riscure

Attacking AUTOSAR using Software and Hardware Attacks

Pascal Nasahl
Graz University of Technology

Niek Timmers
Riscure

Introduction

Introduction

- Niek Timmers
  - Principal Security Analyst @ Riscure

Introduction

- Niek Timmers
  - Principal Security Analyst @ Riscure

- Analyzing and testing of embedded technologies

Introduction

- Niek Timmers
  - Principal Security Analyst @ Riscure
- Analyzing and testing of embedded technologies
- Research
  - Automotive, secure boot, fault injection, etc.
  - More at niektimmers.com and riscure.com

Introduction

- Niek Timmers
  - Principal Security Analyst @ Riscure
- Analyzing and testing of embedded technologies
- Research
  - Automotive, secure boot, fault injection, etc.
  - More at niektimmers.com and riscure.com

Please visit Riscure's booth for more information! ☺

Today’s Agenda

Today’s Agenda

- Brief introduction to AUTOSAR Classic

Today's Agenda

- Brief introduction to AUTOSAR Classic
- Attacks on AUTOSAR

Today's Agenda

- Brief introduction to AUTOSAR Classic
- Attacks on AUTOSAR
- Case study

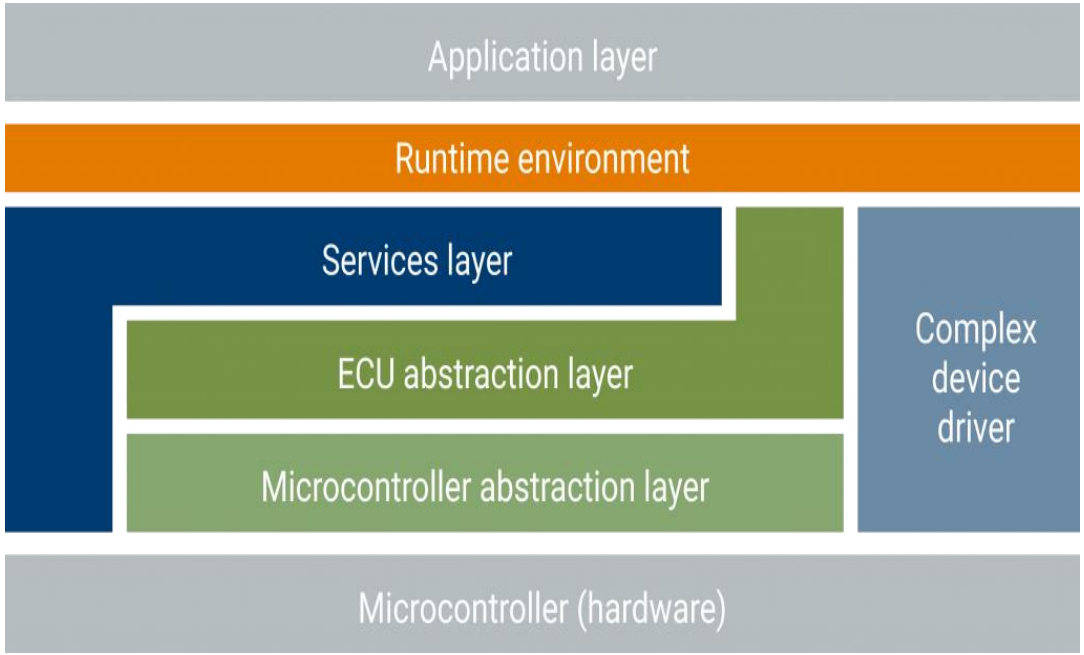
Today's Agenda

- Brief introduction to AUTOSAR Classic
- Attacks on AUTOSAR
- Case study
- Wrap-up

Today's Agenda

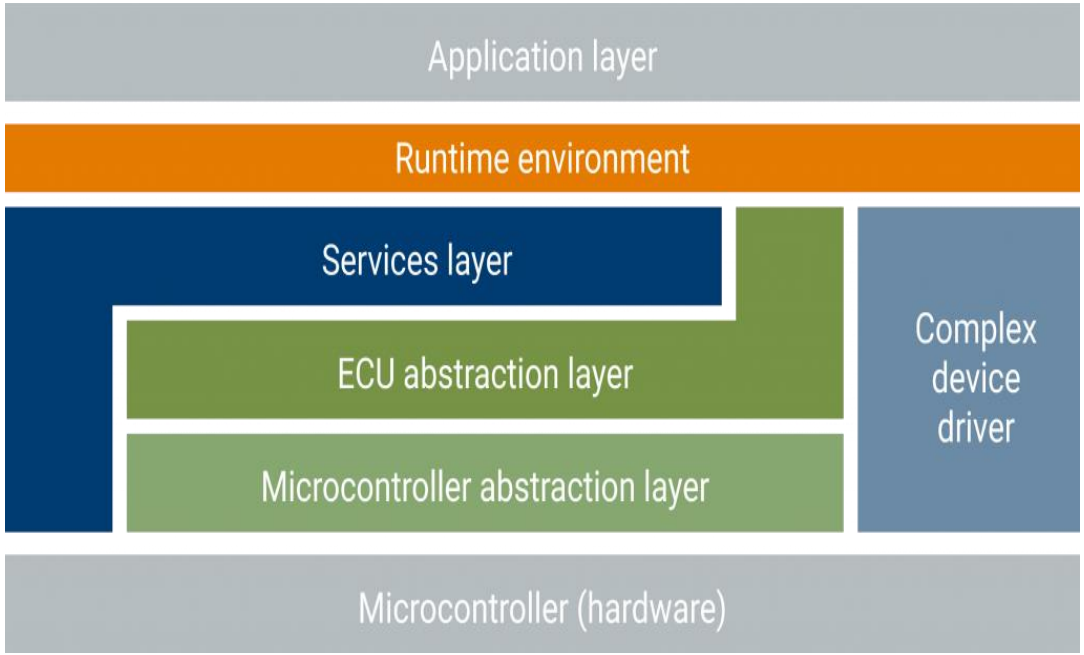
- Brief introduction to AUTOSAR Classic
- Attacks on AUTOSAR
- Case study
- Wrap-up
- Q&A

