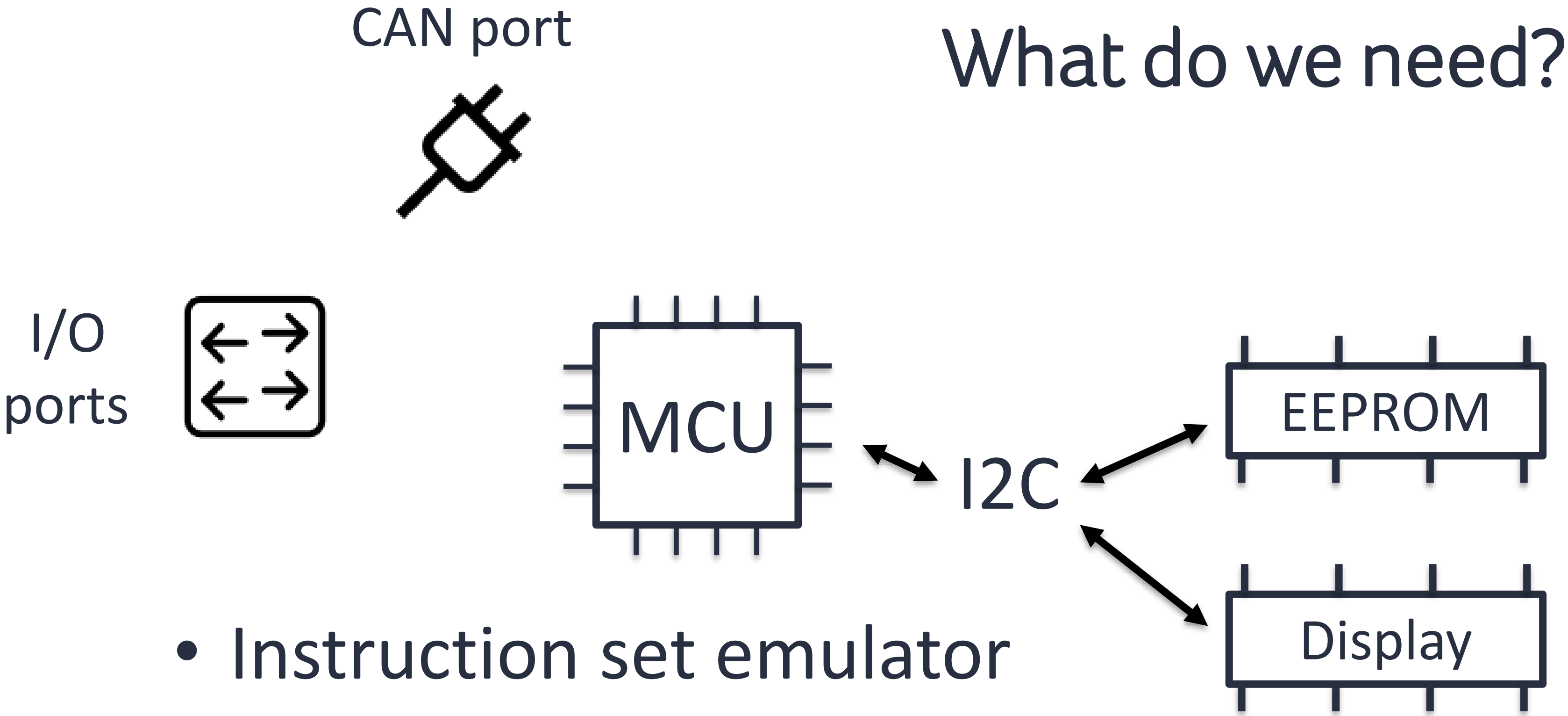


# What do we need?



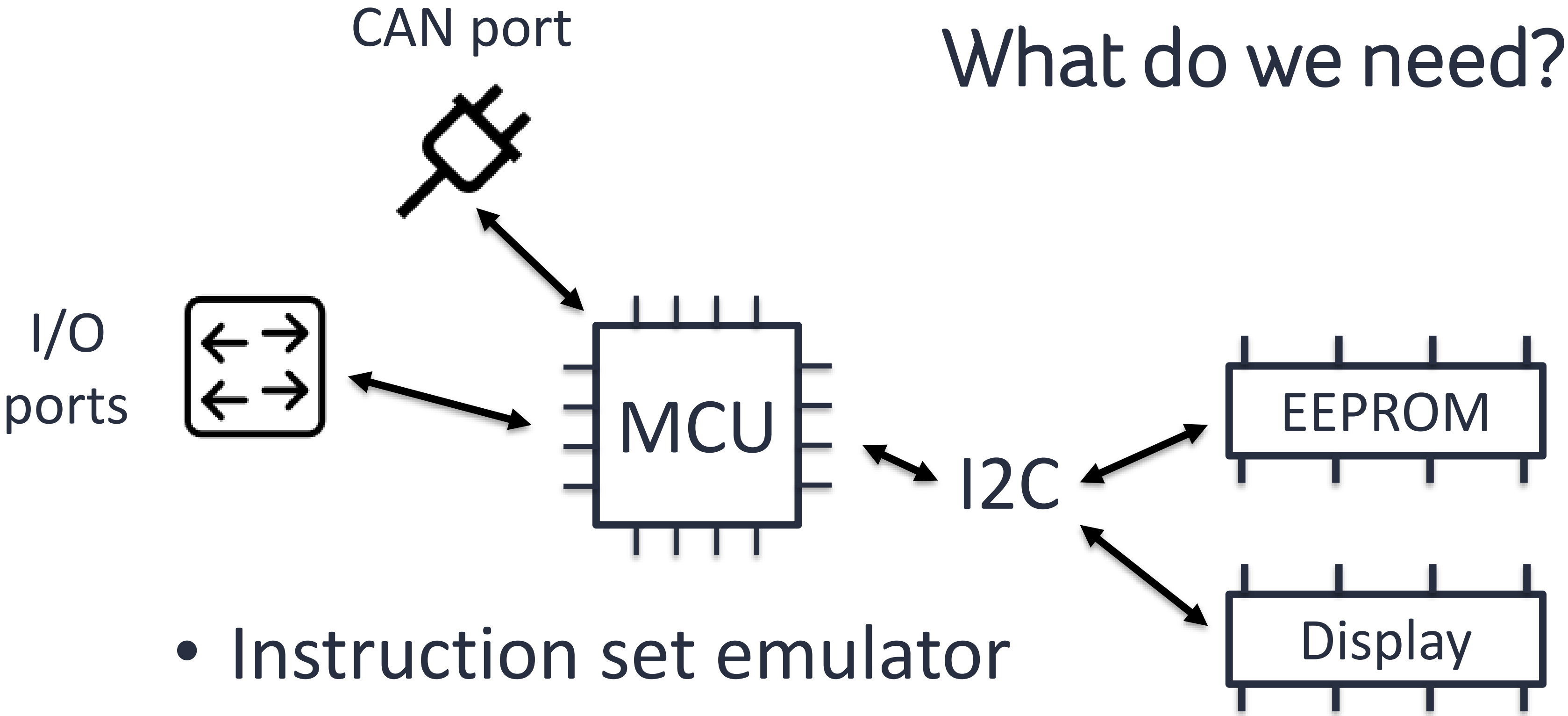
- Instruction set emulator
- Timers, interrupts, ...

# What do we need?



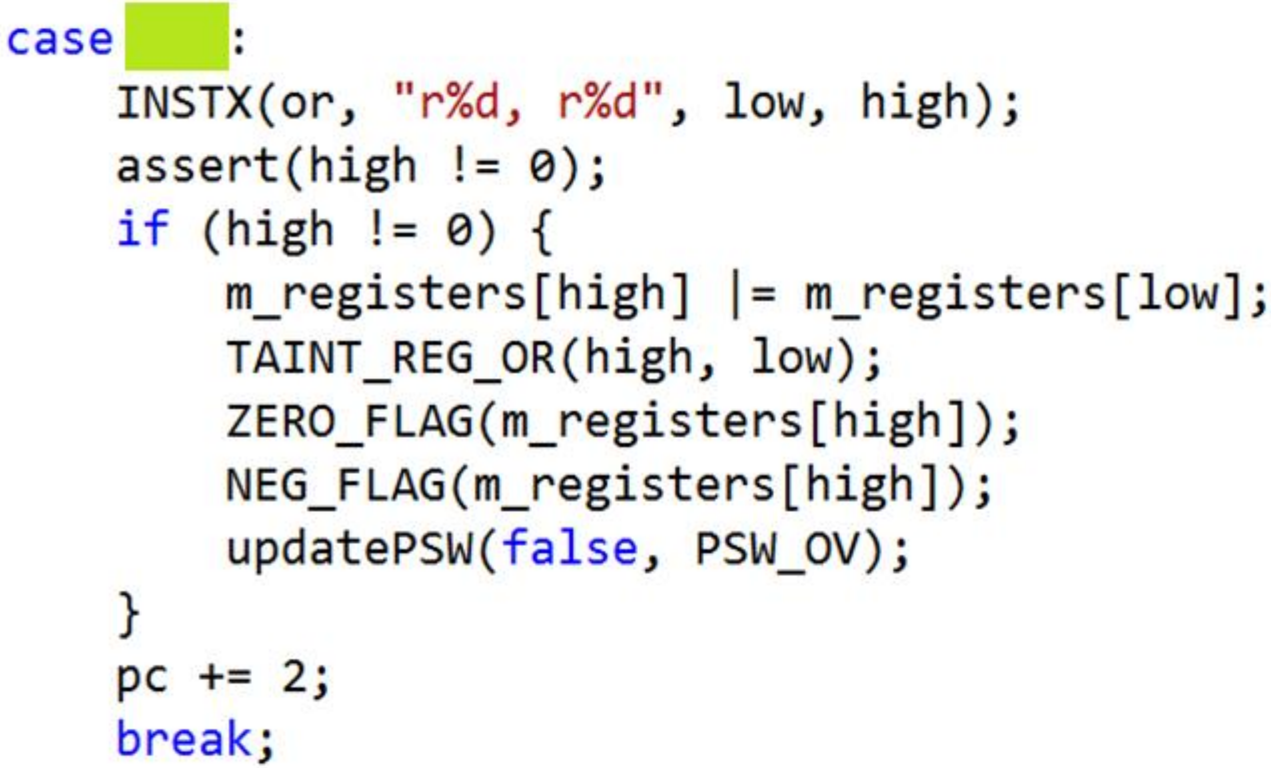
- Instruction set emulator
- Timers, interrupts, …

