WooKey: Episode VII

The Force Awakens

Philippe Teuwen

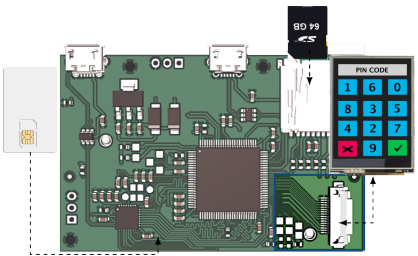
19 November 2021, (almost) Grenoble

Quarkslab

Doegox

- A Team Leader of
  Crypto & Embedded Security at Quarkslab
- White-box crypto “grey box” attacks
- EEPROM/RFID “tear-off” attacks
- Funky microsoldering on Google Titan M
- One of Libnfc and RRG/Proxmark3 Maintainers
- Hardware-oriented CTFs
- International Journal of PoC||GTFO

WooKey in brief



Secure USB Mass Storage developed by ANSSI

- Encrypted SD Card
- Screen + touchscreen (randomized keypad)
- ISO7816 reader + tokens (JavaCards)

WooKey nominal usage

1. Insert AUTH token
2. Type PetPIN
3. Check PetName
4. Type UserPIN
5. Now seen as Mass Storage

Why Pet Pin and Pet Name ?
To avoid PIN phishing attempts with a fake WooKey

WOOKEY

ANSSI

Context

- ANSSI: French National Cybersecurity Agency
- Awards CSPN certifications to security products
- Based on private evaluation labs: CESTI/ITSEF
- Quarkslab: certified software CESTI
- ANSSI organizes regular inter-CESTI challenges

Last inter-CESTI challenge

Based on WooKey

- Open source (hw/sw) project by ANSSI
- Large evaluation plan shared across 10 CESTI
- Mixing hardware and software skills

What is this talk about?

A bit about the WooKey

- We already got many talks on it

Mostly about how to set up cheap hardware attacks

1. Timing attack against smartcard
2. UI automation over SPI (and a bit of crypto)
3. Electromagnetic Fault Injection test bench

Timing attack against smartcard

Quarkslab

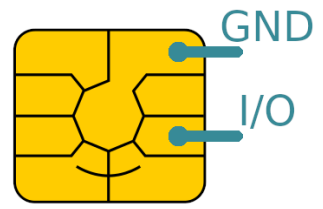
The Plan

- Check if UserPIN validation doesn't leak timing information
  - Correct PIN (e.g. 1234) timing probably different, ok
  - What if 1999 vs 9999 ?
  - What if 9994 vs 9999 ?
  - What if 9999 vs 99999999 ?

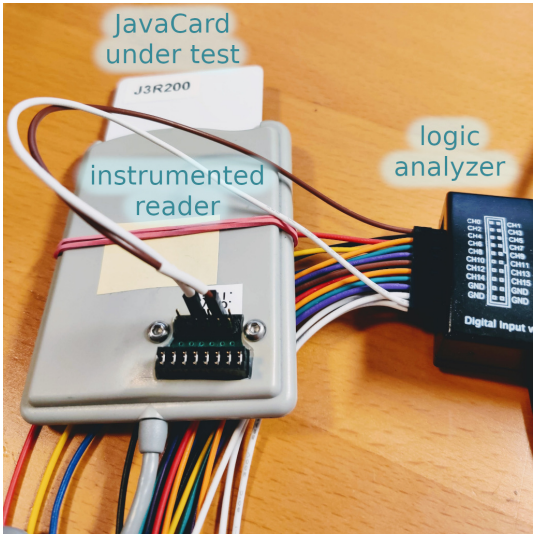
Code review: they used "OwnerPIN.check" JavaCard API
Still, let's see how to do it...

