This work is distributed under the CC BY-SA license, unless stated otherwise.

Hardware Security

12. Fault attacks (tutorial)

Ján Jančár

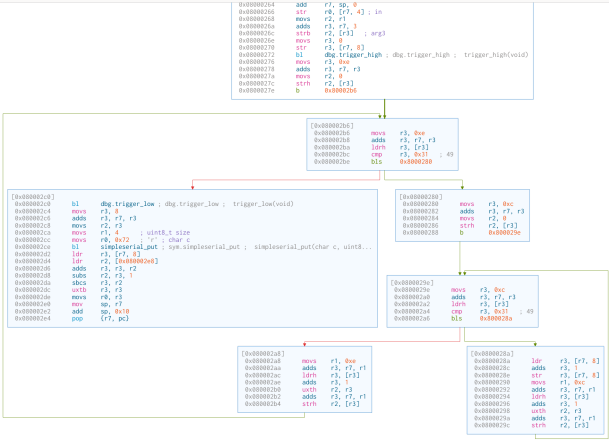
Summer 2026

Today

- Targets
- Voltage glitching
- ChipWhisperer Nano

Targets

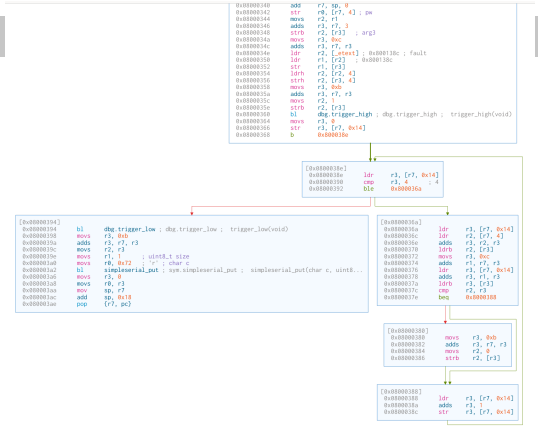
Loop



```
cnt = 0;
trigger_high();
for(i=0; i<10; i++){
    for(j=0; j<10; j++){
        cnt++;
    }
}
trigger_low();
simpleserial_put("r", 4, (uint8_t*)&cnt);
return cnt == 100;
```

Targets

Password

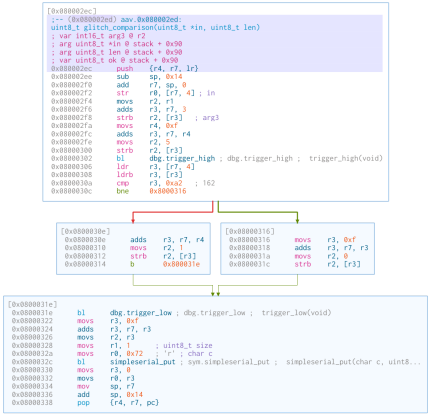


```
char passwd[] = "fault";
char passok = 1;
int cnt;

trigger_high();
for(cnt = 0; cnt < 5; cnt++){
    if (pw[cnt] != passwd[cnt]){
        passok = 0;
    }
}
trigger_low();
simpleserial_put("r", 1, (uint8_t*)&passok);
return passok;
```

Targets

Comparison

```
uint8_t ok = 5;
trigger_high();
if (*in == 0xA2){
    ok = 1;
} else {
    ok = 0;
}
trigger_low();
simpleserial_put("r", 1, (uint8_t*)&ok);
return ok;
```

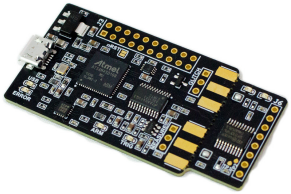


Targets

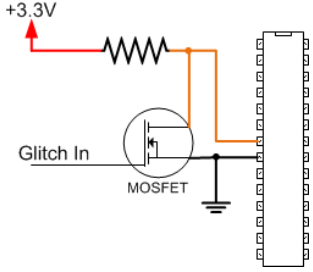
Infinite loop

```
volatile uint8_t a = 0;

trigger_high();
trigger_low();

while(a != 2){
    ;
}

// Blink
```

ChipWhisperer Nano



- Can only do voltage glitching
- Temporarily connects target VCC to ground



ChipWhisperer Nano

Glitch parameters

- `scope.glitch.ext_offset`: The offset of the glitch (from trigger)
- `scope.glitch.repeat`: The length of the glitch

ChipWhisperer Nano

SimpleSerial

- Serial communication interface
- **Command**
  - Single character command id
  - Payload
  - CRC
- **Response**
  - Single character response id (`"r"`)
  - Payload
  - CRC

Faulting

- Outcomes?
  - Normal behavior (fault did not manifest)
  - Faulty behavior (response is incorrect)
  - Reset (target did not respond at all)
- Too "*little*" fault → normal behavior
- Too "*much*" fault → reset
- **Lots** of trial and error, unfortunately

Tools

- **arm-none-eabi-gcc**: Building and binary examination
- **Rizin** + **Cutter**: Binary analysis

Tasks

Explore

- **1.1**: Explore fault parameter space for **Loop** target
- **1.2**: Determine relationship between glitch offset and sample index
- **1.3**: Explore glitches in different places in the computation

Mitigate

- **2.1**: Implement and test a countermeasure for the **Loop** target

Make notes so we can discuss your solutions together.

Tips and tricks

- Lecture
- Riscure guidance on SCA/FI
  - `FAULT.CONSTANT.CODING`
  - `FAULT.DETECT`
  - `FAULT.FLOW`
  - `FAULT.DOUBLECHECK`
  - `FAULT.LOOPCHECK`
- Hardware Hacking Handbook – Chapter 14
- Check that the compiled code looks right
  - Uses `volatile` variables to avoid optimization

Countermeasures

Example loop

```
cnt1 = 0;
cnt2 = 0
trigger_high();
for(i=0; i<10; i++){
    for(j=0; j<10; j++){
        cnt1++;
    }
}
for(i=0; i<10; i++){
    for(j=0; j<10; j++){
        cnt2++;
    }
}
trigger_low();
if (cnt1 != cnt2) {
    //Fault
}
return cnt1 == 100;
```

Takeaways

- What are **you** taking away from this tutorial?
- What surprised you?