Machine Learning For Security: The Case of Side-Channel Attack Detection at Run-time

Maria Mushtaq, Ayaz Akram, Muhammad Khurram Bhatti, Maham Chaudhry, Muneeb Yousaf, Umer Farooq, Vianney Lapotre, Guy Gogniat

Machine Learning For Security: The Case of Side-Channel Attack Detection at Run-time

Maria Mushtaq[1], Ayaz Akram[2], Muhammad Khurram Bhatti[3], Maham Chaudhry[3]
Muneeb Yousaf[3], Umer Farooq[4], Vianney Lapotre[1], and Guy Gogniat[1]
[1]Lab-STICC, University of South Brittany (UBS), Lorient, France
[2] University of California, Davis, USA
[3] ECLab, Information Technology University, Lahore, Pakistan
[4] Department of E&C Engineering, Dhofar University, Salalah, Oman

This paper presents experimental evaluation and comparative analysis on the use of various Machine Learning (ML) models for detecting Cache-based Side Channel Attacks (CSCAs) in Intel's x86 architecture. The paper provides performance evaluation of ML models based on run-time detection accuracy, speed, computational overhead, and distribution of error in terms of false positives and false negatives. Experiments are performed using state-of-the-art CSCAs namely; Flush+Reload and Flush+Flush attacks, under realistic load conditions on RSA and AES crypto-systems. The paper provides quantitative & qualitative analysis of at least 12 ML models being used for CSCA detection for the first time.

***Index Terms*—Cache-based Side-Channel Attacks, Cryptography, RSA, AES, Flush+Reload, Flush+Flush, Detection, Machine Learning.**

I. Introduction

SECURITY has already become the first-class design constraint in modern computing systems. Access-driven Cache-based Side-Channel Attacks (CSCAs) are strong cryptanalysis techniques used to break the otherwise strong cryptographic algorithms by targeting cryptographic implementations [1], [2], [3] for unauthorized retrieval of information. In recent years, Intel's x86 architecture has been exposed to high resolution and stealthy CSCAs, such as Flush+Reload [1], Flush+Flush [4], Spectre and Meltdown [5]. Modern-day processors do extensive sharing and de-duplication for performance benefits. CSCAs exploit sharing vulnerabilities in caches [6], [7] to retrieve information. Such attacks rely on the presence of specialized instructions to manoeuvre the state of shared caches.

Both software- and hardware-based protection techniques have been proposed against SCAs [8], [9] in recent years. Such attacks can be prevented at different levels such as system-level, application-level, and hardware-level [8]. At the system level, physical and logical isolation approaches exist [10], [11]. At the application level, the proposed countermeasures tend to target the source of information leakage and mitigate it [12]. At the hardware level, mitigation techniques are rather difficult as they are not applicable on commodity systems. Hardware solutions, nevertheless, suggest to have new secure caches, changes in prefetching policies and either randomization or complete removal of cache interference [13].

Despite diverse efforts, protection techniques against SCAs are not perfect. These techniques generally protect against any given specific vulnerability and an *all-weather* protection against such attacks is often performance costly. Evidence suggest that attacks are becoming sophisticated and stealthier such as Spectre & Meltdown. Therefore, detection techniques can be used as a *first line of defense* against such attacks. For detection-based prevention strategy to be effective, detection need to be highly accurate, should incur minimum system overhead at run-time, and be capable of early-stage detection, i.e., before the attack completes. Thus, the evaluation metrics for detection techniques is tough.

Machine learning techniques have proved their effectiveness across many application domains. Recently, various ML techniques are also being used in information security domain such as in [14], [15], [16], [17], [18], [19]. In this work, we have analyzed the effectiveness of various ML models in cache-based SCA detection using the aforementioned evaluation metrics. We provide experiments & results using state-of-the-art attack techniques such as Flush+Reload & Flush+Flush, on Intel's x86 architecture under RSA & AES encryption algorithms.