AUTOSAR Classic



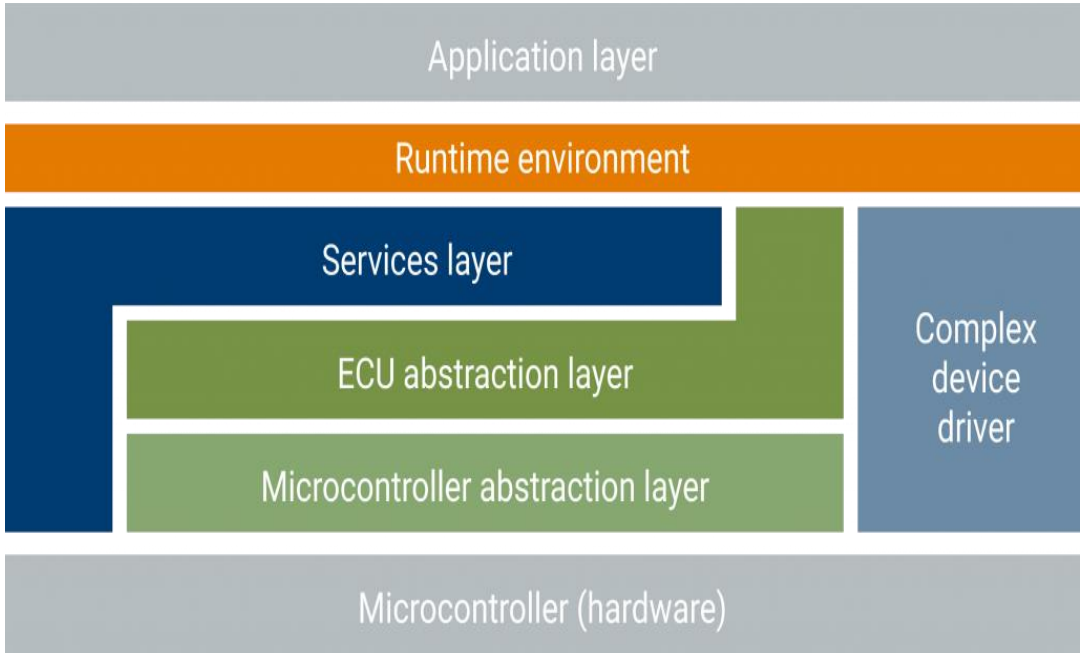
AUTOSAR Classic

- Layered software architecture



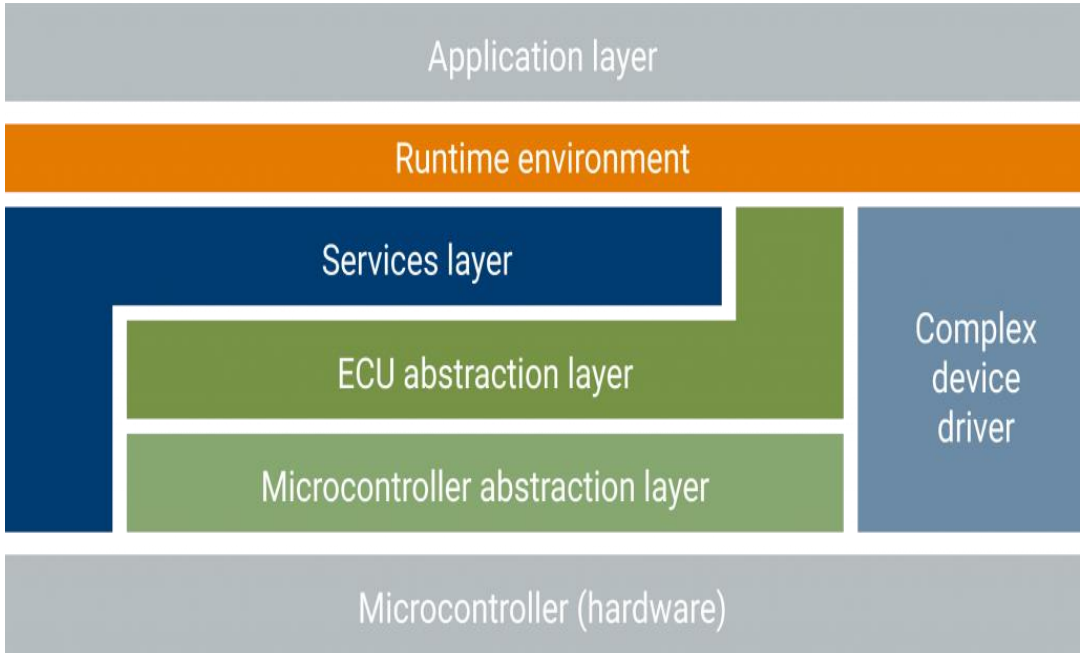
AUTOSAR Classic

- Layered software architecture
- Most layers are independent from the Microcontroller

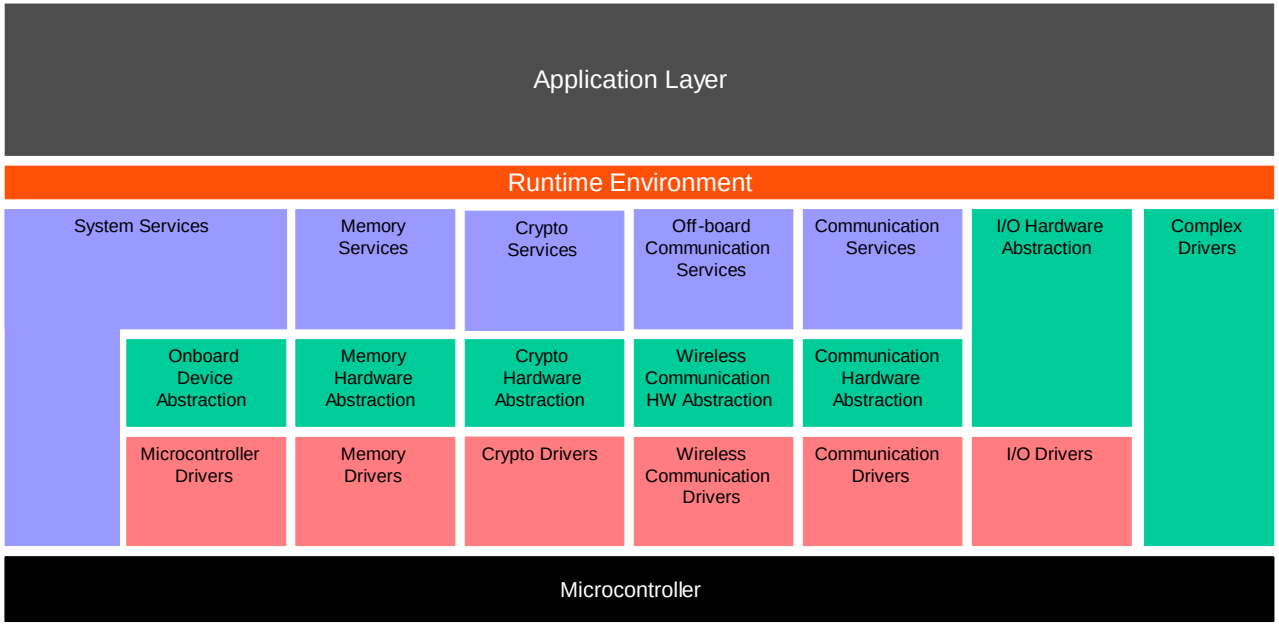


AUTOSAR Classic

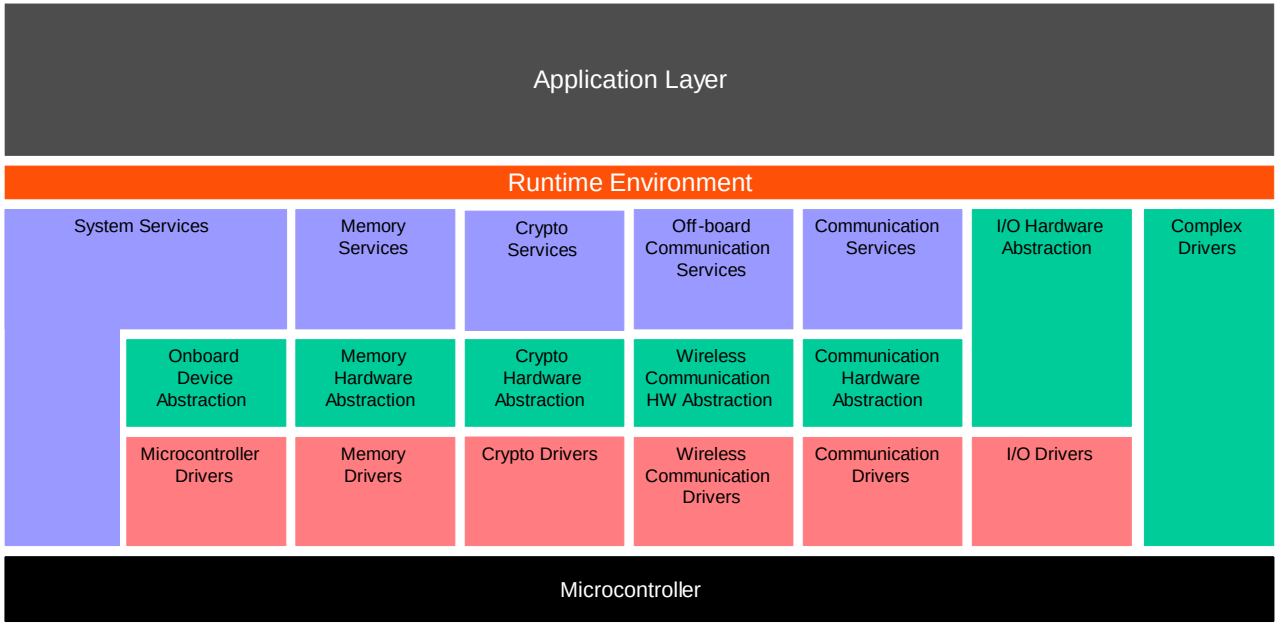
- Layered software architecture
- Most layers are independent from the Microcontroller
- Improve software reusability



AUTOSAR Classic



AUTOSAR Classic



Vulnerabilities can be introduced in any layer!

Summary of AUTOSAR

Summary of AUTOSAR

- Complex software; will contain bugs/vulnerabilities

Summary of AUTOSAR

- Complex software; will contain bugs/vulnerabilities
- Made by different vendors / developers
  - Do you trust your suppliers?

Summary of AUTOSAR

- Complex software; will contain bugs/vulnerabilities
- Made by different vendors / developers
  - Do you trust your suppliers?
- Mature code due to safety requirements
  - i.e. MISRA-C

Summary of AUTOSAR

- Complex software; will contain bugs/vulnerabilities
- Made by different vendors / developers
  - Do you trust your suppliers?
- Mature code due to safety requirements
  - i.e. MISRA-C

Mature! But not guaranteed secure...

What can go wrong?

