Seon Ha
Security Researcher

Email : seonha@unist.ac.kr
https://h0tsun.github.io

Education

- **Ulsan National Institute of Science and Technology** — Ulsan, Korea
  PH.D of Science in Computer Science; — *Mar. 2021 –*
- **Ulsan National Institute of Science and Technology** — Ulsan, Korea
  Master of Science in Computer Science; — *Mar. 2019 – Feb. 2021*
- **University of Science and Technology** — Daejeon, Korea
  Master of Information Security Engineering; — *Mar. 2018 – Aug. 2018*
- **Dalian Ocean University** — Dalian, China
  Exchange Student; — *Aug. 2016 – Feb. 2017*
- **Pukyong National University** — Busan, Korea
  Bachelor of Engineering in information and communication engineering; — *Mar. 2013 – Feb. 2018*

Research Interest

- **Building secure software system**: I am particularly interested in secure KVS(Key-value store) system. I am interested in improving the security of the system.
- **Software compartmentalization**: Software compartmentalization is an approach that reduces the impact of one vulnerability by splitting software into pieces.
- **Hardware extension for security**: Hardware extensions for secure system.

Experience

- **ETRI(Electronics and Telecommunications Research Institute)** — Daejeon, Korea
  Software Engineer — *Jul. 2017 - Oct. 2017*
  - **Indoor Localization**: I improved the accuracy of beacon recognition indoors.

Publications

- **Protecting Kernel Code integrity with PMP on RISC-V** — WISA 2023
  Seon Ha and Hyungon Moon
  - **Kernel code integrity**: Kernel code integrity is the foundation of the security of the entire system. Attackers are motivated to compromise the kernel code integrity because it gives them the highest possible privilege on the system, allowing them to take the full control of it. They can perform the attack by either modifying the kernel code directly or tricking the kernel to execute from data pages. Existing kernels and processors are working together to defeat this threat, but their reliance on the page table leaves the attackers leeway to bypass the protection. Existing solutions aiming to tackle this limitation, the reliance on the page table integrity, are either too expensive or require custom hardware. In this paper, we present a software-only design of a kernel code integrity protection mechanism for RISC-V-based systems that implement the Physical Memory Protection (PMP). We show that, despite the lack of direct support for kernel code protection, the kernel and the machine mode firmware can work together to leverage the PMP to defeat the advanced kernel code integrity-compromising attacks by dynamically switching the memory protection policies on user-kernel switches. The performance estimation using our prototype shows that the proposed mechanisms incur moderate (¡24%) overhead on system call latencies. The security evaluation using synthetic advanced attacks also demonstrates that the proposed mechanism can effectively prevent the page table-corrupting kernel code injection attacks.
- **Kernel Code Integrity Protection at the Physical Address Level on RISC-V** — ITEE Aceess
  Seon Ha, Minsang Yu, Hyungon Moon, Jongeun Lee
  - **Kernel code integrity**: An operating system kernel has the highest privilege in most computer systems, making its code integrity critical to the entire system's security. Failure to protect the kernel code integrity allows an attacker to modify the kernel code pages directly or trick the kernel into executing instructions stored outside the kernel code pages. Existing prevention mechanisms rely on the memory management unit in which certain memory pages are marked as not-executable in supervisor mode to prevent such attacks. However, an attacker can bypass these existing mechanisms by directly manipulating the page table contents to mark the memory pages with

malicious code as supervisor-executable. This paper shows that a small architectural extension enables a physical address-level mechanism to stop this threat without relying on page table integrity. PrivLock lets, at boot time, the kernel specifies the physical address ranges containing its code. At run time, PrivLock ensures that the content within the range is not manipulated and that only the instructions from those pages are executed while the processor runs in supervisor mode. Despite this protection, the kernel can still create new code pages (e.g., for loadable kernel modules) and make them executable with the help of PrivLock's secure loader. The experimental results show that PrivLock incurs low performance (¡0.5%), area (0.14–0.3%), and energy/power (0.053–2%) overhead.

Awards

- **Korea information security BOB Idea cup**
  KETRI(Korea Information Technology Research Institute), Korea *Feb. 2015*
  Netizen special prize: Create excellent information protection ideas and raise public awareness of security.

Activities

- **CERT-IS** Pukyong National University Security Club
  Member *Mar. 2014 – Feb. 2018*
- **Microsoft Student Partners 8th** Microsoft Korea
  Member *Aug. 2014 – Aug. 2015*

Programming Skills

- **Languages**: Scala, C++, C, Python, CHISEL/FIRRTL, verilog