Real-time detection of hardware trojan attacks on General-Purpose Registers in a RISC-V processor

ShiWei Yuan¹, Lei Li¹, Ji Yang², Yuanchang He³, WanTing Zhou¹, and Jin Li¹

Abstract Hardware trojans (HT) is one of the main threats for hardware security, especially attacks on General-Purpose Registers (GPRs) of processors. This paper presented a novel method to detect HT induced attacks by comparing the states of GPRs with an embedded reference model in real time. Firstly, the instruction sequence was realigned with four principles. Secondly, based on the realigned instruction stream, a reference model for GPRs was built. Finally, HT induced attacks on GPRs can be detected by comparing the expected GPRs’ state from the reference model with the sampled GPRs’ state from processors. We integrated this method into a RISC-V core of PULpino and investigated the feasibility with different programs. The experimental results showed that all the randomly inserted HT attacks can be detected in real time with the latency of two clock cycles.

Keywords: hardware trojan, GPRs, instruction sets, hardware security, RISC-V

1. Introduction

Malicious attacks and revealed hidden backdoors of processors were emerging and becoming a considerable threat [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17]. Hardware trojans (HT) even could be inserted during the fabrication phase and used to attack GPRs of processors directly [18]. Processor hardware security is becoming more and more important and has been well studied. Some works provided physical isolation methods which built safety-first processor system to protect from malicious attacks [9, 20]. Lots of works [19, 20, 21, 22, 23, 24, 25] employed machine learning to detect malicious behaviors based on the internal information of processor. A. Marcelli, et al. [26] used instruction obfuscation to reduce the probability of HT triggers. However, the above studies require complex computation and cannot detect HT induced attacks on GPRs in real time. Based on the instruction sets and microarchitectures of processors, we proposed a simple detection method to supervise HT induced attacks on GPRs of processors in real time, which is firstly presented in the patent of [27] without real implementation. In this paper, we extended our work in [27] and applied it to a RISC-V core.

2. Proposed method

According to the instruction sets architecture and microarchitecture of processors, the states of GPRs are determined by the corresponding instructions. Thus, a reference model can be established by extracting the related information from the instruction sequence and microarchitecture [27]. If a GPR was attacked, the abnormal state can be detected by comparing the expected GPRs’ state from the reference model with the sampled GPRs’ state from processors. In this paper, we applied the proposed method to a RISC-V core, PULpino [28, 29, 30]. Fig. 1 plotted the block diagram of the realization, in which the proposed method was implemented as GPR-State Real-Time Detection Module (GSRTM). GSRTM is consisted of Cycle Decoder, Op Decoder, Realignment unit, State Decoder and Judgement unit.

2.1 Cycle decoder and Op decoder

The life cycle, operation type and instruction type are obtained from the instruction set architecture and microarchitecture of processors. Non-fixed-length instructions are recorded in accordance with the minimum present cycles in which they occur. Based on above information, two decoders of instruction modules are settled. As shown in Fig. 1, Cycle Decoder decodes the inputted instructions into the life cycle information of the corresponding instructions, which is

---

1. Research Institute of Electronic Science and Technology, University of Electronic Science and Technology, Chengdu 611731, China
2. Unit 78102 of the Chinese People’s Liberation Army, Chengdu 610036, China
3. Science and Technology on Communication Security Laboratory, Institute of Southwest Communication, Chengdu 610041, China

a) lilei_uestc@uestc.edu.cn

DOI: 10.1587/elex.18.20210098
Received February 26, 2021
Accepted April 12, 2021
Publicized April 20, 2021
Copyedited May 25, 2021
acquired to realign the current (decoding phase) instruction stream in Realignment module. Next, Op Decoder module is used to obtain the necessary information from the realigned instruction stream, including instruction type and operation type, which will be employed in Judgement module to detect HT induced attacks on GPRs in real time.

2.2 Realignment unit
The function of Realignment shown in Fig. 1 is to output the standard instruction stream signal for the reference model to generate the expected state of GPRs. For in-order processors, instructions are dispatched sequentially and executed sequentially. Generally speaking, subsequent instructions will get stuck in the execution phase to wait for the current instruction (execution or write-back phase) to be completed. As a result, the decoding order of instructions is different from the execution order of instructions (As shown in Fig. 2–4). Besides, the state of GPRs is consistent with the instructions in the execution or write-back phase. Hence, it is necessary to realign the instruction stream in the decoding phase to match the instruction and the state of GPRs completely. Herein, we formulated 4 principles [27] to achieve the realignment of instruction stream.