Potential MCAL vulnerabilities

```
Std_ReturnType Fls_Read(    Fls_AddressType SourceAddress,
                            uint8 *TargetAddressPtr,
                            Fls_LengthType Length)
{
    if(SourceAddress == NULL || TargetAddressPtr == NULL || SourceAddress >= FLS_SIZE) {
        return E_NOT_OK;
    }

    /* potential integer overflow */
    if (Length == 0U || (SourceAddress + Length) > FLS_SIZE) {
        return E_NOT_OK;
    }
    ...
}
```

Potential MCAL vulnerabilities

```
Std_ReturnType Fls_Read(    Fls_AddressType SourceAddress,
                            uint8 *TargetAddressPtr,
                            Fls_LengthType Length)
{
    if(SourceAddress == NULL || TargetAddressPtr == NULL || SourceAddress >= FLS_SIZE) {
        return E_NOT_OK;
    }

    /* potential integer overflow */
    if (Length == 0U || (SourceAddress + Length) > FLS_SIZE) {
        return E_NOT_OK;
    }
    ...
}
```

Potential MCAL vulnerabilities

```
Std_ReturnType Fls_Read(    Fls_AddressType SourceAddress,
                            uint8 *TargetAddressPtr,
                            Fls_LengthType Length)
{
    if(SourceAddress == NULL || TargetAddressPtr == NULL || SourceAddress >= FLS_SIZE) {
        return E_NOT_OK;
    }

    /* potential integer overflow */
    if (Length == 0U || (SourceAddress + Length) > FLS_SIZE) {
        return E_NOT_OK;
    }
    ...
}
```

Who verifies your MCAL for vulnerabilities?

What about MISRA-C?!

What about MISRA-C?!

Dir 4.1	Run-time failures shall be minimized
	C90 [Undefined 15, 19, 26, 30, 31, 32, 94] C99 [Undefined 15, 16, 33, 40, 43–45, 48, 49, 113]
Category	Required
Applies to	C90, C99

Dir 4.14	The validity of values received from external sources shall be checked
	C90 [Undefined 15, 19, 26, 30, 31, 32, 94] C99 [Undefined 15, 16, 33, 40, 43-45, 48, 49, 113]
Category	Required
Applies to	C90, C99

What about MISRA-C?!

Dir 4.1 Run-time failures shall be minimized

C90 [Undefined 15, 19, 26, 30, 31, 32, 94]
C99 [Undefined 15, 16, 33, 40, 43–45, 48, 49, 113]

Category Required
Applies to C90, C99

Dir 4.14 The validity of values received from external sources shall be checked

C90 [Undefined 15, 19, 26, 30, 31, 32, 94]
C99 [Undefined 15, 16, 33, 40, 43-45, 48, 49, 113]

Category Required
Applies to C90, C99

A directive is a guideline for which it is not possible to provide the full description necessary to perform a check for compliance. Additional information, such as might be provided in design documents or requirements specifications, is required in order to be able to perform the check. Static analysis tools may be able to assist in checking compliance with directives but different tools may place widely different interpretations on what constitutes a non-compliance.

You cannot conform to directives automagically...

What else?

Vulnerabilities in complex software

Vulnerabilities in complex software

Vulnerabilities in complex software

Who verifies your communication stack?

Mitigating software vulnerabilities

Mitigating software vulnerabilities

- Minimize the low hanging fruit
  - *Secure coding standard, code checkers, …*

Mitigating software vulnerabilities

- Minimize the low hanging fruit
  - *Secure coding standard, code checkers, …*
- Find vulnerabilities yourself before attackers do
  - *Continuous security code reviews, …*

Mitigating software vulnerabilities

- Minimize the low hanging fruit
  - *Secure coding standard, code checkers, …*
- Find vulnerabilities yourself before attackers do
  - *Continuous security code reviews, …*
- Make it harder to exploit software vulnerabilities
  - *Software exploitation mitigations, …*

Sufficiently complex software has vulnerabilities.

Sufficiently complex software
has vulnerabilities.

Finding them is not always trivial...

Sufficiently complex software
has vulnerabilities.

Finding them is not always trivial...

What if an attacker cannot find any
or they are too difficult to exploit?

Hardware attacks!?

Hardware attacks!?

- Attacker needs physcial access the ECU

Hardware attacks!?

- Attacker needs physcial access the ECU
- Attacker often needs to open the ECU

Hardware attacks!?

- Attacker needs physcial access the ECU
- Attacker often needs to open the ECU
- Different types of HW attacks:
  - E.g. PCB-level, Fault injection, Side Channels, etc.

Hardware attacks!?

- Attacker needs physcial access the ECU
- Attacker often needs to open the ECU
- Different types of HW attacks:
  - E.g. PCB-level, Fault injection, Side Channels, etc.
- Often a stepping stone for more scalable attacks

Case study: FI on AUTOSAR

“Using FI to take control of an AUTOSAR-based ECU.”

Case study: FI on AUTOSAR

“Using FI to take control of an AUTOSAR-based ECU.”

- Demonstration ECU implemented using:
  - STM32F4 development board
  - Arctic Core for AUTOSAR v3.1

Case study: FI on AUTOSAR

"Using FI to take control of an AUTOSAR-based ECU."

- Demonstration ECU implemented using:
  - STM32F4 development board
  - Arctic Core for AUTOSAR v3.1
- Attacking using a previously described FI fault model

Case study: FI on AUTOSAR

"Using FI to take control of an AUTOSAR-based ECU."

- Demonstration ECU implemented using:
  - STM32F4 development board
  - Arctic Core for AUTOSAR v3.1
- Attacking using a previously described FI fault model

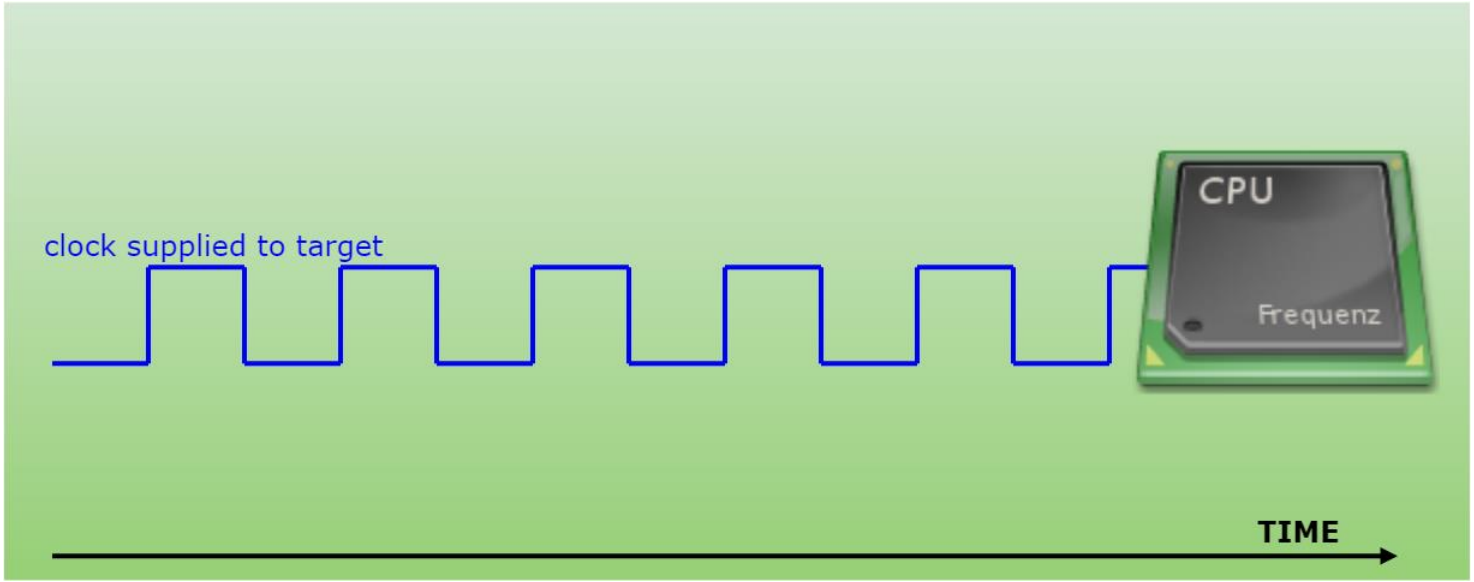
Fault Injection? Fault model?

Voltage Fault Injection

"Introducing faults into a chip in order to change its intented behavior."

