riscure

escar
Embedded Security in Cars

Fault Injection on Diagnostic Protocols

Niek Timmers
Principal Security Analyst, Riscure
timmers@riscure.com / @tieknimmers

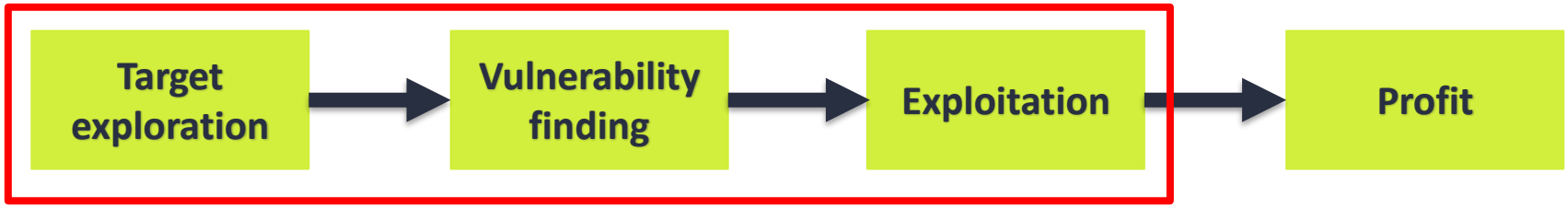
21-06-2018

Big shout out to Ramiro and Santiago!

- Security Analysts at Riscure's ***Automotive Security Team***
- **Riscure**
  - Services
    - E.g. Penetration Testing, Security Architecture Review, and more.
  - Tools
    - E.g. Automotive Security Test Tools, Fault Injection, and more.
  - Training
    - E.g. Embedded Security for Automotive, Fault Injection, and more.
- Offices in the Netherlands, USA and China

Combining services and tools for fun and profit!

The hacker's approach for hacking ~~embedded systems~~ ECUs



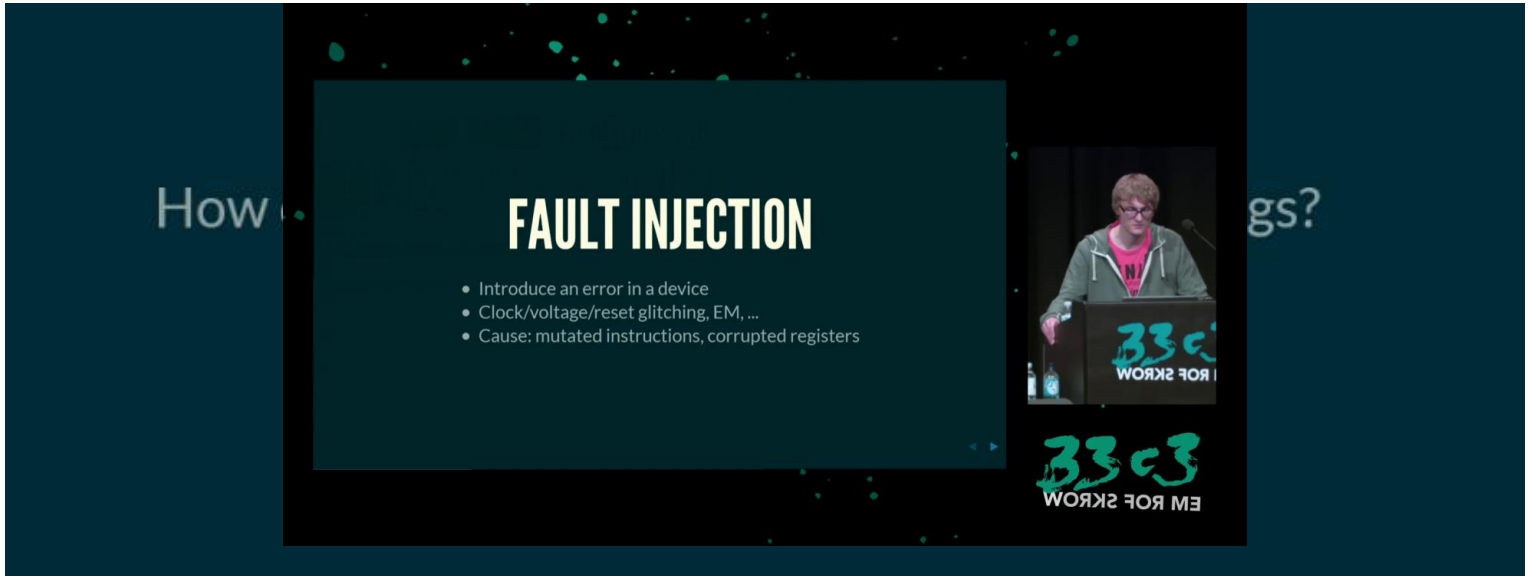
Access to the firmware is a tremendous convenience for an attacker!