1. The instruction stream will not be realigned with continuous single-cycle instructions or a single-cycle instruction following a multicycle instruction and another single-cycle instruction. (Load and store instructions in this paper are regarded as single-cycle instructions if they access Data RAM.)

2. If a load or store instruction accessing Peripherals is next to a single-cycle instruction is followed by continuous load or store instructions, the realignment process of instruction stream is shown in Figure 2.

3. If current instruction is a load or store instruction (access Peripherals) next to a multicycle instruction is followed by a single-cycle instruction, the realignment process of instruction stream is shown in Figure 3.

4. If current and next instructions both are load or store instructions (access Peripherals), the third is a multicycle instruction, and the fourth is a single-cycle instruction, the realignment process of instruction stream is shown in Figure 4.

For instructions with uncertain life cycle, the study integrates valid signal in each pipeline stage of the processor, the pipeline conflict and the data conflict indicator signal to solve the problem.

2.3 State decoder
State Decoder (Fig. 1) is the unit for accessing, processing and simplifying the state of GPRs. State Decoder compares sampled values of GPRs in real time with the values of GPRs in preceding clock cycle to determine the state of GPRs. In order to facilitate the subsequent judgment on abnormal state of GPRs, a 2-bit signal is used to simplify the state of the GPRs corresponding to instruction:

1. 2'b00: The instruction does not write a value to the target register, and other registers are not written.
2. 2'b01: The instruction does not write a value to the target register, and other registers are written.
3. 2'b10: The instruction does write a value to the target register, and other registers are not written.
4. 2'b11: The instruction does write a value to the target register, and other registers are written.

2.4 Judgement unit
Judgement (Fig. 1) is used to determine whether GPRs have been attacked by HT. Firstly, Judgement unit uses the realigned instruction stream, life cycle of instruction, operation type and cycle type of instruction to generate the expected state of GPRs (2-bit). Secondly, we compare the state of GPRs generated by the reference model with the actual state of GPRs with using the signal of write port for auxiliary judgment. If they are inconsistent, it meant that the corresponding GPR has been attacked by HT. Finally, the corresponding instruction, PC value and clock cycles will be provided to the processor for warning and self-protection.

3. Experimental results
We injected several HTs into GPRs module of processor to maliciously tamper a random register of GPRs during program operation. Five programs [30] with different functions were used for test, and the test results were shown in Table I. The experimental results showed that all the randomly inserted HT attacks could be detected only with the alignment process of instruction stream is shown in Figure 3.

For instructions with uncertain life cycle, the study integrates valid signal in each pipeline stage of the processor, the pipeline conflict and the data conflict indicator signal to solve the problem.

### Table I Detection results of abnormal state of GPRs

| Code.c | HT number | DT number | Percentage |
|--------|-----------|-----------|------------|
| testALU | 50        | 50        | 100%       |
| testCnt | 38        | 38        | 100%       |
| testClip | 18       | 18        | 100%       |
| testMUL | 82        | 82        | 100%       |
| testDIVREM | 17      | 17        | 100%       |

1 The number of effective Hardware Trojans implanted.
2 The number of abnormal states of GPRs detected.
latency of two clock cycles. The proposed method employs the pipeline structure and can detect HT induced attacks on GPRs each clock cycle, while the work [9] has at least 100us check interval length for HT detection with a performance loss range from 0.98% to 22.71%.

4. Conclusion

This paper presented a real-time detection method for abnormal GPRs’ state of processor. GSRTM was embedded in the RISC-V core of PULpino for test. The detection results were 100% accurate with the latency of two clock cycles. In conclusion, the presented method for detecting hardware trojan attacks was effective on GPRs in RISC-V processors.

Acknowledgments

This work was supported by Sichuan Science and Technology Program (No.2021YJ0082) and the National Defense Science and Technology Key Laboratory Fund (No.6142103190310).

