Physical Attacks on Secure Systems (NWI-IMC068)

Tutorial 4: Fault Injection(FI)

Omid Bazangani (omid.bazangani@ru.nl)

Goals: After completing this tutorial successfully, you should understand what a hardware FI (Clock and Voltage glitch) is, What requirements and parameters are needed to make this attack reproducible, and how these attacks can bypass security measures.

Before you start: To complete this tutorial, your teacher will provide you with a package that contains the Chipwhisper-lite board. You will collect the traces with Chipwhisperer as you did in the previous tutorials.

1 Introduction to FI

Hardware fault injection is a technique used to intentionally cause faults in a system's hardware components in order to compromise its security. This can be achieved through a variety of methods, including voltage or clock glitches, electromagnetic interference, or laser-induced faults.
One common type of hardware fault injection is called glitching, which involves introducing a momentary fault by manipulating the system's power supply or clock signal. By carefully timing these glitches, an attacker can cause the system to behave unpredictably or enter into an insecure state. For example, an attacker could introduce a glitch in the power supply of a microcontroller to bypass its security measures and gain access to sensitive data.
This tutorial implements clock glitches on the ChipWhisperer platform to bypass password checking. Clock fault injection is a technique used to induce faults in a digital circuit by manipulating its clock signal [1]. In digital circuits, the clock signal is used to synchronize the various components of the circuit, and any manipulation of the clock signal can result in unintended behavior.
Typically, a quartz crystal oscillator generates a clock signal that produces a precise frequency. This frequency is used as a reference for all the other components of the circuit. By manipulating the clock signal, an attacker can introduce timing errors that can cause the circuit to behave in unexpected ways.
There are several ways to perform clock fault injection, including:

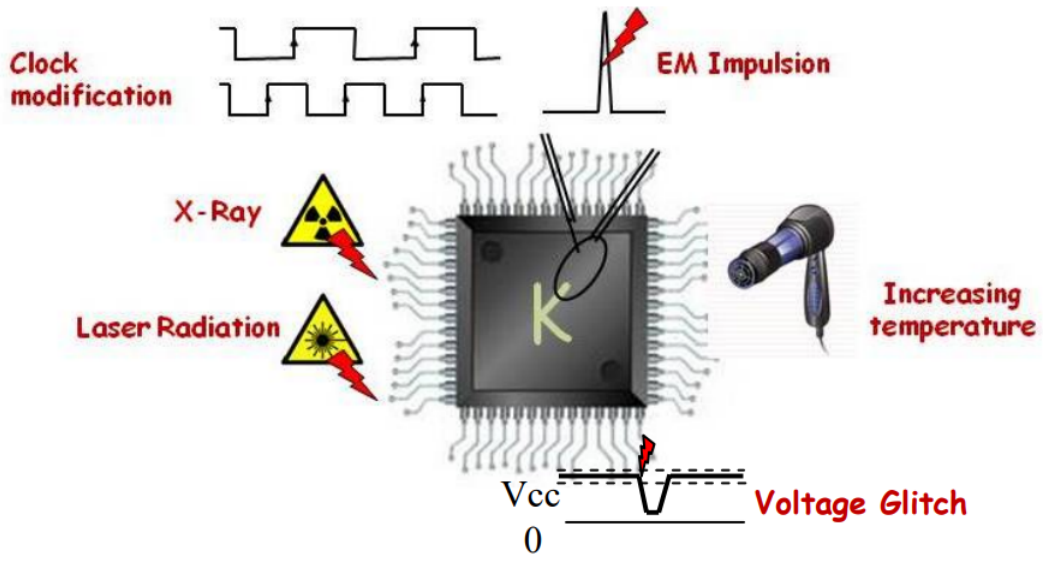

Figure 1: Various hardware FI attacks
pictureLink.

1. **Clock signal glitching:** This involves briefly interrupting the clock signal by introducing a glitch. This can cause the circuit to execute a different set of instructions than intended.

2. **Clock signal delay:** This involves introducing a delay in the clock signal. This can cause the circuit to miss a critical timing window, leading to incorrect results.

3. **Clock signal frequency change:** This involves changing the frequency of the clock signal. This can cause the circuit to execute at a different speed than intended, leading to incorrect results.

4. **Clock signal phase shift:** This involves shifting the phase of the clock signal. This can cause the circuit to execute at a different time than intended, leading to incorrect results [2].

Clock fault injection can exploit system vulnerabilities, such as buffer overflows, stack overflows, and race conditions. It can also bypass security mechanisms, such as code signing and authentication. As such, it is a severe threat to the security and reliability of digital systems.

1.1 What's a glitch, and what's the glitch effect?

All CPUs and MPUs need a reliable clock to guarantee their performance. Usually, A reliable or normal clock is a square wave with a 50% duty cycle and a specific duty cycle. A glitch is a clock disruption added to the normal clock (Fig 2).

