Cache-Based Side-Channel Intrusion Detection using Hardware Performance Counters

Maria Mushtaq, Ayaz Akram, Muhammad Khurram Bhatti, Vianney Lapotre, Guy Gogniat

ABSTRACT

We present a novel run-time detection approach for cache-based side channel attacks (SCAs). It constitutes machine learning models which take real-time data from hardware performance counters for detection purpose. We have performed our experiments with two state-of-the-art cache-based side channel attacks namely, Flush+Reload and Flush+Flush to evaluate the effectiveness of our detection approach. We have provided the experimental evaluation using real time system load conditions and analyzed the results on detection accuracy, detection speed, system-wide performance overhead and confusion matrix for used models. Proposed detection mechanism uses three different machine learning models, namely LDA, LR, SVM model for intrusion detection. We collect data related to the real-time behavior of running processes through selected CPU events, which are stored in registers called Hardware Performance Counters (HPCs). This data is used as features for our machine learning models. The method is designed for run-time detection of cache-based SCAs on RSA and AES crypto-systems. The proposed method uses carefully selected unique hardware events in a multiplexed fashion to reduce false positives and false negatives. We perform experiments on Intel's Core i5 i7 machines under No load, Average load, and Full load system conditions. Our results show detection accuracy of up to 99.51%in the best case for Flush+Reload and 99.97% in the best case for Flush+Flush SCA. Our detection approach shows considerably high detection efficiency under realistic system load conditions.

1. CONTEXT

Side-channel attacks (SCAs) target micro-architectural features which exploit mathematically sound cryptographic implementations like RSA, AES and ECC etc. SCAs observe the pattern of memory accesses and timing of data-dependent cryptographic operations which is a major source of information leakage in attacks. In the last decade, numerous cache-based SCAs have been proposed as well as some mitigation techniques for such attacks too. Some famous techniques of published attacks are Prime+Probe, Flush+Reload, Flush+Flush, Evict+Time and Evict+Reload [1], [2], [3], [4], [5]. Whereas, there are many software and hardware mitigation techniques are proposed in the recent past [6], [7]. Some pragmatic solutions to mitigate SCAs are; disabling hardware threading [8], auditing [9], cache flushing [10], cache coloring and partitioning [11], [12], scheduling-based obfuscation [13], and hardware cache partitioning [14]. These techniques are designed to provide protection against specific vulnerability exploited in information leakage channel. But on the other hand, they provide with an excessive blow in code size, execution time, resource utilization and performance overhead. Since, devising all weather mitigation techniques are hard to propose and applying mitigation techniques in all cases is expensive too. Therefore, sophisticated detection mechanisms can help to analyze the necessity of need-based mitigation.

In the recent past, an emerging area for mitigating cache-based SCAs is detection around malicious activities. Some researches [3], [15], [16], [17], [18], [19], [20] have been done to detect side channels by implementing user-level processes to observe the execution of other processes or by providing light-weight patches in operating systems. These mechanisms do not modify underlying hardware. These detection based techniques focussed on detection by incoming information used by hardware performance counters with and without integrating machine learning approaches. These detection mechanisms are focused on different leakage vulnerabilities of cache and have been validated on different cryptographic implementations.

Hardware Performance Counters (HPCs) are specially designed hardware registers which are utilized mainly for performance monitoring, timing execution information and distinguishing bottlenecks in program's execution. HPCs disclose run-time behavioral information of software which is using specific hardware events e.g. cache references, cache misses, cache hits, branch miss prediction, retired instructions, CPU cycles etc. These events use dedicated hardware, so, they can be accessed very fast without affecting target software.

We propose a detection mechanism using HPCs and different machine learning models; Linear Discriminant Analysis (LDA), Logistic Regression (LR), Support Vector Machine (SVM) [21] to identify the malicious activity at a sophisticated and fine grain level. Figure 1, represents an abstract view of proposed methodology using LDA, LR and SVM models. Our detection approach is based on training the machine learning models, run-time profiling of hardware events and clasification and detection of anomaly.

We create two experimental case studies to demonstrate detection of F+R and F+F attacks. In each case study, we evaluate the performance of our ML models under realistic system load conditions. To do so, we vary the system load from *No Load* (NL), *Average Load* (AL), to *Full Load* (FL) conditions by using selected SPEC benchmarks that offer memory-intensive computations such as; *gobmk*, mcf, $omnetpp$, and *xalancbmk*, to run in the background as both attacks would be targeting/affecting caches and they would

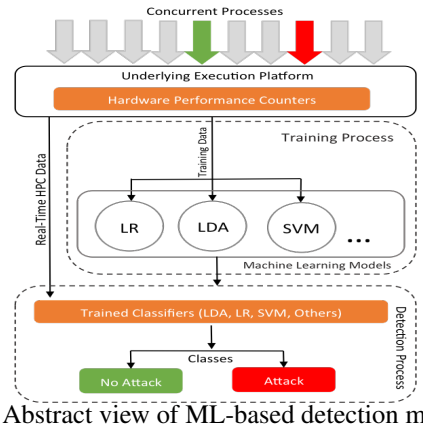

Figure 1: Abstract view of ML-based detection mechanism.

create realistic execution scenarios for evaluation. A NL condition involves only Victim and Attacker processes running, an AL condition involves at least two SPEC benchmarks running along with Victim & Attacker processes, and a FL condition involves at least four SPEC benchmarks running along with Victim & Attacker processes. Proposed detection approach works for access-driven cache-based side-channel attacks on modrn processors and it uses multiple ML models for detection purpose. These models use real-time data from hardware performance counters to determine cache-based intrusions on RSA and AES crypto-systems. We provide detailed analysis of results on detection accuracy, speed, system-wide performance overhead and confusion matrix for used models. One of the strengths of this work is that we provide experimental evaluation using realistic system load conditions. We use standard SPEC benchmarks to create No Load, Average Load, and Full Load conditions to train and evaluate ML models. NIGHTs-WATCH shows considerably high detection efficiency under variable system load conditions.