# What do we need?



- Instruction set emulator
- Timers, interrupts, …

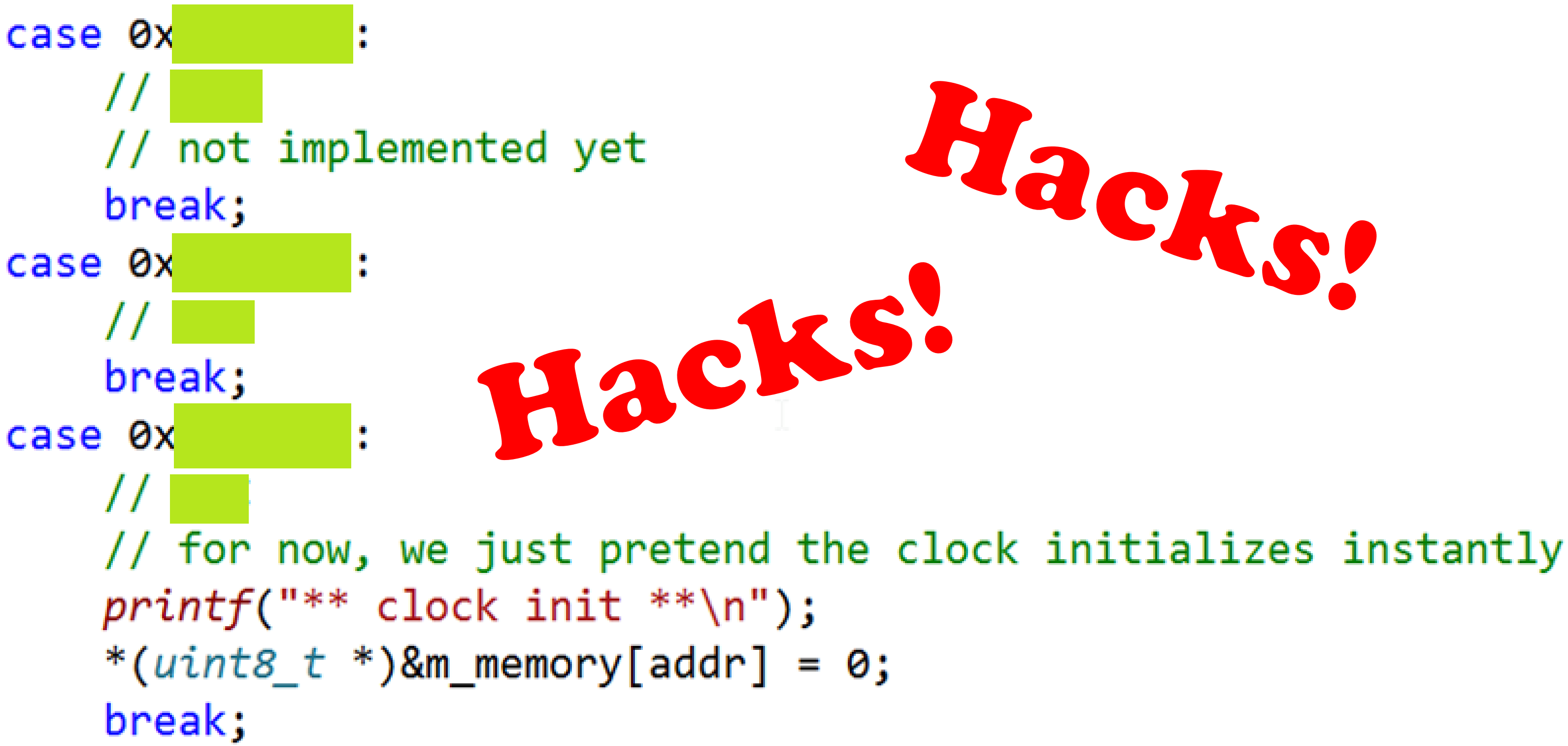
# Emulating the CPU architecture



```
case [redacted]:
    INSTX(or, "r%d, r%d", low, high);
    assert(high != 0);
    if (high != 0) {
        m_registers[high] |= m_registers[low];
        TAINT_REG_OR(high, low);
        ZERO_FLAG(m_registers[high]);
        NEG_FLAG(m_registers[high]);
        updatePSW(false, PSW_OV);
    }
    pc += 2;
    break;
```

# “Implementing” peripherals

# “Implementing” peripherals



```
case 0x[redacted]:
    // [redacted]
    // not implemented yet
    break;
case 0x[redacted]:
    // [redacted]
    break;
case 0x[redacted]:
    // [redacted]
    // for now, we just pretend the clock initializes instantly
    printf("** clock init **\n");
    *(uint8_t *)&m_memory[addr] = 0;
    break;
```

Hacks!

Hacks!

# How difficult was it?

# How difficult was it?

~ **1 (sleepless) week** of work

(for a hacker experienced in writing emulators)

# How difficult was it?

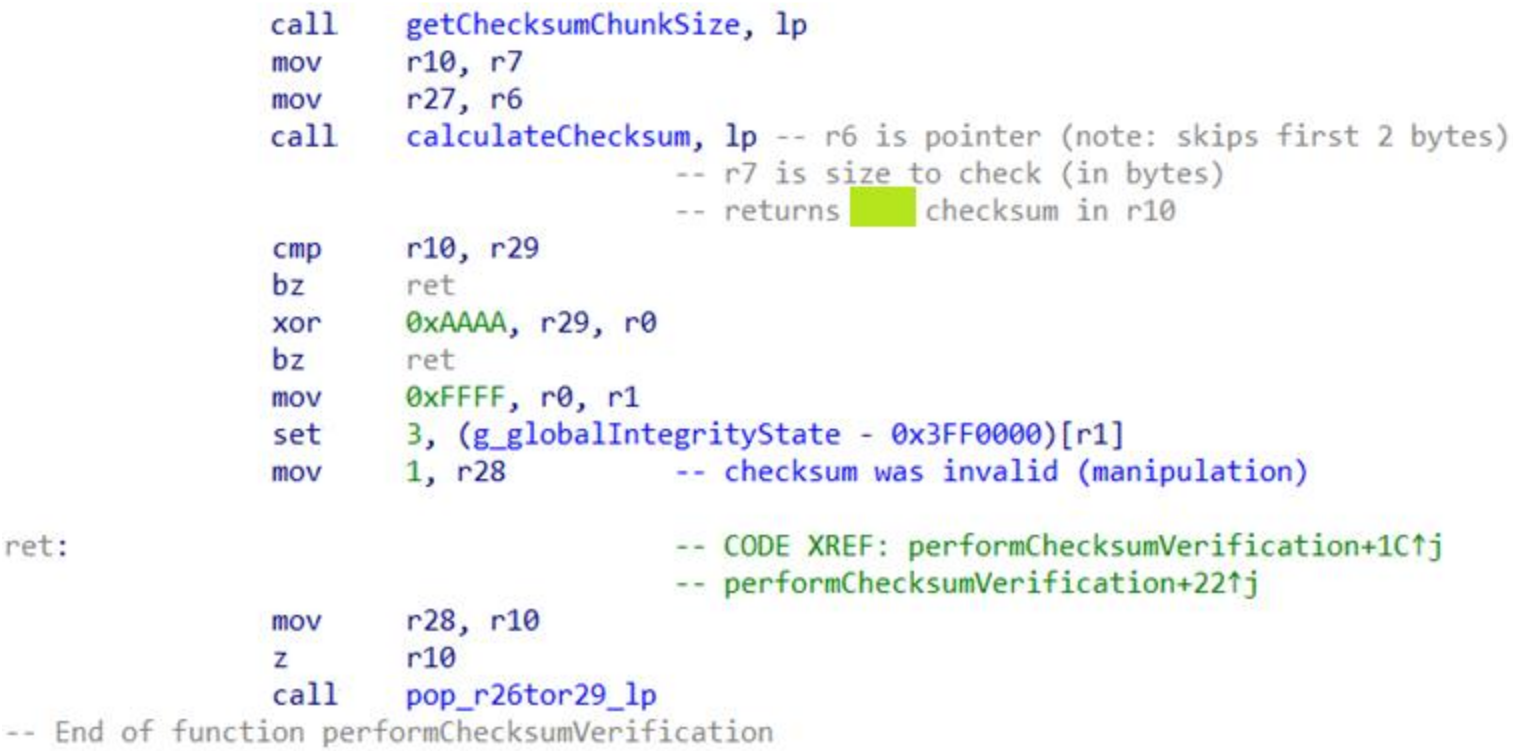
~ **1 (sleepless) week** of work
(for a hacker experienced in writing emulators)

~ **3000 lines** of (terrible) code
(excluding support tooling)

# Why write an emulator?

- Debugging (e.g. GDB stub)
- SocketCAN
- **Execution tracing**
- **Taint tracking**

# Execution tracing



```
                call    getChecksumChunkSize, lp
                mov     r10, r7
                mov     r27, r6
                call    calculateChecksum, lp -- r6 is pointer (note: skips first 2 bytes)
                                          -- r7 is size to check (in bytes)
                                          -- returns [redacted] checksum in r10
                cmp     r10, r29
                bz      ret
                xor     0xAAAA, r29, r0
                bz      ret
                mov     0xFFFF, r0, r1
                set     3, (g_globalIntegrityState - 0x3FF0000)[r1]
                mov     1, r28            -- checksum was invalid (manipulation)

ret:                                      -- CODE XREF: performChecksumVerification+1C↑j
                                          -- performChecksumVerification+22↑j
                mov     r28, r10
                z       r10
                call    pop_r26tor29_lp
-- End of function performChecksumVerification
```