Under the constraints such as; real-time requirements, early-stage detection, and minimal performance overhead, the domain of CSCA detection becomes particularly challenging and interesting application domain for Machine Learning. This work provides various selection metrics for ML models, supported by our extensive experimental results, under these constraints. Specifically, the contributions of this paper are as following:

1) We demonstrate successful detection of state-of-the-art cache-based SCAs, Flush+Reload & Flush+Flush attacks on RSA and AES crypto-systems, respectively.
2) We provide experimental evaluation and comparative analysis on the use of various Machine Learning (ML) models for detecting CSCAs on Intels x86 architecture.
3) We provide selection metrics for machine learning models to perform run-time CSCA detection under real-time, high-speed, and minimum overhead constraints.

Rest of the paper is organized as follows. Section II presents necessary background on detection with the help of machine learning models, selection of hardware events which can be

Manuscript received June 30, 2018; revised Date. Corresponding author: M. Mushtaq (email:maria.mushtaq@univ-ubs.fr).

used as useful features and selection of machine learning models for experimental evaluation of detection accuracy. Section III presents that how machine learning models can be helpful to provide security and detect certain CSCAs. Section IV concludes this paper.

II. Background & Related Work

This section provides the constitutional background on detection of cache-based side channel attacks, hardware performance counters, and machine learning models.

A. Machine Learning for Detection

There is very limited research work that focuses on detecting CSCAs implemented on user-level and kernel-level processes. Authors in [15] use machine learning techniques to detect cache-based side channels. Neural networks have been used in this detection mechanism to construct models by having values of HPCs which encounter benign and spy process during detection of stealth F+R attack. [20] proposed a malware detector for side channel attacks. It uses a case study on P+P attacks and utilizes values of HPCs for training of machine learning models such as; K-Nearest Neighbor (KNN) [21], Decision Trees [22], Random Forest and Aritficial Neural Netwroks (ANN) [23]. [16] proposed a technique *HexPADS* which does not involve machine learning and determines a threshold for a certain number of cache misses to detect an attack. They validated F+R and P+P attacks but the authenticity of determining such thresholds with no sophisticated learning is questionable. This malware detection mechanism detects P+P without false positives. [17] trained three machine learning models namely; Neural Networks [23], Decision Trees [22] and K-Nearest Neighbor (KNN) [21] to detect AES cryptosystem. [24] also proposed a threshold-based detection mechanism named *CloudRadar*. It detects side channels in cloud systems by relating crypto-systems executing on virtual machines. *CloudRadar* considers that it performs detection when victim is running crypto-application (already known). It monitors untrusted VMs by sampling performance events at regular intervals and if the difference crosses a certain threshold, attack is detected. This mechanism also detected P+P and F+R and shows 100% accuracy. But in real systems its hard to claim when victim is running crypto-application and taking into account performance events on a certain threshold may have less precision. [25] relies on Intel cache monitoring technology (CMT) and HPCs. It uses gaussian anomaly detection for cache-based side channel attacks on VMS. The proposed mechanism shows very good accuracy in isolated conditions but suffer from high false positives in noisy conditions/realistic load conditions. [18] proposed three step detection mechanism for cache and branch predictor based side channels (detection of anomaly, class of anomaly and correlation of malicious process with victim). Authors in [14] present a novel technique named as NIGHTs-WATCH to detect cache-based side channel attacks. It comprises of different machine learning models, which use real-time data from hardware performance counters for detection at run-time. It detects two state-of-the-art attacks namely; F+R and F+F. To be noted F+F being a stealth and high frequency attack, has never been detected in the past researches. This is the only research published with detection of F+F attack. Both the attacks are detectable on run-time with realistic load conditions and detection accuracy of 99.51% (F+R) and 99.97% (F+F). Performance overhead of NIGHTs-WATCH with highest detection speed is $< 2\%$. This paper is the extension of NIGHTs-WATCH, providing with comparative analysis of machine learning models which perform best to detect side channels.