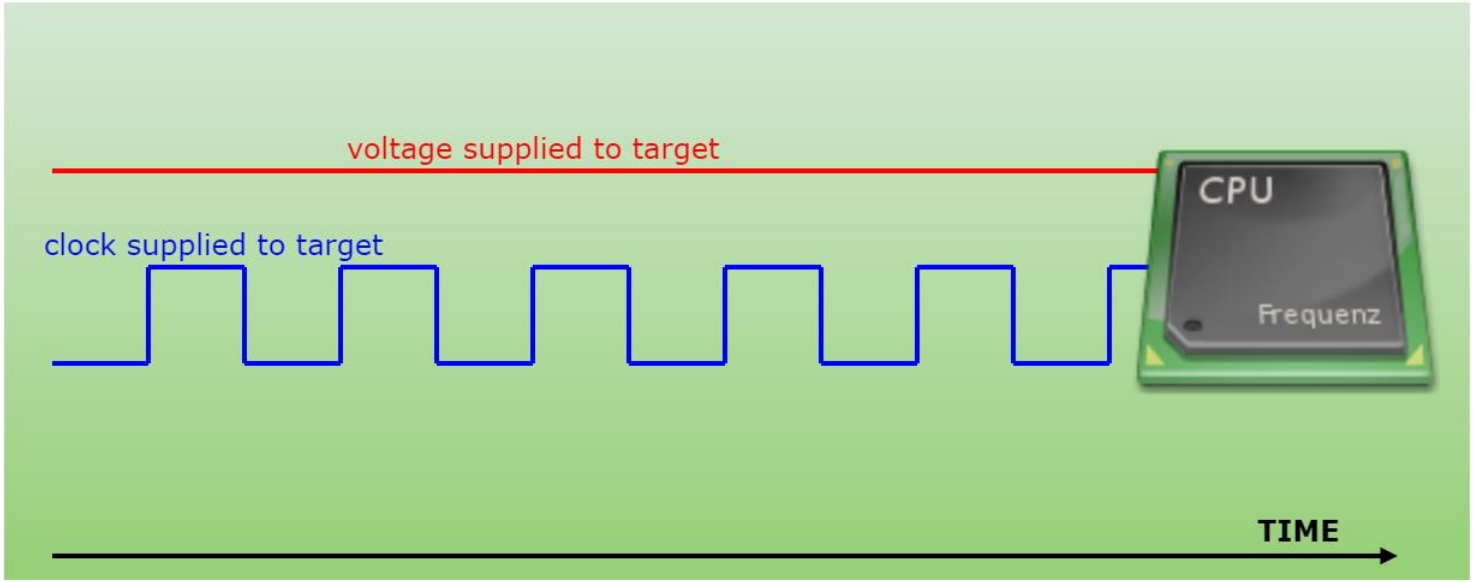
Voltage Fault Injection

"Introducing faults into a chip in order to change its intented behavior."



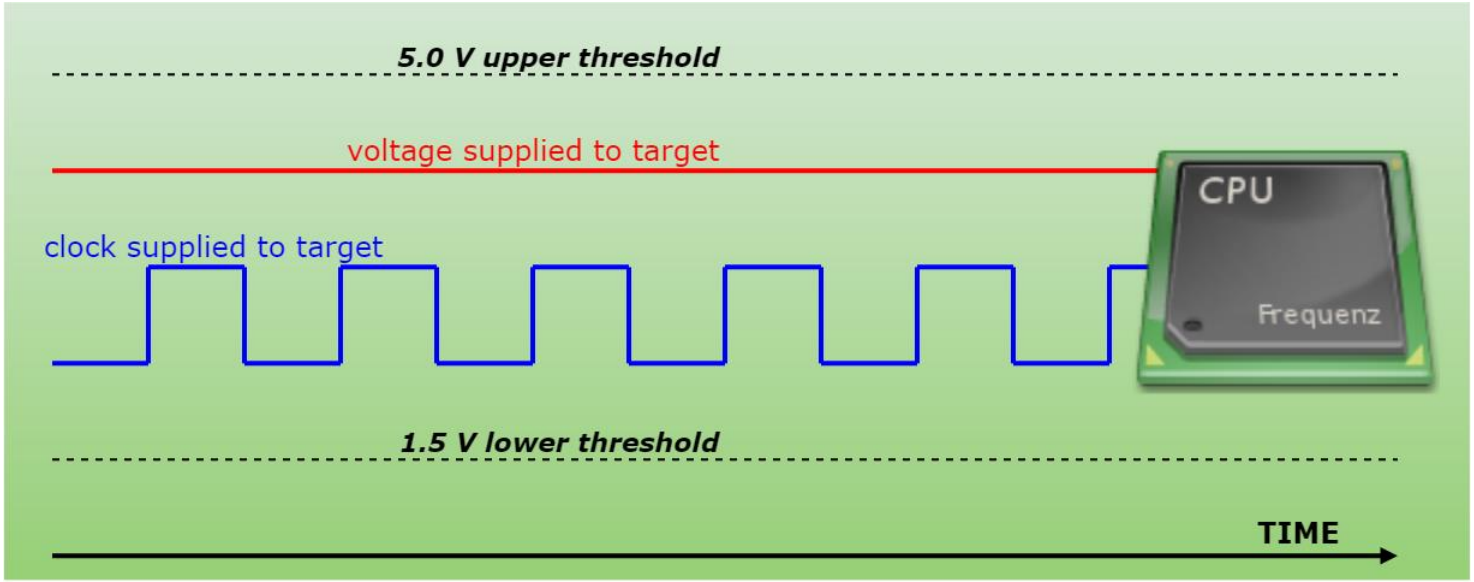
Voltage Fault Injection

"Introducing faults into a chip in order to change its intented behavior."



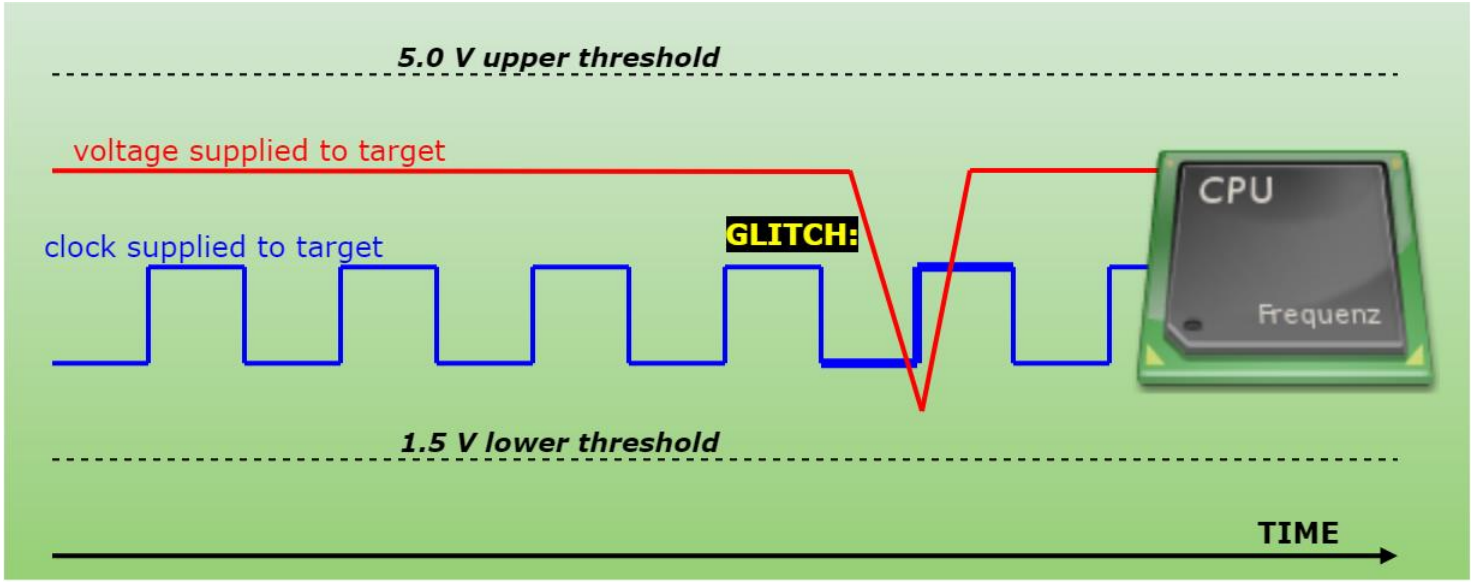
Voltage Fault Injection

"Introducing faults into a chip in order to change its intended behavior."



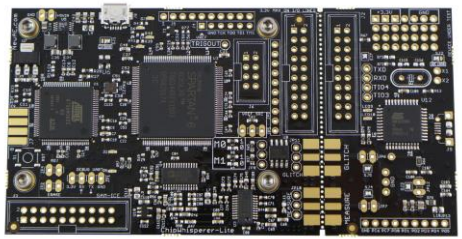
Voltage Fault Injection

"Introducing faults into a chip in order to change its intented behavior."