# Execution tracing

```
0x02920
0x02922 (jump)
0x02926
0x02928
0x0292c
0x02930
```

# Execution tracing

**0x02920**

**0x02922 (jump)**

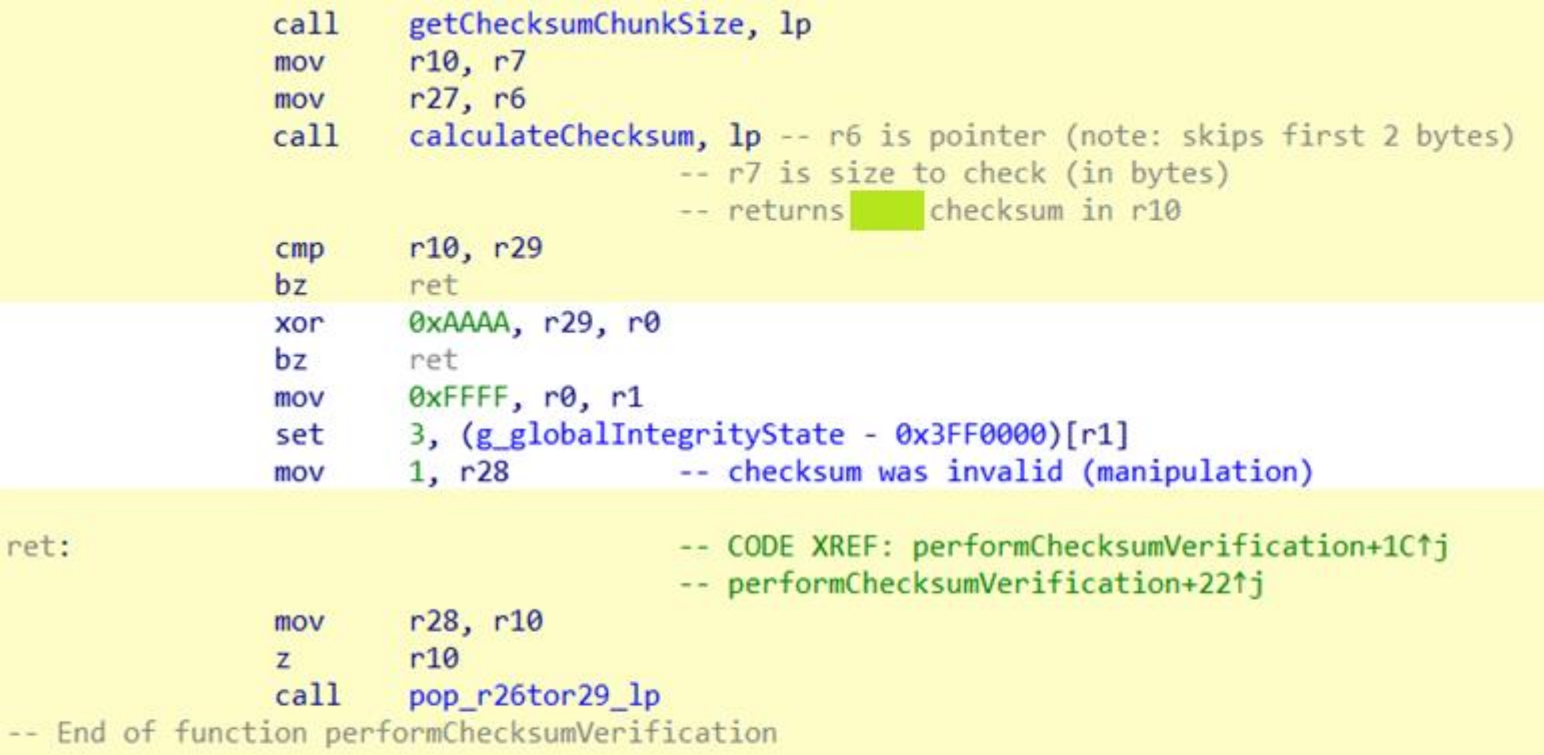
0x02926

0x02928

0x0292c

**0x02930**

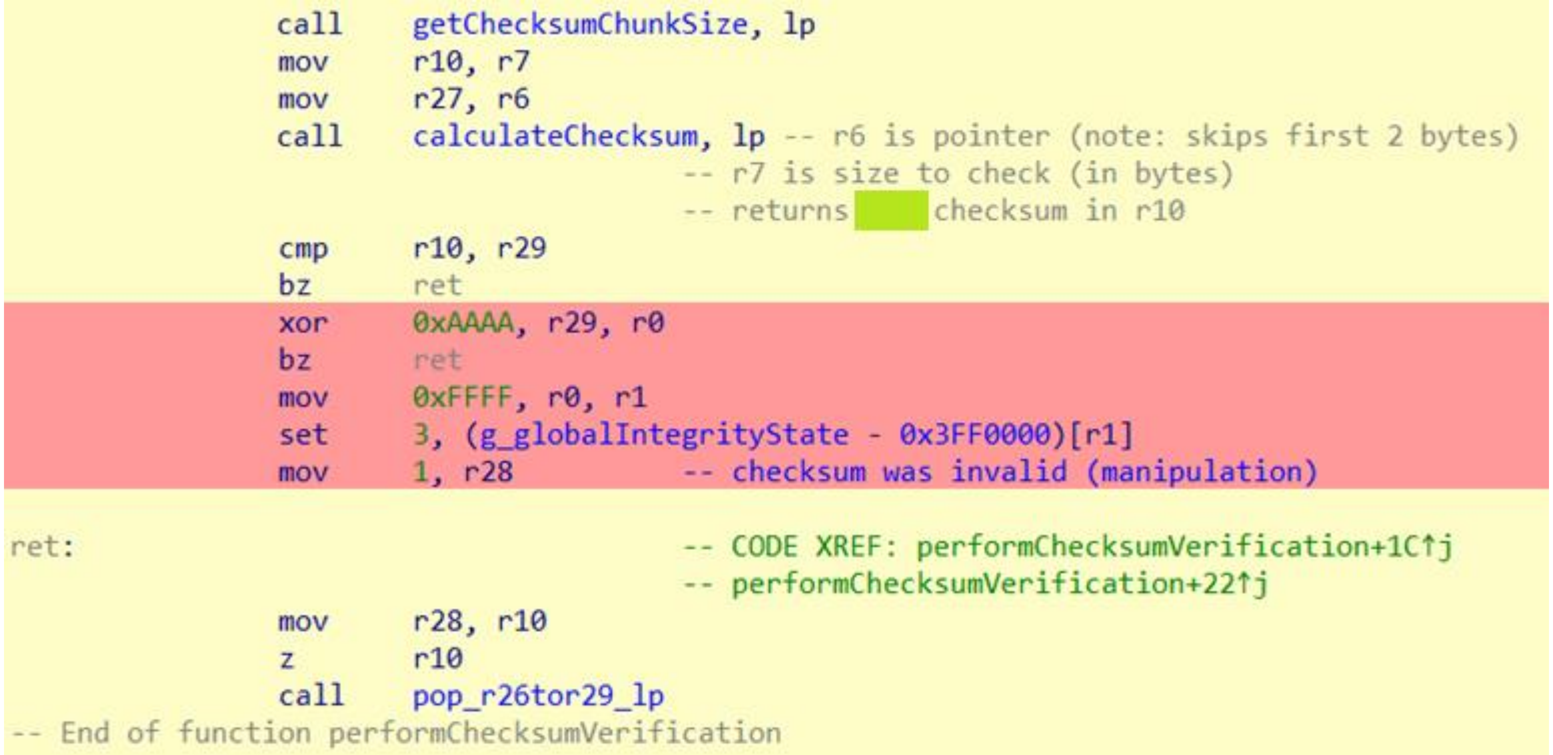
# Execution tracing



```
              call    getChecksumChunkSize, lp
              mov     r10, r7
              mov     r27, r6
              call    calculateChecksum, lp -- r6 is pointer (note: skips first 2 bytes)
                                         -- r7 is size to check (in bytes)
                                         -- returns [redacted] checksum in r10
              cmp     r10, r29
              bz      ret
              xor     0xAAAA, r29, r0
              bz      ret
              mov     0xFFFF, r0, r1
              set     3, (g_globalIntegrityState - 0x3FF0000)[r1]
              mov     1, r28             -- checksum was invalid (manipulation)

ret:                                     -- CODE XREF: performChecksumVerification+1C↑j
                                         -- performChecksumVerification+22↑j
              mov     r28, r10
              z       r10
              call    pop_r26tor29_lp
-- End of function performChecksumVerification
```

# Execution tracing



```
                call     getChecksumChunkSize, lp
                mov      r10, r7
                mov      r27, r6
                call     calculateChecksum, lp -- r6 is pointer (note: skips first 2 bytes)
                                               -- r7 is size to check (in bytes)
                                               -- returns [redacted] checksum in r10
                cmp      r10, r29
                bz       ret
                xor      0xAAAA, r29, r0
                bz       ret
                mov      0xFFFF, r0, r1
                set      3, (g_globalIntegrityState - 0x3FF0000)[r1]
                mov      1, r28                -- checksum was invalid (manipulation)

ret:                                           -- CODE XREF: performChecksumVerification+1C↑j
                                               -- performChecksumVerification+22↑j
                mov      r28, r10
                z        r10
                call     pop_r26tor29_lp
-- End of function performChecksumVerification
```

# Taint tracking

| | |
|---|---|
| **1** | **??** |
| 2 | ?? |
| 3 | ?? |
| 4 | ?? |
| 5 | ?? |
| 6 | ?? |
| 7 | ?? |
| 8 | ?? |

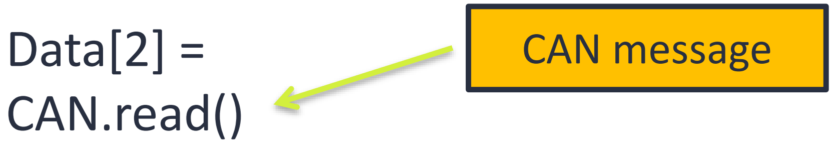
# Taint tracking

| | |
|---|---|
| **1** | **??** |
| 2 | ?? |
| 3 | ?? |
| 4 | ?? |
| 5 | ?? |
| 6 | ?? |
| 7 | ?? |
| 8 | ?? |

CAN message

# Taint tracking

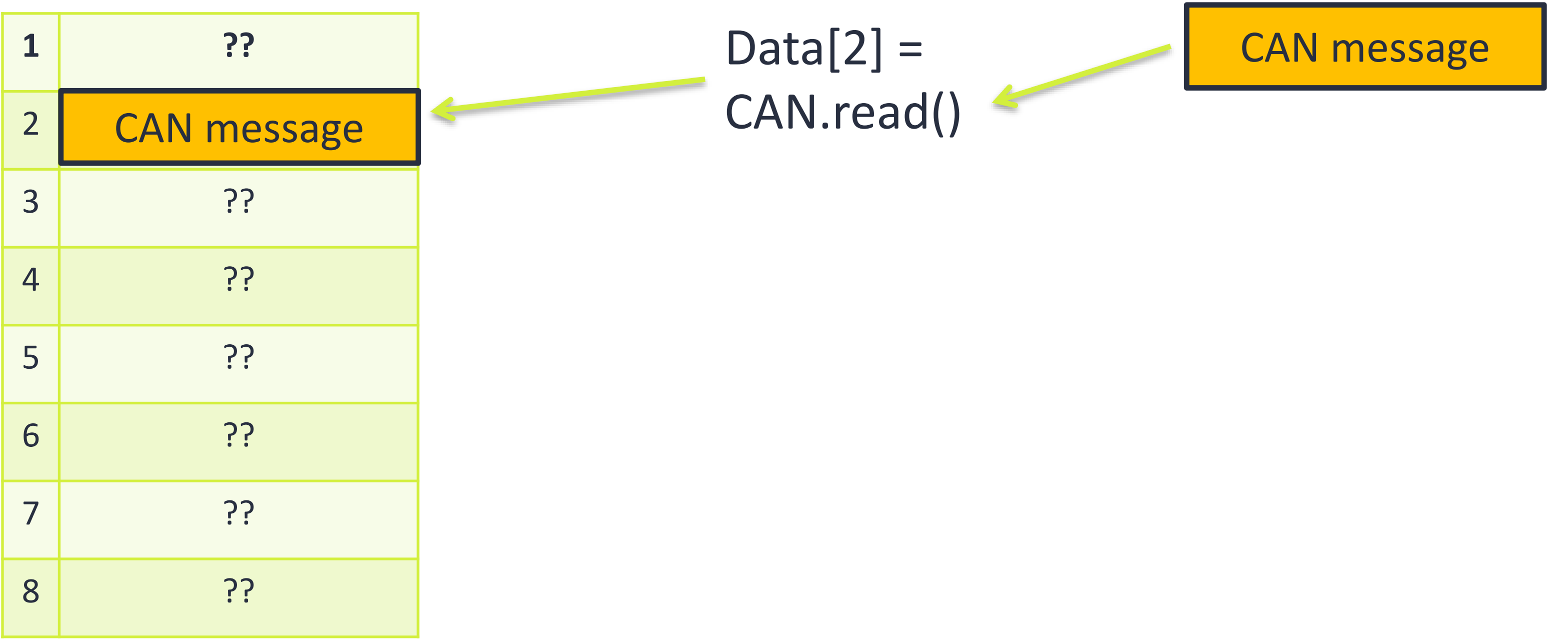
| | |
|---|---|
| **1** | **??** |
| 2 | ?? |
| 3 | ?? |
| 4 | ?? |
| 5 | ?? |
| 6 | ?? |
| 7 | ?? |
| 8 | ?? |



Data[2] = CAN.read()

