Towards a Black-Box Security Evaluation Framework

Mosabbah Mushir Ahmed, Youssef Souissi, Oualid Trabelsi, Sylvain Guilley, Antoine Bouvet, Sofiane Takarabt

Towards a Black-Box Security Evaluation Framework

Mosabbah Mushir Ahmed[2], Youssef Souissi[2], Oualid Trabelsi[2], Sylvain Guilley[2,3], Antoine Bouvet[1⋆] and Sofiane Takarabt[2,3]

[1]Secure-IC S.A.S., Z.A.C. des Champs Blancs, 35510 Cesson-Sévigné, France
firstname.lastname@secure-ic.com
[2]Secure-IC S.A.S., Tour Montparnasse, 75015 Paris, France
firstname.lastname@secure-ic.com
[3]Télécom Paris, 91120 Palaiseau, France
firstname.lastname@telecom-paris.fr

Abstract. Injection of faults has been studied in various research works since last decades. Several hardware targets have been studied with respect to the efficiency of fault injections. In this paper we address the security evaluation of embedded systems in constrained environments called black-box analyses. This is not considered by standards of evaluation as they require conducting the analysis in the most relaxed conditions, often called white-box analysis which focuses on specific security modules provided that the finer details are available. However, black-box analysis has a much larger view by focusing on all the system as potential target. It is closer to a real world attacker. This allows measuring the impact of real attack scenarios, and therefore thinking and building the most adequate protections. We put forward a six steps evaluation methodology along with a practical use-case on a real end-user device. This shall give a better understanding and also an evaluation framework of black-box analysis.

Keywords: Security evaluation · Black-box analysis · Embedded systems · Physical attacks · Methodology · Laboryzr^{TM} tool.

1 Introduction

Today, security of user data is becoming worth more than devices. Therefore, the evaluation of any device holding and manipulating sensitive data is mandatory before being used. Fortunately, international standards exist as testing frameworks to assess the robustness of security modules. We cite the Common Criteria [4], FIPS 140 [12] and ISO 17825 [19]. Even better, more strict regulation has been recently put in place thanks to GDPR [3] that is a regulation in European Union law on data protection and privacy. The methodologies proposed for the security evaluation of the embedded systems are usually conducted in a

⋆ Security Science Factory (SSF) Sponsor at Secure-IC Rennes, and corresponding author of this cybersecurity research paper.

white-box context through comfortable analysis conditions. This often requires the maximum of details about the target of evaluation. Those details range from technical specifications, architecture implementations or source codes, to conception schemes and more [8]. Moreover, the scope of the analysis view is limited to a target module that can be for instance a cryptographic library run by some CPU or a secure element run by hardware. In addition to this, a white-box analysis is usually run on testing boards that are well prepared and conditioned to get a direct access to the target module. This approach has limitations: indeed, disclosing all the details might raise a problem of trust between the provider and certifications laboratories (because the former should describe all the security features in a document called a Protection Profile). Moreover, a white-box analysis focuses on a small piece of the puzzle that is the whole system, called the scope of analysis. All the inputs and outputs of that scope are provided for evaluation independently from the rest of pieces. In fact, security might be broken at the frontier of the analysis scope. The perimeter of that scope is sometimes confusing and complex specifically when it comes to a combination of software, hardware with storage requirements. Furthermore, a white-box analysis does not aim at assessing the robustness of devices against reverse-engineering as the access to the target is by default not considered or it should be opened and well documented anyways.

Contributions. In this context, we propose to push upward a testing framework to assess the security of a system in its integrity. We expose the so-called black-box methodology that is followed by advanced attackers to harm end-user devices:

- **Testing framework.** In the first phase of this study we have analyzed a high-level description about various modes of targets in terms of their accessibility by an attacker.
- **Practical use-case.** The second phase of this study elaborates a practical use-case scenario that shows the experimentation setup, results and analysis from the experiments. These experiments are carried out to assess the security of a black-box target against physical attacks like fault injection.

Our goal is to shed light on the danger around this practical and powerful analysis mode, and to put forward a black-box testing framework for standardization purposes. Explaining the ways an experienced attacker behaves should help to design better protections. Besides, it should guide manufacturers in testing their final products.