Our results show detection accuracy of 99.51%, 99.50% and 99.44% for F+R attack in case of NL, AL and FL conditions, respectively, with performance overhead of $< 2\%$ at the highest detection speed, i.e., within 1% completion of a single RSA encryption round. In case of Flush+Flush (stealthier) attack, our results show 99.97%, 98.74% and 95.20% detection accuracy for NL, AL and FL conditions, respectively, with performance overhead of $< 2\%$ at the highest detection speed, i.e., within 12.5% completion of 400 AES encryption rounds needed to complete the attack.

We also experimented with tree-based ML models like Random Forest (RF) in this work, which shows good detection accuracy for both attacks. Although, tree-based models achieve good accuracy, their implementation complexity makes it harder to use them in real-time attack detection mechanisms like our proposed detection technique. Moreover, in order to prove portability, we performed experiments on different hardware such as Intel's core $i3-2120$ CPU running on Linux Ubuntu $4.4.0-116-generic$ at 3.30-MHz. Our results show consistency. Timely detection of intrusion using actual variations in process's execution, measured directly from hardware (particularly caches), can help the operating system to take preventive measures such as halt or killing of identifiable malicious process, completely isolated or critical section-first execution of victim process, or relocation of co-located VMs etc. In future, we intend to integrate more ML models in our detection module and apply it on other cache-based SCAs.

2. REFERENCES

[1] F. Liu, Y. Yarom, Q. Ge, G. Heiser, and R. B. Lee, "Last-level cache side-channel attacks are practical," in *IEEE Symp. on S&P*, pp. 605–622, 2015.

[2] Y. Yarom and K. Falkner, "Flush+reload: A high resolution, low noise, l3 cache side-channel attack," in *USENIX Security 14*, pp. 719–732.

[3] D. Gruss, C. Maurice, K. Wagner, and S. Mangard, "Flush+flush: A fast and stealthy cache attack," in *DIMVA*, pp. 279–299, 2016.

[4] D. A. Osvik, A. Shamir, and E. Tromer, "Cache attacks and countermeasures: The case of aes," *CT-RSA*, pp. 1–20, 2006.

[5] D. Gruss, R. Spreitzer, and S. Mangard, "Cache template attacks: Automating attacks on inclusive last-level caches," in *USENIX*, pp. 897–912, 2015.

[6] Q. Ge, Y. Yarom, D. Cock, and G. Heiser, "A survey of microarchitectural timing attacks and countermeasures on contemporary hardware," *IACR Crypt. ePrint Arch.*, p. 613, 2016.

[7] S. A. et al, "Cross-vm cache-based side channel attacks and proposed prevention mechanisms: A survey," *Journal of Network and Computer Applications*, pp. 259 – 279, 2017.

[8] M. A. et al, "Security best practices for developing windows azure applications," *Microsoft Corp*, p. 1, 2010.

[9] Y. Tan, J. Wei, and W. Guo, "The micro-architectural support countermeasures against the branch prediction analysis attack," in *IEEE TrustCom*, pp. 276–283, 2014.

[10] Y. Zhang and M. K. Reiter, "Düppel: Retrofitting commodity operating systems to mitigate cache side channels in the cloud," in *ACM CCS*, pp. 827–838, 2013.

[11] M. . Godfrey and M. Zulkernine, "Preventing cache-based side-channel attacks in a cloud environment," *IEEE Transactions on Cloud Computing*, vol. 2, pp. 395–408, Oct 2014.

[12] T. K. et al, "Stealthmem: System-level protection against cache-based side channel attacks in the cloud," in *USENIX Security 12*, pp. 11–11.

[13] F. Liu, L. Ren, and H. Bai, "Mitigating cross-vm side channel attack on multiple tenants cloud platform.," *JCP*, pp. 1005–1013, 2014.

[14] Z. Wang and R. B. Lee, "New cache designs for thwarting software cache-based side channel attacks," in *ISCA*, pp. 494–505, 2007.

[15] J. Demme, M. Maycock, J. Schmitz, A. Tang, A. Waksman, S. Sethumadhavan, and S. J. Stolfo, "On the feasibility of online malware detection with performance counters," in *ISCA*, 2013.

[16] A. Tang, S. Sethumadhavan, and S. J. Stolfo, "Unsupervised anomaly based malware detection using hardware features," *CoRR*, 2014.

[17] M. A. et al, "Performance counters to rescue: A machine learning based safeguard against micro-architectural side-channel-attacks." Crypt. ePrint Arch., 2017. `https://eprint.iacr.org/2017/564`.

[18] M. Chiappetta, E. Savas, and C. Yilmaz, "Real time detection of cache-based side-channel attacks using hardware performance counters," *Appl. Soft Comput.*, vol. 49, pp. 1162–1174, Dec. 2016.

[19] G. Torres and C. Liu, "Can data-only exploits be detected at runtime using hardware events?: A case study of the heartbleed vulnerability," in *HASP*, pp. 2:1–2:7, 2016.

[20] T. Zhang, Y. Zhang, and R. B. Lee, "Cloudradar: A real-time side-channel attack detection system in clouds," in *RAID 2016*.

[21] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. 2006.