The Material

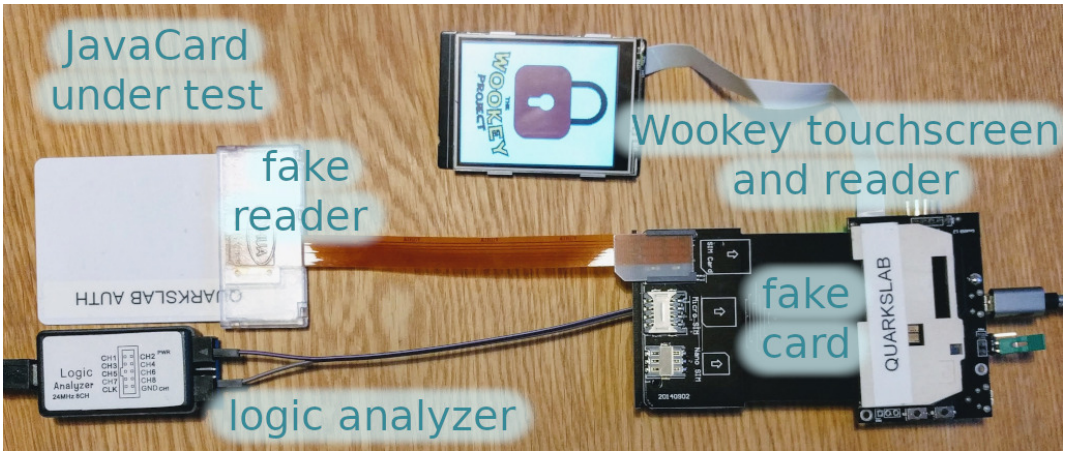
- Logic analyzer
- Way to hook on I/O and GND PINs
- Script
  - Trigger trace acquisition
  - Communicate with the card
  - Acquire trace
  - Rinse and repeat
- Target (that won't kill itself after 3 fails...)



Option 1: instrumented reader



Option 2: reader-card extension



Trace example

PIN length tests example

	Response delay in ms	
Tested PIN length	**Configured PIN length: 4**	**Configured PIN length: 8**
1	9.598	9.611
2	9.617	9.603
3	9.643	9.642
4	9.706	9.604
5	9.661	9.643
6	9.644	9.644
7	9.652	9.642
8	9.598	9.689
9	9.616	9.611
10	9.612	9.609
Correct PIN	12.159	12.156

PIN length tests example: oops...

	Response delay in ms	
Tested PIN length	**Configured PIN length: 4**	**Configured PIN length: 8**
1	9.598	9.611
2	9.617	9.603
3	9.643	9.642
4	9.706	9.604
5	9.661	9.643
6	9.644	9.644
7	9.652	9.642
8	9.598	9.689
9	9.616	9.611
10	9.612	9.609
Correct PIN	12.159	12.156

PIN length: a JavaCard OS issue

- Not a problem of the WooKey
- Found on J3H081 (JCOP3, EAL5+) and J3R200 (JCOP4, EAL6+) JavaCards
- Not a big vulnerability, but still unexpected
- Reported to NXP

UI automation over SPI

Quarkslab

First a bit of crypto

Remember PetPIN & PetName?

- Wookey
  - PetPIN $\rightarrow$ DK = PBKDF2-SHA512(PetPIN, salt)
  - $\rightarrow$ DK sent to token
- Token
  - Contains an encrypted key blob (ELK)
  - KPK $= \mathrm{Dec}_{\mathrm{DK}}(\mathrm{ELK})$
  - $\rightarrow$ KPK sent to WooKey
- Wookey
  If KPK correct:
  - Decrypts a keystore
  - Mounts a Secure Channel with token
  - Gets PetName from token
  - Displays PetName

First a bit of crypto

- DK $\rightarrow$ KPK $= \text{Dec}_{\text{DK}}(\text{ELK})$
- DK & KPK $\rightarrow$ ELK $= \text{Enc}_{\text{DK}}(\text{KPK})$

First a bit of crypto

- $\text{DK} \rightarrow \text{KPK} = \text{Dec}_{\text{DK}}(\text{ELK})$
- $\text{DK \& KPK} \rightarrow \text{ELK} = \text{Enc}_{\text{DK}}(\text{KPK})$
- $\text{DK}^* \text{ \& KPK}^* \rightarrow \text{ELK} = \text{Enc}_{\text{DK}^*}(\text{KPK}^*)$

No need to know the correct DK or PetPIN to extract ELK!

