Defeating Hardware Trojan through Software Obfuscation

Andrea Marcelli, Marco Restifo, Ernesto Sanchez, Giovanni Squillero
Politecnico di Torino
Corso Duca degli Abruzzi 24, 10129, Torino
{andrea.marcelli, marco.restifo, ernesto.sanchez, giovanni.squillero}@polito.it

***Abstract*—In recent years a new kind of threat, known as Hardware Trojan, is affecting the Integrated Circuit industry. Due to the segmentation in the production, untrusted parties involved in the supply chain may illegally inject additional hardware components, that, under specific circumstances, act for malicious purposes. While it is mostly unfeasible to identify malicious hardware tampering with in-lab testing, as a remedy, several countermeasures have been proposed, mostly based on hardware alterations of the original design, with the main drawbacks of increased production costs, increased area and energy consumption. In this paper, we introduce a cost effective solution, completely software-based, that minimize the chance of activation of a multi-stage trigger Hardware Trojan. The proposed approach relies on a software obfuscation mechanism, which exploits evolutionary algorithms to modify an executable program without affecting its original functionalities. Such always-changing, obfuscation routine, can be used to protect critical infrastructures and operations, at a minimum and predictable loss of performances. To show the effectiveness of the proposed technique, we developed a proof-of-concept evolutionary obfuscator and we are going to test it against a well-known real-world hardware attack scenario.**

I. Introduction

The current trend of Integrated Circuits (ICs) of moving to smaller transistors increased the overall hardware performances, while on the other hand dramatically raised the production costs. As a consequence, most of the semi-conductor companies prefer to focus on the circuit design only and outsource to third-parties the fabrication phase. This radical change raises concerns about the trustiness of all the parties involved in the production-chain and the threat of a hardware piracy, through the insertion of a Hardware Trojan Horse (HT) [1], [2], became reality. In order to guard against any type of production error, both intentional and not, companies heavily rely on post-production logic testing. However, since most of the malicious hardware alterations are usually activated by a very low reproducible sequence of events, it is very unlikely that traditional test strategies trigger and detect them.

According to a model presented in [3], the HT conceals its malicious behavior during normal operations: it monitors specific circuit inputs or internal signals and activates the *payload* only when a predefined pattern, known as *trigger*, is detected. However, to decrease the HT chance of being activated during logic testing, usually the trigger is connected to signals with very low controllability, and the payload is attached to signals with very low observability.

In 2016 Yang et al. [4] presented an interesting fabrication-time attack that could lead to a controlled privilege escalation attack. By injecting a single gate and a capacitor in the open spaces of an already placed and routed design, Yang et al. were able to force the flip-flop that holds the privilege bit to a desired value.

In the following we are going to present the threat scenario. Due to the impossibility of verifying any piece of hardware to guarantee its trustworthiness, we conservatively consider any hardware platform as *untrustable* and already *compromised* by malicious parties, according to the model presented in [3]. On the other hand, it is important to remind that the hardware itself only represents the necessary underlying infrastructure to carry out an attack, but it is harmless without a specially crafted software that activates the malicious trigger function. In the same way, we define any software that could be executed on a particular hardware platform as *untrustable* and already *compromised* too.

In order to overcome these limitations, in this paper we propose an innovative approach that completely relies on a software solution that exploits an evolutionary alghorithm to generate a new obfuscated version of the code to be executed. In a previous work [5], we introduced the idea of an evolutionary opcode generator to hide the core functionalities of an always-mutating malicious software, or malware, to challenge antivirus products. In this research, we brought a similar idea to the software-obfuscation of critical code to avoid the triggering of malicious hardware injected components.

To demonstrate the effectiveness of the proposed method, we deployed a proof-of-concept evolutionary obfuscator to challenge the multi-stage trigger introduced by Yang et al. [4]. While detailed experiments are still under analysis, preliminary results show that the software-obfuscation causes the failure of attack at a minimal loss of runtime performances.

The rest of the paper is organized as follows. Section II introduces the proposed approach. Section III shows the current experimental setup, while Section IV discuss the limitations of the solution. Section V concludes the paper.

