.talsec Evidence of App Safety | Quarkslab

Mobile App Security Conference

Making Attackers Sweat: Raising the Bar with Software Protection

Product R&D Manager @ Quarkslab

Béatrice Creusillet

Expert in compilation, software protection
Quarkslab's main representative at GlobalPlatform

Quarkslab

Securing every bit of your data

French cybersecurity company
founded in 2011

€ 8m fundraising in 2020

110+ employees
20% PhDs

324 conferences
247 Blogposts https://blog.quarkslab.com
51 academic articles

Lab

Security audit, pentesting, vulnerability research

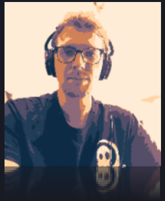
Shield

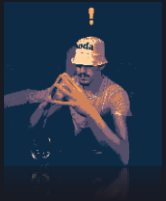
Software protection

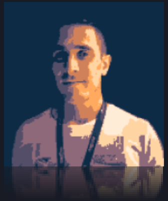
- Obfuscation, RASP, Integrity
- Whitebox cryptography
- Secure data storage

Offices in France and Argentina

Attacking an EV Supply Equipment through its companion app


Ricardo Mori


Alex Chazal


Robin David

Autel MaxiCharger AC Wallbox:

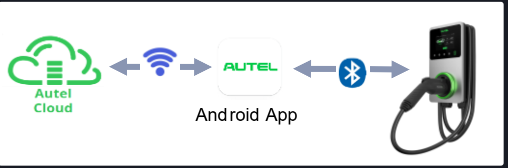
- residential and commercial
- connected (USB, Ethernet, Bluetooth, WiFi, NFC)
- access via apps iOS/Android
- billing features
- FOTA

Compromise the EV charger:

- get **free charging**
- **damage** car/battery
- get **access** to:
  - home/company network
  - nearby devices
  - cloud-based vendor back-end
  - ...

I - Firmware retrieval – 3 days

App reversal to find firmware retrieval URL



- Packed with SecNeo
- Dynamic analysis to retrieve app code
- Static analysis to retrieve URL

II - Firmware decryption – 3 days

Firmware was encrypted:

- Call in a **crypt-analyst**

III – Vulnerability research – 20 days

Firmware analysis:

- No mitigations / protections (ASLR, obfuscation)
- No symbols:
  - use **internal tool** for function similarity -> FreeRTOS
- Static analysis to identify 3 vulnerabilities:
  - Bluetooth stack: 2 vulns
  - USB stack: 1 vuln

IV – Exploitation - 7days

- **2 exploitation chains** (Bluetooth, USB)
- Persistent across firmware updates

- Did not work on day-D: different firmware version

Why it's important to protect IoT devices and their companion apps?

They provide:

- a window into your home/company network
- access to the devices they interact with
- access to cloud-based services

Don't let them be the weakest link of your security posture

	Firmware retrieval	Firmware decryption	Vulnerability discovery	Exploitation	Total
Actual case	**3** p.days	**3** p.days	**20** p.days	**7** p.days	**33** p.days
No protection			**20** p.days with internal tool	**7** p.days	
Light protection	**3** p.days **Packing**	**3** p.days + crypt-analyst **light encryption**			
Strong protection					

	Firmware retrieval	Firmware decryption	Vulnerability discovery	Exploitation	Total
Actual case	**3** p.days	**3** p.days	**20** p.days	**7** p.days	**33** p.days
No protection	A few hours	**0** p.days	**20** p.days with internal tool	**7** p.days	**27** p.days
Light protection	**3** p.days **Packing**	**3** p.days + crypt-analyst **light encryption**			
Strong protection					

	Firmware retrieval	Firmware decryption	Vulnerability discovery	Exploitation	Total
Actual case	**3** p.days	**3** p.days	**20** p.days	**7** p.days	**33** p.days
No protection	A few hours	**0** p.days	**20** p.days with internal tool	**7** p.days	**27** p.days
Light protection	**3** p.days **Packing**	**3** p.days + crypt-analyst **light encryption**	~ **26** p.days **light obfuscation**: • FreeRTOS: a few days • applicative part: a few days	**7** p.days + additional time per version	~ **39** p.days above allocated time
Strong protection					