CAN message

# Taint tracking

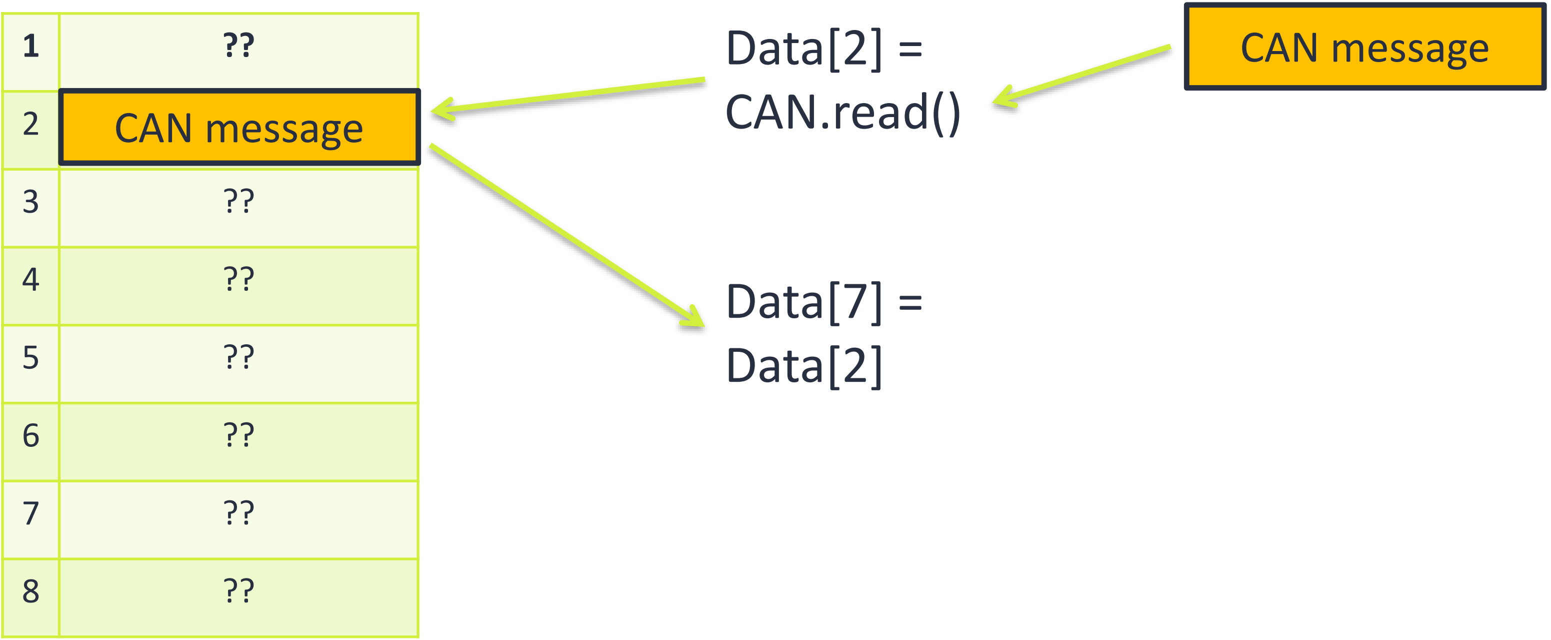


| | |
|---|---|
| 1 | **??** |
| 2 | CAN message |
| 3 | ?? |
| 4 | ?? |
| 5 | ?? |
| 6 | ?? |
| 7 | ?? |
| 8 | ?? |

Data[2] =
CAN.read()

CAN message

# Taint tracking



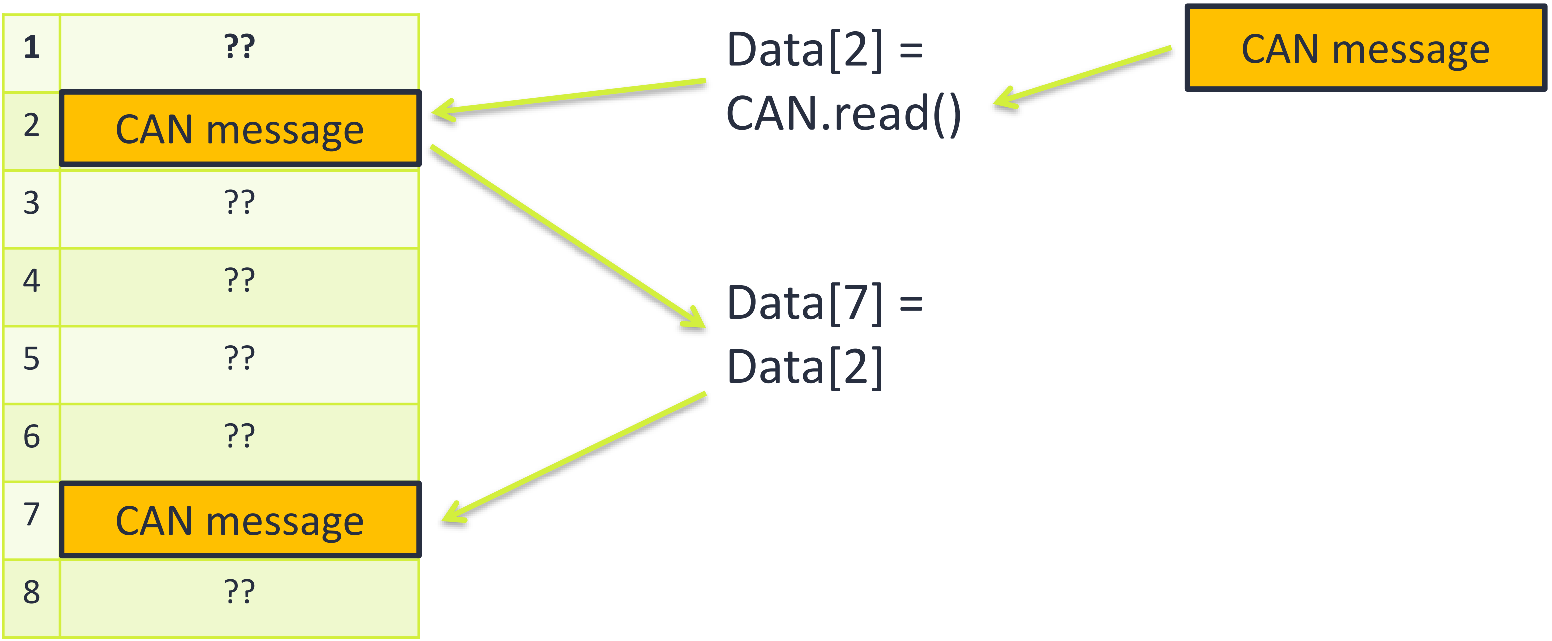
| | |
|---|---|
| 1 | ?? |
| 2 | CAN message |
| 3 | ?? |
| 4 | ?? |
| 5 | ?? |
| 6 | ?? |
| 7 | ?? |
| 8 | ?? |

```
Data[2] =
CAN.read()
```

CAN message

```
Data[7] =
Data[2]
```

# Taint tracking



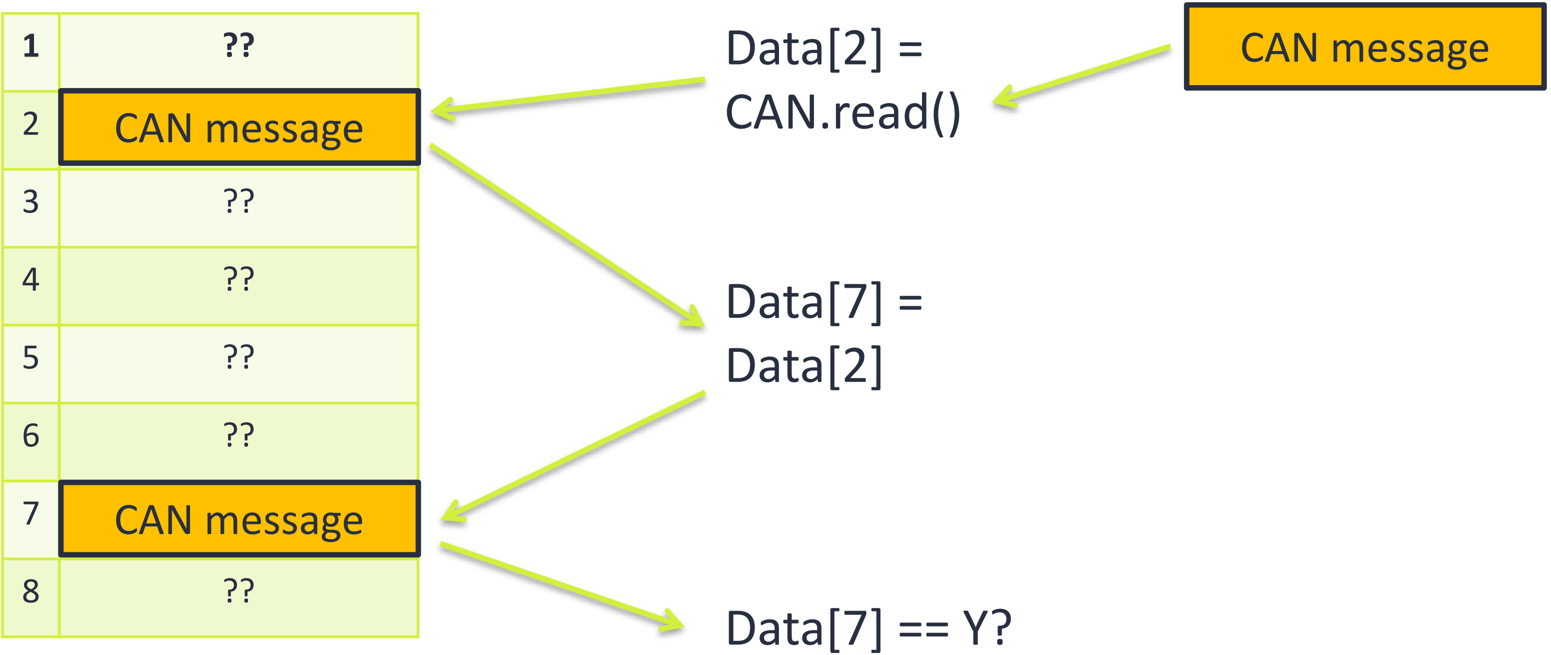
| | |
|---|---|
| 1 | ?? |
| 2 | CAN message |
| 3 | ?? |
| 4 | ?? |
| 5 | ?? |
| 6 | ?? |
| 7 | CAN message |
| 8 | ?? |

```
Data[2] =
CAN.read()
```

CAN message

```
Data[7] =
Data[2]
```

# Taint tracking



| | |
|---|---|
| 1 | ?? |
| 2 | CAN message |
| 3 | ?? |
| 4 | ?? |
| 5 | ?? |
| 6 | ?? |
| 7 | CAN message |
| 8 | ?? |