Outline. The rest of the paper is organized as follows. In Sec. 2, we present the black-box approach along with the other analyses modes as white and grey-box respectively, as well as the electro-magnetic (EM) fault injection (EMFI) method. In Sec. 3, we go through a detailed description of our proposed testing framework. One related use-case on a real end-user device is presented and discussed in Sec. 4. Section 5 discusses advantages and limitations of the Black-box (Bbox) evaluation methodology. Eventually, general conclusions are derived in Sec. 6.

2 Background

2.1 Security evaluation modes

A security testing allows to avoid cybersecurity-related incidents that cause a loss to an organization or individual. Its aim is to demonstrate the possible ways of penetrating into a system. We generally distinguish three main security testing modes, depending on the level of knowledge and access that is granted to the security tester:

- **White-box mode.** The analysis is conducted in the best conditions, with a full knowledge about the target. Every detail should be accessible and documented, including the specification, architecture and implementation of the target module along with the ways to communicate and query it. The inputs and outputs of the target are controlled, which allows to perform all the possible assessment tests [17]. This mode of analysis is often required by certification laboratories, when seeking a certification label according to one standard as Common Criteria or FIPS 140.
- **Grey-box mode.** The analysis is performed with some knowledge about the target. This situation happens for instance when a circuit provider wants to make his product evaluated by some labs without necessarily disclosing all the details [17] (*e.g.* implementation and countermeasures are kept secret).
- **Bbox mode.** The analysis is made with the minimum knowledge about the target. The underlying systems are usually hidden from the end-user or attacker. Generally, the black-box security mode is chosen for reverse-engineering purposes, or when the device manufacturer wants to measure the extent of a real attacker. Another situation is purely defensive, by exploring the device for any hidden backdoor, or robustness testing before deployment in a company with sensitive activity and high security conditions.
  Obviously, there is no standard explaining how to assess and conduct such analysis. Therefore, the black-box mode puts the evaluator in the worst and toughest analysis conditions, even if there is dedicated team, with complementary skills, involved in the analysis.

2.2 Electro-magnetic Fault Injection Attacks

We choose for our experiments the EM interference fault injection [7]. Considered for Radio-Frequency (RF) applications, the design of an EM probe is highly related to the generated EM interference. Therefore, the EMFI setup is characterized by several properties: pulse intensity, spatial distribution, injection time, pulse width, polarity and amplitude, number of pulses, etc. All these parameters need to be considered for a successful fault injection.

Many studies have been conducted to characterize embedded systems faulty behavior under EMFI [10, 15, 18]. Related results provide a clear evidence of the possibility to fault a system at the hardware or software level [6]. By the analysis of the fault model derived from the faulty behavior [15, 18], recent works point

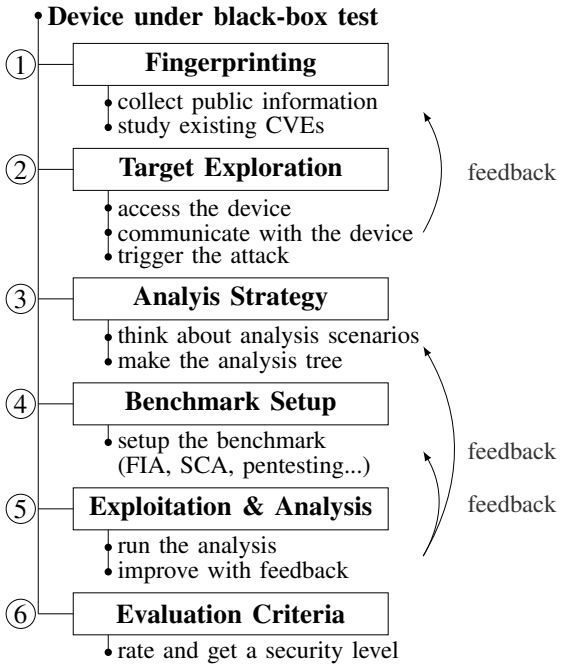

Fig. 1. Illustration of the proposed Bbox testing framework.

out how the program flow of a running device can be disturbed (*i.e.* bypassing authentication process), and where the efficiency of countermeasures is to be reconsidered [9, 21].

