Phased-Guard: Multi-Phase Machine Learning Framework for Detection and Identification of Zero-Day Microarchitectural Side-Channel Attacks

Han Wang[1], Hossein Sayadi[2], Gaurav Kolhe[1], Avesta Sasan[3], Setareh Rafatirad[3], and Houman Homayoun[1]
[1]University of California, Davis, CA, USA
[2]California State University, Long Beach, CA, USA
[3]George Mason University, Fairfax, VA, USA
[1]{hjlwang, gskolhe, hhomayoun}@ucdavis.edu, [2]{hossein.sayadi}@csulb.edu,[3]{asasan, srafatir}@gmu.edu

***Abstract*—Microarchitectural Side-Channel Attacks (SCAs) have emerged recently to compromise the security of computer systems by exploiting the existing processors' hardware vulnerabilities. In order to detect such attacks, prior studies have proposed the deployment of low-level features captured from built-in Hardware Performance Counter (HPC) registers in modern microprocessors to implement accurate Machine Learning (ML)-based SCAs detectors. Though effective, such attack detection techniques have mainly focused on binary classification models offering limited insights on identifying the type of attacks. In addition, while existing SCAs detectors required prior knowledge of attacks applications to detect the pattern of side-channel attacks using a variety of microarchitectural features, detecting *unknown (zero-day)* SCAs at run-time using the available HPCs remains a major challenge. In response, in this work we first identify the most important HPC features for SCA detection using an effective feature reduction method. Next, we propose *Phased-Guard*, a two-level machine learning-based framework to accurately detect and classify both known and unknown attacks at run-time using the most prominent low-level features. In the first level (SCA Detection), *Phased-Guard* using a binary classification model detects the existence of SCAs on the target system by determining the critical scenarios including system under attack and system under no attack. In the second level (SCA Identification) to further enhance the security against side-channel attacks, *Phased-Guard* deploys a multiclass classification model to identify the type of SCA applications. The experimental results indicate that *Phased-Guard* by monitoring only the victim applications' microarchitectural HPCs data, achieves up to 98% attack detection accuracy and 99.5% SCA identification accuracy significantly outperforming the state-of-the-art solutions by up to 82% in zero-day attack detection at the cost of only 4% performance overhead for monitoring.**

***Index Terms*—Side-Channel Attacks, Machine Learning, Zero-Day Attacks, Detection, Identification**

I. Introduction

Cache hierarchy, memory de-duplication, branch prediction unit, and speculative execution are a few of the key strategies proposed to enhance the performance of computer systems in the past few decades. Even though such strategies have been beneficial in reducing the main memory usage and latency of accessing the data and improving performance, they have also become the major source of vulnerabilities. Recently, these hardware vulnerabilities have been exploited by the microarchitectural Side-Channel Attacks (SCAs) in conjunction with the software-based methods to infer sensitive information about the victim's data or code blocks by monitoring the shared on-chip cache memory state at different levels of the memory hierarchy [3], [26], [29], [34]. Compared to malware or software-based attacks, such attacks can pose a significant security threat on a broader range of systems and are more difficult to eliminate due to the invisible behavior and passive nature of SCAs [34]. The recent works on microarchitectural SCAs [9], [12], [16], [19], [27], [32] have outlined various potential threats that have been crafted to leak important information from the system. Thus, there is an emerging need to address the security threats posed by such attacks in modern computer systems.

Prior studies on detecting microarchitectural side-channel attacks leverage the Machine Learning (ML) algorithms on the low-level microarchitectural data captured by Hardware Performance Counter (HPCs) registers to detect the known SCAs [3], [6], [21], [28], [30], [34]. The ML-based SCAs detectors have demonstrated promising results in determining the side-channel attacks with lower latency ranging from several milliseconds to seconds. For instance, the work in [6] proposes an HPC-based monitoring model to detect the SCAs by using the performance counters events collected by running both victim and SCA applications. The mechanism deploys the correlation metric between the events of victims' and attacks' HPC traces. Another work proposed in [3] profiles the victim's application under two different scenarios, i.e, applications with no attacks, and applications with attacks. Although this detection technique helps to mitigate various SCAs, it can not detect the unknown SCAs.

There has been much progress in terms of securing the processors against various microarchitectural SCAs in last few years. However, there are still two major challenges involved with contemporary SCA detectors. First, the fast-paced development of the emerging intricate SCAs to circumvent the current detection techniques has not been properly addressed. Hence, the existing techniques are not able to detect and identify the unknown (zero-day) attacks. Second, the existing side-channel attacks detectors have limited their study on detecting whether the running application is "under attack" condition or not. In particular, such detectors provides no

2576-6996/20/$31.00 ©2020 IEEE
DOI 10.1109/ICCD50377.2020.00111

TABLE I. COMPARISON OF THE RECENT WORK ON MICROARCHITECTURAL SCA ATTACK DETECTION

Detection Work	Dynamic vs Static Analysis	Victim-based only vs Victim+SCAs	Known Attack	Zero-day Attack	Multi-Class Classification
SCADET [23]	Static Analysis	SCAs	✓	✗	✗
MASCAT [15]	Static Analysis	SCAs	✓	✗	✗
Gorka.et al [11]	Static Analysis	SCAs	✓	✗	✗
CloudRadar [34]	Dynamic Analysis	Victim+SCAs	✓	✗	✗
Chiappetta.et al [6]	Dynamic Analysis	Victim+SCAs	✓	✗	✗
Nights-watch [21]	Dynamic Analysis	Victim-based only	✓	✗	✗
★ *Phased-Guard*	Dynamic Analysis	Victim-based only	✓	✓	✓

information on which type of vulnerability exploited by the attack applications. Knowing the type of SCAs ahead of time could facilitate crafting an effective mitigation technique to alleviate the influence of side-channel attacks on the performance of target system.