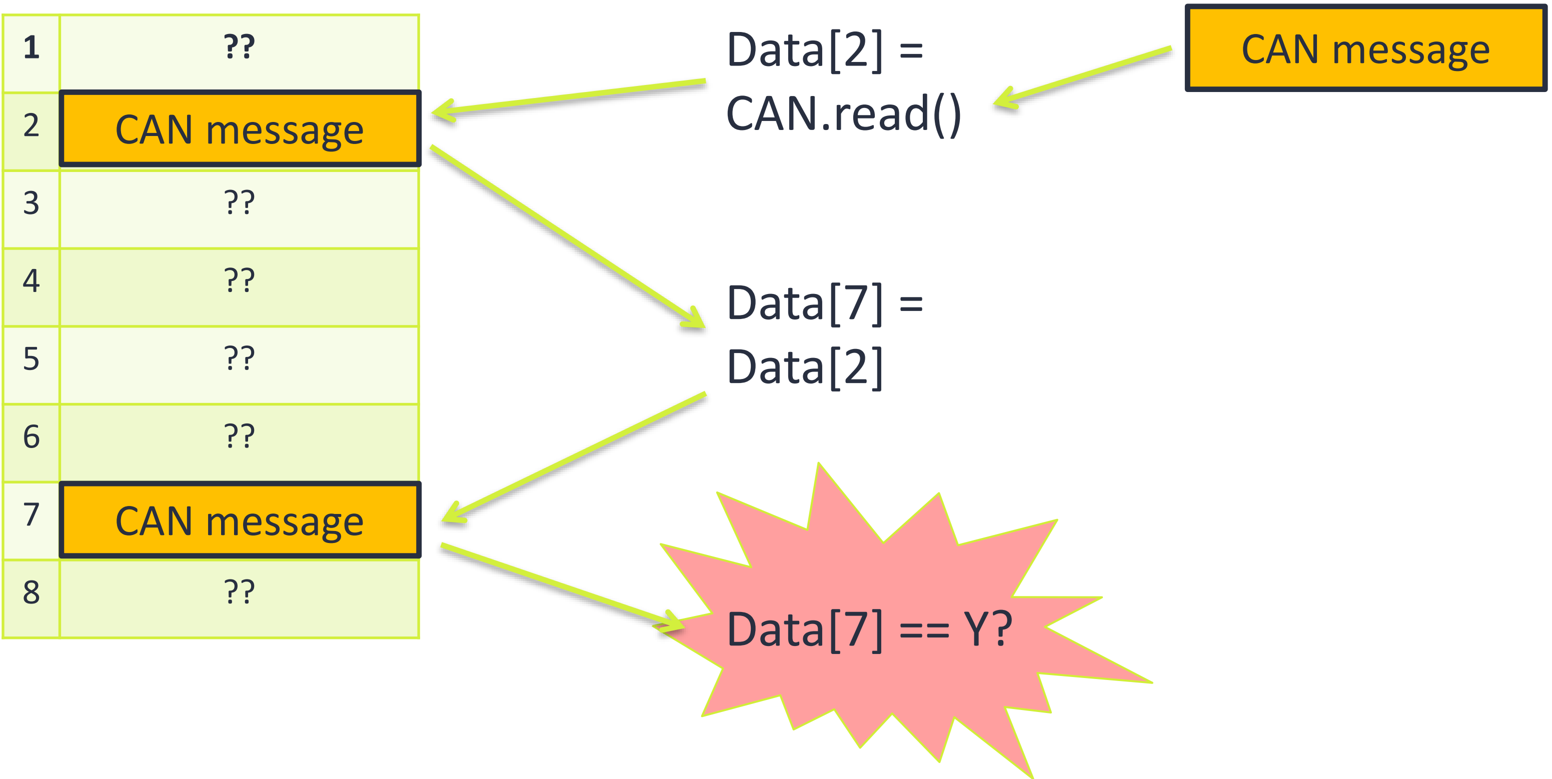
CAN message

Data[2] =
CAN.read()

Data[7] =
Data[2]

Data[7] == Y?

# Taint tracking



| | |
|---|---|
| 1 | ?? |
| 2 | CAN message |
| 3 | ?? |
| 4 | ?? |
| 5 | ?? |
| 6 | ?? |
| 7 | CAN message |
| 8 | ?? |

Data[2] = CAN.read()

CAN message

Data[7] = Data[2]

Data[7] == Y?

# Taint tracking UDS Security Access

| | |
|---|---|
| **1** | **??** |
| 2 | ?? |
| 3 | ?? |
| 4 | ?? |
| 5 | ?? |
| 6 | ?? |
| 7 | ?? |
| 8 | ?? |

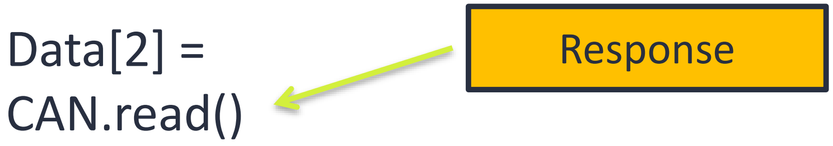
# Taint tracking UDS Security Access

| | |
|---|---|
| **1** | **??** |
| 2 | ?? |
| 3 | ?? |
| 4 | ?? |
| 5 | ?? |
| 6 | ?? |
| 7 | ?? |
| 8 | ?? |

Response

# Taint tracking UDS Security Access

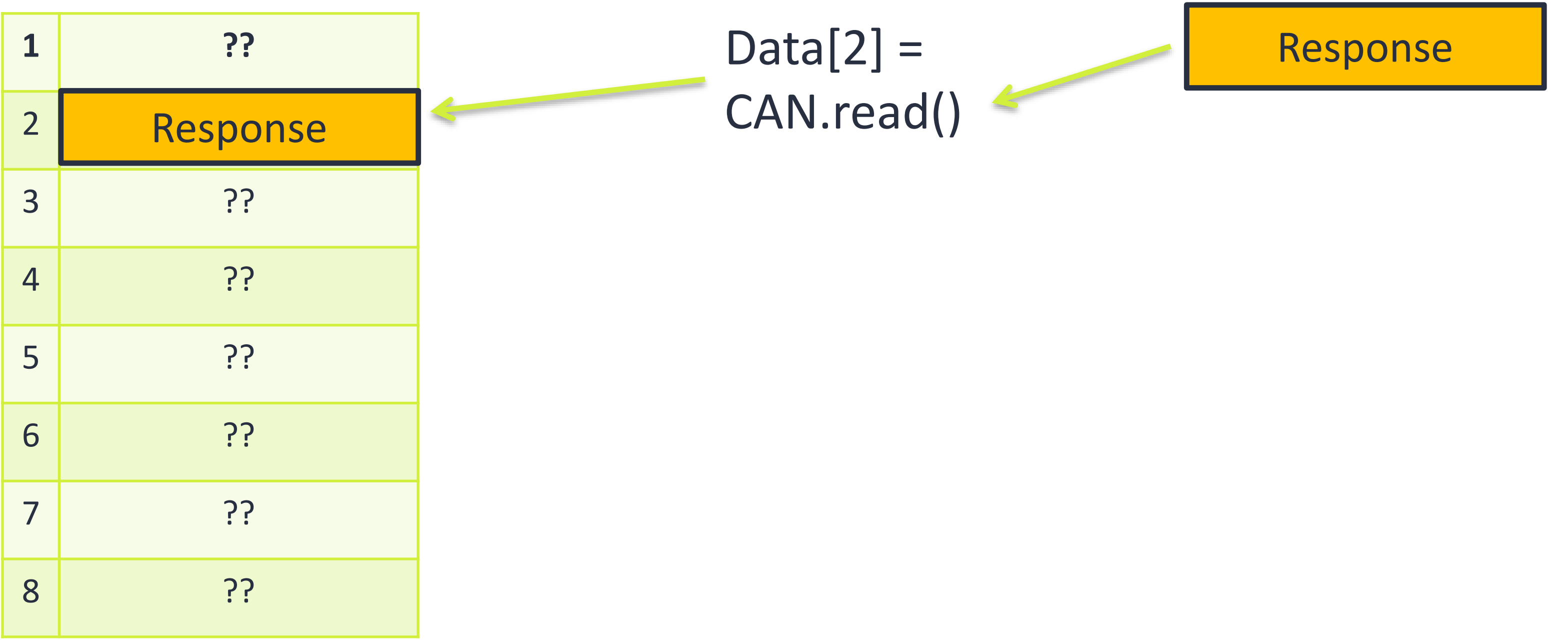
| | |
|---|---|
| 1 | ?? |
| 2 | ?? |
| 3 | ?? |
| 4 | ?? |
| 5 | ?? |
| 6 | ?? |
| 7 | ?? |
| 8 | ?? |



Data[2] =
CAN.read()

Response

# Taint tracking UDS Security Access

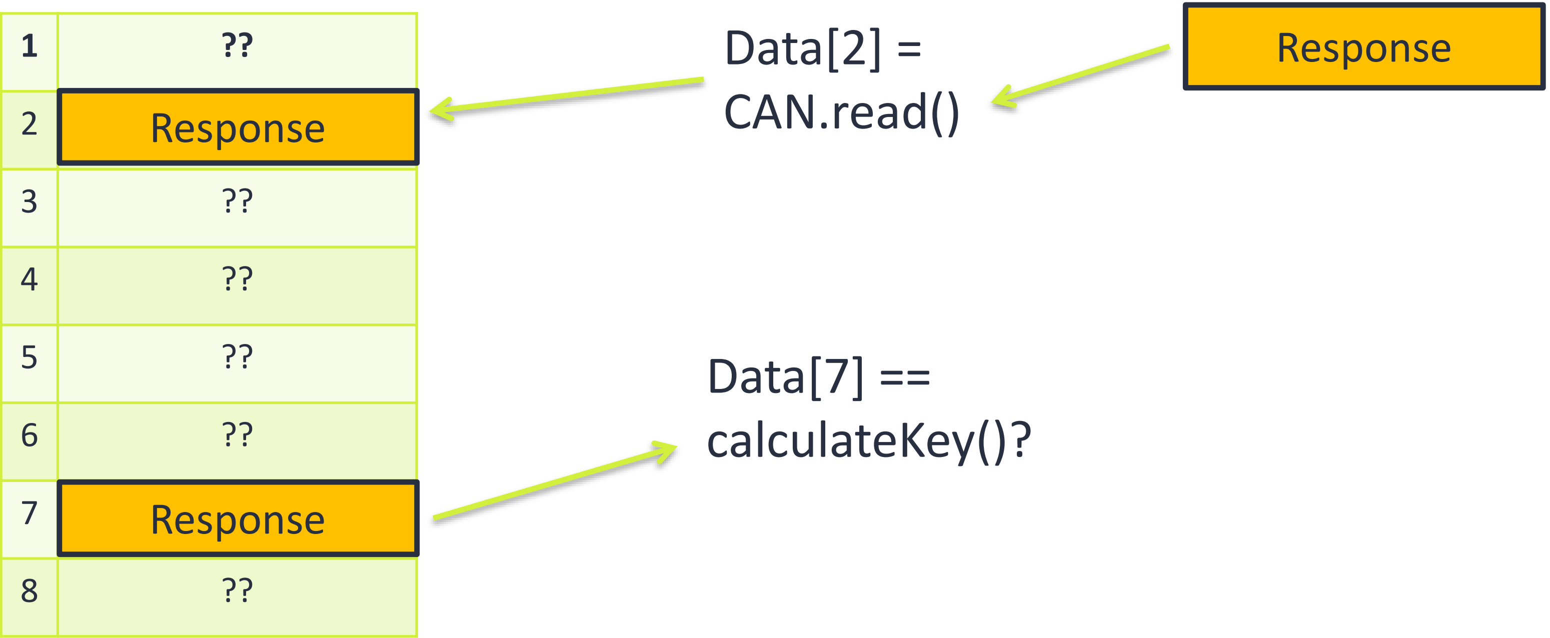


| | |
|---|---|
| 1 | ?? |
| 2 | Response |
| 3 | ?? |
| 4 | ?? |
| 5 | ?? |
| 6 | ?? |
| 7 | ?? |
| 8 | ?? |