3 Proposed testing framework

It is clear that there is a need to elaborate the mechanisms with which a black-box mode of testing can be done. To accomplish so, we propose a black-box testing framework which consists of six main and generic steps, summarized in Fig. 1.

3.1 Fingerprinting

As the target is a Commercial Off-The-Shelf (COTS) device, the only input an attacker might have is the user product sheet. It is obvious that an attacker needs to find and explore more inputs. Publicly available information – documentation, datasheets, references and discussions available in the common and accessible web – are first explored. Advanced attackers might look at the deepest and darkest side of the web to extract non-public resources such as private datasheets or firmware specifications. Moreover, during this theoretical exploration phase,

the attacker might check the existing Common Vulnerabilities and Exposures (CVEs), publicly available databases that disclose devices' vulnerabilities. This is a very important point as the goal of the attacker is to shorten the time-to-attack, involve affordable resources and perform the exploit in the more efficient way.

At this point, the attacker makes a synthesis of his knowledge about the target by trying to find answers. From the evaluator viewpoint, he should have the maximum possible knowledge that a best attacker can acquire during this phase:

- **Components enumeration.** Those are soldered on the target as CPUs, memories or System-on-Chip (SoC). In addition to this, the attacker shall be interested not only in the electrical part, but mechanical and maybe the optical parts should be considered too.
- **Information about components manufacturers.** This allows knowing about CPU's type, namely its frequency and architecture along with the internals of available memories, by having a precise exploration of their datasheets. A very basic example is to check the pads of a flash memory. This allows knowing the ground (GND) pad which itself is needed to identify the GND pad of the UART debugging port. After identifying all the UART pads, the debugging port will be used of course to access and interact with the target device. Other information can be the model of available MIPS or fingerprint sensors.
- **Availability of hardware or software sources.** Having the source code or even a binary form of it is always useful. Lets take for instance an Internet of Things (IoT) manufacturer that leaves the firmware of the main chip available on the Internet for upgrading purposes. This can be used by an attacker to reverse the code and explore sensitive information like user or even root passwords.
- **Existing vulnerabilities (CVEs).** The attacker often studies and explores the existing vulnerabilities looking for potential leakages. It is the shortest way to conduct an exploit. The CVEs are generally held in the public databases. That is different from zero-day vulnerabilities that exist but never made public. They can be intentionally created as backdoors by the manufacturer or its providers; or just a security flaw exploited by advanced attackers.
- **Possible security mechanisms.** The manufacturer might show some claims of security as certification labels which help the attacker to make a first view about implemented security. For instance, a FIPS 140 level-3 certification requires tamper resistant mechanisms as putting epoxy on the top of the circuit.

3.2 Target Exploration

The fingerprinting phase is complemented by a practical approach that is an active exploration. In other words, the attacker starts looking at how to access the device. For this purpose, several points can be addressed:

- **How to open the device?** Some devices do not present any difficulties, as for smart-cards of which the main circuit is totally exposed. In some other cases, only what an attacker has to do is to use a screwdriver to have a full access to the PCB board and all electronic components of the target like for hard drivers. In other situations, the target circuit is often hidden by a cooling fun system like for personal computers.
- **How to bypass anti-tamper protections?** More and more devices have anti-tamper mechanisms. It can be for instance a light sensor that detects that the device is opened; a coating (*e.g.* epoxy, acrylic or silicone) or encapsulation layer at the top of the SoC; a radiated sensor that detects an X-ray scanning or Focused Ions Beam (FIB) for probing, or also a fuse bit system for on-chip memories protection [1]. Another aspect can be a suicide kill activation process when the tamper action is detected.
- **How to communicate with the target?** Communication with the target can be time consuming as it might differ from one target to another. This depends on many factors as the availability of left debugging ports, the control of inputs and outputs (*e.g.* plaintexts and ciphertexts), the possibility to upload and download data from the device, or the number of possible iterations with the target before locking it. More precisely, the debugging ports such as JTAG and UART are considered as first option. Those ports are initially used for testing and verification purposes, but when left accessible after device production, then they would be exploited. With a UART port for instance, we can follow the boot process along with the interaction between the main process, the Flash and dynamic memories. Moreover, if not protected, then a root access might be obtained which allows a deeper penetration of the installed software. The JTAG could be more harmful as it allows a dynamic interaction with the processor and memory. For instance, with the JTAG, one may spy the internal registers, halt the processor and then dump a copy of the dynamic memory.
- **How to trigger the attack?** Irrespective of the targeted layer (hardware, software or network), triggering the attack consists in finding a short time window of opportunity to target some running sensitive process.