B. Selected Hardware Events as Useful Features

Almost all modern processors support hardware performance counters. Their type and count, however, can vary for different processor families. PAPI library provides access to $100+$ events for Intel's core $i7$ family. Such events are helpful to access per-core, per-CPU and system wide profiling for executing processes. Since our interest is to detect access-driven cache-based side channel attacks, we consider only the events which are plausible to be affected by these attacks. In order to best select these events, we performed experiments on a set of 12 best suited events as listed in Table I. We collected a system-wide profile for different hardware events under state-of-the-art CSCAs, namely the F+R attack on RSA encryption algorithm as shown in Figure 1 and F+F attack on AES encryption algorithm as shown in Figure 2, to form a data set of benign and malicious behavior on the system. Shown sample events are collected under no load conditions, i.e., when no process other than encryption and attack processes is executing.

TABLE I: Selected events related to cache-based SCAs

Scope of Event	Hardware Event as Feature	Feature ID
L1 Caches	Data Cache Misses	L1-DCM
	Instruction Cache Misses	L1-ICM
	Total Cache Misses	L1-TCM
L2 Caches	Instruction Cache Accesses	L2-ICA
	Instruction Cache Misses	L2-ICM
	Total Cache Accesses	L2-TCA
	Total Cache Misses	L2-TCM
L3-Caches	Instruction Cache Accesses	L3-ICA
	Total Cache Accesses	L3-TCA
	Total Cache Misses	L3-TCM
System-wide	Total CPU Cycles	TOT_CYC
	Branch Miss-Predictions	BR_MSP

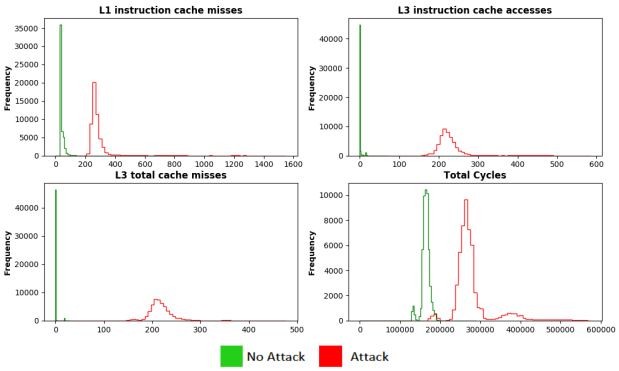

Fig. 1: Selected hardware events under Zero Load conditions for RSA encryption algorithm: With & Without F+R Attack

C. Selected Machine Learning Models

Table II enlists machine learning models being used in experimental evaluation in this work. We have experimented with a total of 12 models, 6 each from linear & non-linear category. In favor of space, we do not provide details. on models. However, the theoretical review of these ML models can be found in other sources such as [26], [27], [28].

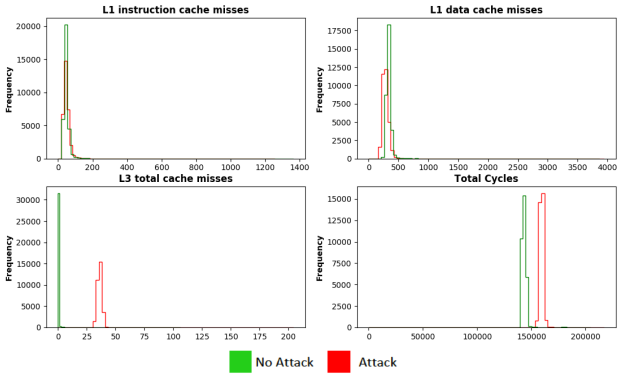

Fig. 2: Selected hardware events under Zero Load conditions for AES encryption algorithm: With & Without F+F Attack

TABLE II: List of Machine Learning Models for CSCA Detection (Non-exhaustive)