Obtaining ECU Firmware

- Firmware is available through **official channels**
- Firmware is stored in an **external memory chip**
- Firmware **upgrades are not encrypted**
- Firmware is **leaked / distributed illegally**
- **Code protection** features are **not enabled**
- Firmware is extracted using a **software-based attack**

What if all of the above is not applicable? Attackers will resort to something else…

Others came to similar conclusions…



Reference: *https://derrekr.github.io/3ds/33c3/#/18*

Hackers nowadays use Fault Injection!

Fault Injection – Introduction

"Introducing faults in a target to alter its intended behavior."

```
...
if( key_is_correct ) <-- Glitch here!
{
  open_door();
}
else
{
  keep_door_closed();
}
...
```

How can we introduce these faults?

Fault Injection – Techniques

Clock

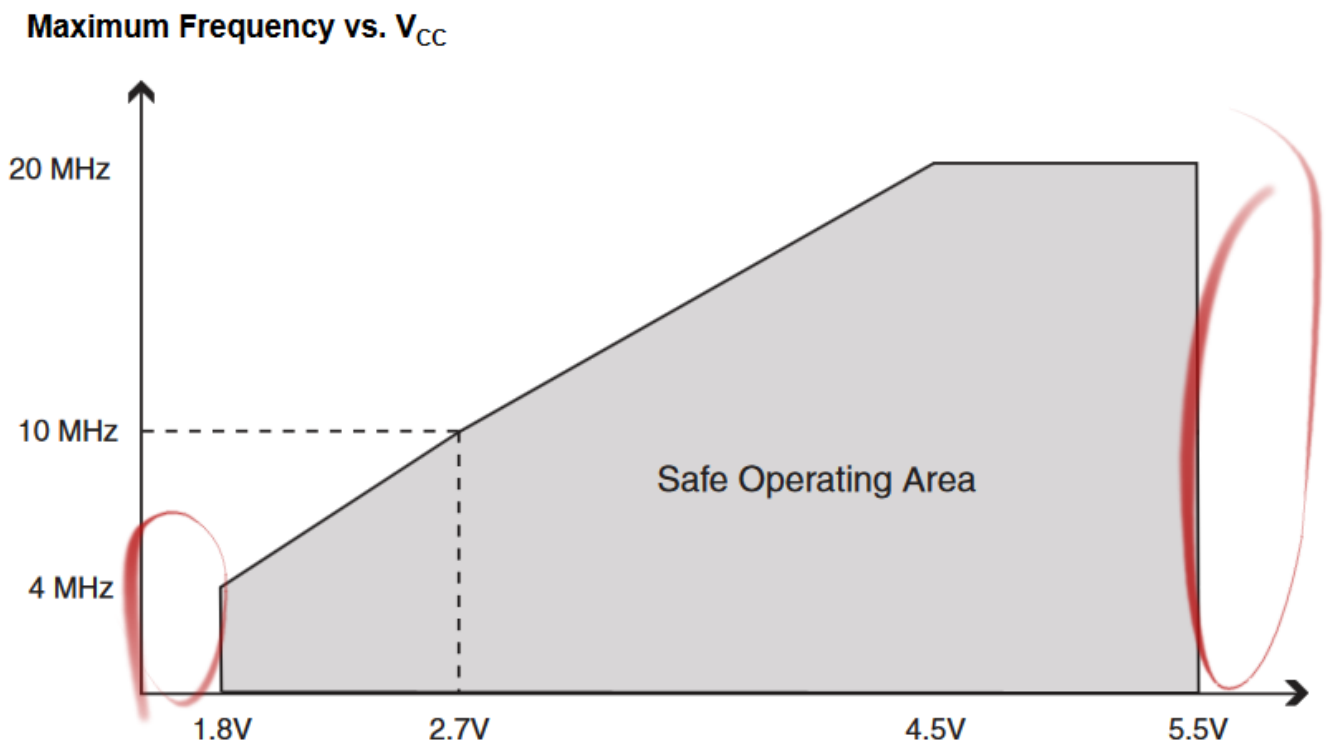
Voltage

Electromagnetic

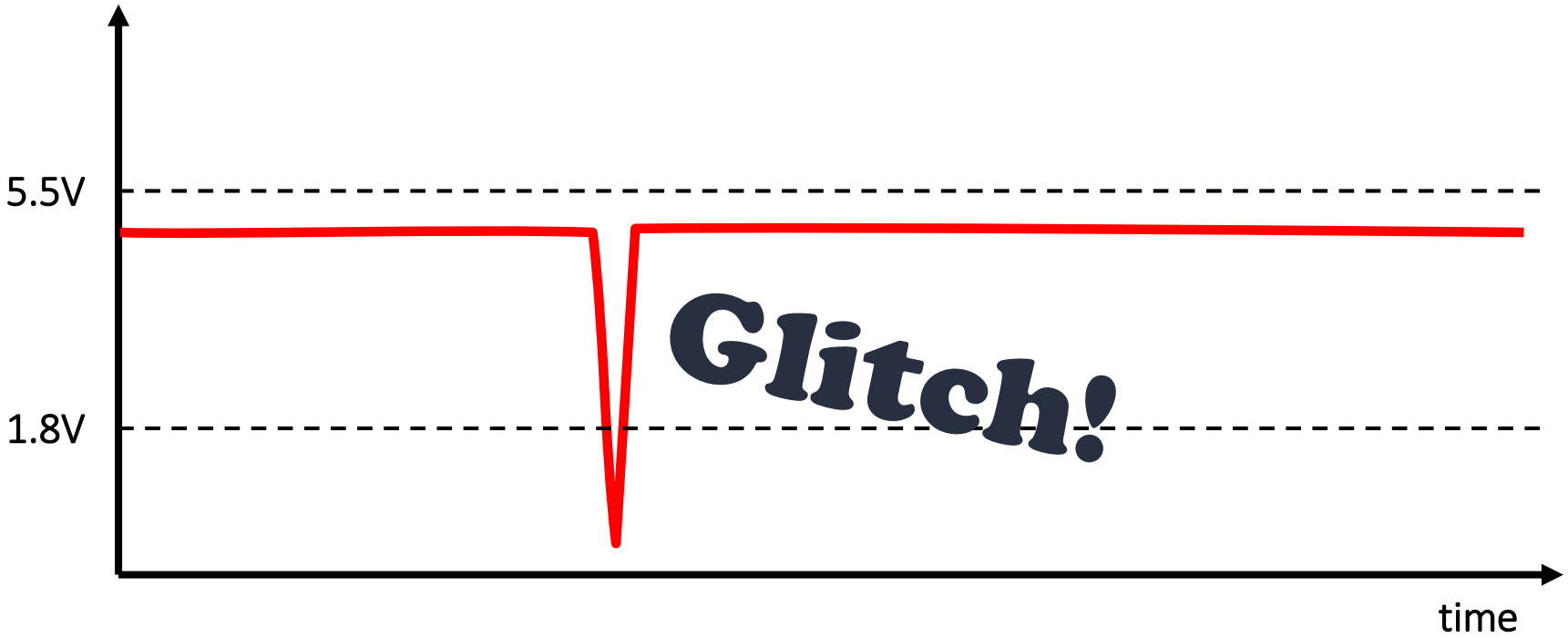
Laser

- A controlled environmental change leads to altered behavior in the target
- They leverage a vulnerability in hardware

Fault Injection – Why does it work?



