Oops..! I Glitched it Again!
How to Multi-Glitch the Glitching-Protections on ARM TrustZone-M

Marvin Saß, Richard Mitev, Ahmad-Reza Sadeghi
System Security Lab
Technical University of Darmstadt, Germany
Motivation - Countermeasures

**DCFG_CC_SOCU**: Credential constraints:

DCFG_CC_SOCU is a bit mask that specifies debug access rights. It is derived from combination of PFR words CMPA_CC_SOCU_DFLT, CMPA_CC_SoCU_PIN, CFPA_CC_SOCU_DFLT_NS, CFPA_CC_SoCU_PIN_NS:

- Lower half-words of these PFR words define the functionality.

**48.10.6.24** IDXBLK_L_DP register

This register is duplicate of IDXBLK_L register and provides protection against malicious.

**48.10.6.21** Index blocking duplicate register (IDX8 - IDX15)

This register is duplicate of IDXBLK_H register and provides protection against malicious.

**47.4.74** Master secure level anti-pole register

This register is inverse of MASTER_SEC_LEVEL register above. Secondary register with

**47.4.77** Secure control duplicate register

This register is duplicate of MISC_CTRL_REG. A secondary register with duplicate programing is implemented to provide better protection against malicious hacking attacks such as glitch attack.
Consequence

Modify Register  →  Direct Memory Access  →  Read Secret Keys  →  Access TrustZone
Duplicated Registers

```
0x500acff8:  0x00000000
0x500acfffc: 0x00000000

ldr    r1, =0x500acff8
movw   r2, #0xaaa5
str    r2, [r1]
...
```

```
ldr    r1, =0x500acfffc
movw   r2, #0xaaa5
str    r2, [r1]
```
Duplicated Registers

0x500acff8: 0x00000000
0x500acfffc: 0x00000000

ldr r1, =0x500acff8
movw r2, #0xaaa5
str r2, [r1]
...

ldr r1, =0x500acfffc
movw r2, #0xaaa5
str r2, [r1]
Duplicated Registers

0x500acff8: 0x0000aaa5
0x500acff8: 0x0000aaa5

```
ldr r1, =0x500acff8
movw r2, #0xaaa5
str r2, [r1]
...
ldr r1, =0x500acff8
movw r2, #0xaaa5
str r2, [r1]
```
Duplicated Registers

`0x500acff8: 0x00000000`  
`0x500acfffc: 0x00000000`

```
ldr r1, =0x500acff8
movw r2, #0xaaa5
str r2, [r1]
ldr r1, =0x500acfffc
movw r2, #0xaaa5
str r2, [r1]
```

`time`  
`voltage`

Diagram:

```
ldr r1, =0x500acff8
movw r2, #0xaaa5
str r2, [r1]
...  
ldr r1, =0x500acfffc
movw r2, #0xaaa5
str r2, [r1]
```
Duplicated Registers

0x500acff8: 0x00000000
0x500acffc: 0x00000000

ldr r1, =0x500acff8
movw r2, #0xaaa5
str r2, [r1]

ldr r1, =0x500acffc
movw r2, #0xaaa5
str r2, [r1]

...
Duplicated Registers

0x500acff8: 0x00000000
0x500acfffc: 0x0000aaa5

ldr r1, =0x500acff8
movw r2, #0xaaa5
str r2, [r1]

ldr r1, =0x500acfffc
movw r2, #0xaaa5
str r2, [r1]
Duplicated Registers

0x500acff8: 0x00000000

0x500acff8: 0x0000aaa5

Original != Duplicate

ldr r1, =0x500acff8
movw r2, #0xaaa5
str r2, [r1]
...
ldr r1, =0x500acff8
movw r2, #0xaaa5
str r2, [r1]
Can these Fault Injection countermeasures be evaded by “simply” injecting another fault?
How secure are “glitch protected embedded processors” against multiple fault injection?
Related Work

- **Kömmerling et al. USENIX**
  Design Principles for Tamper-Resistant Smartcard Processors

- **Barenghi et al. WESS**
  Countermeasures against fault attacks on software implemented AES: effectiveness and cost

- **O’Flynn**
  Fault Injection using Crowbars on Embedded Systems

- **Zussa et al. HOST**
  Analysis of the fault injection mechanism related to negative and positive power supply glitches using an on-chip voltmeter

- **Bozzato et al. TCHES**
  Shaping the Glitch: Optimizing Voltage Fault Injection Attacks

- **Roth 36C3**
  TrustZone-M(eh): Breaking ARMv8-M's security

- **Kenjar et al. CCS**
  V0LTpwn: Attacking x86 Processor Integrity from Software
Most Related Work