No.	Machine Learning Model	Category
1	Linear Regression (LR)	Linear
2	Linear Discriminant Analysis (LDA)	Linear
3	Support Vector Machine (SVM)	Linear
4	Quadratic Discriminant Analysis (QDA)	Non-linear
5	Random Forest (RF)	Non-linear
6	K-Nearest Neighbors (KNN)	Non-linear
7	Nearest Centroid	Linear
8	Naive Bayes	Linear
9	Perceptron	Linear
10	Decision Tree	Non-linear
11	Dummy	Non-linear
12	Neural Networks	Non-linear

III. Machine Learning for Security

Machine learning models are mathematical functions that tend to find such patterns from the data, which can best explain the input output relationship. Classification of the data is an important application of Machine Learning. CSCA detection is basically one such binary classification problem from machine learning prospective. In this section, we present a comparative analysis of various ML models used for detection of CSCAs in Intel's x86 architecture. Most of these models are used for the first time in CSCA detection in this work. This analysis is based on experimental data collected at the hardware level using Hardware Performance Counters (HPCs). The data provides real-time behavioral information of concurrent processes running on Intel's x86 architecture such as; Instruction and data cache misses & hits for L1/L2/LLC, total CPU cycles, branch miss-predictions, and total cache accesses etc. We create two experimental case studies for our analysis of F+R and F+F attacks. In each case study, we evaluate the performance of ML models under realistic system load conditions. To do so, we vary the system load from *Zero Load* (ZL), *Medium Load* (ML), to *Heavy Load* (HL) conditions by using selected SPEC benchmarks [29] that offer memory-intensive computations such as; $gobmk$, mcf, $omnetpp$, and $xalancbmk$, to run in the background. A ZL condition involves only Victim and Attacker processes running, a ML condition involves at least two SPEC benchmarks running along with Victim & Attacker processes, and a HL condition involves at least four SPEC benchmarks running along with Victim & Attacker processes. We have performed experiments on Intel's core $i7-4770$ CPU running on Linux Ubuntu $16.04.1$ with kernel $4.13.0-37$ at 3.40-GHz.

Most of the existing machine learning classifiers can be divided into two basic categories of linear and non-linear models. We experimented with a set of 12 popular classification machine learning models, 6 each from both categories. A list of ML models being used in this work is provided in Section II-C. Here we characterize and compare these machine learning models to solve the classification problem at hand i.e. detection of CSCA. We describe the rationale and criteria behind selection of any particular models for run-time CSCA detection through experimentation. Run-time detection constraints have been discussed in Section I. We keep the following constraints to assess selected ML models: classification accuracy, implementation feasibility for run-time detection (in terms of performance overhead), distribution of error (false positives & false negatives), and detection speed.

The classification accuracy of the studied machine learning models is shown in Figure 3 and 4 for Flush+Reload (on RSA) and Flush+Flush (on AES) attacks, respectively, under Zero-Load (ZL), Medium-Load (ML) and Heavy-Load (HL) conditions. As shown in Figure 3, for detection of Flush+Reload attack most of the machine learning models show pretty good accuracy (close to 100% for all load conditions) except Nearest-Centroid and Dummy Classifiers. However, for detection of Flush+Flush attack, more number of classifiers exhibit low accuracy (like Dummy, Perceptron, Neural Network). Moreover, under Heavy-Load (HL) condition, the classification accuracy of all machine learning models degrades. The degradation of performance of classifiers for Flush+Flush attack can be explained by the stealth nature of the attack as discussed before. Unlike F+R, for F+F attack Nearest-Centroid manifests good classification accuracy. Since, any detection mechanism should only employ those machine learning models which show acceptable accuracy for all types of attacks the detection framework is supposed to work for, we argue that the models that can be used for detection of CSCAs out of the studied ones based on their accuracy include: LDA, LR, SVM, QDA, Naive-Bayes, KNN, Decision Tree, Random Forest and QDA.