Fault Injection Setup

Fault Injection Setup

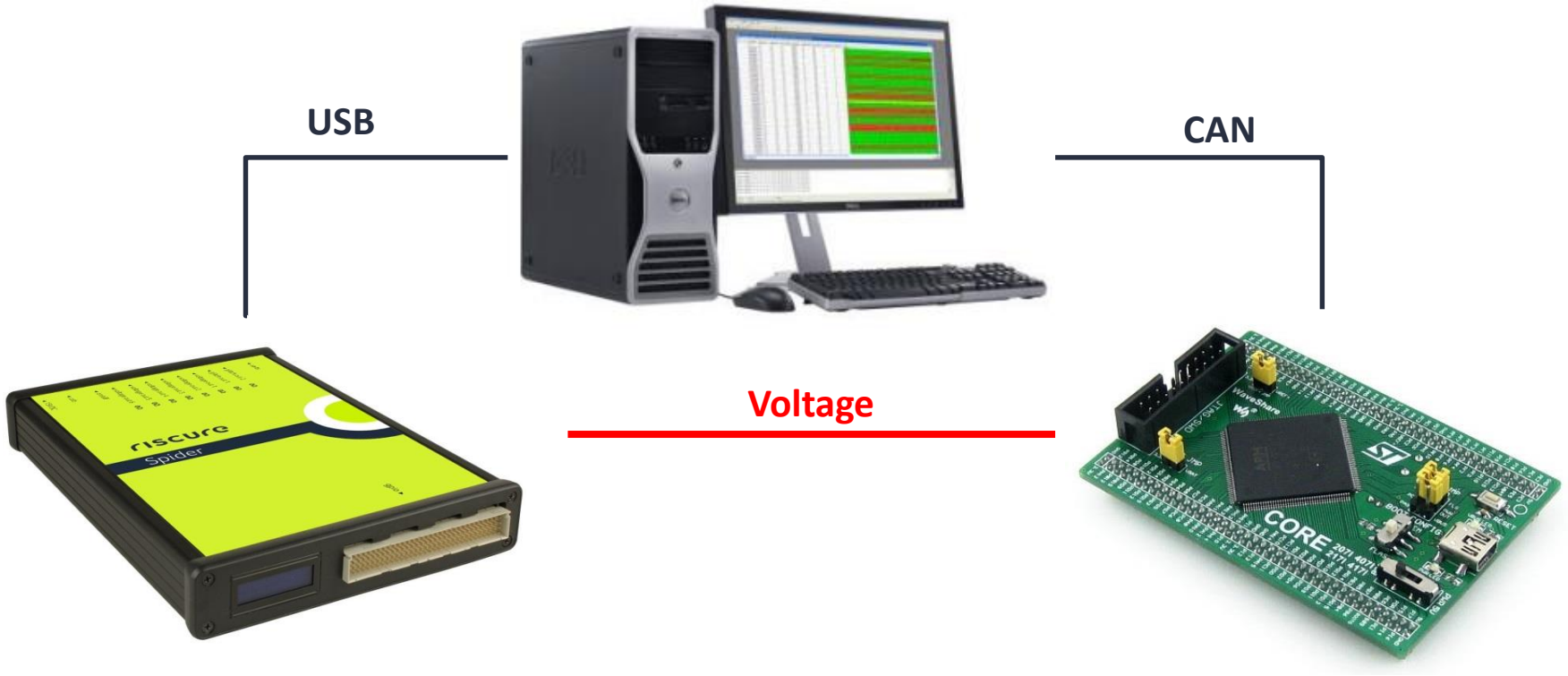


Fault Injection Setup

Fault Injection Setup

Fault Injection Setup

Fault Injection Setup



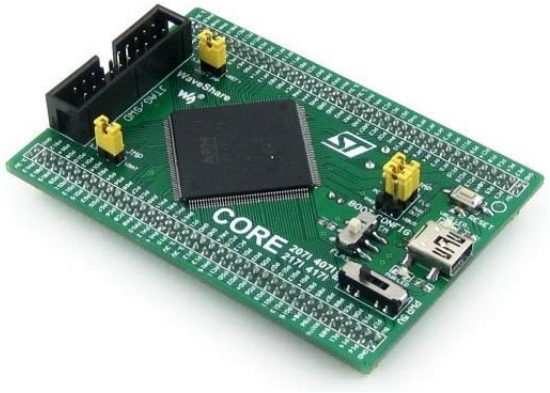
Fault Injection Setup

USB

CAN

Voltage

Reset



What can we do with fault injection?

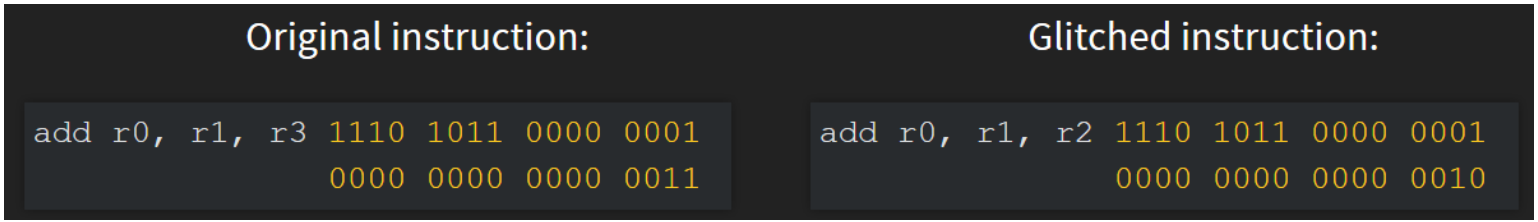
Fault Injection Fault Model

“Instruction corruption.”

Fault Injection Fault Model

“Instruction corruption.”

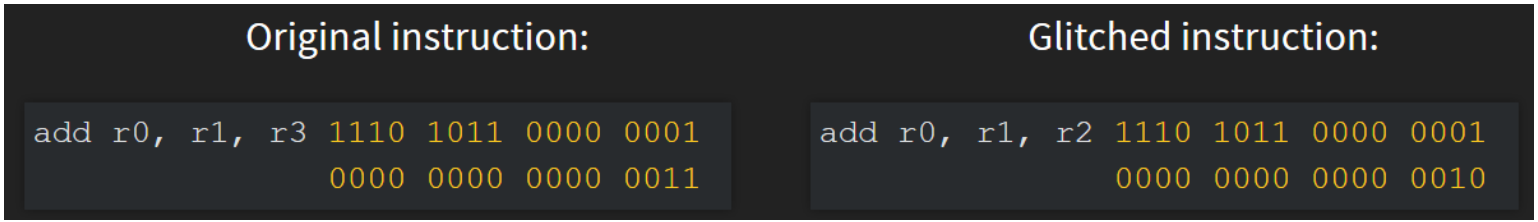
- Glitches can be used to modify instruction



Fault Injection Fault Model

"Instruction corruption."

- Glitches can be used to modify instruction

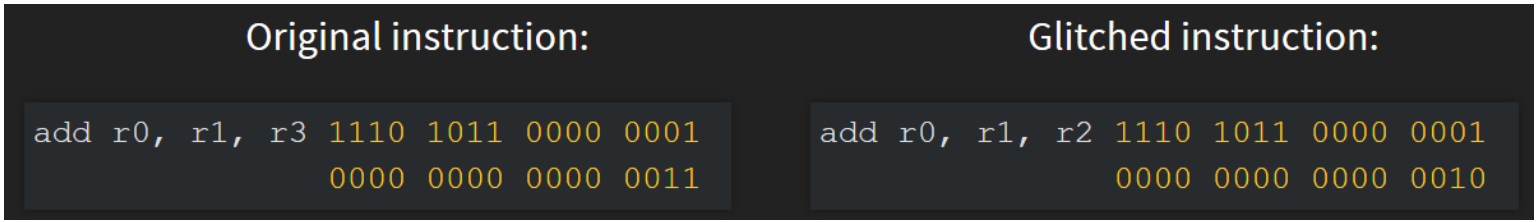


- In other words, we can modify software

Fault Injection Fault Model

"Instruction corruption."

- Glitches can be used to modify instruction



- In other words, we can modify software
- Fault injection breaks any software security model

How can we use this to attack AUTOSAR-based ECUs?