	Firmware retrieval	Firmware decryption	Vulnerability discovery	Exploitation	Total
Actual case	**3** p.days	**3** p.days	**20** p.days	**7** p.days	**33** p.days
No protection	A few hours	**0** p.days	**20** p.days with internal tool	**7** p.days	**27** p.days
Light protection	**3** p.days **Packing**	**3** p.days + crypt-analyst **light encryption**	~ **26** p.days **light obfuscation**: • FreeRTOS: a few days • applicative part: a few days	**7** p.days + additional time per version	~ **39** p.days above allocated time
Strong protection	~ **6-11** p.days **RASP**	~ **7-14** p.days **strong encryption** ~ **5-20** days **WB encryption**	**not possible in allocated time:** -> change of attack techniques **heavy obfuscation**	?? + greater additional time per version	above allocated time

Application protection

PROTECT

Behavior

- safety, security, revenues

Sensitive data

- credentials, keys,...
- operating parameters

Algorithms

- IP/proprietary technology

THREATS

- Static binary analysis
- Dynamic binary analysis
- Symbolic execution
- Fault-injection
- Side-channel attacks
- Vulnerability exploitation

ENSURE

Integrity

- Application signature
- Code integrity
- Compiler/linker options
- RASP

Confidentiality

- Code obfuscation
- Whitebox cryptography

Obfuscation: make your code harder to understand

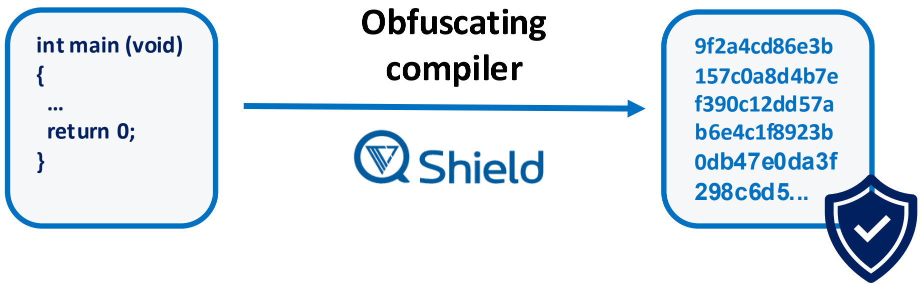
Complexify / Hide

- Application/program structure (call graph)
- Functions structure (control flow graph)
- Instructions/operations
- Constants
  - meaningful scalars
  - strings, arrays,…

Diversify

- Protect different versions differently
- Protect several instances of a given version differently

Obfuscation: make your code harder to understand *automatically*

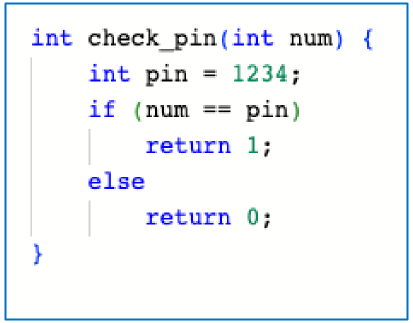


```
int main (void)
{
  ...
  return 0;
}
```

```
9f2a4cd86e3b
157c0a8d4b7e
f390c12dd57a
b6e4c1f8923b
0db47e0da3f
298c6d5...
```

Obfuscation: make your code harder to understand

Simple PIN code check
Source Code

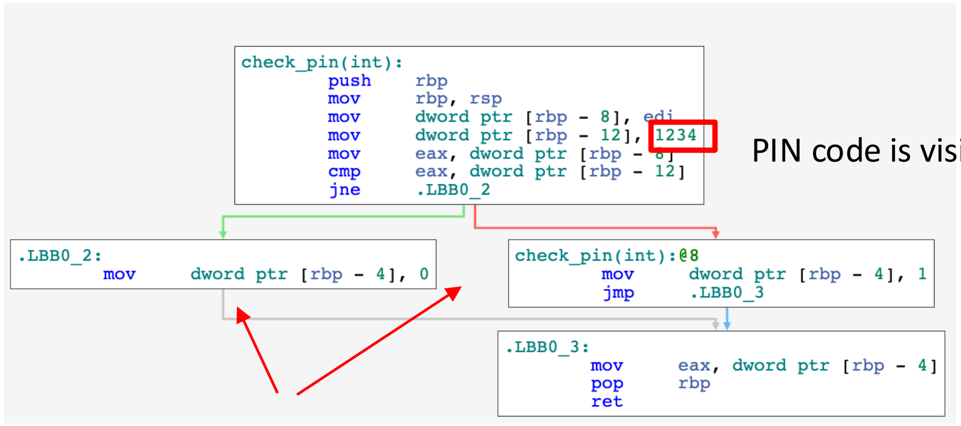


```
int check_pin(int num) {
    int pin = 1234;
    if (num == pin)
        return 1;
    else
        return 0;
}
```