To address the limitations of the existing SCA detectors, this work we propose *Phased-Guard*, a multi-phase machine learning framework to accurately detect and identify both known and unknown attacks at run-time using the most prominent microarchitectural features. To this aim, we first preprocess the collected microarchitectural data with Dynamic Time Warping (DTW) time series algorithm to calculate the similarities among two HPC-based traces. The first trace is obtained by profiling the victim applications during training phase, and the second trace is obtained during testing phase. In the first phase, *Phased-Guard* with the aid of binary classification models accurately detects the existence of SCAs on the target system. Next, in the second phase, *Phased-Guard* is specialized using a multiclass classification model to identify the type of side-channel attack applications (e.g. Flush+Flush, Flush+Reload, etc.). The prediction results of the identification phase could be utilized by the administrators to take certain safeguard measures and further for devising or deploying effective mitigation strategies. To the best of our knowledge, the is the first work that proposes a machine learning-based approach for zero-day aware side-channel attack detection and identification relying on processors' microarchitectual features. A detailed comparison of *Phased-Guard* with prior microarchitectural side-channel attack detection techniques are listed in Table I. The main contributions of our proposed work are summarized below:

- To facilitate an efficient SCAs detection by choosing an optimal set of available HPCs, an effective feature reduction method is deployed to determine the most prominent microarchitectural events.
- We propose *Phased-Guard*, a machine learning-based microarchitectural SCAs detection and identification framework which comprised of a binary classifier in the first level for distinguishing between known and unknown SCAs existence or benign application followed by a multiclass classifier that predicts the microarchitectural side-channel attack type.
- Unlike prior works, *Phased-Guard* accurately detects SCAs based on differentiating HPCs data of only the victim applications under two conditions including victim under attack and victim under no attack. Using the victim's data makes the proposed approach more robust against the crafted SCAs.

II. BACKGROUND AND RELEVANT WORKS

A. Microarchitectural Side-channel Attacks

The emergence of different hardware components such as cache memory, branch predictor, etc. to enhance the performance of the modern microprocessors, have led to exposure of new hardware vulnerabilities in the systems. This makes a unique opportunity for the attackers to deduce sensitive information by exploiting such vulnerabilities.

1) Flush+Reload: The researches in [5], [31] exploits the vulnerability of page de-duplication technique by monitoring the memory access lines in the shared pages. This attack targets the Last-Level Cache in the CPU and flushes out victim applications' data in the cache and waits for the victim application to execute. After flushing the cache, the attacker tries to access the data and measures the accessing time (latency). Shorter accessing time denotes that the victim application has accessed the data; otherwise, it has not been accessed.

2) Flush+Flush: This SCA relies on the execution time of the flush instruction [12]. Unlike prior attacks, Flush-Flush does not make any memory accesses, nor it relies on the access latency of the data. The execution time of flush instruction depends on whether the data is stored in the cache. Flush-Flush uses the execution time of the subsequent flush instruction following the victim application's execution. The large execution time of the flush instruction is indicative of the fact that, the corresponding data was brought to the cache and later accessed by the victim application.

3) Prime+Probe: Without the memory de-duplication restriction, Prime+Probe [18] could be applied to more systems. This type of SCA consists of two different stages: Prime and Probe. In the Prime stage, the attacker builds the eviction sets which are group cache sets causing potential conflict with victim applications and then fills the cache with the eviction sets. Next, the attacker waits for the execution of the victim application and then re-accesses the eviction sets (Probe stage). If the accessing time is long enough, it means the victim application has accessed the data; otherwise, the victim application does not access it.

B. Dynamic Time Warping (DTW)

The idea of Dynamic Time Warping (DTW) was first introduced in classification and time series mining problems by Berndt and Clifford [2], where dynamic programming was used to align sequences with different lengths. DTW is used to find an optimal match between two sequences by allowing a nonlinear mapping of one sequence to another to minimize the distance between two sequences. Considering two time

series of A (a_1, a_2,..., a_n) and B (b_1, b_2,..., b_m), DTW finds an optimal warping path between A and B by using dynamic programming to calculate the minimal cumulative distance γ (n, m), where γ (i, j) is defined below:

$$\gamma(i,j) = (a_i, b_j)^2 + \min \begin{cases} \gamma(i-1,j) \\ \gamma(i-1,j-1) \\ \gamma(i,j-1) \end{cases} \quad (1)$$

This allows the DTW to practically match similar shape sequences together even though the sequences may be shifted out of the phase. This qualification has made DTW viable for multitude of time series classification problems.

C. Relevant Works on Side-Channel Attacks Detection

To cope with the challenge of microarchitectural SCAs, prior studies have mainly considered two major categories of developing SCAs detector: static code analysis and dynamic analysis based detection techniques. For static analysis based detectors, SCADET [23] proposes analyzing the executable binary file and detect if there is any potential risk for Prime Probe attacks. It shows 100% attacks detection accuracy with 7.4% false positive rate. However, it only evaluates with Prime Probe and analysis time was at the granularity of minute. CaSym [4] first gets a program from a compiler IR and performs symbolic execution. Then it monitors the programs and cache status. The cache status is used to build a formula that is fed to SMT solver to analyze whether side-channels exist. It can help to identify whether cryptographic applications have side-channel vulnerabilities and then apply preloading to mitigate such vulnerabilities.