Fault Injection – Basic concept



Fault Injection – Typical faults

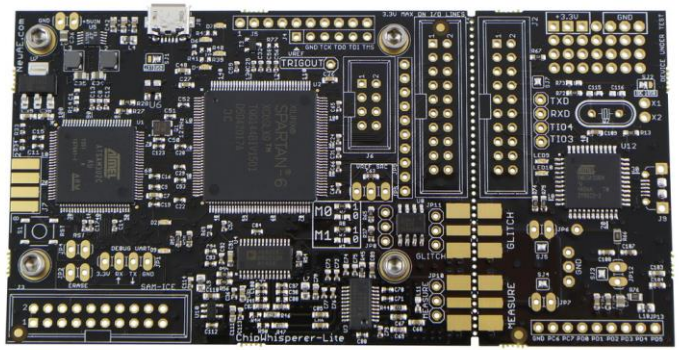
- **Instruction corruption**
  - Executing different instructions
  - Skipping instructions

- **Data corruption**
  - Reading different data
  - Writing different data

These faults change the intended behavior of software!

Fault Injection – Tooling

Open source



ChipWhisperer®

Commercial

Inspector Fault Injection

Fault Injection tooling is available to the masses!

Fault Injection – Examples

riscure

black hat
EUROPE 2016

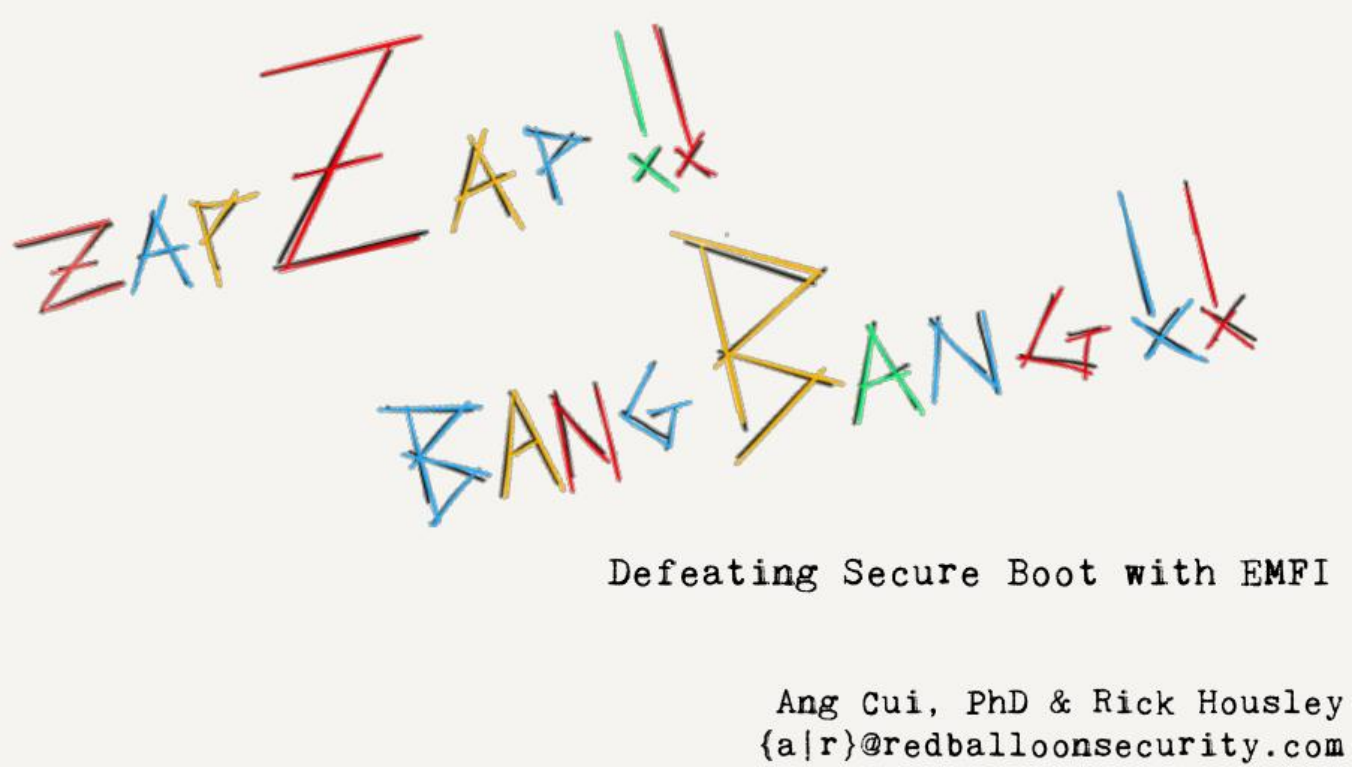
Bypassing Secure Boot using Fault Injection

Niek Timmers
timmers@riscure.com
(@tieknimmers)

Albert Spruyt
spruyt@riscure.com

November 4, 2016

Fault Injection – Examples



Fault Injection – Examples

KERNELFAULT:

R00ting the Unexploitable using Hardware Fault Injection

Niek Timmers
Senior Security Analyst
@tieknimmers / niek@riscure.com

Cristofaro Mune
Product Security Consultant
@pulsoid / c.mune@pulse-sec.com

We can inject faults and alter software…

What software?