Core

MPU
SAU
IDAU

Other Bus-Master

TZ-Wrapper

AHB

Securable AHB Slaves

TZ-Gate
RAM
TZ-Gate
Flash
TZ-Gate
Peripherals

Address
Privilege Level
Security State

(e.g., STM32L5, M2351, SAML11)
Most Related Work

(e.g., STM32L5, M2351, SAML11)
The last SAU configuration example combines both previous approaches together with an AHB Secure controller. The advantage of basic SAU configuration together with enabled AHB Secure controller is a cross checking of the bus transaction. The first check is done at the core level (SAU/IDAU) while the second check is performed at the system level (AHB Secure controller). If some inconsistency is detected between the SAU and AHB Secure controller configurations (due to some software error or glitch attack), access to specific resource is blocked. So, by employing both SAU and AHB Secure controller for TrustZone isolation makes isolation more robust when compared to a basic SAU configuration shown in Figure 266 "TrustZone isolation".

(e.g., STM32L5, M2351, SAML11)
Multiple Fault Injection is required to fully break TZ-M!

(e.g., STM32L5, M2351, SAML11)
Multiple Voltage Fault Hardware
MVFI Engine

Due to the multiplexers, we are able to change the number of injected faults even after hardware synthesis.
MVFI Parameter Search
Brute-Force Double-Fault Parameter Search
Brute-Force Double-Fault Parameter Search

Problem
Search space increases exponentially!
Brute-Force Double-Fault Parameter Search

**Problem**
Search space increases exponentially!

**Solution**
Differentiate hit fault targets!
Our Approach

1. Overall Success Function
2. Partial Success Functions
3. Sweep through search space
4. Translation
5. Fuzzyfication
6. Integration
7. Evaluation
Our Approach

OSF determines if all fault targets have been hit (Overall Success)
Our Approach

PSFs determine if a specific fault target has been hit (Partial Success)
Our Approach

Use a single voltage fault to sweep through search space, record partial successes
Our Approach

Translate absolute parameters into relative ones
Our Approach

Fuzzify parameter ranges to counter uncertainty
Our Approach

Combined brute-force search on small ranges generated by fuzzification
Our Approach

Evaluate findings from integration, check for repeatability
System Overview

TZ-M Setup Code
Partial Success Function

MFI Engine
Glitch Signal
Trigger
Target I/O

DuT

V

MFI Engine
Glitch Signal
Trigger

DuT

Re-use found parameters

TZ-M Setup Code
Our attack: Disabling TZ-M
Finding Parameters

![Graph showing voltage fault width (10ns) vs. voltage fault offset (10ns)]

- **Unaffected**
- **Crashes**
- **Original Register**
- **Duplicate Register**
Finding Parameters

- Unaffected
- Crashes
- Original Register
- Duplicate Register
Finding Parameters

Voltage Level before first fault

Voltage Level before second fault

Slow recover after FI
Attacking NXPs LPC55SXX & RT6XX

Diagram showing the structure of the system with components such as Core, MPU, SAU, IDAU, Other Bus-Master, TZ-Gate, TZ-Wrapper, AHB, RAM, Flash, and Peripherals. The diagram includes color codes for Address, Privilege Level, and Security State.
Attacking NXPs LPC55SXX & RT6XX

- MPU
- SAU
- IDAU
- Other Bus-Master
- TZ-Gate
- TZ-Gate
- TZ-Gate
- TZ-Wrapper
- AHB
- Securable AHB Slaves
  - RAM
  - Flash
  - Peripherals

Legend:
- Address
- Privilege Level
- Security State
Attacking NXP's LPC55SXX & RT6XX

Core

MPU

SAU

IDAU

Other Bus-Master

TZ-Wrapper

AHB

Securable AHB Slaves

TZ-Gate

RAM

TZ-Gate

Flash

TZ-Gate

Peripherals

Address Privilege Level Security State
Attacking NXPs LPC55SXX & RT6XX

- Core
- MPU
- SAU
- IDAU
- Other Bus-Master
- TZ-Wrapper
- AHB
- Securable AHB Slaves
  - TZ-Gate
  - RAM
  - TZ-Gate
  - Flash
  - TZ-Gate
  - Peripherals

- Address
- Privilege Level
- Security State
Attacking NXPs LPC55SXX & RT6XX

- MPU
- SAU
- IDAU
- TZ-Wrapper
- Core
- AHB
- Other Bus-Master
- TZ-Gate
- RAM
- Flash
- Peripherals

Address
Privilege Level
Security State
Attacking NXPs LPC55SXX & RT6XX

- MPU
- Other Bus-Master
- Core
- SAU
- IDAU
- TZ-Wraper
- AHB
- SAU:
  - 45.1%
- SAU & Orig.:
  - 2.52%
- SAU & Orig. & Dupl.:
  - 0.23%
- SAU & Orig. & Dupl. & Priv.:
  - 0.0003%

Security State

- Address
- Privilege Level

Securable AHB Slaves

- RAM
- Flash
- Peripherals
Thanks!