Attacking AUTOSAR

Attacking AUTOSAR

- Our goal is to execute arbitrary code

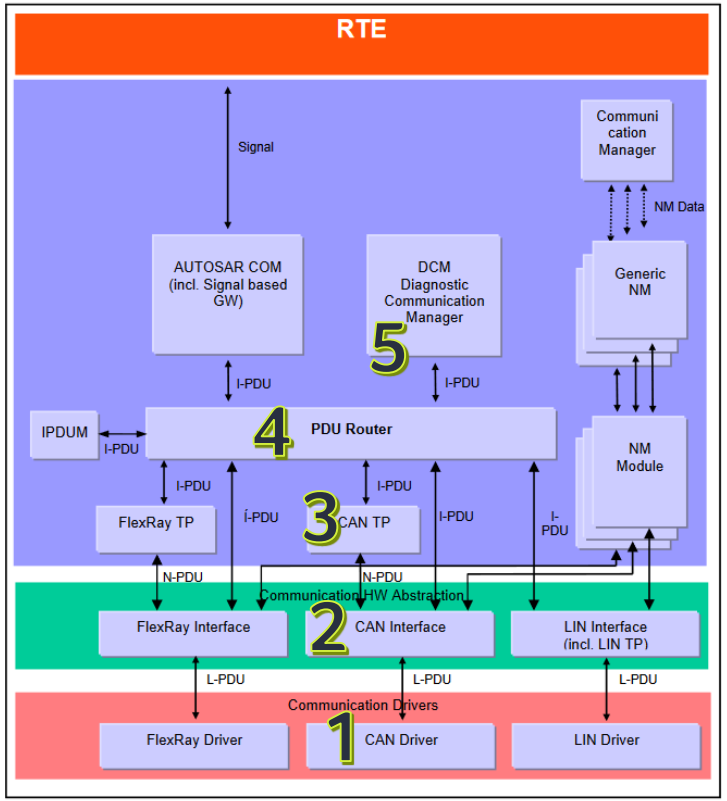
Attacking AUTOSAR

- Our goal is to execute arbitrary code
- Our only entry into the device is the CAN bus

Attacking AUTOSAR

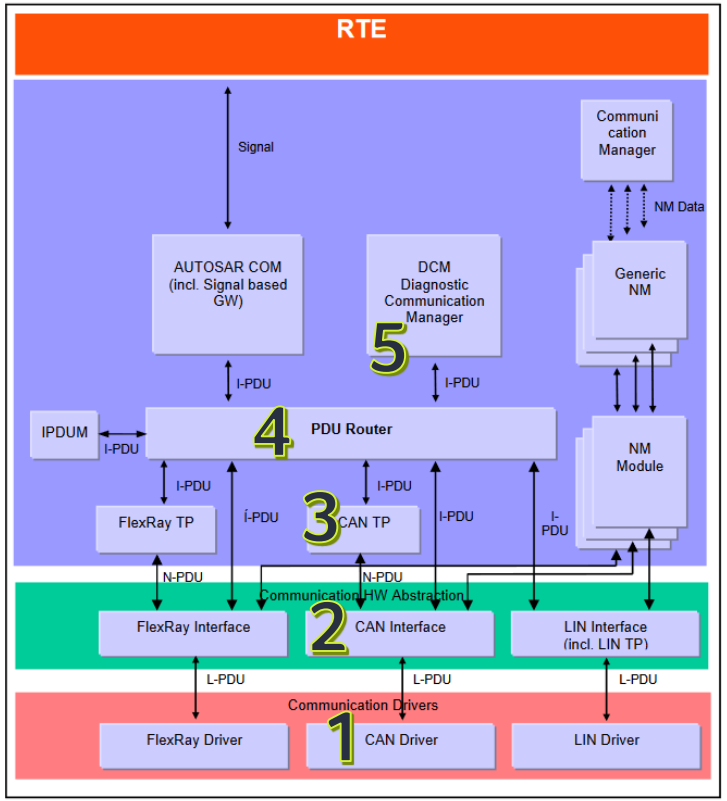
- Our goal is to execute arbitrary code
- Our only entry into the device is the CAN bus
- Of course, we have physical access...

