Analyzing the security of a Fitbit wearable

Bechir BOUALI & Baptiste Neveu

June 26, 2019

Plan

1. Project Objectives
2. Fitbit hardware access
   a. Reach the Printed Circuit Board
   b. Connection to the Fitbit
   c. Dump of the firmware
3. Static analysis of the firmware
4. Dynamic analysis of the firmware
   a. GDB
   b. Avatar²
   c. Demo
5. Conclusion & Further work

Project Objectives

- Get hardware access
- Dump the firmware
- Dynamic analysis using Avatar²

Reach the PCB

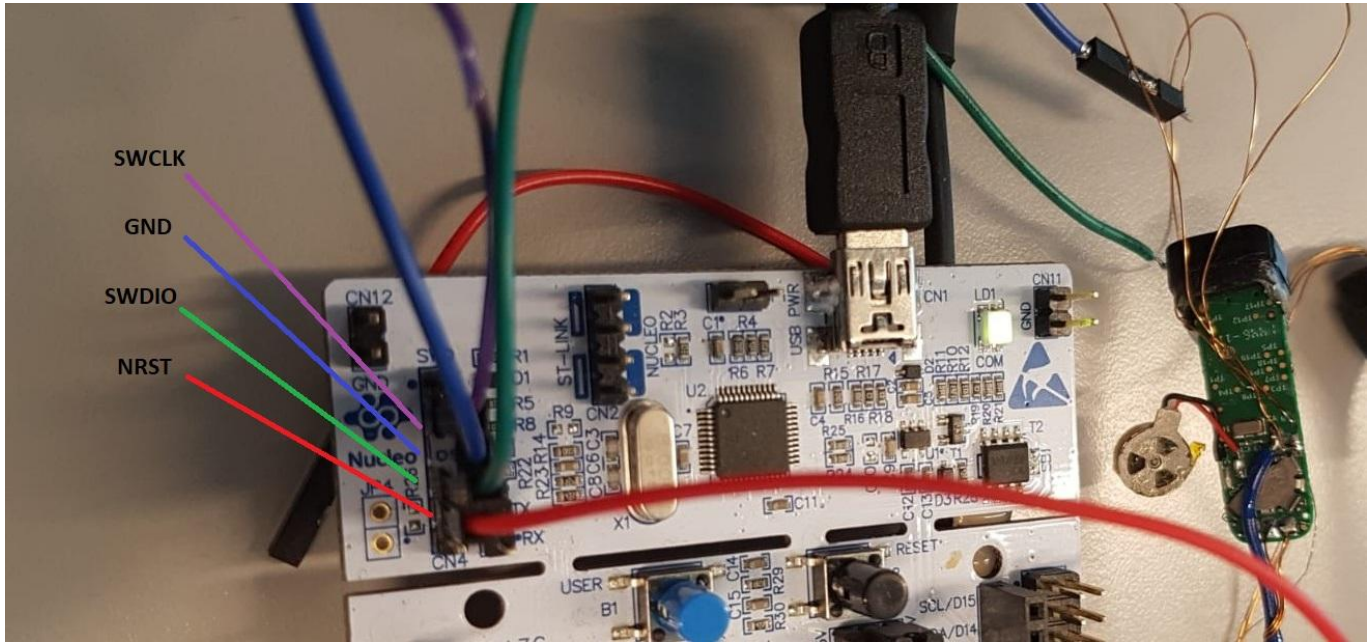
- Melt the plastic that protect the PCB
- Extra attention to :
  - the bluetooth antenna
  - every tiny part of the PCB

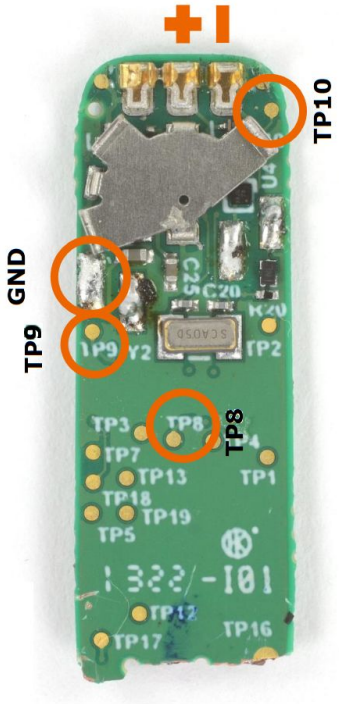
Connection to the Fitbit

- Setup connection between the debug adapter and the board

SWD pins	ST-LINK-V2-1 pins	Fitbit test points
SWDCLK	Pin 2	TP8
SWDIO	Pin 4	TP9
GND	Pin 3	GND
NRST	Pin 5	TP10