Data[2] =
CAN.read()

Response

# Taint tracking UDS Security Access

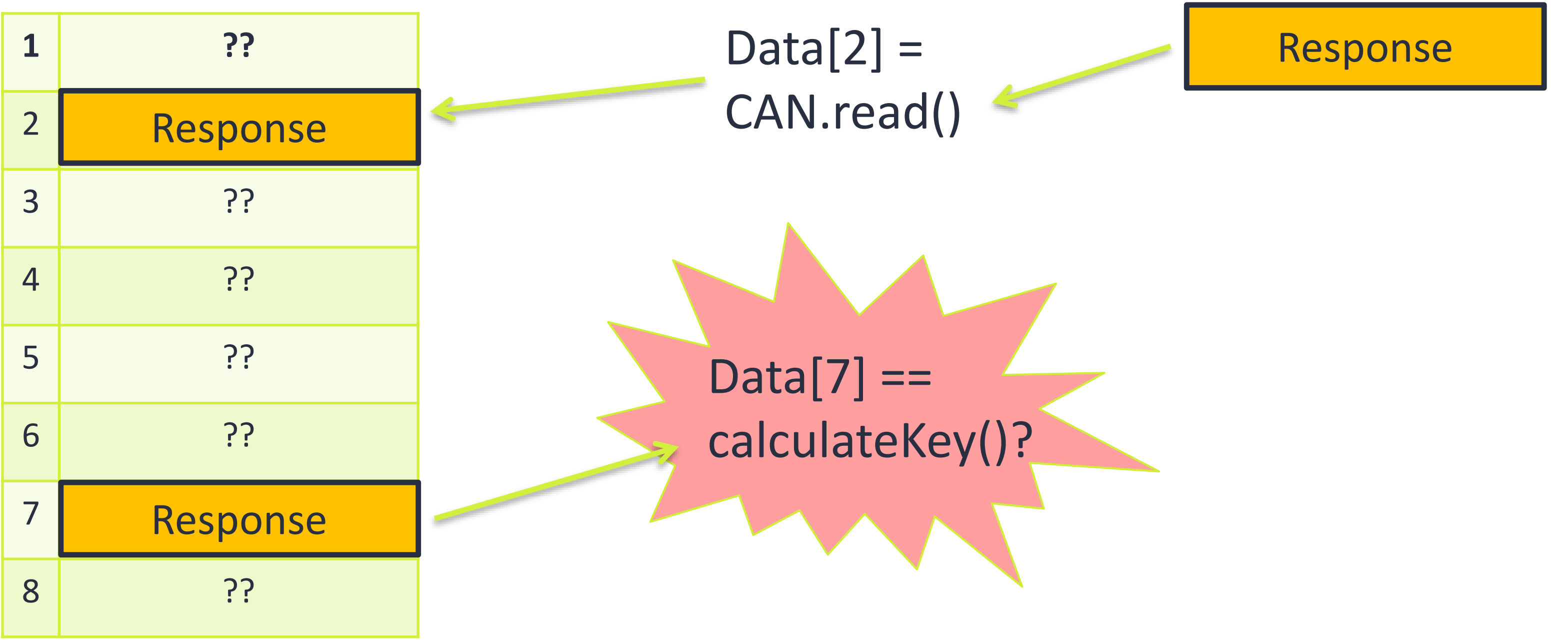


| | |
|---|---|
| 1 | ?? |
| 2 | Response |
| 3 | ?? |
| 4 | ?? |
| 5 | ?? |
| 6 | ?? |
| 7 | Response |
| 8 | ?? |

Data[2] =
CAN.read()

Data[7] ==
calculateKey()?

# Taint tracking UDS Security Access



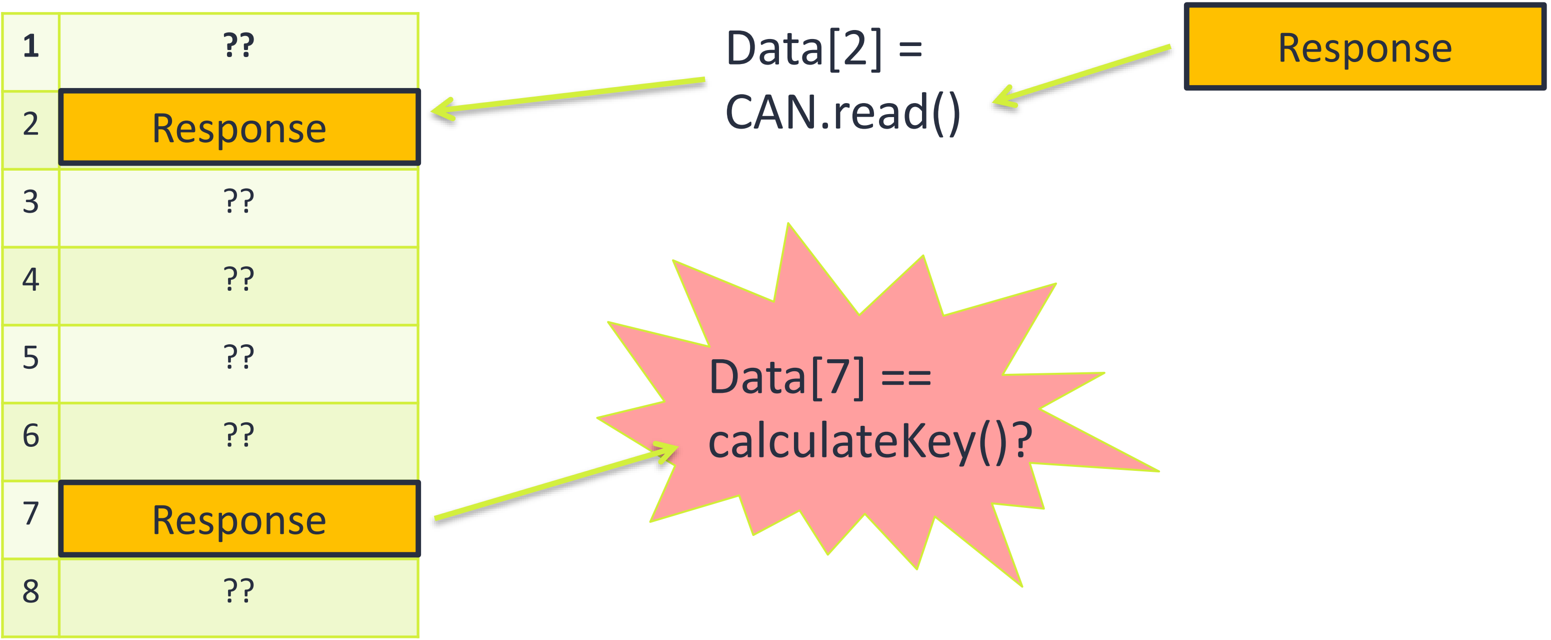
| | |
|---|---|
| 1 | ?? |
| 2 | Response |
| 3 | ?? |
| 4 | ?? |
| 5 | ?? |
| 6 | ?? |
| 7 | Response |
| 8 | ?? |

Data[2] =
CAN.read()

Response

Data[7] ==
calculateKey()?

# Taint tracking UDS Security Access



|   |   |
|---|---|
| 1 | ?? |
| 2 | Response |
| 3 | ?? |
| 4 | ?? |
| 5 | ?? |
| 6 | ?? |
| 7 | Response |
| 8 | ?? |

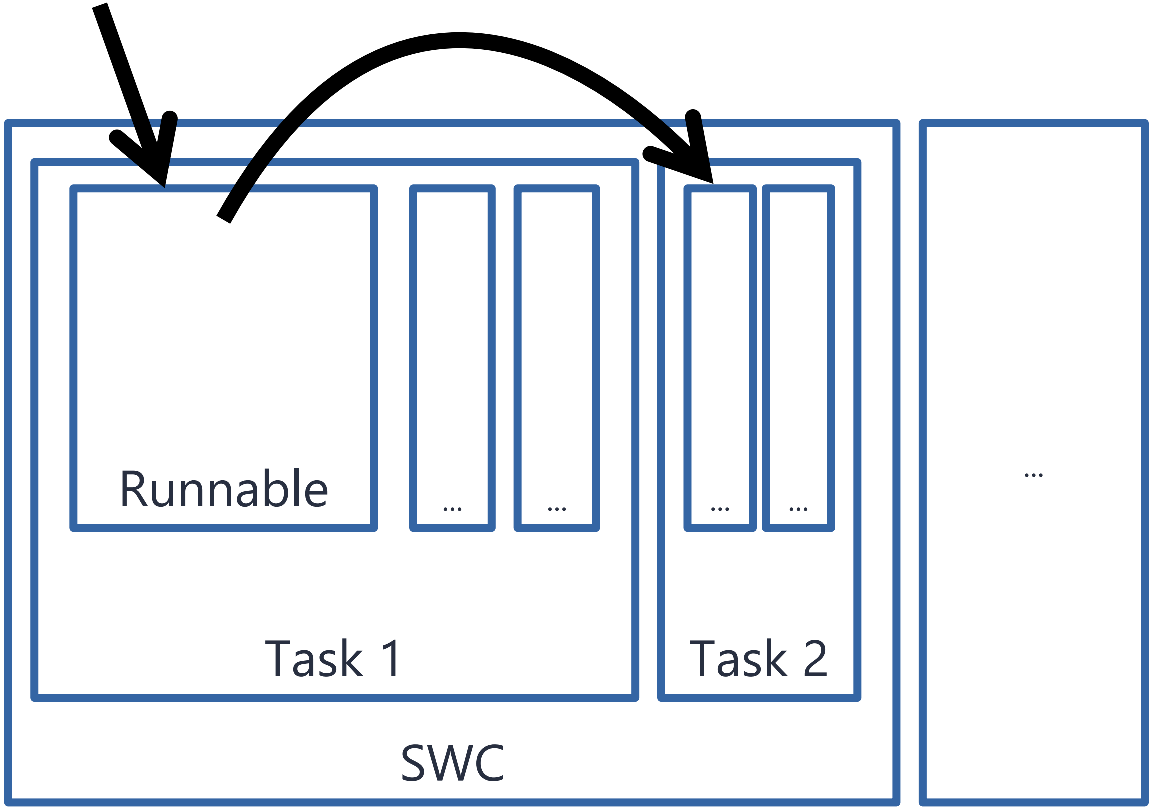
Data[2] =
CAN.read()

Response

Data[7] ==
calculateKey()?

We found the *calculateKey* function!