AUTOSAR's PDU Router



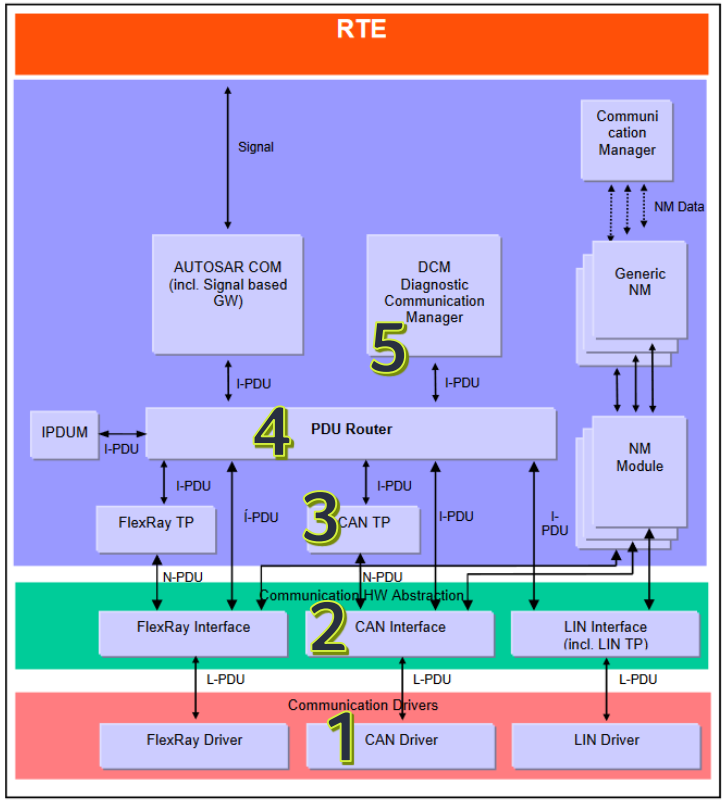
AUTOSAR's PDU Router

1. CAN driver receives 8-byte CAN frame



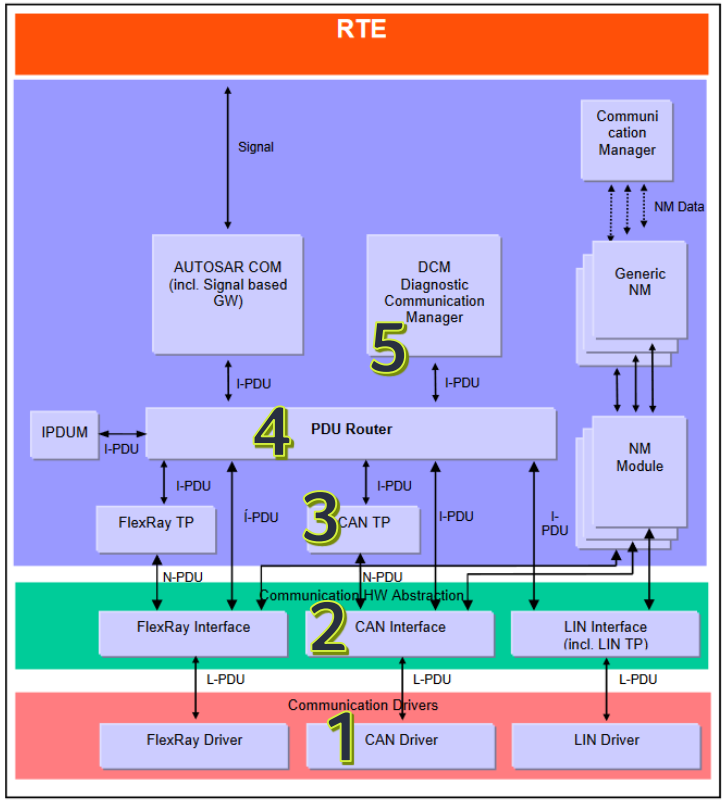
AUTOSAR's PDU Router

1. CAN driver receives 8-byte CAN frame
2. Frame passes the CAN interface



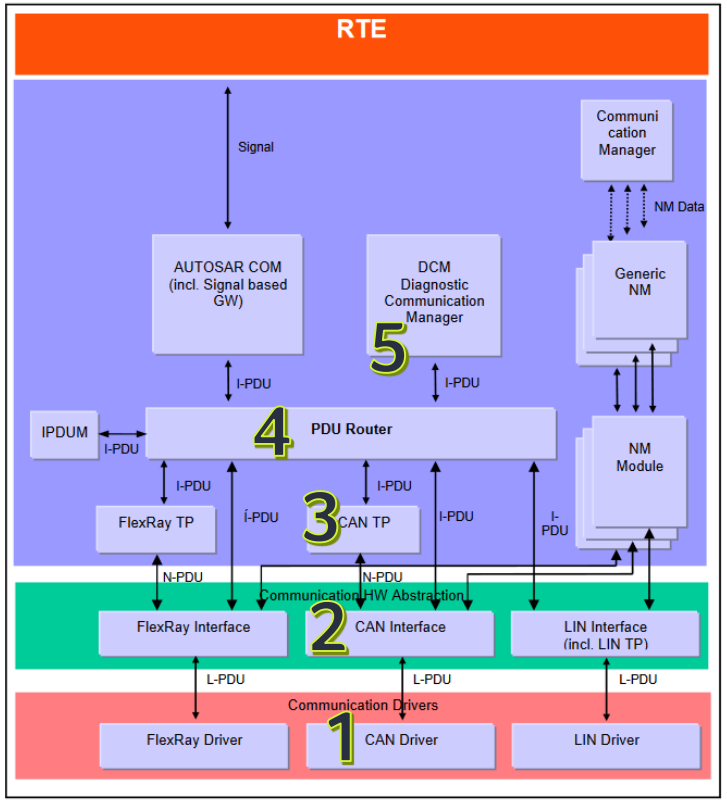
AUTOSAR's PDU Router

1. CAN driver receives 8-byte CAN frame
2. Frame passes the CAN interface
3. Payload is reassembled by ISO-TP



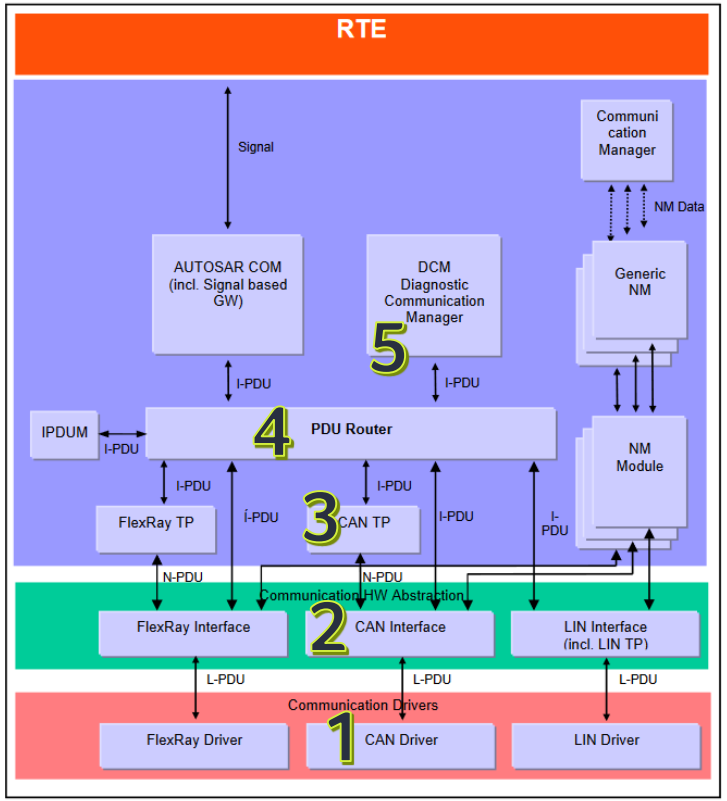
AUTOSAR's PDU Router

1. CAN driver receives 8-byte CAN frame
2. Frame passes the CAN interface
3. Payload is reassembled by ISO-TP
4. Payload is copied to COM or DCM



AUTOSAR's PDU Router

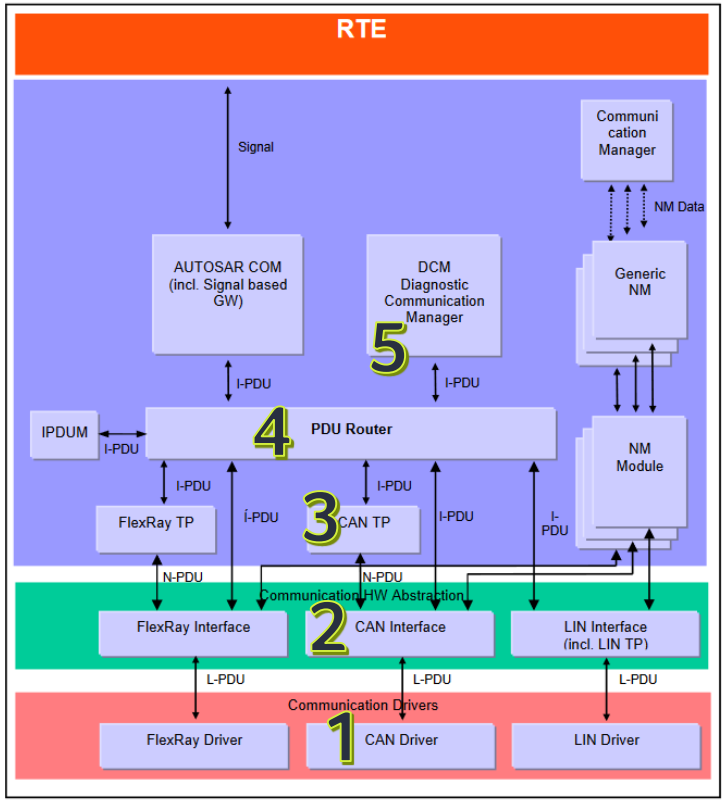
1. CAN driver receives 8-byte CAN frame
2. Frame passes the CAN interface
3. Payload is reassembled by ISO-TP
4. Payload is copied to COM or DCM
5. COM or DCM handles the payload



Where do we attack?!

AUTOSAR's PDU Router

1. CAN driver receives 8-byte CAN frame
2. Frame passes the CAN interface
3. Payload is reassembled by ISO-TP
4. **Payload is copied to COM or DCM**
5. COM or DCM handles the payload



Attacking AUTOSAR’s PDU router

Attacking AUTOSAR's PDU router

- **Step 1:** Send an ISO-TP CAN message (< 4096 bytes)

Attacking AUTOSAR's PDU router

- **Step 1:** Send an ISO-TP CAN message (< 4096 bytes)

Attacking AUTOSAR's PDU router

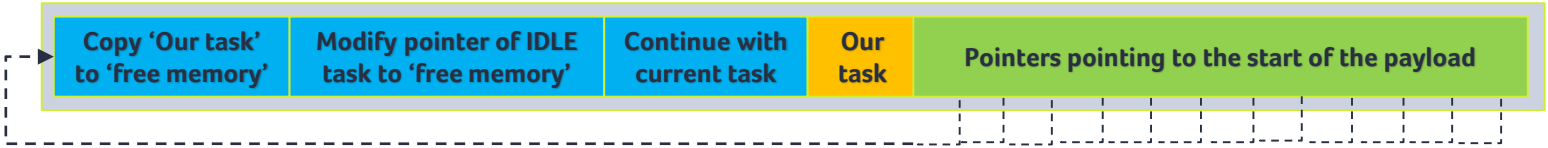
- **Step 1:** Send an ISO-TP CAN message (< 4096 bytes)

Attacking AUTOSAR's PDU router

- **Step 1:** Send an ISO-TP CAN message (< 4096 bytes)

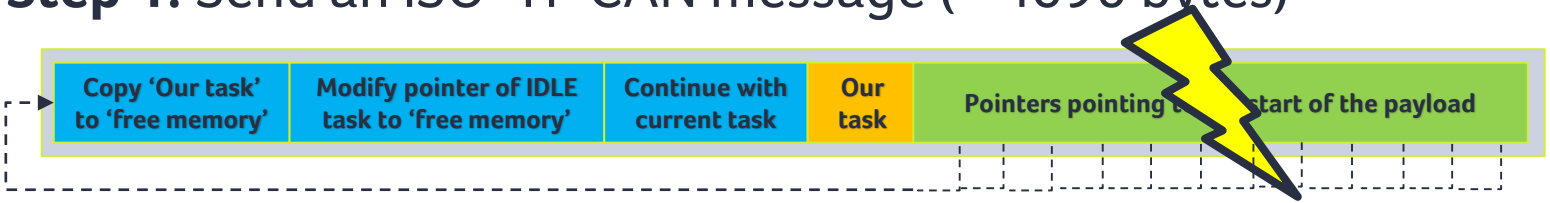
Attacking AUTOSAR's PDU router

- **Step 1:** Send an ISO-TP CAN message (< 4096 bytes)



Attacking AUTOSAR's PDU router

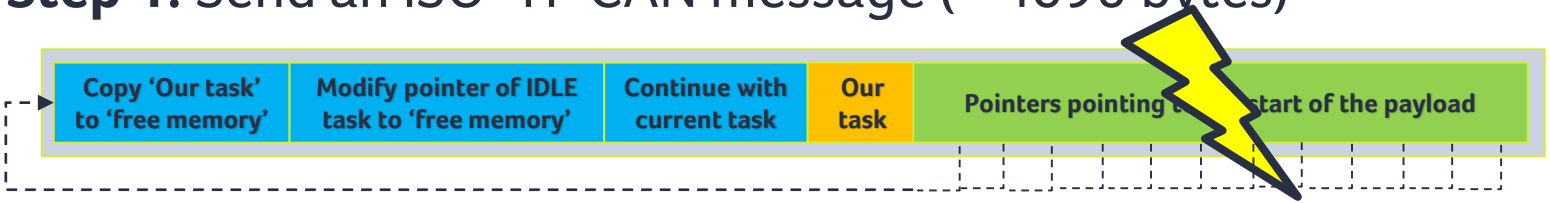
- **Step 1:** Send an ISO-TP CAN message (< 4096 bytes)



