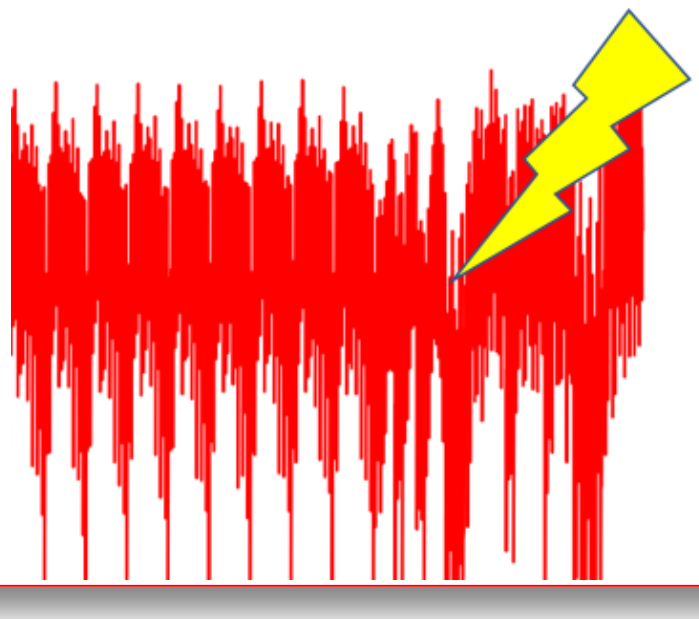
IoT Security - Part 20 (101 - Introduction to Fault Injection Attack (FI))

asmita-jha

12-December-2020



This blog is part of the IoT Security series, where we discuss the basic concepts about the IoT/IIoT ecosystem and its security. If you have not gone through the previous blogs in the series, I will urge you to go through those first. In case you are only interested in the fault injection (FI) introduction, feel free to continue.

IoT Security - Part 1 (101 - IoT Introduction And Architecture)

IoT Security - Part 19 (101 - Introduction to Side Channel Attack (SCA))

In this blog, we will give a basic overview of Fault Injection (FI) attacks. This blog is an introductory, conceptual overview of FI. In future blogs we will discuss details on attack implementation.

Introduction

Fault injection (FI) attack is a physical attack on the device to inject the fault in the system deliberately to change its intended behaviour. It can bypass system security features, change system behavior to accomplish malicious intents, or extract the secret information, key, or even firmware by analyzing the erroneous outputs. There are various ways to inject fault including, voltage glitching, clock glitching, laser injection, electromagnetic (EM) injection, etc. When we say a glitch, it means injecting a short-lived fault in the system, i.e., creating a sudden change in the system that’s not part of its normal operation. This may lead to brick the device, but an analyzed and target glitch with precision can create a potential security threat. It is an active attack. It is performed during the system’s ongoing operations and sometimes invasive when the chip needs to be decapsulated, and electrical contact with the chip is required. Many research has been done using this technique to bypass secure boot, bypass the security of crypto wallet, hack casino slot machine, find various vulnerabilities in ESP32, cryptanalysis to extract the

secret key of the implemented cryptographic algorithm. More related research is mentioned in the reference section of this blog. These attacks can have severe impacts as they can altogether bypass the device's security mechanism if successful.

Fault injection techniques

There are various ways in which faults could be injected into the system. It could be either via the hardware or the software targeting the logic faults. As embedded devices have a relatively small codebase, software fault injection might not provide a broader attack surface. Hardware fault injections have more severe impacts as they can completely break the security features and most devices are still not resistant to such attacks. These techniques are used to perform attacks like differential fault attack (DFA) that we had discussed in the previous SCA blog. Here, we will discuss a few techniques used for fault injection via the hardware.

Clock glitch

This technique involves the tampering of the system clock to change its intended behavior. The integrated circuits (ICs) internally have combinatorial logic blocks and flip-flops registers that share the same clock. The data is latched at either the rising or at the falling edge of the clock. It gets transferred between the two edges, and there is a propagation delay involved to transfer the data through logic blocks. For the circuit to operate correctly, the clock's time period should be greater than the max propagation delay plus some offset timings. Tampering with the clock signal affects the clock period that can potentially affect the device's ongoing operation. The attacker should have direct control over the clock line to change the clock length, i.e., the clock glitching attack cannot work for the devices with internal oscillators. By introducing short pulses in the circuit, the attacker can inject the fault when any specific instruction is being executed to bypass it. If this fault is injected precisely at the right moment with accuracy, it can even attack the cryptographic implementations, skip the instructions to bypass the security checks, etc.