3.3 Analysis Strategy

After having properly gathered all the possible knowledge about the target, the evaluator starts thinking about an analysis strategy. The strategy is generally motivated by the attackers goals as finding a precise secret or getting an access with full privileges. It allows having a general picture with a top level view of the possible analysis paths and the time needed for each of them.

Basically, a strategy can be illustrated by a tree or a diagram. Whatever the illustration is, it should be complete by showing attack surfaces, each possible path and the analysis conditions. Obviously, there is no generic strategy. Actually, it depends on the device itself and the knowledge about it. Moreover, a full analysis strategy should address all the possible attacks. But this also depends

on the experience and skills of the evaluator, that is why a dedicated team of evaluators with complementary skills is required.

3.4 Benchmark Setup

Performing a Bbox analysis is not an easy task which could be time consuming and expensive. Generally, the benchmark used for a Bbox analysis can mix between several platforms and tools as follows:

- **A Side-Channel Attack (SCA) platform.** SCA [14] allows the exploitation, during some sensitive running process like encryption or authentication, of a physical property such as the EM emanations, the power or current flows, or the acoustic waves. In fact, side-channel waves are able to disclose sensitive patterns directly related to the intermediate computations. More importantly, such attacks do not make brute force attacks on the entire secret, but split it and make assumptions on smaller chunks which is tricky and much more faster than a full brute force attack.
- **A Fault Injection Attack (FIA) platform.** Fault injections are powerful attacks that can be mounted easily. Their impact can be bypassing an authentication process, dumping a memory, secret key recovery, denial of service and more. A FIA is generally based on two phases: a fault injection phase and then an exploitation phase. The injection phase could be performed by several modes of injection as clock [13] or power [20] glitching, EM or Laser [16] injections. The exploitation phase often comes with statistical techniques to analyze the temporal and spatial location of the fault. Some techniques aim at recovering the secret key as Differential Fault Analysis (DFA) [5].
- **A scanning platform.** This platform aims at inspecting the PCB circuit looking for an abnormal manufacturing or behavior. Actually, a physical backdoor can be for example a tiny mounted surface component that is on the PCB to infiltrate connected devices [2]. For this purpose, techniques as thermal imaging, infra-red or X-ray can be used. Another aspect consists in scanning and analyzing the communication protocols with the target. Tools as spectrum analyzer or RF signals scanners might be used.
- **A pentesting platform.** This is useful especially to improve the target exploration phase as opening the device and accessing its components. It can be basically composed of a screwdriver kit, a heat gun, an optical microscope and a soldering material support. Pentesting is not a new topic. In fact, several tools exist in the literature as network sniffing, fuzzing, and databases testing. Moreover, we can find dedicated OS distributions to deal with security testing as Kali Linux or formerly Linux Backtrack. In addition to this, interfacing tools and boards might be needed to communicate with debugging ports as UART and JTAG.

3.5 Exploitation & Analysis

A full analysis strategy tree will allow the evaluator to gather all analysis paths including all possible attacks. Now the evaluator could run the analysis and improve it with a continuous feedback to the analysis strategy to precisely tune the exploitation benchmark. Analyzing a real device is not an easy task as the evaluator might play with several benchmarks and attacks in the same time. As a matter of fact, Fault Injection and Side-Channel Attacks can run hand in hand. Actually, a SCA can serve as signal trigger to activate the fault injection. The most commonly classes of analysis can be: scanning, probing, sniffing, fuzzing, web attacks, binary static and dynamic analyses, cache memory attacks, passwords recovery, malware analysis, SCA, FIA, unauthorized debugging and more.