The work in [6] proposes monitoring HPCs of both victims and attacks applications and then training a machine learning based detector with the HPCs data. Similarly, in [34] the authors present CloudRadar which aims at detecting cross-VM side-channel attacks by analyzing HPC patterns. The research in [22] proposes a detection system containing one analytic server and one or more monitored computing devices to detect SCAs including Spectre and Meltdown. The work in [17] proposes an online detection of Spectre by monitoring microarchitectural features using time series classification.

Overall, the prior works on microarchitectural SCAs detection have mainly ignored the challenges associated with detection of zero-day attacks to address the growth of newly developed SCAs. In addition, the existing side-channel attacks detectors have limited their study on detecting whether the running application is an SCA. These works also could be bypassed when SCAs manipulate HPCs values according to HPC monitoring challenges discussed in recent work [8].

III. Motivations

A. Detecting SCAs based on Victim Applications Data

Prior studies on SCA detection have mostly focused on profiling both the victim and attack applications to collect the HPC data. This data is used for detecting whether the victim application is under attack or not [1], [6], [7], [21], [34]. Recent microarchitectural SCAs [12], [18], [32] mainly target access pattern of victim applications' cache or branch predictor by flushing/priming cache, mistraining branch predictors, and then observe accessing time of the cache sets, which changes caching victims' data and microarchitectural behaviors of victim applications [33]. This provides the opportunity of detecting SCAs by observing the alteration in microarchitectural behaviors. Furthermore, our experimental results as shown in Figure 1 indicate a clear difference between the behavior of victim under attack and victim under no attack conditions. In this case study, the HPC traces of L1 HIT feature for the tested victim application (RSA) under no attack (RSA) and under L3 Flush Reload attack (RSA with FR) are illustrated. As seen, the L1 HIT of victim under attack shows a significantly different trend compared to that of victim under no attack. This observation clearly highlights the effectiveness of using HPCs data of only victim applications (excluding the impact of attack applications' HPCs) for detecting the behavior of SCAs.

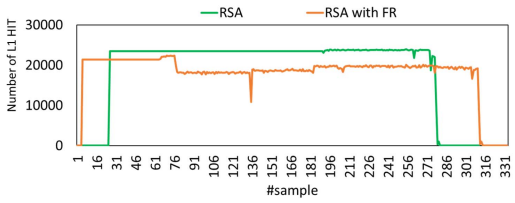


Fig. 1. L1 HIT of RSA and RSA under Flush Reload attack

B. The Necessity of SCAs Type Identification

As mentioned before, existing microarchitectural side-channel attacks detectors have mainly focused on conducting binary classification for detecting SCAs [6], [21], [34]. However, the majority of mitigation strategies could only alleviate one type of SCAs [10]. For instance, obsoleting flush instruction could only protect the system against Flush+Reload and Flush+Flush attack [32]. Given the limits of current SCAs mitigation methods, there is an urgent need for identifying the type of attacks compromising the security of the computer system. As a result, a customized SCAs detector with a multi-class classification model is required to identify the class of SCAs assisting the administrator to take required security measures to mitigate the attacks.

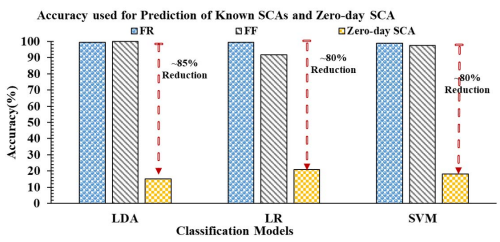


Fig. 2. Detection accuracy of Mushtaq et.al [21] for FR, FF and zero-day SCA (Prime+Probe) across different ML classifiers (LDA: Linear Discriminant Analysis, LR: Logistic Regression (LR), and SVM: Support Vector Machine)

C. Challenge of Detecting Zero-Day Microarchitectural SCAs

While the prior studies on SCAs detection using HPCs' sum values achieve high detection accuracy, they lack the capability for zero-day (unknown) side-channel attacks which are more dangerous SCAs as the attack signature does not exist in the training dataset of ML-based detectors. In order to highlight this point, in Figure 2 we evaluated the detection accuracy results for a recent work Mushtaq et al. [21] by deploying different ML models including J48, OneR, MLP, and BayesNet to detect both known and unknown SCAs including FR (Flush+Reload), FF (Flush+Flush), and zero-day (Prime+Probe). It can be observed that Mushtaq et al. [21] achieves above 95% detection accuracy for detecting FR and FF attacks across different ML classifiers. Whereas, the accuracy of the detector significantly drops by at least 80% for the zero-day SCA. This observation clearly indicates that such methods are not able to protect the target computer systems against dangerous zero-day SCAs, urging the importance of a novel detection methodology to address the challenge.

IV. Phased-Guard Framework Design

In this section, we present the details of the proposed ML-based micorarchitectural side-channel attacks detection and identification framework. Figure 3 illustrates an overview of *Phased-Guard* that is classified into three major components: 1) HPCs data collection and feature representation, 2) Phase I: binary classification for SCAs detection ("under attack" or "under no attack"), and 3) Phase II: multi-class classification for SCAs identification.