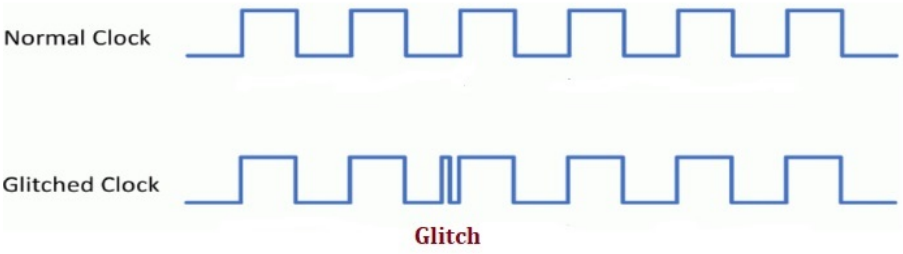

Figure 2: Normal clock vs. Glitched clock
Figure from: Link

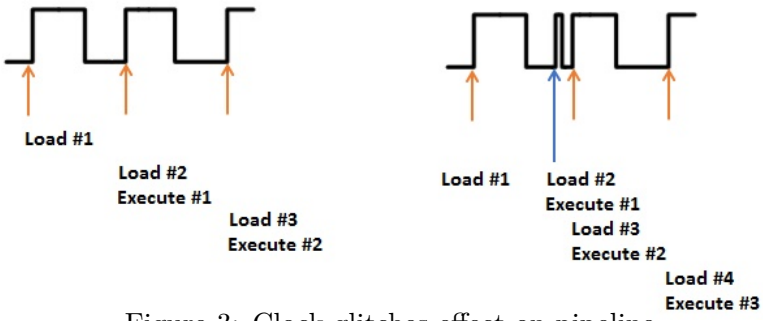

Figure 3: Clock glitches effect on pipeline
Figure from: Link

Clock glitches can affect the standard procedure of the pipeline to skip instructions. For instance, Fig 3 shows how instruction `#1` didn't get enough time to be executed due to the clock glitching.
This can affect the software to skip code lines. For example, in the code snippet shown in Fig 4, password checking is bypassed if a clock glitch skips the `passok = 0`.

1.2 Clock glitch attack implementation

Chipwhisperer provides a password-checking firmware called `simpleserial-glitch`. We apply a clock glitch attack on this firmware to bypass password checking. A Jupyter notebook file (`.ipynb`) is provided to guide you to conduct the test step by step. You can download the file from the link. After downloading the file, you need to place it in the ChipWhisperer directory. For example, in Windows, it can be like: "`C:\Users\USER\ChipWhisperer_xx\cw\home\portable\chipwhisperer\jupyter\experiments`." [1]
This experiment consists of a few Steps to follow while the goal is to attack the password-checking code introduced in Fig 4. After the attack, you will see the correct configuration on the clock glitch unit can bypass the password checking, and the password function would return the value "1" for the `passok` variable. The correct configuration for the

[1] In Linux or MAC, the path can be different. You must put the notebook file in the correct directory based on the operating system.

```
uint8_t password(uint8_t* pw)
{
    char passwd[] = "touch";
    char passok = 1;
    int cnt;

    trigger_high();

    //Simple test - doesn't check for too-long password!
    for(cnt = 0; cnt < 5; cnt++){
        if (pw[cnt] != passwd[cnt]){
            passok = 0;
        }
    }

    trigger_low();

    simpleserial_put('r', 1, (uint8_t*)&passok);
    return passok;
}
```

Figure 4: password checking code snippet
Figure from: Link

clock glitch unit is related to finding the exact timing that the clock glitch needs to be applied and how long it needs to stay to have a successful instruction bypassing. These parameters must be configured in the glitch controller using the following APIs[2]

1. `gc.set_range("width", ?, ?)`
2. `gc.set_range("offset", ?, ?)`
3. `gc.set_range("ext_offset", ?, ?)`

In the Jupyter Notebook file, we preset the range for the glitch controller parameter. You need to narrow these parameters based on the pre-settings to find more success in the attack.
Note: If you find this tutorial interesting, You can try it yourself to apply a voltage glitch attack on the target. ChipWhisperer is equipped with an advanced glitch controller that provides all you need to apply a voltage glitch attack. You can find more information on the examples page of the website here.

[2]You can learn more about these APIs by taking a look at ChipWhisperer website here or using `help(scope.glitch)` command in the Jupyter Notebook file.

2 Questions

After successfully conducting the test, you can challenge yourself by answering the following questions.

1. Is the clock glitch attack repeatable?
2. Is the glitch configuration related to the chip frequency?
3. Can you explain how you can find the glitch configuration if you didn't have access to presets?
4. Explain the effect of clock glitch on the targeted C code.
5. Can you think of other FI attacks on the targeted C code?

References

[1] Dusart, J. and Danger, J.L. (2013). Fault injection techniques and tools for embedded systems reliability evaluation. *IEEE Transactions on Instrumentation and Measurement*, 62(4), 707-716.

[2] Tsai, T.Y., Zhang, L., and Huang, W. (2014). Differential clock fault attacks on implementations of cryptographic algorithms. In *Proceedings of the 2014 International Symposium on Hardware Oriented Security and Trust* (pp. 87-92). IEEE.

[3] Blomer, J., Grosse, K., and Paar, C. (2018). Voltage fault attacks on RSA-CRT signatures: Concrete results and practical countermeasures. *IEEE Transactions on Dependable and Secure Computing*, 15(5), 834-847.

[4] Karroumi, M., Ferradi, H., El Hajji, S., and Machhout, M. (2019). Fault injection attacks against RSA public key cryptosystem based on voltage glitches. In *Proceedings of the 2019 International Conference on Advanced Technologies for Signal and Image Processing* (pp. 1-6). IEEE.