II. Proposed Framework

In this paper, we propose an evolutionary-based solution to mitigate the possible activation of a multi-stage trigger

HT inserted in a processor core, by modifying the processor software. In particular, the proposed method is able to slightly modify the processor software in order to eliminate the required sequence of instructions that may activate a HT possibly inserted in the targeted processor. According to the model we developed, each software that needs to be executed on a particular hardware platform, must be preprocessed by the evolutionary obfuscator. The core of the software obfuscator is a Turing-complete evolutionary algorithm able to generate completely new variant of the original software without affecting its functionality.

The obfuscation process requires to substitute some sequence of assembly instructions and small blocks of code with complementary ones. Since even a few instructions (i.e., a sequence of 3 or 4 instructions) may trigger the activation of HT, it is necessary to partially shuffle some assembly instructions in the original code, although this has to be done carefully to prevent the potentially modifying of the program functionality. Actually, we propose to modify the original code by applying some of the following mutations to the original code: - Insert a NOP instruction before or after a selected instruction - Substitute the selected instruction with a set of equivalent ones providing the same functionality - Insert a couple of meaningless instructions before or after the selected instruction. - Swap a couple of instructions. This mutation requires a deeper analysis of the program flow in order to do not modify the source program functionality.

In order to efficiently evaluate a candidate solution, the Jaccard Similarity is used to assess the candidate fitness value. Aiming to destroy the trigger capabilities through program diversity, the process is iterated for a given number of generations, or until the Jaccard Index is lower than an experimentally-defined threshold.

Since the application of the software obfuscation introduces a minimum but predictable loss of performances, the run time execution may be affected by the new instruction substitution that could not be optimized for the task under execution. However, as far as the introduced delay is known, it could be added as a constraint during the evolutionary process.

III. Experimental Setup

To demonstrate the effectiveness of the proposed methodology, we implemented several variants of a multi-stage trigger Hardware Trojan on two open source processor cores: the miniMIPS and the OR1200. The choice was dictated by their similarity with the models that are commonly available in the embedded market.

The miniMips [6] is a MIPS-like, 32-bit RISC processor core. It also includes a 5-stage pipeline with hazard detection, a pipeline interlock, a branch prediction unit and a system co-processor. The synthesized version of the processor counts about 46k equivalent gates, while the instruction set is made of 52 instructions, which includes load and store, arithmetic, logic, and branch related instructions.

On the other hand, the OR1200 [7] is a 32-bit RISC processor that adopts the Harvard architecture, a 5 stage of integer pipeline, virtual memory support (MMU) and basic DSP capabilities. It also supports the integration of the cache in the system design, indeed the processor supports a 1-way direct-mapped 8KB data cache and a 1-way direct-mapped 8KB instruction cache, each one with a 16-byte line size. The synthesized version of the processor counts about 50k equivalent gates.

In order to emulate behavior of a HT, both of the processors were manipulated by inserting a special module able to emulate the behavior of a multi-stage HT.

As a software choice, we selected a set of open source benchmarks, specially developed for embedded systems [8].

In particular, we choose benchmark programs related to the automotive industry, among the ones available in the MiBench collection, called *basicmath*, *bitcount*, *qsort*, and *susan*. The MiBench programs include two different set of input values, that is, for every benchmark program is possible to run a small or a large portion of experiments.

IV. Limitations

Similarly to other security solutions that rely on a probabilistic approach, the proposed method is far to be completely reliable. While the proposed evolutionary obfuscator extremely minimize the possibility of triggering malicious hardware components, there exist a rare possibility that, after the obfuscation process, the new assembly instructions will still trigger the activation of the inserted Hardware Trojan. Moreover, it is theoretically feasible that a clean software could be transformed into a malicious one, if the new sequence of instructions perfectly coincide with the triggering sequence. Although both scenarios exist in theory, we are firmly confident that in practice these are so rare, to be considered negligible. In addition, several attacks require a specific timing sequence, that is, instructions to be executed after the hardware payload has been activated. As an example, in the attack described by Yang et al. [4], the change of the privilege mode should be treated accordingly, in order to access the privileged processor state.

Finally, it should be noted that any hardware alteration, both intentional and not, causes an irremediable modification to the hardware structure. Consequently, the original design is forever compromised, resulting in an unpredictable behavior if specific triggering conditions are met. Neither the software obfuscator, nor any other software-based solution, could ever overtake the limitations of a non-proper-working hardware.