- **Step 2:** We inject the glitch when the pointers are being copied

Attacking AUTOSAR's PDU router

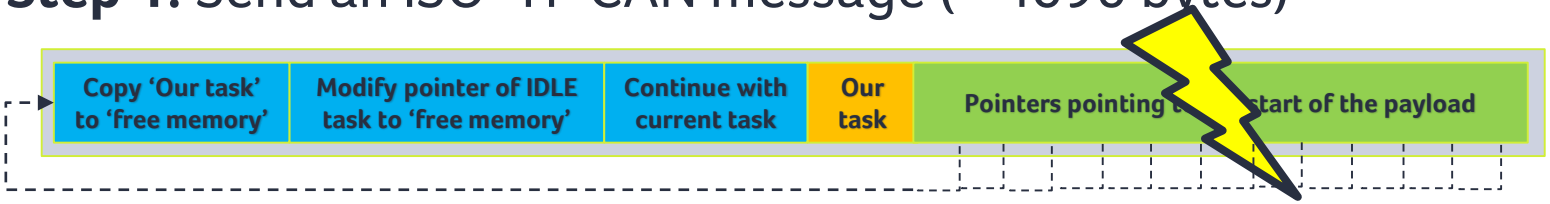
- **Step 1:** Send an ISO-TP CAN message (< 4096 bytes)



- **Step 2:** We inject the glitch when the pointers are being copied
- **Step 3:** Successful glitches load a pointer into the PC register

Attacking AUTOSAR's PDU router

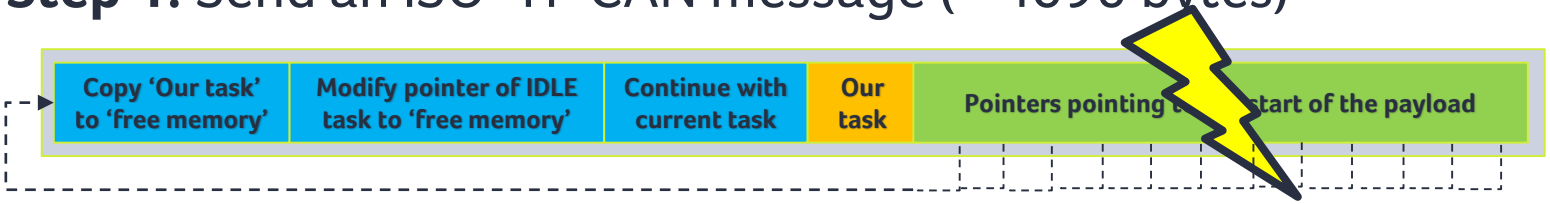
- **Step 1:** Send an ISO-TP CAN message (< 4096 bytes)



- **Step 2:** We inject the glitch when the pointers are being copied
- **Step 3:** Successful glitches load a pointer into the PC register
- **Step 4:** MCU will execute the ISO-TP message (blue blocks)

Attacking AUTOSAR's PDU router

- **Step 1:** Send an ISO-TP CAN message (< 4096 bytes)



- **Step 2:** We inject the glitch when the pointers are being copied
- **Step 3:** Successful glitches load a pointer into the PC register
- **Step 4:** MCU will execute the ISO-TP message (blue blocks)
- **Step 5:** Wait for IDLE task to be scheduled and execute our task

Why does this work?

Attacking AUTOSAR's PDU Router

```
BufReq_ReturnType PduR_CanTpCopyRxData(..., PduInfoType* p, ...) {
    ...
    memcpy(p->SduDataPtr, source, p->SduLength);
    ...
}
```

Attacking AUTOSAR's PDU Router

```
BufReq_ReturnType PduR_CanTpCopyRxData(..., PduInfoType* p, ...) {
    ...
    memcpy(p->SduDataPtr, source, p->SduLength);
    ...
}
```

Disassembled memcpy()

```
<memcpy>:
  mov     ip, r0
  orr.w   r3, r1, r0
  ands.w  r3, r3, #3
  bne.n   8008300 <memcpy+0xe8>
  subs    r2, #64 ; 0x40
  bcc.n   80082ac <memcpy+0x94>
  ldr.w   r3, [r1], #4
  str.w   r3, [r0], #4
  subs    r2, #64 ; 0x40
  bcs.n   8008228 <memcpy+0x10>
  adds    r2, #48 ; 0x30
  bcc.n   80082d4 <memcpy+0xbc>
```

Attacking AUTOSAR's PDU Router

```
BufReq_ReturnType PduR_CanTpCopyRxData(..., PduInfoType* p, ...) {
    ...
    memcpy(p->SduDataPtr, source, p->SduLength);
    ...
}
```

Disassembled memcpy()

```
<memcpy>:
  mov     ip, r0
  orr.w   r3, r1, r0
  ands.w  r3, r3, #3
  bne.n   8008300 <memcpy+0xe8>
  subs    r2, #64 ; 0x40
  bcc.n   80082ac <memcpy+0x94>
  ldr.w   r3, [r1], #4
  str.w   r3, [r0], #4
  subs    r2, #64 ; 0x40
  bcs.n   8008228 <memcpy+0x10>
  adds    r2, #48 ; 0x30
  bcc.n   80082d4 <memcpy+0xbc>
```



```
<memcpy>:
  mov     ip, r0
  orr.w   r3, r1, r0
  ands.w  r3, r3, #3
  bne.n   8008300 <memcpy+0xe8>
  subs    r2, #64 ; 0x40
  bcc.n   80082ac <memcpy+0x94>
  ldr.w   pc, [r1], #4
  str.w   r3, [r0], #4
  subs    r2, #64 ; 0x40
  bcs.n   8008228 <memcpy+0x10>
  adds    r2, #48 ; 0x30
  bcc.n   80082d4 <memcpy+0xbc>
```

Attacking AUTOSAR's PDU Router

```
BufReq_ReturnType PduR_CanTpCopyRxData(..., PduInfoType* p, ...) {
    ...
    memcpy(p->SduDataPtr, source, p->SduLength);
    ...
}
```

Disassembled memcpy()

```
<memcpy>:
  mov     ip, r0
  orr.w   r3, r1, r0
  ands.w  r3, r3, #3
  bne.n   8008300 <memcpy+0xe8>
  subs    r2, #64 ; 0x40
  bcc.n   80082ac <memcpy+0x94>
  ldr.w   r3, [r1], #4
  str.w   r3, [r0], #4
  subs    r2, #64 ; 0x40
  bcs.n   8008228 <memcpy+0x10>
  adds    r2, #48 ; 0x30
  bcc.n   80082d4 <memcpy+0xbc>
```



```
<memcpy>:
  mov     ip, r0
  orr.w   r3, r1, r0
  ands.w  r3, r3, #3
  bne.n   8008300 <memcpy+0xe8>
  subs    r2, #64 ; 0x40
  bcc.n   80082ac <memcpy+0x94>
  ldr.w   pc, [r1], #4
  str.w   r3, [r0], #4
  subs    r2, #64 ; 0x40
  bcs.n   8008228 <memcpy+0x10>
  adds    r2, #48 ; 0x30
  bcc.n   80082d4 <memcpy+0xbc>
```

We take control of the Program Counter (PC) during the copy!

We have our own task. Now what?!

Post Exploitation

Post Exploitation

- Extract information (secrets)

Post Exploitation

- Extract information (secrets)
- Analyze firmware dynamically

Post Exploitation

- Extract information (secrets)
- Analyze firmware dynamically
- Perform additional attacks (e.g. side channel attack)

Post Exploitation

- Extract information (secrets)
- Analyze firmware dynamically
- Perform additional attacks (e.g. side channel attack)
- Add (malicious) and/or change functionality

Is all hope lost?

Hardening AUTOSAR-based ECUs

Hardening AUTOSAR-based ECUs

- Adhere to (automotive) security guidelines/standards

Hardening AUTOSAR-based ECUs

- Adhere to (automotive) security guidelines/standards
- Make use of strong (hardware-based) security

Hardening AUTOSAR-based ECUs

- Adhere to (automotive) security guidelines/standards
- Make use of strong (hardware-based) security
- Minimize attack surface and increase attack complexity

Hardening AUTOSAR-based ECUs

- Adhere to (automotive) security guidelines/standards
- Make use of strong (hardware-based) security
- Minimize attack surface and increase attack complexity
- Consult internal/external embedded security experts

To wrap up...

Takeaways

Takeaways

- Devices (incl. AUTOSAR-based ECUs) will be hacked

Takeaways

- Devices (incl. AUTOSAR-based ECUs) will be hacked
  - Not AUTOSAR's fault!

Takeaways

- Devices (incl. AUTOSAR-based ECUs) will be hacked
  - Not AUTOSAR's fault!
- No (known) software vulnerabilities ≠ secure

Takeaways

- Devices (incl. AUTOSAR-based ECUs) will be hacked
  - Not AUTOSAR's fault!
- No (known) software vulnerabilities ≠ secure
- Hardware attacks are efficient and do scale

Thank you. Questions?

Niek Timmers
niek@riscure.com / @tieknimmers

Riscure B.V.
Frontier Building, 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@riscure.com

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

inforcn@riscure.com

riscure

Challenge your security