A. Data Collection and Experimental Setup

In this work, all experiments are conducted on an Intel I5-3470 desktop with four cores, 8GB DRAM, and a three-level cache system. The operating system is Ubuntu 16.0.4 LST with Linux kernel 4.13. Victim applications and side-channel attacks are selected from Mastik [31]. Furthermore, MiBench [13] benchmark suite is used to represent benign applications. There are three attack conditions for each victim in our data collection process, i.e, RSA under Flush+Reload, RSA under Flush+Flush, and RSA under Prime+Probe conditions. Each pair of a victim under attack and victim under no attack conditions executes 100 times resulting in 4590 records of RSA under no attack samples, and 10000 records for RSA under each of the three SCAs. We further employ a PAPI-based customized tool [20] to collect hardware performance counters based on Model-Specific Registers (MSRs). This performance monitoring tool collects HPCs per processor at microsecond scale with privileged access. Based on the behavior and functionality of studied SCAs, 16 HPC features are considered in this work for further analysis, as listed in Table II. These hardware performance counters data are collected using the four available HPC registers in the experimented I5 processor at every 500 microseconds. A standard 70%-30% non-biased percentage split is used for training and testing the ML binary classification models. Furthermore, in order to model the zero-day SCAs detection, the ML models will also be tested with data from the unknown attack.

TABLE II. The collected HPC features and their ranking

Ranking	HPC Name	Ranking	HPC Name
1	L3 MISSES	9	BRANCHES_MISPREDICTED
2	L2_HIT	10	ITLB_MISSES
3	L1_MISS	11	DTLB_LOAD_MISSES
4	L1 HIT	12	L3 HIT
5	INST_RETIRED.ANY	13	.ALL_BRANCHES_RETIRED
6	UOPS_RETIRED.ALL	14	DTLB_STORE_MISSES
7	L2_MISS	15	BR_TAKEN_CONDITIONAL
8	BR_NONTAKEN_CONDITIONAL	16	TAKEN_INDIRECT_NEAR_CALL

B. Threat Model

Phased-Guard is designed to secure multi-core computing processors against microarchitectural SCAs that employ inclusive and non-inclusive caches on the Intel architectures in which malicious and benign processes use shared libraries. In addition, *Phased-Guard* considers that the applications are executing in a Linux-based single OS environment where both victims and attacks reside in the physical machine, on different processing cores. The system can face with different types of microarchitectural SCAs including shared-memory based SCAs such as Flush+Flush, Flush+Reload, and none shared-memory based SCAs such as Prime+Probe. In order to create the known and unknown threat conditions, two out of the three considered SCAs are used as known attacks and the third attack is used for modeling unknown attacks. As a result, the known attacks can be profiled and corresponding run-time HPCs information is stored in the database for training the ML models. Next, the unknown attack features are deployed to examine the effectiveness of ML-based detectors built in *Phased-Guard* for detecting zero-day SCAs.

C. Similarity Calculation using DTW Method

We leverage the similarity metric between different temporal HPC sequences to train the ML classification model in *Phased-Guard*. Despite previous approaches on microarchitectural SCAs detection that have used the raw HPC values to train the classification models with only known SCAs detection ability [6], [21], [34], *Phased-Guard* captures similarities between HPC's sequences, and provides high detection accuracy for both known and unknown SCAs. As for the similarity metric between the two HPC temporal sequences, we use the Euclidean distance between the two sequences. Furthermore, as demonstrated in Figure 4 the dynamic time warping method is employed to calculate the similarities among victim under no attack HPC sequences and victim under attack HPC sequences. For following sections, the two distances are calculated when victim application are under no attack and attack are abbreviated as VNAD and VAD. We repeat the DTW similarity calculation process for all the 16 HPC features to prepare the training dataset. Next, during the testing process the DTW is only conducted for the most prominent HPC features that are identified by the feature selection process described in below.

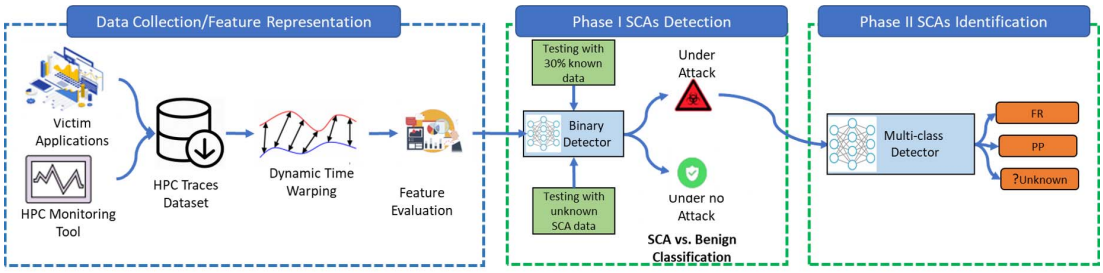


Fig. 3. Overview of *Phased-Guard*, the proposed multi-phase machine learning framework for detection and identification of zero-day microarchitectual SCAs

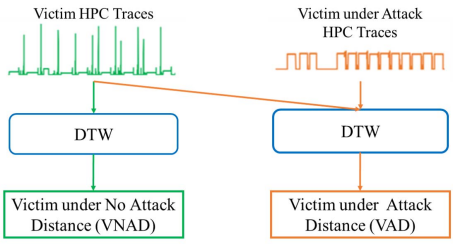


Fig. 4. Similarity calculation using dynamic time warping based on victim HPCs data sequences in *Phased-Guard*

D. HPC Features Selection

Given that there exists a limited number of HPC registers available in modern microprocessors that can be collected simultaneously [24], [25], it is necessary to identify the most important HPCs for classifying the victim under attack and victim under no attack conditions. To select the most prominent HPC features we employ Correlation Attribute Evaluation ($CorrelationAttributeEval$ in Weka [14]) with its default settings to calculate the Pearson correlation between attributes (HPC features) and class (victim under attack and victim under no attack conditions). Correlation attribute evaluation algorithm calculates the Pearson correlation coefficient between each attribute and class, as given below:

$$\rho(i) = \frac{cov(Z_i, C)}{\sqrt{var(Z\_i)\ var(C)}} \quad i = 1, ..., 16 \qquad (2)$$

where ρ is the Pearson correlation coefficient. Z_i is the input dataset of event i ($i = 1, \ldots, 16$). C is the output dataset containing labels, i.e., "Under Attack" or "Under No Attack" in our case. The $cov(Z_i, C)$ measures the covariance between input data and output data. The $var(Z_i)$ and $var(C)$ measure variance of both input and output datasets, respectively. Next, the HPCs will be ranked according as shown in Table II.