V. Conclusion

In this paper, we introduced a novel software obfuscation technique that effectively increase the security of a system in the presence of a hardware affected by malicious alterations. Among the existing Hardware Trojan variants, previous researches showed how the multi-stage trigger currently represents the most hiding, hence powerful, type of attack. Exploiting a pure software evolutionary approach, we are able to prevent, at a minimum and predictable loss of performances, such a helpless scenario.

In order to test the effectiveness of the introduced methodology, we selected some real-world attacks and developed a proof-of-concept of the software obfuscator. While detailed experiments are still under analysis, preliminary results show that the evolutionary obfuscator was successful in preventing privilege escalation attacks.

ACKNOWLEDGMENT

Andrea Marcelli's Ph.D. program at Politecnico di Torino is supported by a fellowship from TIM (Telecom Italia Group).

REFERENCES

[1] X. Wang, M. Tehranipoor, and J. Plusquellic, "Detecting malicious inclusions in secure hardware: Challenges and solutions," in *Hardware-Oriented Security and Trust, 2008. HOST 2008. IEEE International Workshop on*. IEEE, 2008, pp. 15–19.

[2] M. Tehranipoor and F. Koushanfar, "A survey of hardware trojan taxonomy and detection," *IEEE Design and Test of Computers*, vol. 27, no. 1, pp. 10–25, 2010.

[3] S. Dupuis, G. Di Natale, M.-L. Flottes, and B. Rouzeyre, "Identification of hardware trojans triggering signals," in *First Workshop on Trustworthy Manufacturing and Utilization of Secure Devices*, 2013.

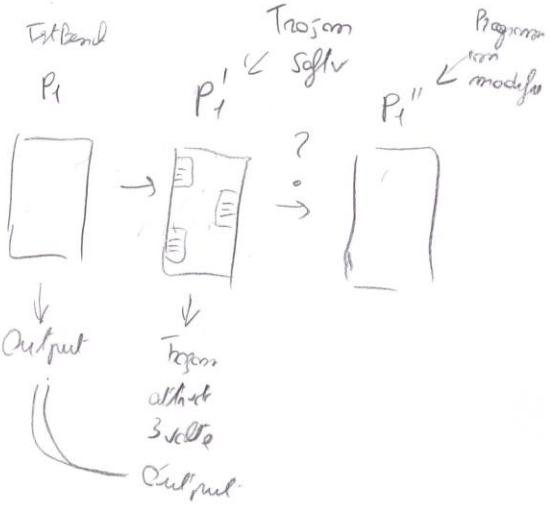
[4] K. Yang, M. Hicks, Q. Dong, T. Austin, and D. Sylvester, "A2: Analog malicious hardware," in *Security and Privacy (SP), 2016 IEEE Symposium on*. IEEE, 2016, pp. 18–37.

[5] M. Gaudesi, A. Marcelli, E. Sanchez, G. Squillero, and A. Tonda, "Challenging anti-virus through evolutionary malware obfuscation," in *European Conference on the Applications of Evolutionary Computation*. Springer, 2016, pp. 149–162.

[6] "minimips," http://opencores.org/project,minimips, accessed: 15-Sep-2016.

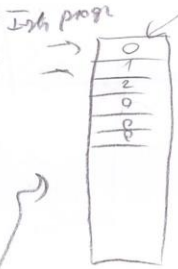
[7] "Openrisc 1200 ip core specification," http://opencores.org/websvn,filedetails?repname=openrisc&path=%2Fopenrisc%2Ftrunk%2For1200%2Fdoc%2Fopenrisc1200_spec_0.7_jp.pdf.

[8] "Mibench," http://vhosts.eecs.umich.edu/mibench//.



Proposta: metrica di distanza (quanto differisce il codice P_1'' da P_1'') MASSIMIZZARE

Goal Tenere output uguale!