UDS!

It’s common and includes convenient functionality!

UDS – Unified Diagnostic Services

- **Diagnostic and Communication Management**
  - Diagnostic Session Control
  - Security Access
- **Data Transmission**
  - Read and Write memory
- **Upload / Download**
  - Read and Program flash
- **Manufacturer proprietary**

ISO14229

Disclaimer: *Manufacturers are free to implement only a subset of the UDS specification!*

UDS – What speaks it?

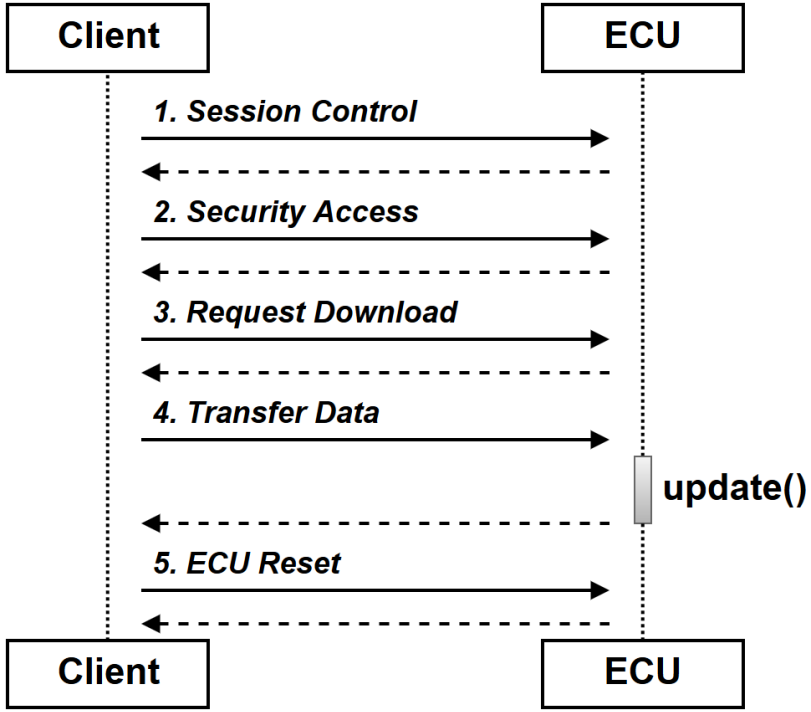
Most ECUs in a modern car speak it using a multitude of protocols

CAN **K-Line** **FlexRay** **Ethernet**

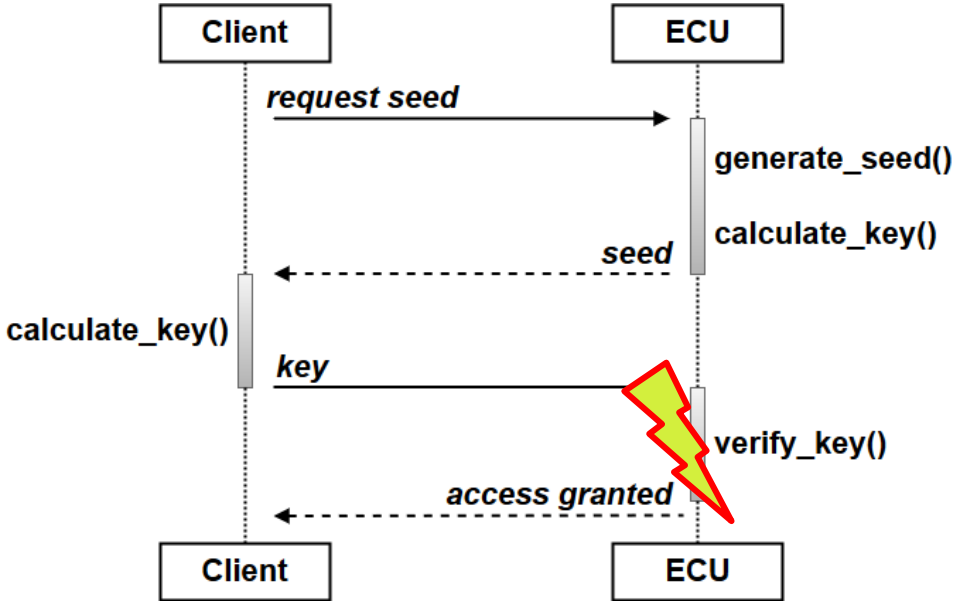
It is unlikely UDS will go away!