Voltage glitch

This technique changes the system's intended behavior by manipulating the voltage at a specific instance to inject the fault. Though the voltage glitching technique is low-cost, the arbitrary change in the voltage may result in unpredictable fault states but might not lead to an advanced exploitable attack.
With proper analysis and a well-timed glitch or device, underpowering can even attack the cryptographic implementation. The properly-timed glitch on the power line can make the device skip specific instructions that can be exploited to bypass security features. Since the Vcc pins of the chip are mostly accessible, it becomes relatively easier to identify the attack point. However, to perform the exploitable attack, the correct settings and the proper external glitch circuit gets tricky to achieve sometimes. It is because when we try to change the voltage level at the target power line, there are other components like decoupling capacitors and voltage regulators that do not allow us to keep the setting accurate and constant. A paper titled, "Fault Injection using Crowbars on Embedded Systems" describes a novel way to use this technique. Another, a very high precision attack using voltage and temperature manipulation has been proposed in one of the research papers titled, "Precise Fault-Injections using Voltage and TemperatureManipulation for Differential Cryptanalysis"

EM glitch

This technique is used to inject faults in the device via an electromagnetic field. This is one of the most used techniques for fault injection as it is mostly non-invasive. For more efficiency, it is preferred to depackage the chip by removing its plastic package. In this case, a high or medium voltage pulse

generator is used to produce an EM pulse to inject the transient fault in the system. The proper positioning of the EM probe is required for this technique. For this purpose, the X-Y table is used to precisely position the EM probe above the target chip's surface. Researchers at Raelize discovered various flaws in ESP32 using EM fault injection. More detailed information can be found on Raelize's website. The attacks may result in secure boot bypass, bypass flash encryption, execution on arbitrary code, access to flash memory contents, etc.

Optical injection

This technique involves the illumination of transistors in the target device that makes it conduct and inject faults in the system. A research paper titled, "Optical Fault Induction Attacks" by Sergei P. Skorobogatov and Ross J. Anderson discusses performing optical attacks on secure microcontrollers and smart cards without using expensive laser equipment via a semi-invasive approach.Compared to the non-invasive attacks, invasive attacks, though expensive but very powerful to resist. The optical injection using a laser or high energy light source like a UV lamp is done by decapsulating the chip using acid for accuracy and precision. These attacks can even reset the microcontroller's internal protection fuses, wipe the particular part of the data in the memory, attack EEPROM and Flash memory, break cryptographic implementation, etc.

Examples of fault injection attacks

Differential fault attacks use any of the fault injection techniques, as mentioned above, to break the device security feature. When combined with power analysis, machine learning, etc., these techniques result in more powerful attacks. Many fault injection attacks have been done by researchers on many cryptographic devices like smart cards. Glitching attacks can break non-crypto devices' security, including the execution of arbitrary code on the device, attack on xbox 360, etc. Apart from general tools like probes, oscilloscope, there are specific fault injection tools and techniques available to perform these attacks including chipwhisperer, chipshouter, Riscure FI tools, Raiden, glitchsink, etc. A few examples of fault injection attacks are discussed below.

CVE-2019-15894

- Espressif ESP32: Bypassing Secure Boot
- An attacker can bypass the Secure Boot verification at the startup of the ESP32 CPU via fault injection.
- Leads to the execution of unverified code from flash.
- It was identified that if the flash encryption is enabled, this attack could be mitigated. Hence the flash encryption was then later implemented.
- Reference: Espressif Advisory, CVE Mitre

CVE-2020-15048

- Espressif ESP32: Bypassing Flash Encryption
- An attacker can bypass the Secure Boot and Flash Encryption physical security features of the ESP32 CPU via fault injection.
- Leads to the execution of unverified code despite both the secure boot and flash encryption implementation.
- Reference : Espressif Advisory, Raelize