# Demo Time!!!

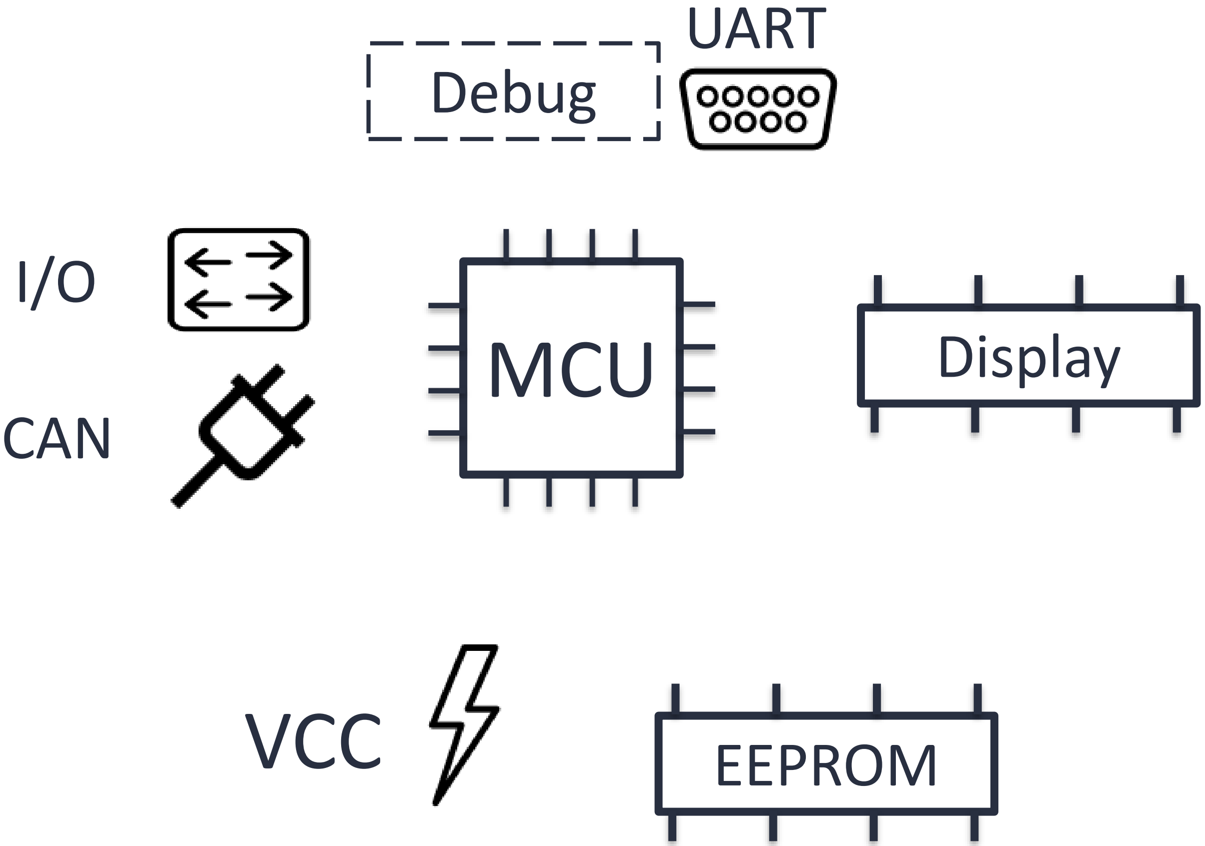


# Wrap up!

- Hardware will betray you!
- Emulating a dashboard is not too tricky?
- Fault injection attacks on UDS are *cool.*

This Fault Injection attack on UDS is **not efficient**!

We can do better...