UDS – Typical use case

Firmware update



Standard Security Access Check



*The **secret** used by the **key calculation algorithm** should be protected!*

Standard Security Access Check

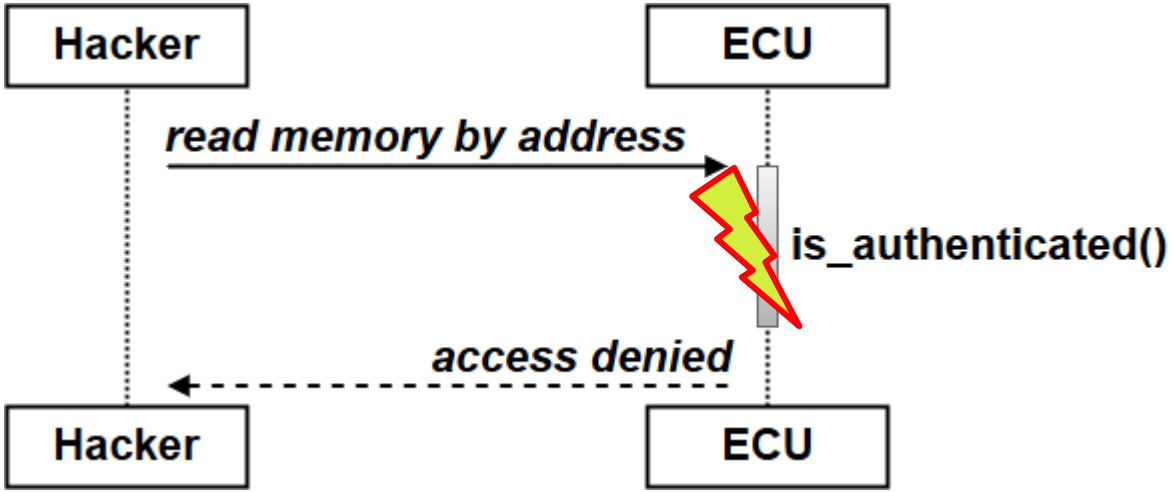
```
int key_verification(. . .) {
    ...
    // key verificatio
    if ( received_key == calculated_key ) {
        access_granted();
        error_code = NONE;
    } else {
        error_code = INVALID_KEY;
    }
    return error_code;
}
```

Standard Security Access Check

- Not successful :’(
- There’s a 10 minute timeout after 3 failed attempts
- Simply not practical for us (or an attacker)

Some times you have to take your losses and move on!

Reading Memory



No restrictions on failed attempts!

Reading Memory

```
int read_memory_by_address(...) {
    ...
    // check if authenticated
    if ( authenticated ) {
        error_code = NONE;
        memcpy(buffer, address, length);
    } else {
        error_code = SECURITY_ACCESS_DENIED;
    }
    return error_code;
}
```

Reading Memory

- Successful on multiple instrument clusters

- Depending on the target
  - Allows reading out **N** bytes from an **arbitrary** address
  - Extraction of the internal memories in **N** days

- Addresses include volatile and non-volatile memories
  - Complete firmware extracted

Extraction of the firmware only has to be done once!

We have the firmware… now what?

Lots of “cool” stuff…

- Reverse engineering
  - Understanding the device
  - Extracting secrets
- Finding vulnerabilities

Please see Alyssa’s presentation on reverse engineering firmware efficiently!