CVE-2020-13468

- Gigadevice GD32F130 devices: Debug interface permissions escalation
- An attacker can exploit the insufficiently physically protected inter-IC bonding wires in Gigadevice GD32F130 devices via fault injection to escalate the debug interface permissions.
- Leads to the extraction of firmware from the devices despite debugging protection implementation, reads the protected flash memory, performs arbitrary modifications on the system, etc.
- Reference : Research paper, CVE Mitre

CVE-2020-15808

- STM32 USB Device Library : Buffer overflow vulnerability exploit
- The buffer overflow vulnerability in the CDC communication interface code exploited using fault injection.
- Leading to the access of sensitive information, keys, obtaining firmware, etc. with the properly crafted USB packets.
- References : Black Hat Asia 2020, Grzegorz Wypych, Raiden

Countermeasures

There is much ongoing research work to find the countermeasures to mitigate fault injection attacks. Resistance to invasive fault attacks, where the chip is decapped and the passivation layer is removed to bypass the shielding, gets comparatively more difficult. A few of them includes:

- Proper shielding to mitigate EM-based attacks, making chip die) resistant to UV light-based attacks, and proper physical hardening.
- Implementing proper fault detection mechanisms like error detection codes, external error checking circuits, etc., to detect the fault immediately and abort the further device operation.
- Implementing the resistance against the side channel attacks as discussed in the previous blog
- Implementing the redundancy for the critical control signals, i.e., implementing the control signals more than once that behave differently to mitigate glitching effects.

Since these are physical attacks at the CPU level, they have high potential risks on the devices. However, these countermeasures cannot completely stop the attacker from performing these attacks but can make it more challenging for them to perform these attacks.

Conclusion

In this blog, we discussed the basic conceptual overview of fault injection and its techniques. We looked at a few attack cases and countermeasures. Hope, you get some idea about the different impact these attacks can have and how the increasing possibilities of these attacks that are comparatively difficult to resist is a threat to the system. Stay tuned for our upcoming blogs, where we will show you how to perform these attacks practically.

References

- https://www.riscure.com/fi/
- https://arxiv.org/pdf/1903.08102.pdf
- http://euler.ecs.umass.edu/research/bbkn-IEEEP-2012.pdf
- https://www.riscure.com/uploads/2017/10/Riscure_Whitepaper_Escalating_Privileges_in_Linux_using_Fault_Injection.pdf
- https://pdfs.semanticscholar.org/bd8b/724cf102a9e9f6347dfe67ff519220b2fbbd.pdf
- https://www.researchgate.net/publication/269270120_Precise_Fault-Injections_using_Voltage_and_Temperature_Manipulation_for_Differential_Cryptanalysis
- http://dl.ifip.org/db/conf/cardis/cardis2010/AgoyanDNRT10.pdf
- https://www.researchgate.net/publication/286813291_On_the_Effects_of_Clock_and_Power_Supply_Tampering_on_Two_Microcontroller_Platforms
- https://www.cl.cam.ac.uk/~sps32/ches02-optofault.pdf
- https://www.eeweb.com/fault-injection-attacks-a-growing-plague/
- https://pure.tugraz.at/ws/portalfiles/portal/23511745/Attacking_AUTOSAR_using_Software_and_Hardware_Attacks.pdf
- https://pure.tugraz.at/ws/portalfiles/portal/3103517/_CARDIS_Protecting_the_Control_Flow_of_Embedded_Processors_against_Fault_Attacks.pdf
- https://raelize.com/posts/raelize-fi-reference-model/

About Payatu

Payatu is a research-powered cybersecurity service and training organization specialized in IoT, embedded, mobile, cloud, infrastructure security and advanced security training. We offer a full IoT/IIoT ecosystem security assessment, including hardware, firmware, middleware and application interfaces. If you are looking for security testing services then let’s talk, share your requirements: https://payatu.com/#getstarted. Payatu is at the front line of IoT security research, with a great team, and in house tools like expliot.io. In the last 8+ years, Payatu has performed, security assessment of 100+ IoT/IIoT product ecosystems and we understand the IoT ecosystem inside out. Get in touch with us. Click on the get started button below.