Poster: SecTAG: Enhancing ARM Platform Security with Hardware Tags

Zhenyu Ning and Fengwei Zhang
COMPASS Lab
Wayne State University
{zhenyu.ning, fengwei}@wayne.edu

I. Abstract

ARM architecture has dominated the mobile device market since the born of smart phones. In recent years, the ARM architecture is further deployed in Internet of Things (IoT) devices [1], [13], [16] and commercial cloud platforms [20]. With the rapid growth of the market share, the security of the ARM platforms is still a critical problem.

In the world of security, there exist some common security problems in both x86 and ARM architectures. For example, the integrity of the pointers is a big concern. In real-world devices, the adversaries have demonstrated a series of attacks to manipulate the pointers at runtime [6], [8]. To defend against the data and code pointer attacks, security researchers present different solutions [7], [19]. However, these solutions are either targeting a specific scenario or introducing a large performance overhead. To solve the problem with a low-performance penalty, ARMv8.3 architecture proposes the hardware-based Pointer Authentication Code (PAC) [2], [12]. However, the only available ARMv8.3 processor on the market at this moment is the Apple A12, which is not open to researchers. Thus, the pointer integrity problem on the ARMv8 architecture, which is the mainstream ARM architecture, is still not solved.

Another common security concern is the memory boundaries. Buffer overflow attack and its variations are frequently used to as the first step of a sophisticated attack. To solve the problem, software-based solutions such as AddressSanitizer [17], SAFECode [5], and SoftBound [14] have been proposed. Similar to the pointer integrity problem discussed above, these software-based solutions introduce high overhead. Intel introduces Memory Protection Extension (MPX) as a hardware-based solution to verify the memory boundaries before access it via additional bound registers. However, the Intel MPX is not production ready [15], [18], and the support of Intel MPX will be removed from GCC compiler and Linux kernel [10], [11].

Dynamic taint analysis is typical used to detect the information leakage at runtime. Normally, it is achieved by compile-time instrumentation or runtime monitoring. The instrumentation-based solution requires fine-grain instruction instrumentation to propagate the taint tags between different hardware registers, and thus introduces high performance overhead. In regard to the monitoring-based solution, the problem would be how to monitor the executed instructions with a low-performance overhead and solve the taint-tag dependencies in real-time. Although both ARM and Intel proposed hardware-based instruction trace solutions [3], [9], the design of the hardware solutions determines that it is not easy to implement a synchronous dynamic taint analysis mechanism with a low runtime performance overhead.

In this poster, we propose SecTAG to solve the above problems with hardware tags in 64-bit ARMv8 architecture. Specifically, we use hardware tags to enable the following three security features on ARM platforms: 1) Pointer Integrity, 2) Memory Boundary Protection, and 3) Dynamic Taint Analysis. In the 64-bit ARMv8 architecture, each memory address contains 64 bits. However, not all these bits are actually used during CPU addressing.

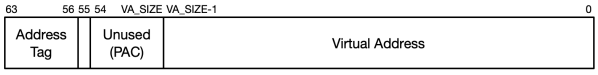

Figure 1: Address in 64-bit ARMv8 architecture.

Figure 1 shows the 64-bit address in ARMv8 architecture. The `VA_SIZE` means the actual size of the virtual address used in the system, whose maximum value is 48. The bits {0...`VA_SIZE`-1} are used as the virtual address during CPU addressing. The bits {54...`VA_SIZE`} is reserved in ARMv8 architecture and used as PAC bits in ARMv8.3 architecture. The top byte (bits {63...56}) is ignored during CPU addressing if the corresponding Top Byte Ignored (`TBI`) bit in `TCR` register [1] is set, while the bit 55 determines which `TBI` bit (`TBI0` or `TBI1`) in `TCR` register should be used. Once the corresponding `TBI` bit is set, we can use the top byte of a 64-bit address as the address tag.

