Improving Confidentiality Against Cache-based SCAs

Maria Mushtaq, Vianney Lapotre, Guy Gogniat
Lab-STICC, University of South Brittany
Lorient, France
{maria.mushtaq, vianney.lapotre,
guy.gogniat}@univ-ubs.fr

M. Asim Mukhtar, Muhammad Khurram Bhatti
ECLab, Information Technology University
Lahore, Pakistan
{asim.mukhtar, khurram.bhatti}@itu.edu.pk

ABSTRACT

Side channels and *covert channels* can give *untrusted* applications access to the trusted and sensitive data in order to retrieve private information.. In this poster, we present a countermeasure called the *Smartflush* against cache-based Side Channel Attacks (SCAs). The Smartflush is a quick-patch countermeasure proposed to counter timing attacks that exploit inclusive caches in Intel's x86 architecture. The proposed countermeasure is tested against recent attacks like Flush+Reload. Results show that it improves the confidentiality of data with minimum or no performance degradation.

Keywords

Cryptography, Side-Channel Attacks, RSA, Countermeasure, Caches.

1. EXTENDED ABSTRACT

Side-channel attacks are a powerful method for breaking theoretically secure cryptographic primitives. In recent years, cache-based SCAs have become an eminent threat to cryptographic algorithms, such as RSA and AES, as they cause inter-process information leakage through measurement of timing variations of cryptographic operations and observation of cache access patterns [1], [2]. The success of cache-based timing SCAs mainly depends on two factors: the ability of *spy* (malicious) process to detect and synchronize itself with the *target* (victim) process and the presence of page sharing mechanisms like Transparent Page Sharing (TPS) or Kernel Same-page Merging (KSM). A recently proposed timing SCA, Flush+Reload [2], exposes these vulnerabilities on Intel's x86 architecture by exploiting its *inclusive* caches. The attack targets RSA cryptographic algorithm's *exponentiation by squaring* implementation, which uses a specific pattern of Square, Multiply, and Subtract operations. The attack uses *clflush* instruction to evict selected cache lines corresponding to these instructions from Last Level Cache (LLC). Due to inclusivity, the instructions are evicted from all other levels as well. The attack then reloads the same cache lines after a prefixed wait time to monitor the timing variations and determines whether the targeted instruction is present in the cache (thus being used by the victim process) or loaded from the main memory. Flush+Reload attack affects the confidentiality by disclosing memory addresses of cryptographic operations performed by RSA.

Our poster presents a countermeasure, called *Smartflush*, to improve confidentiality of cryptographic operations against *Flush+Reload* and similar cache-based timing SCAs. These attacks observe timing variations, i.e., difference between cache hit and miss times, generated by the instructions of targeted operations. Based on these variations, the spy process captures the precise execution pattern of victim process. We propose a mechanism to *confuse* the spy process to a degree that the effort required to extract useful information (i.e., execution pattern of victim) becomes comparable to the brute force attack. We introduce the concept of so-called *Noise* process to be run in parallel to spy and victim processes. Noise process itself does not add any computational overhead. Rather, it only compliments the victim process in hiding its operational details from spy process. Noise process is responsible for generating additional memory accesses (called *positive noise* in this case) or additional evictions (called *negative noise* in this case) on selected instructions, i.e., square, multiply, and subtract. Since the spy process extracts execution pattern based on these operations as shown in Figure 1(A), therefore, the noise process uses various combinations of positive and negative noise on different instructions in order to confuse this pattern as shown in Figure 1(B).

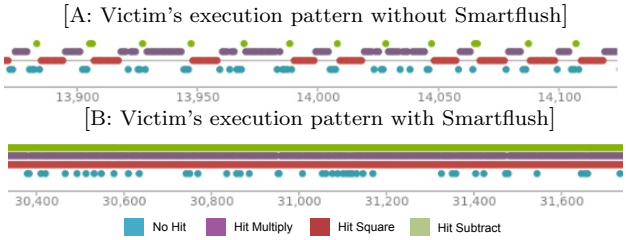

Figure 1: Zoomed view of victim's execution patterns extracted by Flush+Reload with and without Smartflush

The poster will present our experimental results obtained on an Intel Xeon $E5-2643$ processor by generating $10,000$ requests to web-server sequentially and provide a quantitative and qualitative analysis of confidentiality and performance. Our experiments show that Flush+Reload attack cannot extract execution pattern anymore in the presence of Smartflush as the confidentiality is improved substantially with minimum or no performance degradation. Moreover, Smartflush can be used as a *quick patch* countermeasure in user-space without any algorithmic and architectural modifications. Our poster will also present a categorization of countermeasures for cache-based timing SCAs.

2. REFERENCES

[1] Q. Ge, Y. Yarom, D. Cock, and G. Heiser. A survey of microarchitectural timing attacks and countermeasures on contemporary hardware. *J. of Crypto. Engg.*, pp 1–27, 2016.

[2] Y. Yarom and K. Falkner. Flush+reload: A high resolution, low noise, l3 cache side-channel attack. In *Proceedings of the 23rd USENIX Conference on Security Symposium*, pages 719–732, USA, 2014.

WomENcourage, September 6-8, 2017 Barcelona, Spain

DOI: 10.1145/1235