The Plan

- Extract ELK from token
- Create fake token with ELK (and no countermeasure)
- Bruteforce PetPIN with WooKey + fake token
- Use original token and PetPIN $\rightarrow$ get PetName

The Plan

- Extract ELK from token
- Create fake token with ELK (and no countermeasure)
- Bruteforce PetPIN with WooKey + fake token
- Use original token and PetPIN $\rightarrow$ get PetName

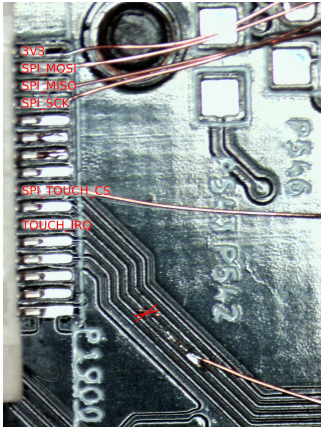
Bruteforce: deal with randomized keypad...

- Sniff SPI commands to screen to reconstruct the keypad
- Inject fake touchscreen IRQ and SPI traffic
- Sniff ISO7816 to watch for a Secure Channel attempt

The Material

- Fake token
- Microcontroller (e.g. a Teensy)
- Logic analyzer (for debugging)
- Attach to SPI bus
- Hijack touchscreen IRQ track

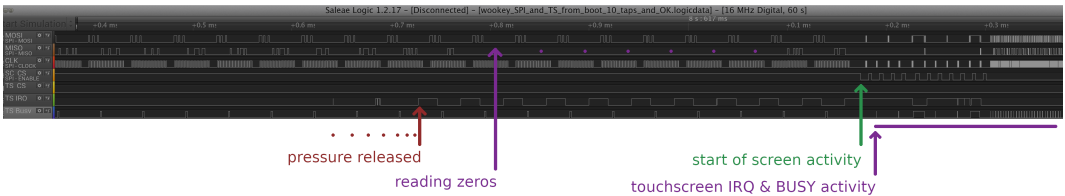
And... very unreliable results...



Tweaks

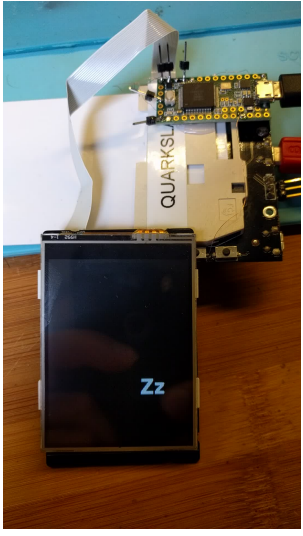
WooKey bugs:

- Touchscreen Chip Select always active
- Touchscreen always sampled 16x even after pressure release → wrong average pos



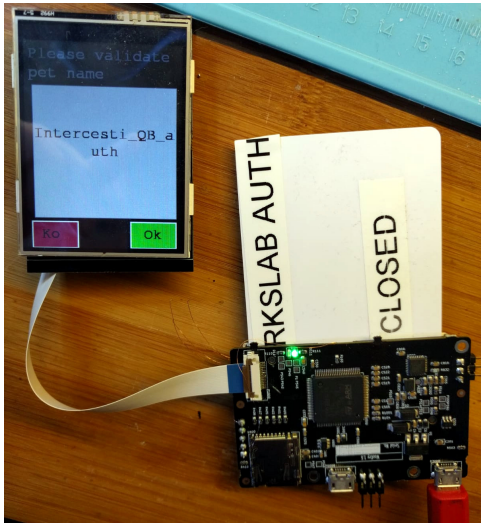
→ Drive touchscreen IRQ and inject fake positions in sync with the full sampling sequence

Action!