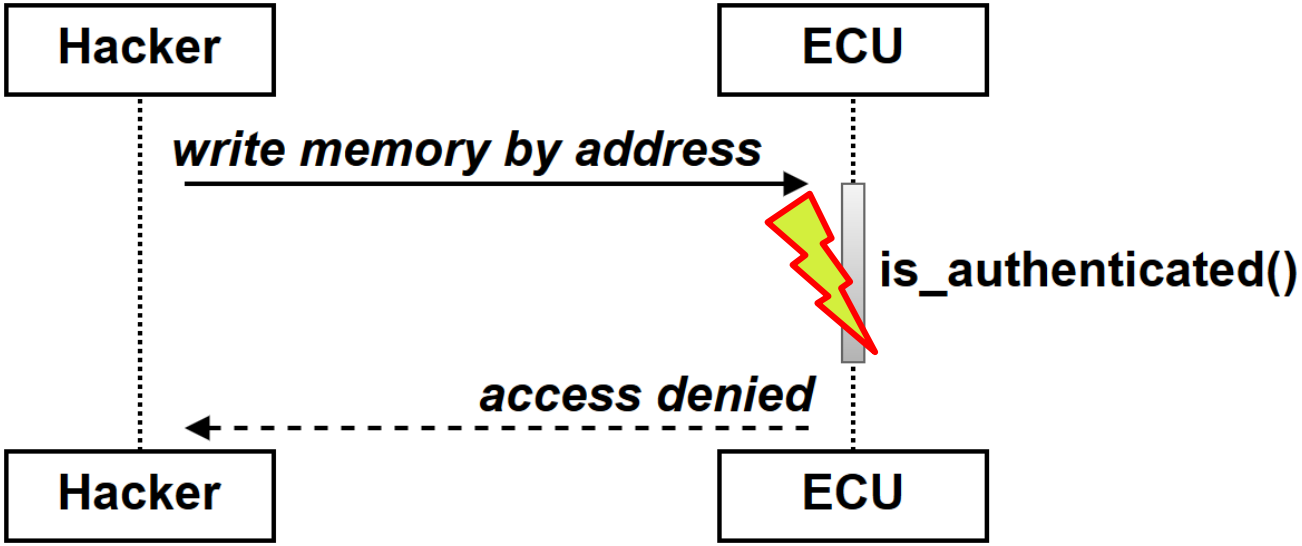
But… does this scale!?

Hardware attacks scale!

- Firmware can be distributed
- Secrets can be distributed
- Vulnerabilities (and exploits) can be distributed
- Vulnerabilities can be exploited remotely

Reading memory is fun!
What about something cooler?

Attack #3: Writing memory



Where should we write to get code execution on the ECU?

Wrapping up…

Why is UDS vulnerable?

- A **robust Security Access check** is **not part of the standard**
- Typical Security Access check based on **pre-shared secrets**
- **No fault injection resistant hardware** used in most ECUs
- **No fault injection resistant software** used in most ECUs

What can you do?

Improving Products

- Include fault injection attacks in your threat model
- Design and implement fault injection resistant hardware
  - Start from an early design
  - Test, test… and test again!
- Implement fault injection resistant software
- Make critical assets inaccessible to software
  - E.g. Using “real” hardware

Fault Injection Hardened Firmware

Not hardened

```
if ( authenticated ) {
    error_code = NONE;
    something_useful();
}
```

Hardened

```
if ( authenticated ) {
    error_code = NONE;
} else { return }

if ( authenticated ) {
    something_useful();
} else { return }
```

Prevent single point of failures for security critical checks! More info [here](#).

Key takeaways

- Fault injection attacks are available to the masses
- Fault injection attacks subvert software security models
- All unprotected devices are vulnerable
- Presented attack not unique; most ECUs affected
- Fault injection attacks result in scalable attacks

Riscure Head Office
Delftechpark 49
2628 XJ Delft
The Netherlands
Phone: +31 15 251 40 90
inforequest@riscure.com

Riscure North America
550 Kearny St., Suite 330
San Francisco, CA 94108
USA
Phone: +1 650 646 99 79
inforequest@rna.riscure.com

Riscure China
Room 2030-31, No. 989, Changle Road
Shanghai 200031
China
Phone: +86 21 5117 5435
inforcn@riscure.com

Thank you!

riscure

Challenge your security

Niek Timmers

Principal Security Analyst

niek@riscure.com / @tieknimmers