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

digital.security
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!
METHODOLOGY
EXISTING METHODOLOGIES

• Rapid7's methodology (7 basic steps)

• OWASP IoT Project (not really mature yet)
PCB REVERSE-ENGINEERING
COMPONENTS IDENTIFICATION
MEMORY EXTRACTION

101000
010001
101011

digital.security
SOFTWARE REVERSE-ENGINEERING
SNIFFING WIRED COMMS.
SNIFFING WIRELESS COMMS.
FIND VULNS & ATTACK!

digital.security
OUR VICTIM SMARTLOCK
digital.security
STEP #1: TEARDOWN
USE THE RIGHT TOOLS
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
COMPONENTS IDENTIFICATION

nRF52832
2.4 GHz Bluetooth
Low Energy capable System-on-Chip

DRV8848
Dual H-Bridge Motor driver
FUNCTIONS VS. COMPONENTS

BLE  Battery  Motor
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
DETERMINE PROTOCOLS USED
SIMPLIFIED SCHEMATICS

- Use Inkscape, Adobe Illustrator, MS Visio, or whatever
- Draw only the interesting stuff, we do not want to counterfeit
digital.security
digital.security
STEP #4: GET FIRMWARE
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
OR DUMP EVERY AVAILABLE STORAGE DEVICE 😎
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
0002c3fd 5B 31 3B 33 32 6D 3A 44 45 42 55 47 3A 65 6E 64 20 66 69 72 6D 77
0002c40a 61 72 65 20 75 70 6C 6F 61 64 0A 00 1B 5B 31 3B 33 32 6D 3A 44 45
0002c420 42 55 47 3A 62 69 6E 64 69 66 6B 72 69 6F 6E 73 65 6D 70 74 6F 62
0002c436 6C 6F 61 64 0A 00 1B 5B 31 3B 33 32 6D 3A 44 45 42 55 47 3A 65 6E
0002c44c 64 20 73 69 67 68 61 74 75 72 65 20 75 70 6C 6F 61 64 0A 00 1B 5B
0002c462 31 3B 33 32 6D 3A 44 45 42 55 47 3A 62 69 6E 64 69 66 6B 72 69 6F
0002c478 6E 73 65 6D 70 74 6F 62 6C 6F 61 64 0A 00 1B 5B 31 3B 33 32 6D 3A
0002c48e 44 45 42 55 47 3A 62 69 6E 64 69 66 6B 72 69 6F 6E 73 65 6D 70 74
0002c4a4 6F 62 6C 6F 61 64 0A 00 1B 5B 31 3B 33 32 6D 3A 44 45 42 55 47 3A
0002c4ba 65 6E 64 20 66 69 72 6D 77 61 72 65 20 75 70 6C 6F 61 64 0A 00 1B
```

art firmware upload...

[1;32m:DEBUG:end firmware upload...[1;32m:DEBUG:Start signature upload...[1;32m:DEBUG:End signature upload...[1;32m:DEBUG:read pk...

[1;32m:INFO:invalid cert r1 value: %d...[1;32m:DEBUG:action state is null

---
digital.security
SPARE AREA IS EVIL
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** (ARMv7-M)
  - 275 µA/MHz running from flash memory
  - 150 µA/MHz running from RAM
  - Serial Wire Debug (SWD)

ARM CORTEX-M0 (ARMV7-M)
DOES IT RUN AN OS?

NOPE.
DOES IT USE A FS?

NOPE.
NRF51 SOFTDEVICE

- (Optional) Bootloader
  - Bootloader Vector Table
    - MBR Parameter storage
      - Application
        - Application Vector Table
          - SoftDevice
            - SoftDevice Vector Table
              - MBR
                - MBR Vector Table

- BOOTLOADERADDR
- MBRPARAMADDR
- APP_CODE_BASE
- 0x00001000
- 0x00000000
$ strings firmware-original.bin | grep sdk
/home/benoit/workspace/nrf51/firmware/sdk/sdk13.0/components/
/home/benoit/workspace/nrf51/firmware/sdk/sdk13.0/components/
/home/benoit/workspace/nrf51/firmware/sdk/sdk13.0/components/
/home/benoit/workspace/nrf51/firmware/sdk/sdk13.0/components/
/home/benoit/workspace/nrf51/firmware/sdk/sdk13.0/components/
/home/benoit/workspace/nrf51/firmware/sdk/sdk13.0/components/
/home/benoit/workspace/nrf51/firmware/sdk/sdk13.0/components/
/home/benoit/workspace/nrf51/firmware/sdk/sdk13.0/components/
/home/benoit/workspace/nrf51/firmware/sdk/sdk13.0/components/
/home/benoit/workspace/nrf51/firmware/sdk/sdk13.0/components/
/home/benoit/workspace/nrf51/firmware/sdk/sdk13.0/components/
/home/benoit/workspace/nrf51/firmware/sdk/sdk13.0/components/
/home/benoit/workspace/nrf51/firmware/sdk/sdk13.0/components/
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!
SPECIFY TARGET ARCHITECTURE AND LAYOUT

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

digital.security
We developed our own tool to ease SoftDevice-based firmware reverse-engineering. It helps detecting SoftDevice version and automatically rename SDK exported functions.
NRF5X-TOOLS AVAILABLE ON GITHUB

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


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 sends it to the lock
3. Lock decrypts token and **react accordingly**
The mobile app authenticates the smartlock only by its exposed service UUID:

```java
private void startScan(int paramInt)
{
    this.lastScanStartTime = System.currentTimeMillis();
    Crashlytics.log pstmt, "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
SEARCH BUGS & VULNS

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

digital.security
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»?
I'VE ALREADY SEEN THAT ...

```c
int get_random_number()
{
    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
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] 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`
$ sudo node replay-token.js
BTLE interface up and running, starting scanning ...

[i] Target found, replaying token ...
done
BUG IS NOW FIXED

digital.security
CONCLUSION
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)
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
- ...

digital.security
QUESTIONS?

CONTACT

@virtualabs
damien.cauquil@digital.security