normal compilation

BEFORE protection

unobfuscated assembly code view



```
check_pin(int):
        push    rbp
        mov     rbp, rsp
        mov     dword ptr [rbp - 8], edi
        mov     dword ptr [rbp - 12], 1234
        mov     eax, dword ptr [rbp - 8]
        cmp     eax, dword ptr [rbp - 12]
        jne     .LBB0_2

.LBB0_2:
        mov     dword ptr [rbp - 4], 0

check_pin(int):@8
        mov     dword ptr [rbp - 4], 1
        jmp     .LBB0_3

.LBB0_3:
        mov     eax, dword ptr [rbp - 4]
        pop     rbp
        ret
```

PIN code is visible

Easy to spot 2 execution branches

Obfuscation: make your code harder to understand

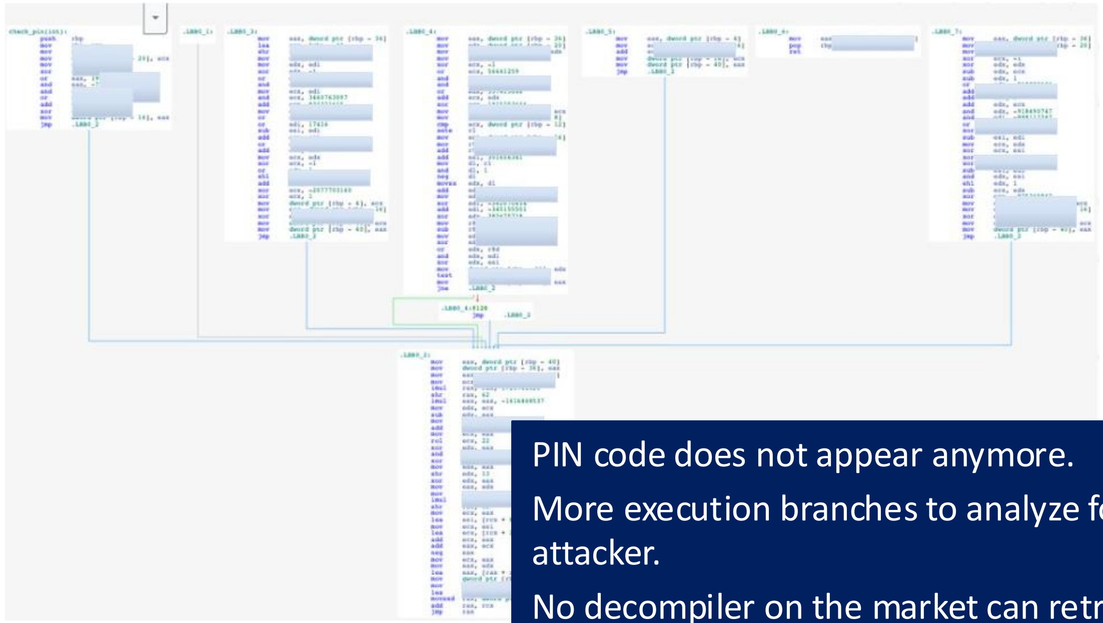
Simple PIN code check
Source Code

```
int check_pin(int num) {
    int pin = 1234;
    if (num == pin)
        return 1;
    else
        return 0;
}
```