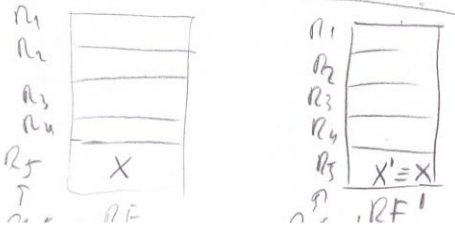
Si prende per P_1' e la si modifica in accordo con le mutazioni polimorfiche

L'evoluzione prende in input questo

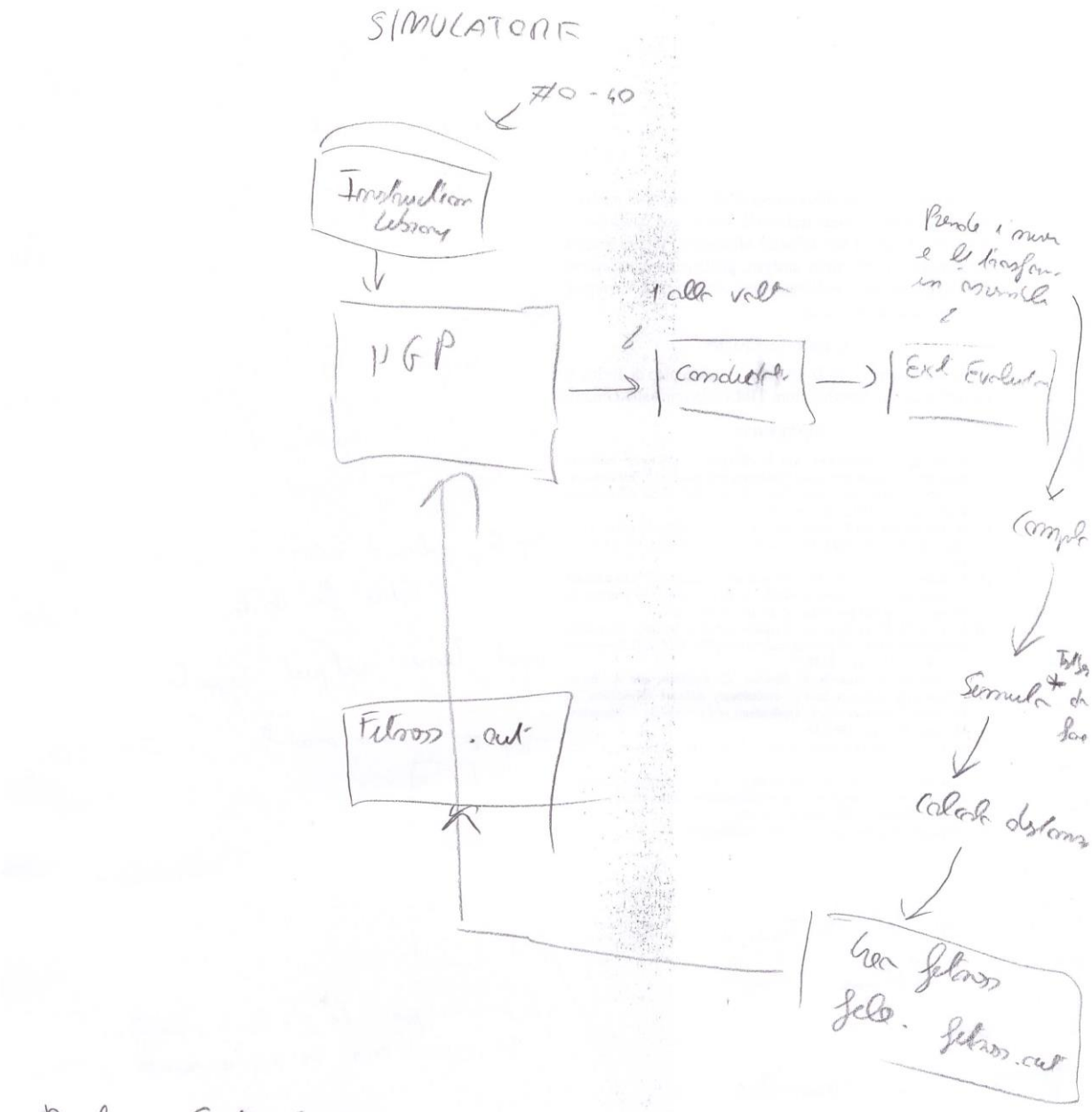
- Inserire una NOP tra I_1, I_2, I_3
- Scambiare Istruzioni
- Fare swap
- Inserire istruzioni equivalenti