Though the most important one, but classification accuracy is not the only parameter to consider while deploying a high-speed run-time CSCA detection mechanism. We reason that the other most important parameter to examine while comparing machine learning models is the implementation feasibility. Machine learning models should be easy to embed in the crypto-system to be protected at run-time. They should also be able to quickly provide their decision of CSCA detection based on the profiling of processes using HPCs. Moreover, the performance overhead caused to the crypto-system's own execution due to embedding of classification patch should be reasonable. On analyzing the short-listed machine learning models on the basis of accuracy, we find that two of those models i.e. Decision Tree and Random Forest would not be easy to implement at run-time due to their tree-based nature. We observed that, based on our experiments, the decision trees/random forest that show good accuracy for our classification problem, also have high depth and a high number of branches. Not only that it

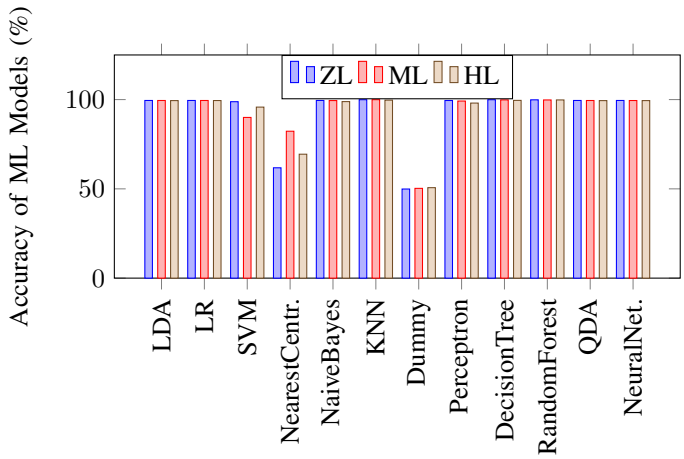

Fig. 3: Accuracy Comparison of ML Models for F+R (RSA)

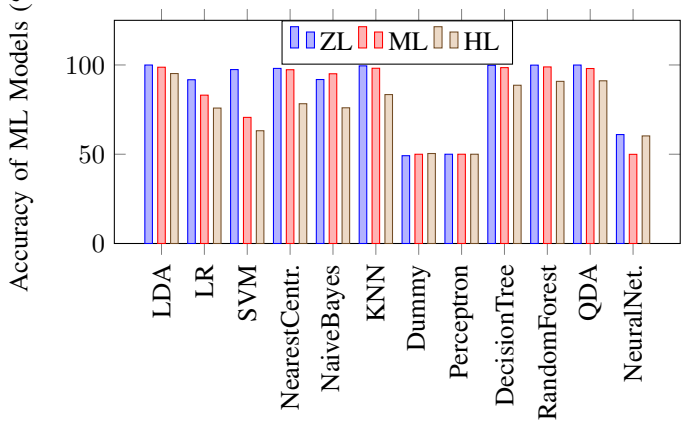

Fig. 4: Accuracy Comparison of ML Models for F+F (AES)

makes their embedding into crypto-system difficult, it can also result into high performance overhead due to a high number of $if-else$ blocks needed to implement them. KNN also shows good classification accuracy for both attacks. However, KNN uses all training data points at run-time to infer a classification decision, which can result into high performance and storage overhead. Since better training would require more data points, KNN leads to a performance vs accuracy trade-off. Out of the remaining machine learning models, since QDA (non-linear) and LDA (linear) are based on Naive Bayes, Naive Bayes can be left out in favor of LDA and QDA. This leaves us with LDA, LR, SVM and QDA as better candidates for use in detection of CSCAs. Our experiments indicate that these four machine learning models show less than 2% performance overhead for detection of Flush+Relaod and FLush+Flush attacks, while capable of detecting way before thresholds of security degradation with with an accuracy of more than 90% under heavy load conditions.