Obfuscation

AFTER protection

obfuscated **assembly code** view

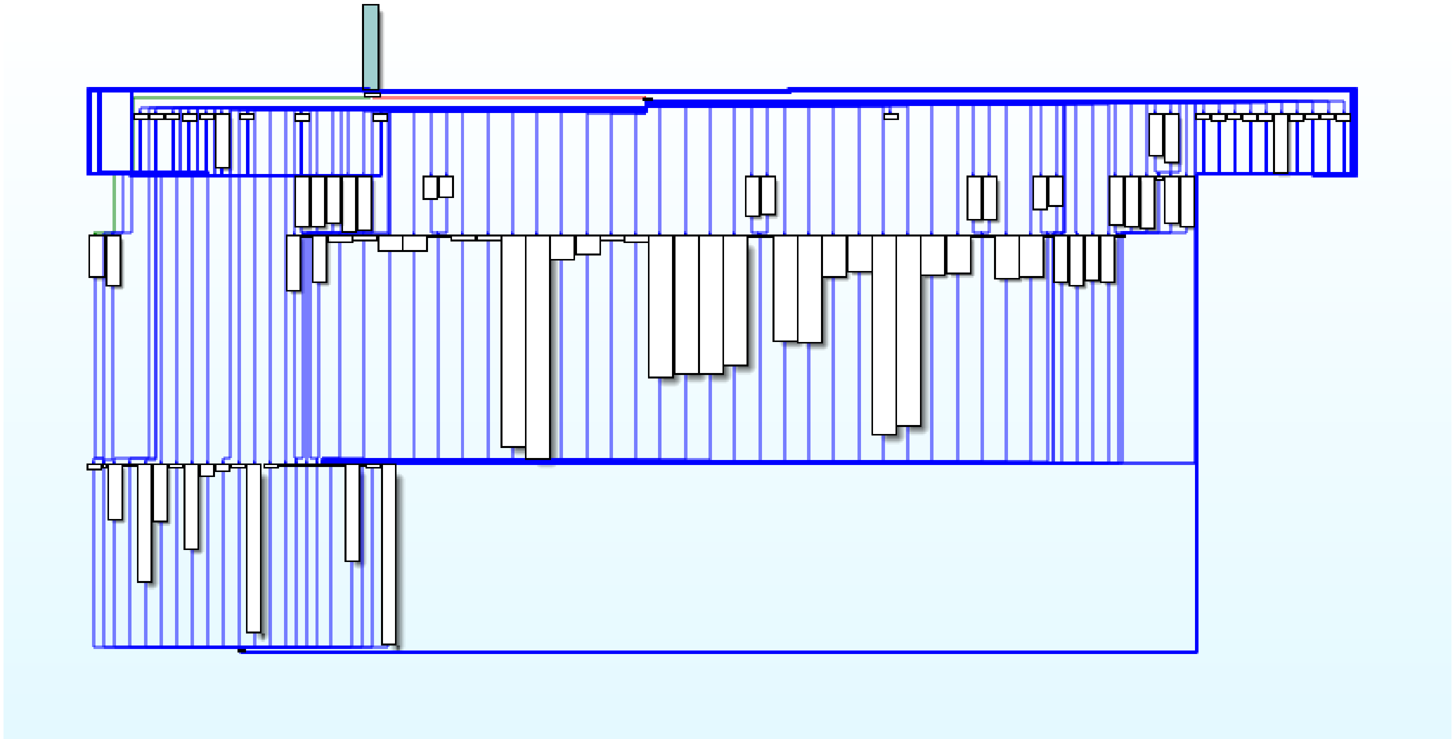


PIN code does not appear anymore.

More execution branches to analyze for an attacker.

No decompiler on the market can retrieve the original code

OBFUSCATION: a more complex example



RAISE THE BAR FOR ATTACKERS

Slow them down, raise the required level of expertise!

… but no technique alone is sufficient to protect code/data:

- **Combine and layer software protections**
  - Obfuscation, RASP, integrity, encryption,…
- **Combine software and hardware protections**
  - HW secure enclaves, memory encryption engines,…
- **Diversify: protect each version of your code differently**
  - Or even each instance when possible
- **Don't just protect sensitive assets**
  - Spread protection everywhere to reduce attack surface

www.quarkslab.com
contact@quarkslab.com

.talsec Evidence of App Safety | Quarkslab

Thank you!