HARDSPLOIT

Framework for Hardware Security Audit

*a bridge between hardware & a software pentester*
Who we are?

- Julien Moinard
  - Electronic engineer @opale-security (French company)
  - Security consultant, Hardware & Software pentester
  - Team project leader of Hardsploit
  - DIY enthusiast

- Yann ALLAIN
  - CEO
  - Blackhat, HackInThebox, HIP, speaker & trainer
  - Cybersecurity veteran (+ 20 years) / (old) electronic engineer
  - Former CSO of ACCOR (software domain)
A PRAGMATIC APPROACH FOR YOUR IT & IoT SECURITY

TRAININGS

CONSULTING

PRODUCTS

WWW.OPALE-SECURITY.COM

CONTACT@OPALE-SECURITY.COM +33 (0)9 53 22 99 64
Internet of Things & Privacy concern?

- Any IoT object could reveal information about individuals
- **Wearable Technology**: clothes, watches, contact lenses with sensors, microphones with cameras embedded and so on
- **Quantified Self**: pedometers, sleep monitors, and so on
- **Home Automation**: connected households using smart fridges, smart lighting and smart security systems, and so on
- ...
• Last news : (you can update this slide every week 😊)

VTech was hacked in November, exposing millions of accounts.

In response, the firm took some essential services offline, meaning products could not be registered on Christmas Day.

Firmware can be read without any problem (SPI memory)
Iot Eco-system (20000 feet view)

• Privacy Risk level : Where?

- HF communication (ISM Band) + Wifi + 3G-5G, Bluetooth, Sigfox, Lora etc..

Classical wired connections

Central servers, User Interface, API, Backoffice etc.
Security speaking, hardware is the new software?

**SOFTWARE**
To secure it:
- Security products (Firewall, Antivirus, IDS, ...)
- Security services (Pentest, Audit, ...)
- Tools (Uncountable number of them)

**HARDWARE**
To secure it:
- Few or unimplemented solutions (Encryption with key in a secure area, anti-replay mechanisms, readout protection, ...)
Hardsploit & hardware hacking
basic procedure

• 1/ Open it
• 2/ Fingerprint all the component if you can else automatic brute forcing
• 3/ Use those that may contain data (Online / Offline analysis ?)
• 4/ Perform read | write operation on them
• 5/ Reverse engineering, find vulnerabilities and exploit them
Global Purpose

DUMP ALL THE DATA
Why?

- Because chips contain interesting/private data
  - Passwords
  - File systems
  - Firmware
  - ...

| 0000000 0000 0001 0001 1010 0010 0001 0004 0128 |
| 0000010 0000 0016 0000 0028 0000 0010 0000 0020 |
| 0000020 0000 0001 0004 0000 0000 0000 0000 0000 |
| 0000030 0000 0000 0000 0010 0000 0000 0000 0204 |
| 0000040 0004 8384 0084 c7c8 00c8 4748 0048 e8e9 |
| 0000050 00e9 6a69 00e9 a8a9 00a9 2828 0028 fdfc |
| 0000060 08fc 1819 0019 9898 0098 d9d8 00d8 5857 |
| 0000070 0057 7b7a 007a bab9 00b9 3a3c 003c 8888 |
| 0000080 8888 8888 8888 8888 288e be88 8888 8888 |
| 0000090 3b83 5788 8888 8888 7667 778e 8828 8888 |
| 00000a0 d61f 7abd 8818 8888 467c 585f 8814 8188 |
| 00000b0 8bd0 e8f7 88aa 8388 8b33 88f3 88bd e988 |
| 00000c0 8a18 889c e841 c988 b328 6871 688e 958b |
| 00000d0 a948 5862 5884 7e81 3788 1ab4 5a84 3ee5 |
| 00000e0 3d86 dc8b 5cbb 8888 8888 8888 8888 8888 |
| 00000f0 8888 8888 8888 8888 8888 8888 8888 0000 |
| 0000100 0000 0000 0000 0000 0000 0000 0000 0000 |
| * |
| 0000130 0000 0000 0000 0000 0000 0000 0000 0000 |

000013e
How?

• A hardware pentester needs to know electronic buses and he needs to be able to interact with them.

PARALLEL

- SPI
- UART
- I²C
- 1-Wire
- CAN
- JTAG / SWD
- Custom
Hardsploit framework

Same hardware but a software update is needed to add a new protocols
Hardsploit bus indentification & scanner
(in progress, not published yet)

Hardsploit → Database of components → Module (I2C, SPI, etc.)

- Database of patterns
- IO hardware mixer
- Input / Output
- IoT target

Scanner

Click to hack audit hardware
# Tool of trade

<table>
<thead>
<tr>
<th>FUNCTIONALITIES</th>
<th>BUSPIRATE</th>
<th>JTAGULATOR</th>
<th>GOODFET</th>
<th>HARDSPLOIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>UART</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td>SPI</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td>PARALLEL</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td>I2C</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td>JTAG / SWD</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td>MODULARITY</td>
<td>Microcontroller</td>
<td>Microcontroller</td>
<td>Microcontroller</td>
<td>uC / FPGA</td>
</tr>
<tr>
<td>EASE OF USE</td>
<td>Cmd line + datasheet</td>
<td>Command line</td>
<td>Command line</td>
<td>Official GUI / API / DB</td>
</tr>
<tr>
<td>I/O NUMBER</td>
<td>&lt; 10</td>
<td>24</td>
<td>&lt; 14</td>
<td>64 (plus power)</td>
</tr>
<tr>
<td>WIRING</td>
<td>TEXT (but MOSI = SDA)</td>
<td>TEXT / AUTOMATIC identification</td>
<td>TEXT</td>
<td>LED / TEXT/ AUTOMATIC identification</td>
</tr>
</tbody>
</table>
Hardsploit: Communication

