Off-the-shelf Embedded Devices as Platforms for Security Research

Lucian Cojocar
Kaveh Razavi
Herbert Bos

Vrije Universiteit Amsterdam
Background

- **Embedded devices are ubiquitous**
  - IoT buzzword
  - Old devices, out-dated code
- **Apply/develop security related techniques**
Background

- **Embedded devices are ubiquitous**
  - IoT buzzword
  - Old devices, out-dated code
- Apply/develop security related techniques

*Reuse and test these devices*
Motivation/Problem

- Develop generic techniques
- Test security oriented ideas
- Deploy security mechanism to old devices
Motivation/Problem

- Develop generic techniques
- Test security oriented ideas
- Deploy security mechanism to old devices
Motivation/Problem

- Develop generic techniques
- Test security oriented ideas
- Deploy security mechanism to old devices
Solution

- Reusing an off-the-shelf embedded device with the goal of testing security related frameworks
- Examples:
  - Avatar: Dynamic firmware analysis (Zaddach et al.)
  - Firmalice: Detection of Authentication Bypass (Shoshitaishvili et al.)
  - PoC back-doors on printers, HDDs, IPCamera etc.
Goal: code execution
Goal: firmware analysis

```assembly
.CODE

EXPORT main

main

PUSH {R4,LR}
LDR R4, =(_GLOBAL_OFFSET_TABLE_ - 0x846C)
LDR R0, =(aHelloFromS - 0x90E8)
LDR R1, [R1]
ADD R4, PC ; _GLOBAL_OFFSET_TABLE_
ADD R0, R4, R0 ; "Hello from: %s\n"
BLX printf
MOVS R0, #0 ; timer
BLX time
BLX srand48
LDR R3, =(aD - 0x90E8)
ADD R4, R4, R3 ; "%d\n"

loc_847E

BLX lrand48 ; CODE XREF: main+30h
MOVS R1, R0
MOVS R0, R4 ; format
BLX printf
MOVS R0, #1 ; seconds
BLX sleep
B loc_847E

; End of function main

ALIGN 4
off_8494
DCD _GLOBAL_OFFSET_TABLE_ - 0x846C ; DATA XREF: main+2r

00000460 00008460: main
```
Goal: debugging channel
Goal

Running new (or partially new) code and communicate with the target device.
SSD and PLC

• SSD details
  - Crucial MX100 – 128GB SATA 6Gb/s

• PLC details
  - Siemens SIMATIC S7-1200

• Focus on the SSD
SSD – recon

- Crucial MX100
  - 128GB SATA 6Gb/s
- Pictures of the PCB on-line
- PCB is very light
  - Not many components
- FU is available
SSD – code execution

• JTAG candidate
  – Checked the ground pins
  – Matched *standard* ARM pinout

• Memory read and write is working

• Code execution successfully tested
  – Tested with a tight loop over a set of NOPs
  – Halt the CPU via JTAG
SSD – communication channel (1)

• Dump the memory through JTAG:
  – Debug strings are present in memory
  – *Error logging routine* → MMIO address

```c
while (1) putc(c++);
```

Run our own code and use the oscilloscope.
SSD – communication channel (2)

- Default config:
  - one stop bit
  - no parity
  - LSB order
  - The pulse width 8.8μS

The position of the RX signal, on the PCB, is obvious.
Results

• SSD
  - JTAG → code execution/debugging channel
  - UART interface → communication channel

• PLC
  - Reprogramming the bootloader → code execution
  - UART interface → communication/debugging
Challenges

• Accessing the flash
  – MiTM setup, ICP
Challenges

- Accessing the flash
  - MiTM setup, ICP
- Watchdog timer interference
  - Static analysis/JTAG

```c
while (*p) {
    *0xBA00430C = 0x0190;
    uart_putc(*p++);
}
```
Challenges

- Accessing the flash
  - MiTM setup, ICP
- Watchdog timer interference
  - Static analysis/JTAG
- Physical signals pinout
  - Program and poke
Enabled research

- SSD
  - Secure data deletion
  - Logging/auditing
  - Backdoor

- PLC
  - Fuzzing
  - Rootkit
  - IDS
Conclusion

- We show how to reuse devices
- We enable research on these two targets

https://github.com/cojocar/embedded-reveng-research