3.6 Evaluation Criteria

After performing the exploitation and analysis phase, it is time to make a synthesis about the assessment results. As standards of evaluation, Common Criteria and FIPS 140, we propose to address a table to rate the black-box security level. As mentioned, we are interested in the whole system with all layers and not focusing on some parts or specific security modules. For this purpose, our framework comes with a rating table with a minimal list of evaluation actions. The more the number of successful actions is, the less the device is secure against black-box analysis. In fact, those phases allow gathering all the possible theoretical and practical actions to tamper with the device in a black-box context. The criteria table that we propose consists of four columns as follows:

1. **Bbox actions.** Defines the set of all possible actions and events in a Bbox analysis context. A minimal list can be addressed and that is common to most of embedded devices. The list can be extended depending on the fingerprinting, exploration and strategy phases.
2. **Tested (Te).** Is a rating column. Rate +1 if the test is performed successfully. It represents a high risk of exploitable leakage.
3. **Not tested (Nt).** Is a rating column. Rate +1 if the test is not performed but still theoretically and practically possible. It represents a potential exploitable leakage. It is likely to be successful in practice.
4. **Not possible (Np).** Is a rating column. Rate +1 if the test is not possible but still theoretically possible on similar devices. In other words a protection or countermeasure exists. It represents a low risk of exploitable leakage.

The way we propose to evaluate a device against Bbox attacks can be described by a generic table that covers most of possible Bbox actions. It can serve as a basis for the evaluation. However, the evaluator might refine the table by editing more or less actions depending on the three first phases of the methodology. We note that only theoretically possible actions are considered. For instance, if our target is standalone and does not have any connectivity with the outside

world, then network attacks actions should not be considered by the evaluation table as they could not exist.

The Bbox testing framework criteria table is illustrated with real experiments in Sec. 4, table 2.

4 Experiments on a real device: Door-lock Unlock

We illustrate the discussion about the Bbox analysis with a practical use-case scenario. The proposed testing framework is used for assessing the security of a black-box target – namely, a smart Door-lock – against physical attacks like FIAs. Basically, the aim is to force the smart Door-lock to unlock without knowing the necessary Personal Identification Number (PIN) using EMFI. The obtained results highlight the possibility to break the security of a system without knowing the underlying features, structure or details of the system. Further, it validates the fact that EMFI is a potent technique against Bbox security also.

Fingerprinting. The knowledge of target's microcontroller has been accessed through the datasheet of the vendor. This assisted in estimating the type of software instructions that could be installed in the target. Establishing the knowledge of the existing setup, the further work has been made to determine if such targets are vulnerable to any type of implementation attacks such as fault injection.

Target Exploration. This phase is executed as part of the experimentation. Once the target details are determined, we physically access the target by removing the outer covering of the target without changing anything to the underlying hardware setup (or IC). More physical analysis details make it easier to inject faults at right spot over the microcontroller chip.

Analysis Strategy. This phase determines the strategy that is adopted in order to bypass the security mechanism of the Door-lock. For the implementation, we have regarded various types of attack scenarios which can be achieved with EMFI. Each scenario has been evaluated in terms of its ease of exploitation, repetition and effectiveness of the practical experimentation.

EMFI. The methodology of implementing the EMFI on the smart Door-lock depends upon the selection of the appropriate hardware setup and their parameters. The injection of faults through EM is based on coupling induction between the target and the injection source. The goal of the attack path is to bypass the checks so that any entered PIN is accepted as a valid one.