Graphical interface (QT)

GUI
- RUBY
  - High level layer communication

API
- RUBY
  - Low level layer communication

PC
- USB 2.0 Full Speed (12 Mbits/s)

Microcontroller
- SPI 15Mhz
- Control SPI 15Mhz

FPGA
- SPI 15Mhz
- Serial Configuration Device (AS mode)
- Bridge between any module and SPI Control & Data
- For microcontroller communication

Wiring module
- Compatible buses
  - Set 64 led statut
  - Get 64 led statut

Input / Output & Led for Assistance wiring
  - Sniffing
  - Read
  - Write
  - Custom Commands

TARGET
- 64 GPIO
Prototype making

• Applying soldering paste (low budget style)
Prototype making

• Manual reflow oven (DIY style)
Prototype making (with a budget)

• The rebirth
The board – Final version

• 64 I/O channels
• ESD Protection
• Target voltage: 3.3 & 5V
• Use a Cyclone II FPGA
• USB 2.0
• 20cm x 9cm
Hardsploit organization
Chip management

- Search
- Create
- Modify
- Interact
Wiring helper

Datasheet representation

Hardsploit Wiring module representation

GUI <-> Board interaction
Settings

Hardsploit - I²C settings

24LC64 PARAMETERS
Base address (W): A2
Base address (R): A3
Frequency (Khz): 400
Total size: 8192
Bus scan: Launch

Hardsploit - Bus settings

25LC640 PARAMETERS
Page size: Total size (8 bits word): 4096
Frequency (Mhz): 1.00 Mode: 1
SPI command read: 3

Hardsploit - Parallel settings

P33-65 nm PARAMETERS
Total size: 120000
Read latency: 1600
Write latency: in nanoseconds
Word size: 8 bits 16 bits
Page size: 0

Cancel  Save
Command editor
What are available on github (Open) ?

- Microcontroller (c)
- API (ruby)
- GUI (ruby)
- Create your own Hardsploit module : VHDL & API (ruby)
Already available (github)

Parallel non multiplexed memory dump
  • 32 bits for address
  • 8/16 bits for data

Helping wiring
I2C 100Khz 400Khz and 1 Mhz
  • Addresses scan
  • Read, write, automatic full and partial dump

SPI mode 0,1,2,3 up to 25 Mhz
  • Read, write, automatic full and partial dump

SWD interface (like JTAG but for ARM core)
  • Dump and write firmware of most ARM CPU

GPIO interact / bitbanging (API only for the moment)
  • Low speed < 500Hz  read & write operations on 64 bits
More to come (see online roadmap)...

- Automatic bus identification & Scanner (@30%)
- Component & commands sharing platform (@90%)
- TTL UART Module with automatic detection speed (@80%)
- Parallel communication with multiplexed memory
- I2C sniffing (shot of 4000 bytes up to 1 Mhz)
- SPI sniffing (shot of 8000 / 4000 byte half / full up to 25Mhz)
- RF Wireless transmission training plateform (Nordic NRF24, 433Mhz, 868Mhz transcievers)
- Metasploit integration (module) ??
- JTAG
- 1 Wire
- CanBUS (with hardware level adapter)
- ...
Concrete case

• An electronic lock system
• 4 characters pin code A – B – C – D
  • Good combinaison – Door opens, green L.E.D turn on
  • Wrong combinaison – Door closes, red L.E.D turn on
Concrete case: Open it
Concrete case: Fingerprint

- **STM32F103RBT6**
- **SPI MEMORY 25LC08**
- **I2C MEMORIES 24LC64**
Concrete case: Online / Offline analysis?
Concrete case: hardsploit scenario

1. Open Hardsploit to create the component (if not exist)
2. Connect the component to Hardsploit (wiring helping)
3. Enter and save the component settings (if not exist)
4. Dump the content of the memories (1 click)
5. Change the door password by using commands (few clicks)
6. Try the new password on the lock system (enjoy)
Concrete case:
Read | Write operation, I2C, SPI, SWD ...

• Time for a live demo?
Parallel bus memory
Concrete case: Fingerprint
Concrete case: Offline analysis
Concrete case: Ready to dump the content
Conclusion

- IoT Device are (also) prone to vulnerabilities help you to find them
- Security policy need to be adapted, nowadays, it is not so difficult to extract data on IoT
- Designers need to design with security in mind
- Skills related to pentest a hardware device is mandatory for Security Experts (but training exist)
- Industry need to take care about device security
Thank you!

Hardsploit board is available at shop-hardsploit.com (250 € / 277 USD / 370 CAD excluding VAT)
To learn more about Hardsploit and follow the development

Hardsploit.io & Opale-Security.com

• Yann ALLAIN (CEO)
  • yann.allain@opale-security.com
  • +33 6 45 45 33 81

• Julien MOINARD (Project leader of Hardsploit)
  • julien.moinard@opale-security.com
  • +33 9 72 43 87 07

Hardware & Software, Pentest, Audit, Training