References

[1] N.G. Tsoutsos, et al.: “Fabrication attacks: zero-overhead malicious modifications enabling modern microprocessor privilege escalation,” IEEE T EMERG TOP COM 2 (2014) 81 (DOI: 10.1109/TETC.2013.2287186).
[2] Y. Jin, et al.: “Exposing vulnerabilities of untrusted computing platforms,” ICCD Papers (2012) 131 (DOI: 10.1109/ICCID.2012.6378629).
[3] R. Karri, et al.: “Trustworthy hardware: identifying and classifying hardware Trojans,” Computer 43 (2010) 39 (DOI: 10.1109/MC.2010.299).
[4] L. Duflo: “CPU bugs, CPU backdoors and consequences on security,” J. Comput. Virol. 5 (2009) 91 (DOI: 10.1007/s11416-008-0109-x).
[5] L. Zhou, et al.: “Hardware-based on-line intrusion detection via system call routine fingerprinting,” DATE Papers (2017) 1546 (DOI: 10.23919/DATE.2017.7927236).
[6] J. Sutherland, et al.: “CPU covert channel accessible from JavaScript,” CyberForensics (2014).
[7] C. Domas: “Hardware Backdoors in x 86 CPUs,” (2018).
[8] P. Kocher, et al.: “Spectre attacks: exploiting speculative execution,” SP Papers (2019) 1 (DOI: 10.1109/SP.2019.00002).
[9] L. Liu, et al.: “Jintide®: a hardware security enhanced server CPU with Xeon® cores under runtime surveillance by an in-package dynamically reconfigurable processor,” HCS Papers (2019) 1 (DOI: 10.1109/HOTCHIPS.2019.8875682).
[10] C. Bolchini, et al.: “HATE: a HArdware Trojan Emulation environment for microprocessor-based systems,” IOLTS Papers (2019) 109 (DOI: 10.1109/IOLTS.2019.8854414).
[11] T. Hoque, et al.: “Hardware Trojan attacks in embedded memory,” VTS Papers (2018) 1 (DOI: 10.1109/VTS.2018.8368630).
[12] X. Wang, et al.: “Software exploitable hardware Trojans in embedded processor,” DFT Papers (2012) 55 (DOI: 10.1109/DFT.2012.6378199).
[13] B.K. Natarajan and V. Srinivasan: “Logic obfuscation technique using configurable gate diffusion input for improved hardware security,” IEICE Electron. Express 15 (2018) 20180802 (DOI: 10.1587/elex.15.20180802).
[14] M. Tehranipoor and C. Wang: Introduction to Hardware Security and Trust (Springer-Verlag, New York, 2012) 231 (DOI: 10.1007/978-1-4419-8080-9).
[15] R. Wojtczuk, et al.: “Attacking SMM memory via Intel CPU cache poisoning,” Invisible Things Lab Papers (2009).
[16] M. Schwarz, et al.: “Malware guard extension: using SGX to conceal cache attacks,” DIMVA 10327 (2017) 3 (DOI: 10.1007/978-3-319-60876-1_1).
[17] E.D. Mulder, et al.: “INVITED: protecting RISC-V against side-channel attacks,” DAC Papers (2019) 1 (DOI: 10.1145/3316781.3323485).
[18] K. Yang, et al.: “A2: analog malicious hardware,” SP Papers (2016) 18 (DOI: 10.1109/SPW.2016.10).
[19] P. Okane, et al.: “Malware detection: program run length against detection rate,” IET Softw. 8 (2014) 42 (DOI: 10.1049/iet-sen.2013.0020).
[20] D. Meng, et al.: “Built-in security computer: deploying security-first architecture using active security processor,” IEEE Trans. Comput. 69 (2020) 1571 (DOI: 10.1109/TC.2020.3011748).
[21] P.O’Kane, et al.: “SVM training phase reduction using dataset feature filtering for malware detection,” IEEE Trans. Inf. Forensics Security 8 (2013) 500 (DOI: 10.1109/TIFS.2013.2242890).
[22] M. Ozsoy, et al.: “Hardware-based malware detection using low-level architectural features,” IEEE Trans. Comput. 65 (2016) 3332 (DOI: 10.1109/TC.2016.2540634).
[23] F. Li, et al.: “A deep malware detection method based on general-purpose register features,” ICS 11538 (2019) 221 (DOI: 10.1007/978-3-030-22744-9_17).
[24] Y. Ding, et al.: “Control flow-based opcode behavior analysis for Malware detection,” Computers & Security 44 (2014) 65 (DOI: 10.1016/j.cose.2014.04.003).
[25] L. Chen, et al.: “HeNet: a deep learning approach on Intel® processor trace for effective exploit detection,” SPW Papers (2018) 109 (DOI: 10.1109/SPW.2018.00025).
[26] A. Marcelli, et al.: “On the mitigation of hardware Trojan attacks in embedded processors by exploiting a hardware-based obfuscator,” IVSW Papers (2018) 31 (DOI: 10.1109/IVSW.2018.8494850).
[27] L. Li, et al.: China Patent 202110162587.6 (2021).
[28] https://www2.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-161.pdf (2017)
[29] https://pulp-platform.org/docs/pulpino_datasheet.pdf
[30] PULpino Project: https://github.com/pulp-platform/pulpino