Towards Fault Analysis of Firmware Updaters

L. Morel\textsuperscript{1} & M-L Potet\textsuperscript{3}
D. Couroussé\textsuperscript{1} & L. Mounier\textsuperscript{3} & L. Maingault\textsuperscript{2}

lionel.morel@cea.fr

\textsuperscript{1}Univ Grenoble Alpes, CEA, List,
\textsuperscript{2}Univ Grenoble Alpes, CEA, Leti
\textsuperscript{3}Verimag

F-38000 Grenoble, France
We are interested in platforms that:

- **are small** (wearables, in-body, appliances, etc.).
- **have low computational power**: 25 to a few 100s MIPS.
- **have low memory capacity**: few Kb to few Mb.
- Let’s suppose **no HW security** component.
- **No full-fledged OS.**
What’s a Bootloader?

- Minimal piece of software to **start** the platform
- Usually **sets up peripherals, memory, etc.**
- ... and jumps to OS’s or application’s main

```cpp
ResetHandler(){
  timerInit();
  memInit();
  comInit();
  encryptionInit();
  main();
}
```
What’s a Firmware?

Depends ...

It can mean:

- The low-level interface with HW, ie drivers
- The OS-like code around the “functionnal” heart of the application
- The whole code (app + drivers + glue code)

For now, we will consider the update of the whole software present on the platform.
What’s a Firmware Updater (FU)?

- Functionality to update the code of the platform firmware
- New functionalities, bug fixes, vulnerability fixes, etc.
- Implies code within the bootloader and ....
- .... code on a remote system (base station, gateway, etc)

- The attack surface is quite large ... !
Why study FUs?

- Achille’s heel of the whole system: if you control FU, you control the world platform.
- SoA essentially deals with “logical” security, not with physical attacks.
- Good intermediate level of complexity.
Typical Firmware Upgrading Flow

Towards Fault Analysis of Firmware Updaters | DACLE Division | May 29th, 2018 | 7
Corresponding Security Issues

[Atmel2013]

Security issues

Device

Customer

Manufacturer

Download

Transport

Firmware

Firmware

Firmware

Code

Firmware

Firmware

UnAUTHORIZED DEVICE

THIRD-PARTY FIRMWARE

FIRMWARE ALTERATION

REVERSE-ENGINEERING
- Execute firmware on **unauthorized** device
- Load **3rd-party** (malicious?) firmware
- **Alter** the firmware, inc.:
  - Prevent future **updates**
    eg to prevent future protections against security flaws.
  - Bring the device to **DoS**
- **Reverse-engineer** the firmware
## Existing SW Protections

### Integrity
- Check against FW alteration
  - Hash
  - Signature
  - MAC

### Authenticity
- Check against 3rd-party FW
  - Signature
  - MAC
- Execute only on authorized device
  - Encryption

### Confidentiality
- Make reverse-engineering hard
  - Encryption
Towards Fault Analysis of Firmware Updaters | DACLE Division | May 29th, 2018 | 11
Towards Fault Analysis of Firmware Updaters | DACLE Division | May 29th, 2018 | 11
These protections are only efficient to protect against “logical” attacker ....

What about physical attacks?

In particular Fault attacks?
Fault Attacks

Many fault models available:

- Instruction skip
- Control flow
- Variable modifications (registers, memory)
- Attacks on encryption function

⇒ What consequences on the BL-FU process?
Potential fault Attacks on BL-FU

Towards Fault Analysis of Firmware Updaters | DACLE Division | May 29th, 2018 | 14
Potential fault Attacks on BL-FU

Load a code that is not authenticated
Potential fault Attacks on BL-FU

1. Initialization
2. trigger update?
   - yes: Verify Firmware Integrity & Authenticity
   - no: Delete Firmware
3. Start Update
   - exec at 0xFFFFFFFF
4. Wait For Data
5. Data Received?
   - yes: Write Last Page
   - no: End of Transfer?
6. Page Received?
   - yes: Decrypt Page
   - no: Store Data
7. End of Transfer?
   - exec at 0xFFFFFFFF
8. Start Update
   - exec at 0xFFFFFFFF
Provoke DoS
Potential fault Attacks on BL-FU

Prevent further FUs
Potential fault Attacks on BL-FU

Build attacks on crypto functions
Potential fault Attacks on BL-FU

Prevent firmware’s suicide

Start of Transfer?
- Page Received?
  - yes: Decrypt Page
  - no: Write Page

Wait For Data
- Data Received?
  - yes: Store Data
  - no: End of Transfer?

Start Update
- Firmware Integrity & Authenticity
  - OK?
    - yes: Load Firmware
    - no: Delete Firmware

Initialization
- trigger update?
  - yes: Verify Firmware Integrity & Authenticity
  - no: Initialization

Write Last Page
- Data Received?
  - yes: Write Last Page
  - no: End of Transfer?
- Load a code that is not authenticated
- Provoke DoS
- Prevent further FUs
- Build attacks on crypto function
- Prevent Firmware’s suicide
Goals of the IRT-NanoElec
CLAPS project

- **on-going** Develop a BL/FU, starting from simplistic version
- **on-going** Define fault models
- Analyse at source-level with Lazart
- Apply compiler-level counter-measures
- Build mechanisms for attack-detection, based on:
  - machine-learning models
  - runtime verification / monitoring
- Evaluate counter-measures with laser testbeds
Thanks!
Questions?