Pointer Integrity. As aforementioned, ARM introduces PAC to protect the pointer integrity since ARMv8.3 architecture. Although PAC is not available in the ARMv8 platforms, the address tag feature is deployed in these platforms. Thus, we can use the address tag to simulate the PAC to protect the pointer integrity in ARMv8 architecture. Other than the PAC bits in a 64-bit address, ARM also introduces additional instructions to perform QARMA [4] encryption to calculate the PAC. To reduce the performance overhead of additional

[1] In ARM architecture, the Translation Control Register (TCR) is used to configure the address translation.

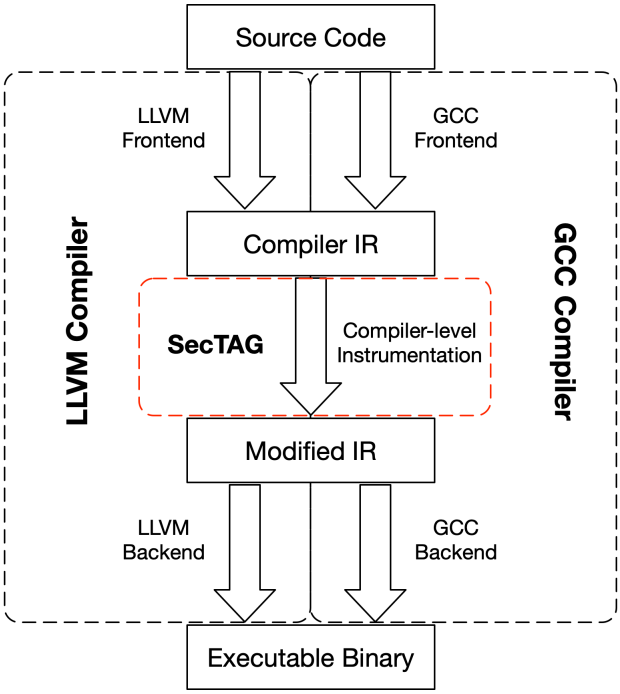

Figure 2: Design of SECTAG.

software implementation of the encryption algorithm, we propose to use the hardware-based AES introduced in ARMv8 architecture instead of QARMA as the encryption algorithm.

Memory Boundary Protection. Inspired by Intel MPX, we may also use the address tag to indicate the high- and low-boundary of a pointer. The boundary is calculated while the pointed memory region is allocated and verified before any usage of the pointer. Unlike the Intel MPX, which uses additional registers and memory to store the boundary, the address tag sticks to the pointer. Thus, the tag-based boundary would not suffer from the Intel MPX issues such as multi-threading problem and heavy performance overhead in bounds table lookup.

Dynamic Taint Analysis. Taint tag and its propagation are critical to dynamic taint analysis. Intuitively, the address tag can also play the role of taint tag in dynamic taint analysis. Since the address tag will be automatically transferred between registers while executing the data processing instructions, we can save a large amount of instruction instrumentation for the taint tag propagation. With the elimination of the additional propagation instructions, the performance of the instrumentation-based dynamic taint analysis would be greatly improved.

Figure 2 shows the design of SECTAG. Specifically, we plan to leverage compiler-level instrumentation to achieve the proposed system, and the LLVM and GCC compiler would be supported with a dedicate compiler plugin. The source code of the application/kernel is first compiled by the LLVM/GCC compiler frontend, and SECTAG will perform instruction instrumentation based on the resulting compiler Intermediate Representation (IR). The backend of the compiler will further transfer the instrumented IR to the executable binary.

In summary, SECTAG would enhance the security of ARM platforms via using the hardware address tags. In particular, it provides novel security functions such as Pointer Integrity, Memory Boundary Protection, and Dynamic Taint Analysis. We consider this hardware-based solution will bring a performance boost on existing security-related protection and detection mechanisms. Moreover, since SECTAG is based on the widely deployed 64-bit ARMv8 devices, it will be practical to deploy our solution on existing product devices.