NON PERDERE FUNZIONALITÀ

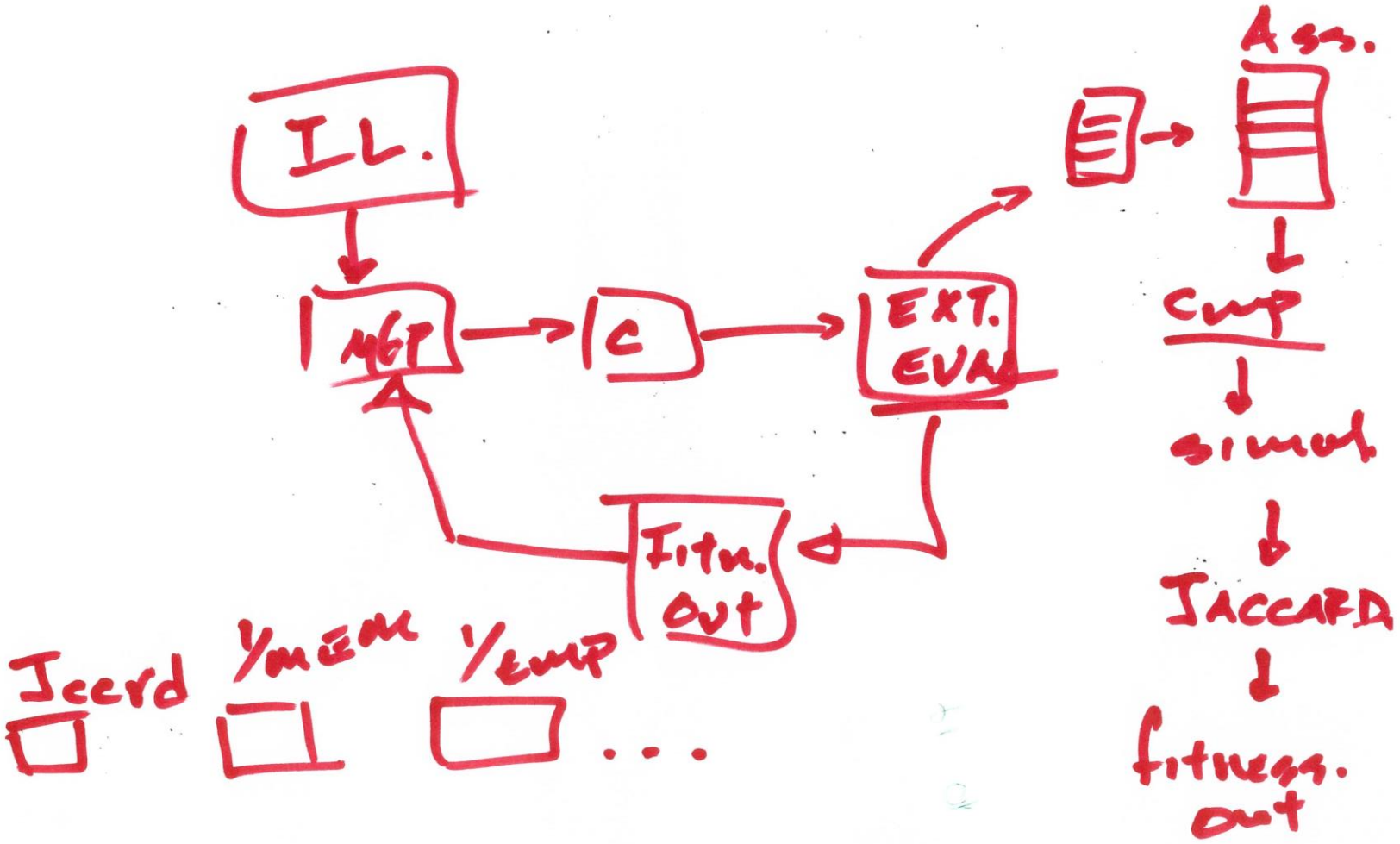
- Verificare output su buon numero di input
- Vedere output uguali
- Applicare una distanza (comando diff su Linux) [calcolare]
- Dimensione & tempo di esecuzione [migliorare]



Il contenuto del Registro Fisico deve essere diverso!!



Da fare Ext Evaluator (è uno script)


IL.
MGP
C
EXT. EVAL
Fitn. out
Jccrd
1/mem
1/emp
...
Ass.
comp
simul.
JACCARD
fitness. out