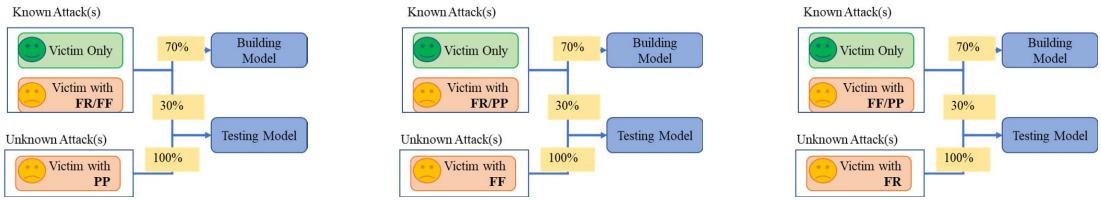
E. Phase I: SCAs Detection using Binary Classification

Initially, the proposed ML-based SCAs detector, *Phased-Guard*, uses binary classification to detect the "under attack" or "under no attack" conditions. For the purpose of thorough analysis of various types of ML classifiers, OneR, MLP (multilayer perceptron), J48, and BayesNet ML algorithms are deployed in *Phased-Guard* as the final binary classification models. The rationale for choosing these machine learning models is that they are from different branches of ML including Bayesian, neural network, decision tree, and rule-based techniques covering a diverse range of learning algorithms.

As mentioned before, only limited HPC features can be collected across different processors at once, due to a limited number of registers available on-chip for storing them (only 4 HPCs on the tested consumer processor such as Intel I5). Hence, reducing the number of HPC features is required for training the ML models in order eliminate the need of multiple runs to capture the required HPCs information. For this purpose, various number of HPCs from 16 to 1 (16, 8, 4, 2, and 1 selected based on the ranking in Table II) are examined to evaluate the influence of reduced HPCs on classification accuracy and highlight the importance and motivation of using a lower number of HPCs for effective runtime SCAs detection in *Phased-Guard*. Four different types of classification algorithms are trained with 70% victims only and victim under known attacks condition. The remaining 30% of victim under no attack, victim under known attack dataset, and all victim under unknown attacks dataset to build the testing dataset for evaluating the effectiveness of the *Phased-Guard*. It is notable that to appropriately model the zero-day attacks impact on the proposed SCAs detector, the considered unknown attacks data is completely separated and do not contribute to the training dataset of the ML classifiers.

F. Phase II: SCA Application Type Identification

As shown in Figure 3, once *Phased-Guard* reports the "under attack" condition in Phase I, the Phase II will be activated to identify the type of the SCA that has attempted to comprise the security of the target system. To this aim, we primarily convert the basic binary SCAs classification to a multiclass classification problem, i.e., with more than two possible discrete outcomes. We propose to predict the behavior of the SCA applications using a multiclass BayesNet technique. In *Phased-Guard*, the output of BayesNet model is corresponding to a set of feasible classes of SCA applications including three individual classes, including "Flush+Flush", "Flush+Reload", and "Prime+Probe" classifying the three analyzed SCA types. It is notable that the proposed multiclass classification can be simply extended to more number of output classes covering the prediction of other types of microarchitectural SCAs. The BayesNet model is trained using an extensive set of HPCs data captured in the profiling step and the inputs to the BayesNet model consists of the top 2 low-level features shown in Table



a) Case 1 b) Case 2 c) Case 3

Fig. 5. Different considered training and testing cases in *Phased-Guard* for building known and unknown victims and attacks dataset

II. During run-time, the probability of each class of application being executed is calculated and the BayesNet classifier then selects the SCA type that achieves the highest probability as the final predicted application type.

V. Experimental Results and Analysis

In this section, we evaluate the effectiveness of *Phased-Guard* across various performance evaluation metrics including detection accuracy, F-measure, and monitoring overhead. To this aim, we consider three different training and testing cases, as shown in Figure 5 for building known and unknown victims and attacks dataset where two SCAs are used for building known attacks data (standard 70%-30% training-testing split) and the third SCA is used to develop the unknown attacks data for testing the ML-based detectors (100% used for testing). The testing results are also presented in terms of detection accuracy for victims alone, victims under known attack, and victim under unknown attacks conditions.

Furthermore, in this section we compare the effectiveness of *Phased-Guard* which is based on similarity distances data calculated by DTW method, with existing traditional machine learning SCAs detection models proposed in previous work [6], [21], [22]. For this purpose, we first demonstrate the accuracy for "under no attack", "under known attack", and "under unknown attack" cases. And then we explore the influence of number of HPCs features on SCAs detection accuracy. Finally we show the robustness of proposed detector with only 2 HPCs.