REFERENCES

[1] Apple, "HomeKit," https://developer.apple.com/homekit/.

[2] ARM, "ARMv8-A reference manual," http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0487a.k/index.html.

[3] ——, "Embedded trace macrocell architecture specification," http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ihi0014q/index.html.

[4] R. Avanzi, "The QARMA block cipher family," *IACR Transactions on Symmetric Cryptology*, 2017.

[5] D. Dhurjati, S. Kowshik, and V. Adve, "SAFECode: Enforcing alias analysis for weakly typed languages," in *Proceedings of the 27th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI'06)*, 2006.

[6] I. Evans, S. Fingeret, J. Gonzalez, U. Otgonbaatar, T. Tang, H. Shrobe, S. Sidiroglou-Douskos, M. Rinard, and H. Okhravi, "Missing the point(er): On the effectiveness of code pointer integrity," in *Proceedings of 36th IEEE Symposium on Security and Privacy (S&P'15)*, 2015.

[7] H. Hu, C. Qian, C. Yagemann, S. P. H. Chung, W. R. Harris, T. Kim, and W. Lee, "Enforcing unique code target property for control-flow integrity," in *Proceedings of the 25th ACM SIGSAC Conference on Computer and Communications Security (CCS'18)*, 2018.

[8] H. Hu, S. Shinde, S. Adrian, Z. L. Chua, P. Saxena, and Z. Liang, "Data-oriented programming: On the expressiveness of non-control data attacks," in *Proceedings of 37th IEEE Symposium on Security and Privacy (S&P'16)*, 2016.

[9] Intel, "Processor tracing," https://software.intel.com/en-us/blogs/2013/09/18/processor-tracing, 2013.

[10] M. Larabel, "GCC 9 looks set to remove Intel MPX support," https://www.phoronix.com/scan.php?page=news_item&px=GCC-Patch-To-Drop-MPX, 2018.

[11] ——, "The Linux kernel might drop memory protection extensions support," https://www.phoronix.com/scan.php?page=news_item&px=Linux-Kernel-Weighing-Intel-MPX, 2018.

[12] H. Liljestrand, T. Nyman, K. Wang, C. C. Perez, J.-E. Ekberg, and N. Asokan, "PAC it up: Towards pointer integrity using ARM pointer authentication," in *Proceedings of the 28th USENIX Security Symposium (USENIX Security'19)*, 2019.

[13] Microsoft, "Azure sphere," https://www.microsoft.com/en-us/azure-sphere/.

[14] S. Nagarakatte, J. Zhao, M. M. Martin, and S. Zdancewic, "SoftBound: Highly compatible and complete spatial memory safety for c," in *Proceedings of the 30th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI'09)*, 2009.

[15] O. Oleksenko, D. Kuvaiskii, P. Bhatotia, P. Felber, and C. Fetzer, "Intel MPX explained: A cross-layer analysis of the intel mpx system stack," in *Proceedings of the ACM 2018 International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS'18)*, 2018.

[16] Samsung, "Artik," https://www.artik.io/.

[17] K. Serebryany, D. Bruening, A. Potapenko, and D. Vyukov, "Address-Sanitizer: A fast address sanity checker," in *Proceedings of the 2012 USENIX Annual Technical Conference (USENIX ATC'12)*, 2012.

[18] K. Serebryany, "Address sanitizer and Intel memory protection extensions," https://github.com/google/sanitizers/wiki/AddressSanitizerIntelMemoryProtectionExtensions, 2016.

[19] C. Song, B. Lee, K. Lu, W. Harris, T. Kim, and W. Lee, "Enforcing kernel security invariants with data flow integrity," in *Proceedings of 23rd Network and Distributed System Security Symposium (NDSS'16)*, 2016.

[20] C. Williams, "Can't wait for ARM to power MOST of our cloud data centers," https://www.theregister.co.uk/2017/03/09/microsoft_arm_server_followup/.