Attacks on encrypted memory and constructions for memory protection

Extended Abstract

Shay Gueron[1,2]

[1] University of Haifa, Israel
[2] Intel Corporation, Israel Deveopment Center, Haifa, Israel
shay@math.haifa.ac.il

Active attacks on memory, and their potential implications

The first part of the talk discusses some potential implications of attacks on the system memory of a computing platform. We show that encryption to protect privacy is not necessarily sufficient to protect against active attacks. This work was demonstrated by Branco and Gueron in [5] (and will be detailed in the proceedings of HOST 2016).

Physical attacks on devices are currently a major concern, especially for the mobile market (risk of theft/loss) and for cloud computing environments where an un-trusted entity has the ability to arbitrarily change memory. Several attacks that leverage capabilities to read and/or write the physical memory have been demonstrated. We give a few examples. Attacks that passively compromise system secrets (e.g., [11]), bypass policies and security mechanisms (e.g., [3] [9] [18]), install malware [22], or even run an entire exploit payload [6]). Different interfaces were used in order to demonstrate the feasibility of reading/writing the physical memory, as in ([4] where it uses a FireWire interface and [19] where it uses the Thunderbolt interface.

Tools that facilitate physical access to the memory serve legitimate purposes: commercial entities and researchers have been using the memory primitives for remote debugging [21], memory capture with forensics purposes [7], and legal interception [23].

The above examples assumed the same adversarial capabilities: the attacker can read values that reside on the memory (DRAM), at least at selected addresses, and can overwrite them with values of his choice. Note that the ability to first read a DRAM value and subsequently overwrite it, allows the attacker to change the DRAM contents at a bit-level precision, thus to change code behavior at will. Such scenarios have a devastating potential.

Several mechanisms to mitigate memory modification attacks have been proposed. One example is to block DMA-capable devices on exposed equipment ports [17]. This overlooks the fact that an attacker may have other means to read/write the DRAM. Another proposed approach is memory encryption (e.g., [12], [14]). Transparent encryption is appealing, compared to encryption with

authentication. It can be viewed as an additional processing step on the CPU-DRAM traffic, without changing the data size, and is therefore relatively simple to implement.

Memory encryption that protects the confidentiality of the DRAM contents, changes, as a byproduct, the capabilities of the active attacker. While the same technical tools are available to the attacker, operating on *encrypted* memory allows him to execute only what we call a Blinded Random Block Corruption (BRBC) attack, where the following limitations apply:

A. (**Blinded**) The attacker does not know the plain-text memory values that he can read from the (encrypted) memory.
B. (**Random (Block) Corruption**) The attacker cannot control or predict the final plain-text that would be used by the system when a modified DRAM value is read from the DRAM, and decrypted. Additionally, any changes affect an entire block in case of block-based ciphers.

Obviously, reducing the active attacker's capabilities to only BRBC attacks is not a theoretically sound protection. Like the clear-text memory attacker, the BRBC attacker is also able to create time-of-check/time-of-use (TOCTOU) [8] race conditions all around the execution environment (e.g., across different processes and parts of the operating system). Such attacks can invalidate results of checks that the code executes, and lead to unexpected security issues. If we assume that code is driven by its input data [20], and that the attacker can change data after the code checks, compromises to a complex system security premisses are possible. Therefore, the following question arises:

> *Does reducing the attacker's capability to only BRBC attacks constitute a "good enough protection" - even if only from the practical viewpoint?*

We show here that the answer is not necessarily positive. This stands in contrast to to the perception (hope) that, given the removal of control over modifications made, and the lack of visibility of current memory contents, the attacker would not be able to leverage his capabilities to gain privileges. That perception exists due to the difficulty of controlling any code modifications, but overlook the potential of data-only attacks. To this end, we demonstrate a BRBC attack where the adversary has the least possible level of control on the platform, and can login as "root" on a locked system, assuming that he has the ability to overwrite the memory.

We show two implementations of this attack. One uses the JTAG interface to overwrite the DRAM, and is fully practical on many systems (if the adversary has the equipment). The other is a *simulated* attack that uses the same principles, but simulates the malicious memory accesses by using a debugger. It can therefore be reproduced without any special (expensive) equipment.

The Memory Encryption Engine

In the second part of the talk, we explore the Memory Encryption Engine [10], which is part of Intels Software Guard Extensions (SGX) technology (see [1], [2], [16], [13], [15]).

SGX is a security technology, designed to allow a general purpose computer platform to run application software in a trustworthy manner, and to handle secrets that are inaccessible to anyone outside the defined trust boundaries. These trust boundaries encompass only the CPU internals, implying, in particular, that the system memory is untrusted

To protect against memory snooping and tampering attacks, SGX is supported by an autonomous hardware unit called the Memory Encryption Engine (MEE), whose role is to protect the confidentiality, integrity, and freshness of the CPU-DRAM traffic over some memory range.