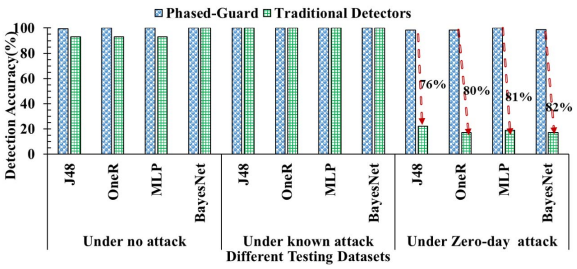


Fig. 6. Detection accuracy comparison of *Phased-Guard* vs. Traditional detectors with total 16 HPCs

A. Detection Accuracy

Figure 6 depicts the detection accuracy of *Phased-Guard* (which uses similarity distances of HPCs data given by DTW) with traditional ML-based detection methods (which are trained using raw HPCs values) [6], [21], [22]. The results are based on the total collected 16 HPCs across different ML classifiers for "no attack", "known attack", and "unknown attack" conditions.

For victim under no attack and victim under known attack conditions with 4 HPCs, it is observed that both *Phased-Guard* and traditional ML-based detection methods achieve high detection accuracy, above 99% across the four classification algorithms (J48, OneR, MLP, and BayesNet). The observations indicate that both DTW-based classification and raw HPCs value-based classifications are able to build effective SCAs detection system for "known attacks" condition where only the victim application is monitored. For victim under unknown attacks, an interesting observation is that while the accuracy of traditional HPC-based detectors significantly drops to less than (20-15)%, the proposed similarity-based classification model, *Phased-Guard*, still achieves a high detection accuracy (99%) outperforming the traditional HPCs value-based classifiers by at least 76%. Furthermore, it can be observed that among the four classification models used for detecting the SCAs, neural network-based MLP classifier outperforms all other models with average accuracy of 99%. By comparison, HPCs value-based classification shows very low accuracy for "under unknown accuracy" across all four classification models, averaging less than 20%. The results clearly highlight the effectiveness of *Phased-Guard* as compared with the state-of-the-art ML-based detection techniques in detecting unknown (zero-day) microarchitectural side-channel attacks.

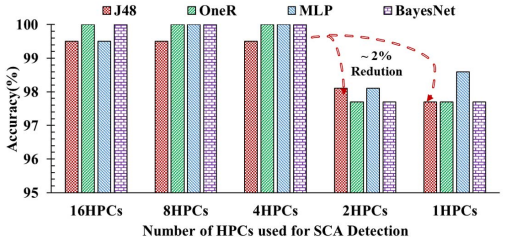


Fig. 7. Detection accuracy of *Phased-Guard* for victim under No attacks

B. Analyzing the Impact of Number of HPC Features

As mentioned in Section IV-D, the number of available registers for collecting the HPC features in modern microprocessors is limited. This has reduced the number of HPC features that can be collected in a single run of applications. Here, we evaluate the detection accuracy of the various ML models with varying number of HPC features (16, 8, 4, 2, and 1) to analyze the impact of fewer HPCs on the detection performance of

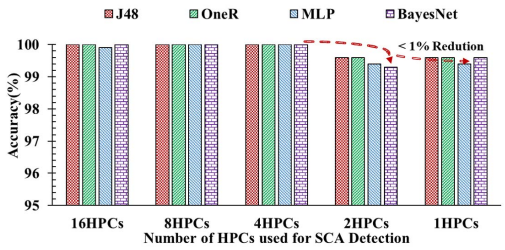


Fig. 8. Detection accuracy of *Phased-Guard* for victim under known attacks

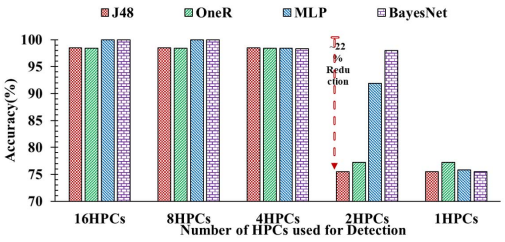


Fig. 9. Detection accuracy of *Phased-Guard* for victim under unknown attacks

SCAs detectors. As shown in Figure 7 and Figure 8, reducing the number of HPCs from 16 to 8 and further to 4 has minimal influence on SCAs detectors in under no attack, under known attacks, and victim under unknown attacks conditions. Interestingly, reducing the number of HPCs to 2 and 1 results in only 2% drop in detection accuracy for no attack condition and less than 1% for victim under known attack. But for victim under unknown attack condition, detection accuracy decreases by 22% for all classification models except BayesNet. Figure 9 further depicts the detection accuracy of *Phased-Guard* for victim under known attacks conditions across different number of HPCs and ML classifiers. As seen, the BayesNet model can still yield high detection accuracy using only 2 HPCs, whereas employing only 1 HPC results in a significantly low SCAs detection accuracy for all four classification models. As a result, due to the limited number of available registers ranging from 2 to 8 in today's microprocessors, *Phased-Guard* employs only the 2 most prominent HPC features (L3 MISSES and L2 HIT) with BayesNet classifier to accurately detect all three conditions (under no attack, under known attack, and under unknown) which makes the proposed solution even compatible for run-time detection of microarchitectural SCAs for resource-limited embedded systems and IoT devices.