UART

Debug

I/O

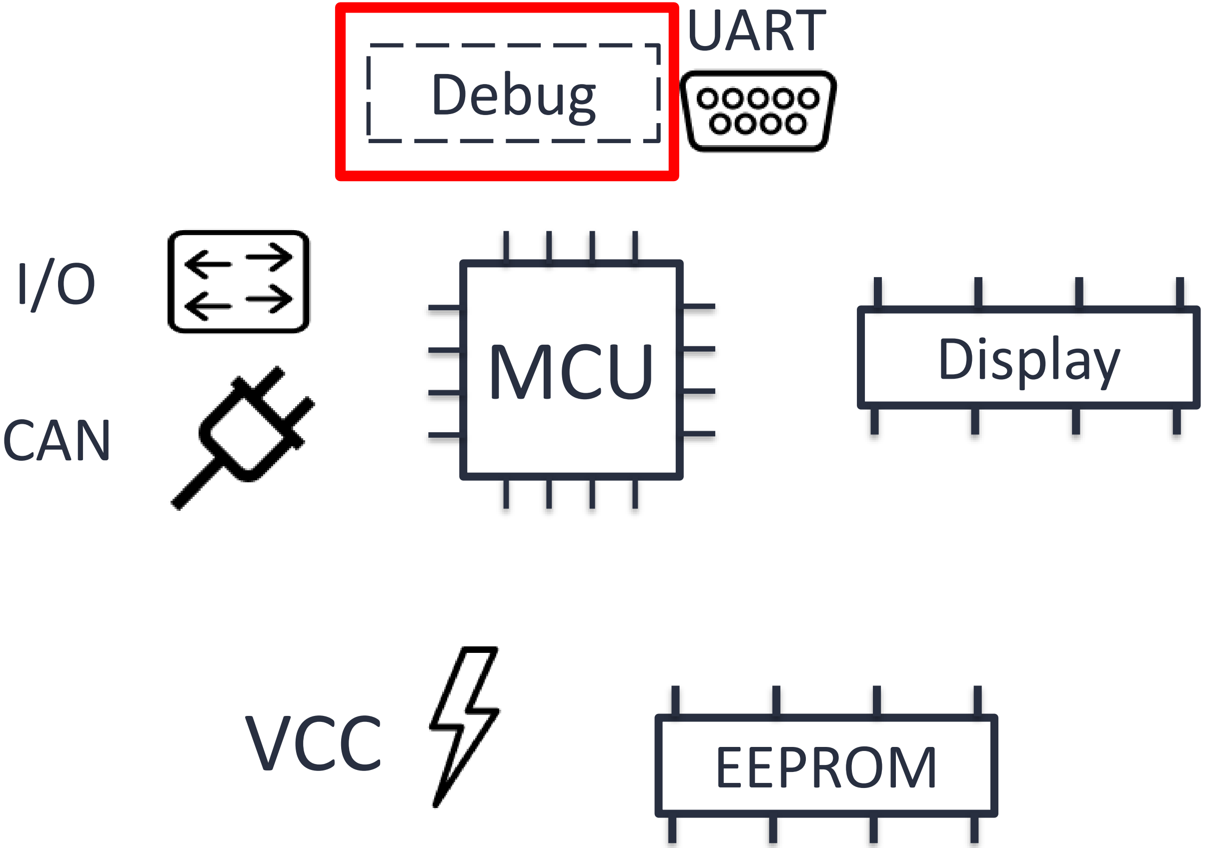
MCU

Display

CAN

VCC

EEPROM


Debug
UART
I/O
CAN
MCU
Display
VCC
EEPROM

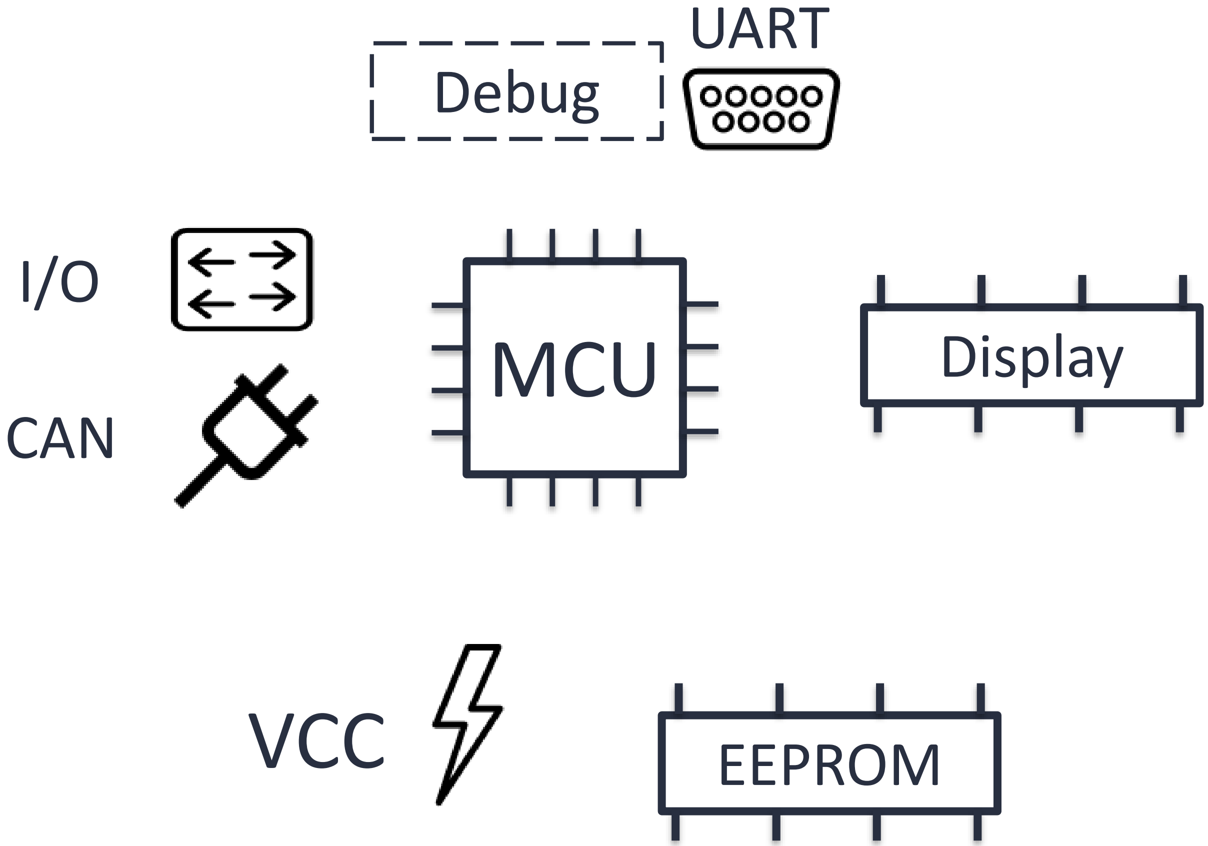
# Debug interfaces

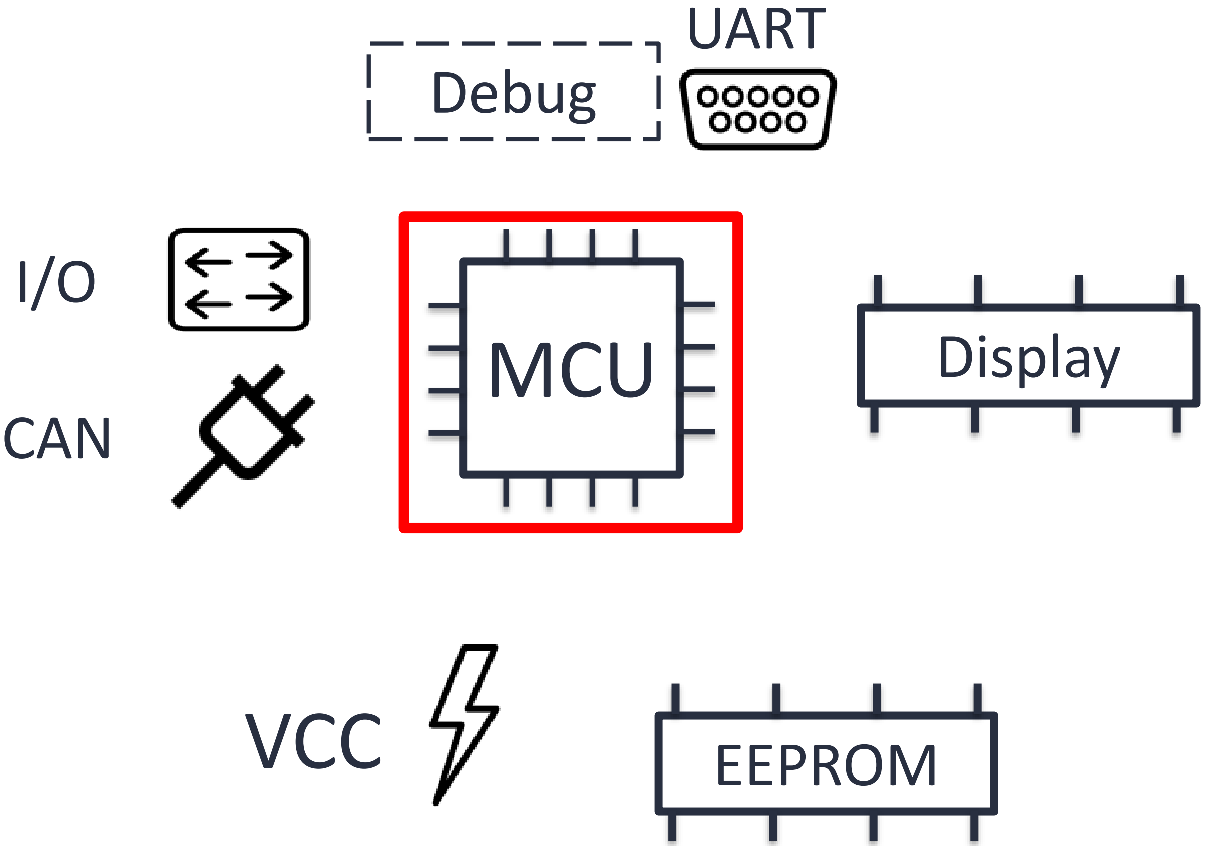
```
if (allow_debug())

{
  open_JTAG();
}
```

# Debug interfaces

```
if (all        bug())
{
  open_JTAG();
}
```


Debug
UART
I/O
CAN
MCU
Display
VCC
EEPROM



UART

Debug

I/O

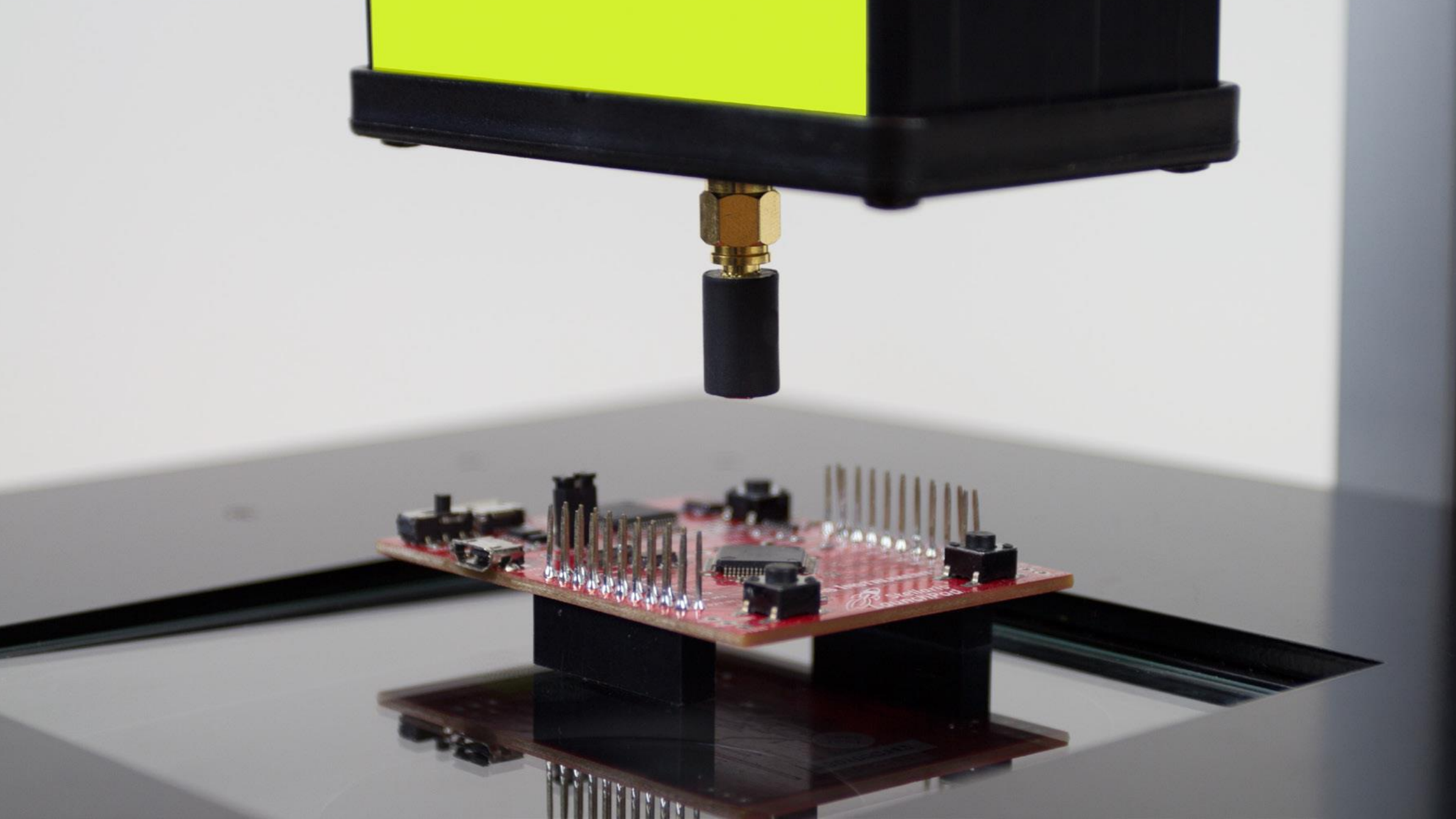
CAN

MCU

Display

VCC

EEPROM


Stellaris
LaunchPad

# Electromagnetic Fault Injection

ARM
FIRE
TO REARM YOU MUST TOGGLE ARM SWITCH ON/OFF CHIPSHOUTER AUTOMATICALLY DISARMS

[**ChipSHOUTER®**]

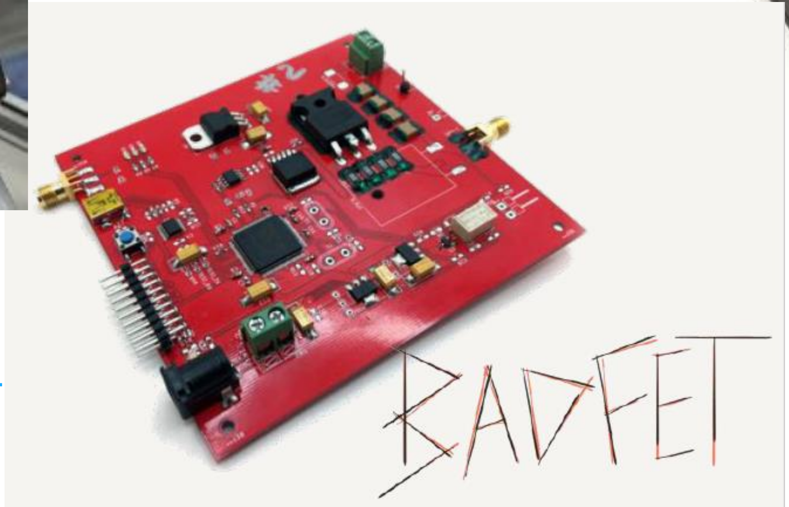
[**Inspector FI**]

# Electromagnetic Fault Injection

Cheap and awesome:

[BADFET](#)

[ChipSHOUTER®](#)



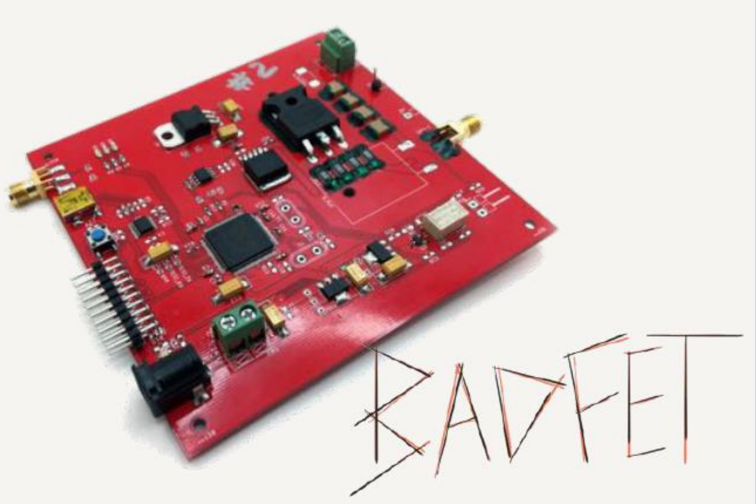
[Inspector FI](#)

# Electromagnetic Fault Injection

ARM
FIRE
TO REARM YOU MUST TOGGLE ARM SWITCH ON/OFF CHIPSHOUTER AUTOMATICALLY DISARMS

Cheap and awesome:

BADFET



ChipSHOUTER®

Inspector FI

***Electromagnetic fault injection available to the masses!***

# Fault Injection is universal

*all devices*

*all CPUs/MCUs*

*all software*

# Can we harden ECUs against fault injection?

# Hardening ECU hardware

# Hardening ECU hardware

- Memory integrity (e.g. ECC)

# Hardening ECU hardware

- Memory integrity (e.g. ECC)
- Processing integrity (e.g. lockstep)

# Hardening ECU hardware

- Memory integrity (e.g. ECC)
- Processing integrity (e.g. lockstep)
    - However, please: *Safety ≠ Security*

# Hardening ECU hardware

- Memory integrity (e.g. ECC)
- Processing integrity (e.g. lockstep)
  - However, please: *Safety ≠ Security*
- Don’t forget: **debug interfaces**

# Hardening ECU software

# Hardening ECU software

- Add <u>**redundancy**</u>:
  - Duplicate code/checks
  - SW-lockstep

# Hardening ECU software

- Add **redundancy**:
  - Duplicate code/checks
  - SW-lockstep
- Be **paranoid**:
  - Control flow integrity
  - Random delays

# Hardening ECU design

# Hardening ECU design

- Don’t expose keys to software:
  **use HW crypto engines** (HSMs)

# Hardening ECU design

- Don’t expose keys to software: **use HW crypto engines** (HSMs)

- Avoid having anything to hide: **use asymmetric cryptography**

# As always, defense in depth is key!

# Key takeaways

- No software vulnerabilities ≠ security
- Understanding firmware is easy (with the right tooling)
- Your firmware **will** be extracted

# Thanks to...

Eloi Sanfelix

Santiago Cordoba

Ramiro Pareja

Nils Wiersma

Our papers are available here, here and here!

*There were glitches… hopefully! ;)*

# Thank you! Any questions?

*Niek Timmers*
*Security Analyst at Riscure*
*@tieknimmers / niek@riscure.com*

*Alyssa Milburn*
*PhD Troublemaker at VUSec*
*@noopwafel / a.a.milburn@vu.nl*