Bruteforce results

PetPIN broken in 15 h
(41 h for full 4-digit space)
→
Use original token to get PetName



Electromagnetic Fault Injection test bench

Quarkslab

The Plan

Send an EM perturbation to affect the chip operations

- E.g. skipping instructions

Characterize target chip with sample code

- Simple loop counters to detect if instructions are skipped
- Try various coils, voltages, durations, XYZ positions,...

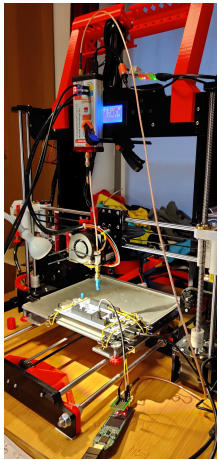
Tune parameters and choose which function to target

- Skip locking and erasing security features
- Target: Loader checking firmware integrity (SHA-256)
- Replace next stage by blinking LEDs (red=alert, blue=success)
- One more parameter: pulse delay

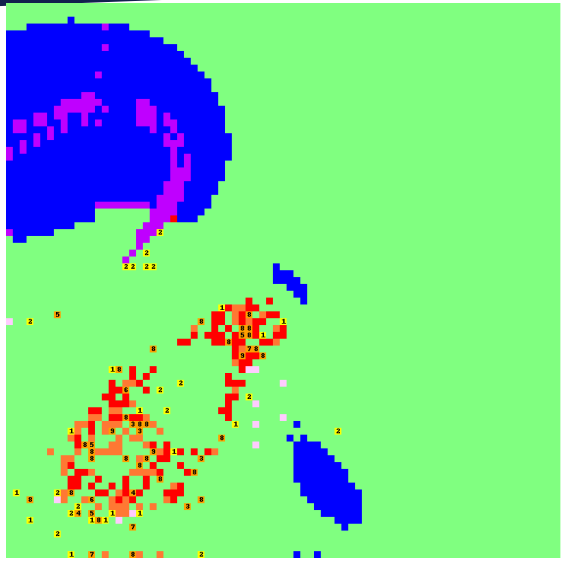
The Material

Script framework orchestrating:

- EMFI injector
- Configurable trigger (FPGA)
- XYZ table (3D printer, CNC,...)
- Communication with target
- Reset of target
- Cartography and test campaigns



Cartography example



Date: 20200219-141012
Tip: 4mmcw-hwtrig
Z: 1.0 mm
Pulse voltage: 500 v
Pulse width: 80 ns
Pulse delay: 2000 ns
X axis: [-0.0, 0.0] mm
Y axis: [-0.0, 0.0] mm
X axis resolution: 0.2 mm
Y axis resolution: 0.2 mm

: normal behavior
: ToE not responsive
: ToE rebooted
: repeated instructions
: skipped < 3 instructions
: skipped < 10 instructions
: skipped < 100 instructions
: skipped many instructions

Results

Integrity check bypass:

- 80 tests/min
- First success after 5000 attempts
- Then with more tuning, success rate of 7.2%

Less expected results...

Got firmware and RAM leaks over UART (disabled in prod)

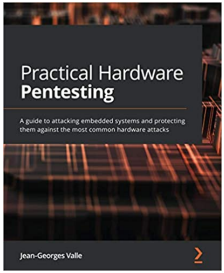
Tried RDP2 bypass, fried a demo board after few hundred thousand tests...

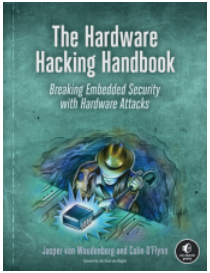
Conclusion

Hardware attacks are fun
Hardware attacks are affordable
Never assume, challenge always

Two very recent books to recommend:

- Practical Hardware Pentesting, by Jean-Georges Valle
- The Hardware Hacking Handbook, by Colin O'Flynn and Jasper van Woudenberg





Thank you

Contact information:

Email: contact@quarkslab.com

Phone: +33 1 58 30 81 51

Website: https://www.quarkslab.com

Quarkslab