TABLE III. DETECTION PERFORMANCE EVALUATION FOR BAYESNET WITH 2 HPCS

Classifiers	TP_Rate	FP_Rate	F-Measure
J48	0.755	0.245	0.861
OneR	0.772	0.228	0.871
MLP	0.919	0.081	0.958
BayesNet	0.980	0.020	0.990

We further evaluate the performance of the proposed methodology with only 2 HPC features by considering various evaluation metrics such as True Positive Rate (TP_rate), False Positive Rate(FP_Rate), and F-Measure. The F-Measure (F-Score) in machine learning is interpreted as a weighted average of the precision (p) and recall (r) which is formulated as 2 × (p×r)/(p+r). As shown in Table III, the BayesNet and MLP are delivering higher detection performance results as compared to the other classifiers with 2 HPCs. As seen, the BayesNet-based SCAs detector in is the best model across all implemented ML classifiers achieving 0.98 true positive rate, 0.02 false positive rate, and close to 1 F-Measure value.

TABLE IV. ACCURACY RESULTS OF SCAS TYPE IDENTIFICATION IN *Phased-Guard*

Actual SCA \ Predicted SCA	Flush+Reload	Flush+Flush	Prime+Probe
Flush+Reload	99.6%	0.4%	0%
Flush+Flush	0.8%	99.2%	0%
Prime+Probe	0%	0.3%	99.7%

C. Application Identification Evaluation

After detecting victim is under attack, the second level of *Phased-Guard* further attempts to identify the type of the microarchitectural SCA. Here, we evaluate the second level of the *Phased-Guard* for identifying the type of side-channel attack, which further could provide valuable information for system administrators to deploy suitable mitigation strategies. Each specialized detector is tested with under known attacks and unknown attacks conditions. Corresponding accuracy and robustness are shown in Table IV. It can be seen that all the three SCAs can be categorized correctly above 99.5% highlighting the effectiveness of *Phased-Guard* in identifying the class of occurred SCA on the target system.

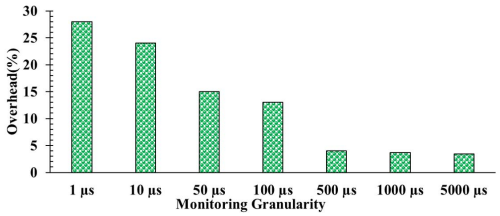


Fig. 10. Performance overhead with various monitoring granularity

D. Monitoring Overhead

Here, we examine the performance overhead caused by the deployed HPCs monitoring tool for collecting both victim and benign applications under various monitoring granularities. As shown in Figure 10, the x-axis represents applied monitoring granularity ranging from 1 μs to 5000 μs, the primary y-axis represents the execution time of victim applications, and the second y-axis represents performance overhead under different monitoring granularity. Execution time under no HPCs monitoring is used to obtain the overhead percentage. It is observed that, generally, the smaller the monitoring granularity is, the larger the performance overhead is. When the monitoring scale is 1 μs, performance overhead is at its highest value reaching to 30%. With 500 μs granularity, the performance overhead is nearly 4% which is close to the under 1000 μs and 5000 μs granularities. That is the reason of choosing 500 μs as the monitoring granularity in *Phased-Guard*.

VI. CONCLUSION

Microarchitectural Side-Channel Attacks (SCAs) exploit side-channel vulnerabilities originated from fundamental

performance-enhancing components (e.g. cache memories and branch predictors) to steal sensitive information. Prior studies have focused on detecting these attacks using Machine Learning (ML) algorithms applied on microarchitectural features captured from Hardware Performance Counters (HPCs). In this work, we have identified important challenges associated with effective run-time zero-day SCAs detection and identification that have been ignored in prior studies. In response, we propose *Phased-Guard*, a two-phase machine learning-based framework to accurately detect and identify both known and unknown attacks at run-time using the most prominent low-phase features. In the first phase, *Phased-Guard* deploys a binary classification model to detect the existence of SCAs on the target system. Next, *Phased-Guard* applies a multiclass classification model to accurately classify the type of microarchitectural SCA applications. The experimental results show that the proposed methodology by monitoring only the victim applications' microarchitectural events, achieves 98% attack detection accuracy even for the unknown SCAs at the cost of only 4% performance overhead for monitoring.

ACKNOWLEDGMENT

This research was supported in part by DARPA SSITH program under the award number 97774952.

REFERENCES

[1] ALAM, M., BHATTACHARYA, S., MUKHOPADHYAY, D., AND BHATTACHARYA, S. Performance counters to rescue: A machine learning based safeguard against micro-architectural side-channel-attacks.

[2] BERNDT, D. J., AND CLIFFORD, J. Using dynamic time warping to find patterns in time series. In *KDD workshop* (1994), vol. 10, Seattle, WA, pp. 359–370.

[3] BRIONGOS, S., IRAZOQUI, G., MALAGÓN, P., AND EISENBARTH, T. Cacheshield: Detecting cache attacks through self-observation. In *Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy* (2018), pp. 224–235.

[4] BROTZMAN, R., AND ET. AL. Casym: Cache aware symbolic execution for side channel detection and mitigation. In *CaSym: Cache Aware Symbolic Execution for Side Channel Detection and Mitigation*, IEEE, p. 0.

[5] CHIAPPETTA, M., SAVAS, E., AND YILMAZ, C. Xlate: https://www.vusec.net/projects/xlate/.

[6] CHIAPPETTA, M., SAVAS, E., AND YILMAZ, C. Real time detection of cache-based side-channel attacks using hardware performance counters. *Applied Soft Computing 49* (2016), 1162–1174.

[7] DEPOIX, J., AND ALTMEYER, P. Detecting spectre attacks by identifying cache side-channel attacks using machine learning. *Advanced Microkernel Operating Systems* (2018), 75.

[8] DINAKARRAO, S. M. P., ET AL. Adversarial attack on microarchitectural events based malware detectors. In *Proceedings of the 56th Annual Design Automation Conference 2019* (2019), pp. 1–6.