Fault Model. On microcontrollers primarily, fault injection is used to skip an assembly instruction set or subroutine calls. Moreover, in general a fault injection is useful in algorithm modification, by inducing safe errors, replacing or skipping

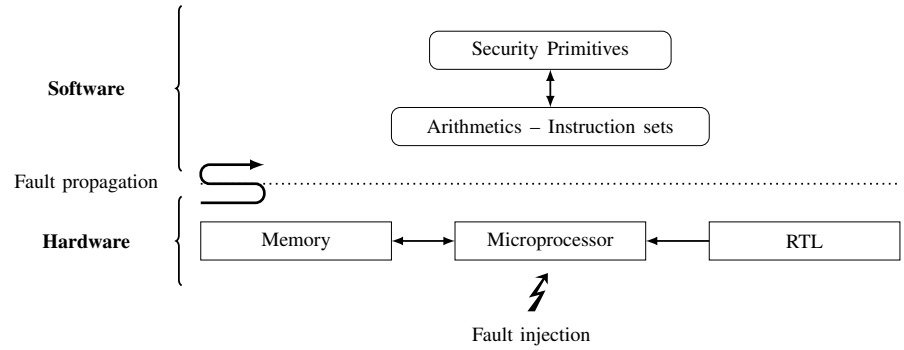

Fig. 2. Fault propagation between hardware and software layers.

instructions executed by the microcontroller [10,11]. As shown in Fig. 2, in order to induce a fault into a Bbox setting it is very important to fix the fault model. The choice depends upon the attacker; he can induce the fault in either hardware or software flow. However, faults like EMFIs are induced in the hardware (bus, registers...), thus the faulty output propagates from the hardware to the software layer, and alters the execution of the instructions.

In order to bypass the authentication phase of the Door-lock PIN, a fault injection is required just after the PIN entry. It can cause the following effects:

- **Skipping instructions**, when an instructions comparison occurs;
- **Bit-flip** (*i.e.* $0 \longrightarrow 1$, and vice-versa): if the entered value is wrong, it can still be coded as correct after the fault injection due to bits flipping.

Benchmark Setup. The setup for the attack has been developed with the EMFI setup (consisted of the equipments, tools, or softwares) effective in injecting the high powered EM signal that can disrupt the normal execution of the target. We have made some iterations in order to determine the best possible location of the probe over the target, or setting up of the target (undisturbed) by external disruptions.

To inject EM pulses, the basic hardware requirements are an EM pulse generator (including a RF amplifier and a delay generator), and probes. To perform the experimentation on the Door-lock provided as a Bbox, we have applied a gated pulse EM injection as a medium to disrupt the normal operation of the device (Fig. 3).

To operate an effective EM attack, we have removed the outer thick covering of the Door-lock such that we could gain good access to the microcontroller. No other modification has been made in the target.

Exploitation & Analysis. This phase determines how the faults injection strategy obtained can be exploited and improved upon. This part is performed

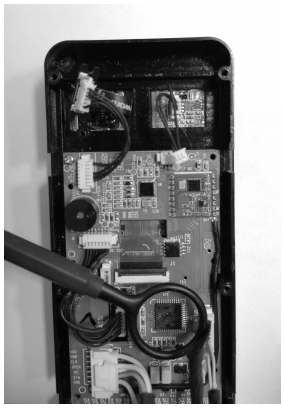

Fig. 3. Backside view of an opened doorlock (reference: `SKY-001-RFPDA-Z`) with an EM H-field probe.

experimentally by fine tuning the various parameters – such as the voltage level or the delay between the trigger and the injection – which are central to EMFI. For instance, the activation of the trigger is actually done by analyzing the chip activity through EM side-channel techniques.

During the preliminary experiments, some alterations have been observed in the functionality of the Door-lock. All of these may not be effective in bypassing its security feature but still can provide us with useful information about the potency of EMFI. Table 1 summarizes the results and thus highlights the robustness of the Bbox target against EMFI:

- **Crash.** The whole system crashes and the Door-lock beeps some alarms. A crash is not exploitable because it takes time to reset target.
- **Language setting access.** The attacker is able to change the language setting without accessing any vital functionality/software code.
- **Stop operation/Target freeze.** With the EM injection, the attacker can freeze the target that can result in causing the system to reset. Under this circumstance, the Door-lock can be left open until the system and access PIN is reset.
- **PIN bypass/Authentication failure.** Apart from these outcomes, on certain instances, we have been able to bypass the security completely, that is open the Door-lock with any PIN.