We explain the engineering challenges involved with adding a hardware unit to the micro architecture of a general purpose processor (in a real product) , describe the MEE design, algorithms and cryptographic properties, and describe some of its performance characteristics.

References

1. Intel ® Software Guard Extensions Programming Reference, 2014. https://software.intel.com/en-us/isa-extensions/intel-sgx.
2. Ittai Anati, Shay Gueron, Simon Johnson, and Vincent Scarlata. Innovative technology for cpu based attestation and sealing. In *Proceedings of the 2nd International Workshop on Hardware and Architectural Support for Security and Privacy*, volume 13, 2013.
3. D. Aumaitre and C. Devine. Subverting Windows 7 x64 Kernel with DMA attacks. Hack In the Box Conference, 2010.
4. M. Becher, M. Dornseif, and C. N. Klein. FireWire all your memory belong to us. CanSecWest Security Conference, 2005.
5. R. Branco and S. Gueron. Blinded random corruption attacks. Poster, IEEE International Symposium on Hardware Oriented Security and Trust (HOST), 2016.
6. R. Breuk and A. Spruyt. Integrating DMA attacks in exploitation frameworks. Technical Report, 2012.
7. B. D. Carrier and J. Grand. A Hardware-Based Memory Acquisition Procedure for Digital Investigations. *Digital Investigation Journal*, 1, 2003.
8. Common Weakness Enumeration. Time-of-check/Time-of-use Race Condition, 2014. Online; accessed 31-October-2015.
9. L. Duflot, Y. Perez, and B. Morin. What if you can't trust your network card? In *Recent Advances in Intrusion Detection*, pages 378–397. Springer-Verlag Berlin Heidelberg, 2011.
10. Shay Gueron. A memory encryption engine suitable for general purpose processors. Cryptology ePrint Archive, Report 2016/204, 2016. http://eprint.iacr.org/.
11. J. A. Halderman, S. D. Schoen, N. Heninger, W. Clarkson, W. Paul, J. A. Calandrino, A. J. Feldman, J. Appelbaum, and E. W. Felten. Lest we remember: Cold Boot Attacks on Encryption Keys. *17th USENIX Security Symposium*, 2008.
12. M. Henson and S. Taylor. Memory Encryption: A Survey of Existing Techniques. *ACM Computing Surveys*, 46, 2013. Online; accessed 31-October-2015.
13. Matthew Hoekstra, Reshma Lal, Pradeep Pappachan, Vinay Phegade, and Juan Del Cuvillo. Using innovative instructions to create trustworthy software solutions. In *Proceedings of the 2Nd International Workshop on Hardware and Architectural Support for Security and Privacy*, HASP '13, pages 11:1–11:1, New York, NY, USA, 2013. ACM.

14. O. S. Hofmann, S. Kim, A. M. Dunn, M. Z. Lee, and E. Witchel. InkTag: Secure Applications on an Untrusted Operating System. *ASPLOS*, 48, 2013. Online; accessed 31-October-2015.
15. Simon Johnson, Vincent Scarlata, Carlos Rozas, Ernie Brickell, and Frank Mckeen. Intel^{TM} Software Guard Extensions: EPID provisioning and attestation services. *White Paper*, April 2016.
16. Frank McKeen, Ilya Alexandrovich, Alex Berenzon, Carlos V. Rozas, Hisham Shafi, Vedvyas Shanbhogue, and Uday R. Savagaonkar. Innovative instructions and software model for isolated execution. In *Proceedings of the 2Nd International Workshop on Hardware and Architectural Support for Security and Privacy*, HASP '13, pages 10:1–10:1, New York, NY, USA, 2013. ACM.
17. Microsoft Corporation. Protecting Bitlocker-encrypted devices from attacks. Microsoft Technical Report, 2015. Online; accessed 31-October-2015.
18. F. L. Sang, V. Nicomette, and Y. Deswarte. I/O Attacks in Intel-PC Architectures and Countermeasures. Technical Report, 2011. Online; accessed 31-October-2015.
19. R. Sevinsky. Adventures in Thunderbolt DMA Attacks. Black Hat USA, 2013.
20. R. Shapiro, S. Bratus, and S. W. Smith. Weird Machines in ELF: A Spotlight on the Underappreciated Metadata. *7th USENIX Workshop on Offensive Technologies (WOOT)*, 2013. Online; accessed 31-October-2015.
21. J. Stecklina. Remote debugging via Firewire. Technische Universitat Dresden - Diploma Thesis, 2009. Online; accessed 31-October-2015.
22. P. Stewin and I. Bystrov. Understanding DMA malware. In *Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 21–41. Springer Berlin Heidelberg, 2013.
23. Wikipedia. FinFisher intrusion tools, 2011. Online; accessed 31-October-2015.

Acknowledgments

This research was partly supported by the Blavatnik Interdisciplinary Cyber Research Center (ICRC) at the Tel Aviv University.
Opinions, findings and conclusions or recommendations expressed in this material are those of the author, and do not necessarily reflect the views of his employers or funding agencies.