[9] DISSELKOEN, C., KOHLBRENNER, D., PORTER, L., AND TULLSEN, D. Prime+ abort: A timer-free high-precision l3 cache attack using intel tsx. In *26th USENIX Security Symposium (USENIX Security 17),(Vancouver, BC)* (2017), pp. 51–67.

[10] FUSTOS, J., FARSHCHI, F., AND YUN, H. Spectreguard: An efficient data-centric defense mechanism against spectre attacks. In *Proceedings of the 56th Annual Design Automation Conference 2019* (2019), pp. 1–6.

[11] GE, J., GAO, N., TU, C., XIANG, J., AND LIU, Z. More secure collaborative apis resistant to flush+ reload and flush+ flush attacks on armv8-a. In *2019 26th Asia-Pacific Software Engineering Conference (APSEC)* (2019), IEEE, pp. 410–417.

[12] GRUSS, D., AND ET.AL. Flush+ flush: a fast and stealthy cache attack. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment* (2016).

[13] GUTHAUS, M. R., AND ET. AL. Mibench: A free, commercially representative embedded benchmark suite. In *Proceedings of the fourth WWC* (2001).

[14] HALL, M., ET AL. The weka data mining software: an update. *ACM SIGKDD explorations newsletter 11*, 1 (2009), 10–18.

[15] IRAZOQUI, G., EISENBARTH, T., AND SUNAR, B. Mascat: Stopping microarchitectural attacks before execution. *IACR Cryptology ePrint Archive 2016* (2016), 1196.

[16] KOCHER, P., AND ET.AL. Spectre attacks: Exploiting speculative execution. *arXiv:1801.01203* (2018).

[17] LI, C., AND GAUDIOT, J.-L. Online detection of spectre attacks using microarchitectural traces from performance counters. In *2018 30th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)* (2018), IEEE, pp. 25–28.

[18] LIU, F., AND ET.AL. Last-level cache side-channel attacks are practical. In *SP* (2015), IEEE, pp. 605–622.

[19] MINKIN, M., MOGHIMI, D., LIPP, M., SCHWARZ, M., VAN BULCK, J., GENKIN, D., GRUSS, D., PIESSENS, F., SUNAR, B., AND YAROM, Y. Fallout: Reading kernel writes from user space. *arXiv preprint arXiv:1905.12701* (2019).

[20] MUCCI, P. J., BROWNE, S., DEANE, C., AND HO, G. Papi: A portable interface to hardware performance counters. In *Proceedings of the department of defense HPCMP users group conference* (1999), vol. 710.

[21] MUSHTAQ, M., ET AL. 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* (2018), ACM, p. 1.

[22] PRADA, I., IGUAL, F. D., AND OLCOZ, K. Detecting time-fragmented cache attacks against aes using performance monitoring counters. *arXiv preprint arXiv:1904.11268* (2019).

[23] SABBAGH, M., FEI, Y., WAHL, T., AND DING, A. A. Scadet: A side-channel attack detection tool for tracking prime-probe. In *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)* (2018), IEEE, pp. 1–8.

[24] SAYADI, H., ET AL. Ensemble learning for effective run-time hardware-based malware detection: A comprehensive analysis and classification. In *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)* (2018), IEEE, pp. 1–6.

[25] SAYADI, H., ET AL. 2smart: A two-stage machine learning-based approach for run-time specialized hardware-assisted malware detection. In *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)* (2019), IEEE, pp. 728–733.

[26] SAYADI, H., ET AL. Recent advancements in microarchitectural security: Review of machine learning countermeasures. In *63rd IEEE International Midwest Symposium on Circuits and Systems (MWSCAS)* (2020), IEEE.

[27] VAN SCHAIK, S., MILBURN, A., ÖSTERLUND, S., FRIGO, P., MAISURADZE, G., RAZAVI, K., BOS, H., AND GIUFFRIDA, C. Ridl: Rogue in-flight data load. In *2019 IEEE Symposium on Security and Privacy (SP)* (2019), IEEE, pp. 88–105.

[28] WANG, H., ET AL. Comprehensive evaluation of machine learning countermeasures for detecting microarchitectural side-channel attacks. In *Proceedings of the 2020 on Great Lakes Symposium on VLSI* (New York, NY, USA, 2020), GLSVLSI '20, Association for Computing Machinery, p. 181–186.

[29] WANG, H., ET AL. Mitigating cache-based side-channel attacks through randomization: A comprehensive system and architecture level analysis. In *DATE'20* (2020), IEEE.

[30] WANG, H., ET AL. Scarf: Detecting side-channel attacks at real-time using low-level hardware features. In *2020 IEEE 26th International Symposium on On-Line Testing and Robust System Design (IOLTS)* (2020), IEEE, pp. 1–6.

[31] YAROM, Y. Mastik: A micro-architectural side-channel toolkit. *Retrieved from School of Computer Science Adelaide: http://cs. adelaide. edu. au/~ yval/Mastik* (2016).

[32] YAROM, Y., AND FALKNER, K. Flush+ reload: A high resolution, low noise, l3 cache side-channel attack. In *USENIX Security Symposium* (2014), vol. 1, pp. 22–25.

[33] ZHANG, T., AND LEE, R. B. Secure cache modeling for measuring side-channel leakage. *Technical Report, Princeton University* (2014).

[34] ZHANG, T., ZHANG, Y., AND LEE, R. B. Cloudradar: A real-time side-channel attack detection system in clouds. In *International Symposium on Research in Attacks, Intrusions, and Defenses* (2016), Springer, pp. 118–140.