Evaluation Criteria. We have drawn the criteria table 2 corresponding to our Door-lock unlock experiments. Since the percentage of high risk is about 36%, one-third of the time an attacker would be able to circumvent the security if the proposed approach is applied in a Bbox target.

The results obtained here validates the fact that EMFI can be useful or effective in breaking the security of a Bbox setup. Further, the basic security

Table 1. Success rate and exploitability of fault injections on Door-lock.

Fault Types	**Exploitation**	**Frequency**
Crash	Not exploitable (4 surfaces)	63%
Language setting access	Exploitable fault	5%
Stop operation/Target freeze	Exploitable fault	11%
PIN bypass/Authentication failure	Imperative fault	21%

Table 2. Door-lock black-box testing framework criteria table.

Bbox actions	Te	Nt	Np
Have one attack surface	+3 (3 surfaces)	-	-
Open the device	+1	-	-
Scanning	-	+1	-
Visually identify components	+1	-	-
Have one debug access	-	+1	-
Communicate with the device	-	+1	-
Get shell access	-	-	+1
Reverse operation sequence (*e.g.* boot process)	-	+1	-
Sniff the communication	-	+1	-
Fuzzing	-	-	+1
Control cryptographic I/O	-	-	+1
Find not public variables	-	+1	-
Exploit one CVE	-	+1	-
Extract the firmware binary	-	+1	-
Analyse the firmware binary	-	+1	-
Fault Injection Attacks	+1	-	-
Side-Channel Attacks	-	+1	-
Compromise authorization (user permission)	+1	-	-
Compromise integrity	+1	-	-
Brute force passwords	-	-	+1 (limited trials)
Bypass authentication	+1 (only for PIN)	-	+1 (fingerprint module)
Attacks combination	+1 (FIA + SCA)	-	-
Secret keys recovery	-	+1	-
Dumping memories	-	+1	-
Protocol attacks	-	+1 (on NFC)	-
Total	10	13	5
Percentage	36%	46%	18%

mechanisms of implementing PIN code are vulnerable to non-invasive fault attacks.

5 Discussion

Bbox security evaluations are obviously not enough to guarantee the security of a device against physical attacks, that is why we do recommend to perform both the usual white-box security evaluation, and the Bbox one following the framework introduced in this paper. Indeed the former method aims at assessing some known vulnerabilities on specific parts of a design, whereas the latter aims at reproducing a more realistic attack on the end-user device. In both cases, there are multiple analysis paths and all of them can not be followed: therefore evaluators have to make choices which obviously have an impact on the final evaluation reports.

6 Conclusion

This study has come with the first initiative to push upward an evaluation framework for end-user devices security testing. By contrast to Common Criteria, FIPS-140 and ISO standards that focus only on analysis performed in the best conditions, known also as white-box analysis, our framework deals with the whole system by considering all its layers and complexity. It is a high level security view of the target which matches with a real impact caused by a real attacker. Our testing framework comes with a six steps-based methodology along with a criteria rating of the security level. Moreover, we have illustrated our study with a real use-case on a real end-user device. This study is a starting point to consider and democratize Bbox analysis in a more elaborated evaluation context as the case for current standards.

References

1. Anti-tamper techniques. elena dubrova royal institute of technology, stockholm, sweden, https://docplayer.net/79391807-Anti-tamper-techniques-elena-dubrova-royal-institute-of-technology-stockholm-sweden.html
2. The big hack, https://www.bloomberg.com/news/features/2018-10-04/the-big-hack-how-china-used-a-tiny-chip-to-infiltrate-america-s-top-companies
3. General Data Protection Regulation GDPR, https://gdpr-info.eu/
4. Common Criteria (2020), https://www.commoncriteriaportal.org/
5. Biham, E., Shamir, A.: Differential Fault Analysis of Secret Key Cryptosystems. In: Jr., B.S.K. (ed.) Advances in Cryptology - CRYPTO '97, 17th Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 1997, Proceedings. Lecture Notes in Computer Science, vol. 1294, pp. 513–525. Springer (1997). https://doi.org/10.1007/BFb0052259, https://doi.org/10.1007/BFb0052259
6. Dehbaoui, A., Dutertre, J., Robisson, B., Tria, A.: Electromagnetic transient faults injection on a hardware and a software implementations of aes. In: 2012 Workshop on Fault Diagnosis and Tolerance in Cryptography. pp. 7–15 (Sep 2012)