IV. Conclusion

Machine learning has proved its effectiveness across many application domains, including information security. In this work, we have provided quantitative & qualitative analysis of various machine learning models while being used in detection of the state-of-the-art cache-based side-channel attacks, Flush+Reload & Flush+Flush, on RSA and AES crypto-systems, respectively. We have analyzed their effectiveness under real-time detection constraints and compared the results for classification accuracy, run-time detection speed, classification overhead, and distribution of error (false positives & negatives). This paper does not conclude in favor of any particular models. Rather, it provides contextualized evidence on the use of these models in CSCA detection. A different and more efficient implementation of these models might result into different results altogether.

References

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

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

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

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

[5] "Spectre & meltdown attacks," 2018.

[6] P. K. et al, "Spectre attacks: Exploiting speculative execution," 2018.

[7] M. Lipp, M. Schwarz, D. Gruss, T. Prescher, W. Haas, S. Mangard, P. Kocher, D. Genkin, Y. Yarom, and M. Hamburg, "Meltdown,"

[8] 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.

[9] 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.

[10] X. Jin, H. Chen, X. Wang, Z. Wang, X. Wen, Y. Luo, and X. Li, "A simple cache partitioning approach in a virtualized environment," in *IEEE ISPA*, p. 519, Aug 2009.

[11] H. Raj, R. Nathuji, A. Singh, and P. England, "Resource management for isolation enhanced cloud services," in *CCSW*, pp. 77–84, 2009.

[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 and R. B. Lee, "Random fill cache architecture," in *MICRO*.

[14] M. Mushtaq, A. Akram, M. K. Bhatti, M. Chaudhry, V. Lapotre, and G. Gogniat, "Nights-watch: A cache-based side-channel intrusion detector using hardware performance counters," in *Proceedings of the 7th International Workshop on Hardware and Architectural Support for Security and Privacy*, HASP '18, (NY, USA), pp. 1:1–1:8, ACM, 2018.

[15] 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.

[16] M. Payer, "Hexpads: a platform to detect stealth attacks," in *International Symposium on Engineering Secure Software and Systems*, pp. 138–154, Springer, 2016.

[17] Z. Allaf, M. Adda, and A. Gegov, "A comparison study on flush+reload and prime+probe attacks on aes using machine learning approachess," *UK Workshop on Computational Intelligence*, pp. 203–213, 2017.

[18] 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.

[19] Z. Allaf, M. Adda, and A. Gegov, "Confmvm: A hardware-assisted model to confine malicious vms," in *UKSim2018: UKSim-AMSS 20th International Conference on Modelling & Simulation*, IEEE, 2018.

[20] J. Demme, M. Maycock, J. Schmitz, A. Tang, A. Waksman, S. Sethumadhavan, and S. Stolfo, "On the feasibility of online malware detection with performance counters," in *ACM SIGARCH Computer Architecture News*, vol. 41, pp. 559–570, ACM, 2013.

[21] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE transactions on information theory*, vol. 13, no. 1, pp. 21–27, 1967.

[22] J. R. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.

[23] R. Lippmann, "An introduction to computing with neural nets," *IEEE Assp magazine*, vol. 4, no. 2, pp. 4–22, 1987.

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

[25] M.-M. Bazm, T. Sautereau, M. Lacoste, M. Sudholt, and J.-M. Menaud, "Cache-based side-channel attacks detection through intel cache monitoring technology and hardware performance counters," in *Fog and Mobile Edge Computing (FMEC), 2018 Third International Conference on*, pp. 7–12, IEEE, 2018.

[26] M. B. Christopher, *PATTERN RECOGNITION AND MACHINE LEARNING*. Springer-Verlag New York, 2016.

[27] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*, vol. 1. Springer series in statistics New York, 2001.

[28] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An introduction to statistical learning*, vol. 112. Springer, 2013.

[29] "https://www.spec.org/benchmarks.html," 2018.