Connection to the Fitbit

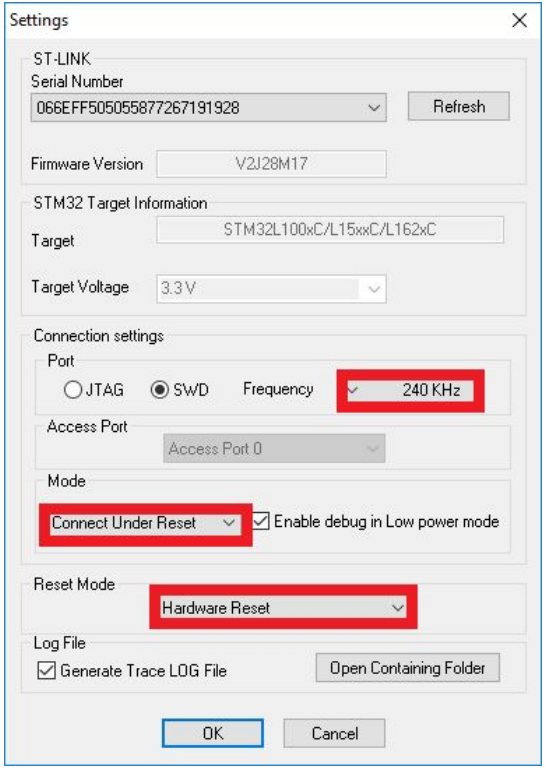




Source: Fitbit firmware hacking by Classen et al.

Dump the firmware

- Using openocd:
  - openocd -f fitbit.cfg
  - telnet 127.0.0.1 4444
  - dump_image firmware.bin 0x0 0x40000
- Using ST-Link Utility:
- Same firmware image



Static analysis of the firmware

- Focus on the bluetooth functions:
  - Library closed to Arduino BLE Breakout Board
  - IDC script to get the address:
    - 0x0800EE62: get_bluetooth_id
    - 0x08012C44: exti_bluetooth_record
    - 0x08018868: rf_record_bluetooth
    - 0x080214A0: printf_bluetooth_id
    - 080187A8: bluetoot_record_sth
    - The bluetooth stack BSL is located at 0x080036E0

Dynamic analysis of the firmware: GDB

- GDBServer hosted by openOCD
  - Run openOCD first
  - Connect to GDB
- Default firmware
  - Device crashes: reconfiguration of GPIO pins
  - Flash a custom firmware with a GDB Backdoor
- Set breakpoints for the past addresses

$arm-none-eabi-gdb

$gdb target extended-remote 127.0.0.1:3333

$gdb monitor halt

$gdb hb *0x0800EE62

$gdb hb *0x08012C44

$gdb hb *0x08018868

$gdb hb *0x080214A0

$gdb continue

Dynamic analysis of the firmware: Avatar²

- Breakpoint avatar² script:
  - Set to the 0x0800EE62 (get_bluetooth_id function)
  - Pairing with the App
  - Three hits:
    - First & second hit: called by the “bluetooth_record” function
    - Third hit: “send_bt_id_c0” function

```
 python avatar-hitbr.py
hardware breakpoint at 0x0800EE62
waitting for the breakpoint
hit it 1 times
My caller function at 0x80187d9
waitting for the breakpoint
hit it 2 times
My caller function at 0x80187d9
waitting for the breakpoint
hit it 3 times
My caller function at 0x800ee55
State transfer finished, emulator $pc is: 0x800ee62
```

Dynamic analysis of the firmware: Avatar²

- Testing bluetooth functions while execution transferred to the emulator
  - Reaching another breakpoint at 0x08012C44: exti_bluetooth_record

```
In [16]: fitbit.set_breakpoint(0x08012C44,hardware=True)
Out[16]: 2

In [17]: fitbit.cont()
Out[17]: True

In [18]: fitbit.get_status()
Out[18]: {'state': <TargetStates.STOPPED: 4>}

In [19]: fitbit.regs.pc
Out[19]: 134294596

In [20]: hex(134294596)
Out[20]: '0x8012c44'
```

Dynamic analysis of the firmware: Avatar²

- Jumping to the instruction at 0x0800EE6C where cpu load the address where the fitbit mac address stored

```
p:0800EE62 ; signed int __fastcall get_bluetooth_id(int mac_addr_dst, unsigned __int8 mac_addr_len)
p:0800EE62 get_bluetooth_id                ; CODE XREF: send_bt_id_c0+2A↑p
p:0800EE62                                 ; bluetoot_record_sth+2C↓p ...
p:0800EE62                 PUSH            {R7,LR} ; returns bluetooth id or r0=0 if error
p:0800EE64                 UXTB            R1, R1  ; Unsigned extend byte to word
p:0800EE66                 CMP             R1, #6  ; Set cond. codes on Op1 - Op2
p:0800EE68                 BCC             loc_800EE76 ; Branch
p:0800EE6A                 MOVS            R2, #6  ; length
p:0800EE6C                 LDR             R1, =bt_mac_address ; Load from Memory
p:0800EE6E                 BL              memcpy2_wrapper ; Branch with Link
p:0800EE72                 MOVS            R0, #0  ; Rd = Op2
p:0800EE74                 B               locret_800EE78 ; Branch
p:0800EE76 ; ---------------------------------------------------------------------------
```

Dynamic analysis of the firmware: Avatar²

- Fitbit mac address: cc:d1:fa:82:9b:03

```
[NEW] Device CC:D1:FA:82:9B:03 Flex
```

```
In [28]: hex(fitbit.regs.pc)
Out[28]: '0x800ee6e'

In [29]: hex(fitbit.regs.r1)
Out[29]: '0x200049f0'

In [30]: hex(fitbit.read_memory(0x200049f0,8,1,False))
Out[30]: '0xccd1fa829b03'

In [31]:
```

Demo time!

Conclusion & Further work

- We were able to cover the majority of the project objectives
- Do another patch to enable pairing between App and fitbit
- Using avatar-panda module

Enough Debugging
Time to be debugged

Questions ?