7. Dumont, M., Lisart, M., Maurine, P.: Electromagnetic fault injection : How faults occur. In: 2019 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC). pp. 9–16 (2019)
8. Ehmer, M., Khan, F.: A comparative study of white box, black box and grey box testing techniques. International Journal of Advanced Computer Science and Applications **3** (06 2012). https://doi.org/10.14569/IJACSA.2012.030603
9. Laurent, J., Beroulle, V., Deleuze, C., Pebay-Peyroula, F., Papadimitriou, A.: On the importance of analysing microarchitecture for accurate software fault models. In: 2018 21st Euromicro Conference on Digital System Design (DSD). pp. 561–564 (Aug 2018)
10. Moro, N., Dehbaoui, A., Heydemann, K., Robisson, B., Encrenaz, E.: Electromagnetic fault injection: Towards a fault model on a 32-bit microcontroller. In: 2013 Workshop on Fault Diagnosis and Tolerance in Cryptography. pp. 77–88 (2013)
11. Moro, N., Dehbaoui, A., Heydemann, K., Robisson, B., Encrenaz, E.: Electromagnetic fault injection on microcontrollers. In: Chip-to-Cloud Security Forum 2013. Nice, France (Sep 2013), `https://hal-emse.ccsd.cnrs.fr/emse-00871686`
12. Morris, J.: Understanding FIPS 140-2 Validation (2000), `https://csrc.nist.rip/nissc/2000/proceedings/papers/901slide.pdf`
13. Ning, B., Liu, Q.: Modeling and efficiency analysis of clock glitch fault injection attack. In: 2018 Asian Hardware Oriented Security and Trust Symposium (AsianHOST). pp. 13–18 (2018)
14. Oswald, E., Mangard, S., Pramstaller, N., Rijmen, V.: A side-channel analysis resistant description of the aes s-box. In: Gilbert, H., Handschuh, H. (eds.) Fast Software Encryption. pp. 413–423. Springer Berlin Heidelberg, Berlin, Heidelberg (2005)
15. Proy, J., Heydemann, K., Berzati, A., Majéric, F., Cohen, A.: A first isa-level characterization of EM pulse effects on superscalar microarchitectures: A secure software perspective. In: Proceedings of the 14th International Conference on Availability, Reliability and Security, ARES 2019. pp. 7:1–7:10 (2019)
16. Roscian, C., Dutertre, J., Tria, A.: Frontside laser fault injection on cryptosystems - application to the aes' last round -. In: 2013 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST). pp. 119–124 (2013)
17. Sasdrich, P., Moradi, A., Güneysu, T.: White-box cryptography in the gray box. pp. 185–203 (03 2016). https://doi.org/10.1007/978-3-662-52993-5_10
18. Trouchkine, T., Bouffard, G., Clédière, J.: Fault injection characterization on modern cpus: From the isa to the micro-architecture. pp. 123–138 (2019)
19. Whitnall, C., Oswald, E.: A critical analysis of iso 17825 ('testing methods for the mitigation of non-invasive attack classes against cryptographic modules'). Cryptology ePrint Archive, Report 2019/1013 (2019), `https://eprint.iacr.org/2019/1013`
20. Yanci, A.G., Pickles, S., Arslan, T.: Characterization of a voltage glitch attack detector for secure devices. In: 2009 Symposium on Bio-inspired Learning and Intelligent Systems for Security. pp. 91–96 (2009)
21. Yuce, B., Ghalaty, N.F., Santapuri, H., Deshpande, C., Patrick, C., Schaumont, P.: Software fault resistance is futile: Effective single-glitch attacks. In: 2016 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC). pp. 47–58 (Aug 2016)