DREAL: Detecting Side-Channel Attacks at Real-time using Low-level Hardware Features

Han Wang[1], Hossein Sayadi[2],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,hhomayoun}@ucdavis.edu, [2]{hossein.sayadi}@csulb.edu,[3]{asasan,srafatir}@gmu.edu

[1] ***Abstract*—Prior studies on detection of SCAs based on low-level microarchitectural features captured from processors' hardware performance counter (HPC) registers have considered collecting hardware events of both victim applications (cryptographic application, e.g. RSA, AES and etc.) and attack applications. However, as shown in recent works the attack HPCs data can be easily corrupted which results in misleading the SCA detection method. Furthermore, the prior works have explored the suitability of a limited number of Machine Learning (ML) algorithms in detecting SCAs without examining the instance level false alarm rate that as we show in this work is a more important evaluation metric for SCA detection techniques. In response, in this paper, we propose DREAL, a customized machine learning-based real-time side-channel attack detection methodology using low level hardware features captured from HPC registers. The experimental results indicate that with the proposed detection methodology, we can achieve up to 97%interval prediction accuracy eliminating the need for profiling SCAs. Furthermore, DREAL detection methodology can obtain 100% attack detection accuracy with 0instance level false alarm rate.**

***Index Terms*—Side-channel Attacks, Mitigation, Frequency, Prefetcher, Randomization, Adaptation.**

I. Introduction

In the last few decades, the complexity of computing devices has been extensively increased to support different functionalities and meet performance demands. Despite the provided performance benefits, they also present a bunch of hardware vulnerabilities which can be exploited by side-channel attacks (SCAs) [10], [17], [20]. Such exploits can be launched by the attacker remotely and require no physical access. There exists an emerging need to address the security risks and challenges posed by such harmful exploits, calling for effective detection methodology for protecting compute system from them with minor overhead.

Prior works on exploits detection such as [6], [21] propose the use of microarchitectural pattern analysis captured through Hardware Performance Counters (HPCs) to detect the SCAs with latency by order of ranging magnitude from milliseconds to seconds. For instance, [6] proposes to detect the SCAs with the usage of both victim and attack applications' HPCs traces. Then based on the obtained HPCs, the correlation between the HPC events of victims' and attacks' traces. Similarly, in [21] the authors present CloudRadar which aims at detecting cross-VM side-channel attacks by making use of HPC patterns. Other work [2] focuses on the victim HPCs traces to detect "under attack" condition to bypass the lack of attack HPCs data circumstances. However, it employs the sum of HPCs as the feature with only one classification model analyzed. Undoubtedly, the prior SCA detection works have made some progress in detecting the attacks. However, they fall short in addressing the challenges defined above as well as several drawbacks, as demonstrated below.

Lack of Robustness: Our comprehensive analysis shows that the majority of previous works on side-channel attacks detection jointly correlate the HPCs traces of victim and attack applications [6]. However, recent studies [7] have demonstrated that current HPC features monitoring methods suffer from the overcounting issue that creates the opportunity for attackers to manipulate HPCs data by slightly changing SCA applications. Hence, current detection techniques relying on the HPCs data of attack applications are facing with great security threats.

Limited Machine Learning Classifiers: With the advancements achieved in machine learning (ML) domain, a wide range of classification and anomaly detection techniques are developed by applying Machine Learning (ML) techniques. However, existing works in particular on SCA detection have primarily focused on one or a few ML techniques for the purpose of attack detection and classification [6], [21]. Such an analysis leaves a void in terms of performance of attack detection, as various ML classifiers yield different performance in detecting various types of attacks [11], [15]. As we show in this work, the detection accuracy and performance of SCA detection vary across different categories of ML algorithms. As a result, it is crucial to comprehensively explore and examine the suitability of various ML classifiers for effective SCA detection.

High False Alarm Rate: Prior studies on real-time HPCbased SCA detection have neglected to examine the instance level (a complete temporal sequence of victim applications' HPCs) false positives of HPC data and have only evaluated the SCA detectors based on the interval level (a subsequence

To address the aforementioned challenges, in this work we propose *DREAL*, a machine learning-based real-time attack detection methodology using low-level hardware features to

[1]DISTRIBUTION STATEMENT A. Approved for public release: distribution is unlimited.

avoid exploit of hardware vulnerabilities. The main contribution of this work can be summarized as below:

- This work proposes *DREAL* to detect attacks based on differentiating HPCs data of only the victim applications under attack and under no attack.
- Various ML classification algorithms are explored.
- A customized set of HPC features is employed to improve the detection accuracy further while lowering the false alarm rate.

II. Motivations and Background

In this section, we present a comprehensive analysis of major motivations and case studies to propose SCA-Hunter framework for detecting side-channel attacks at the hardware level using effective machine learning-based solutions.

A. *SCA Detection based on Victim Applications' HPCs*

As described, SCA-Hunter by using only the HPC features of victim applications could highly accurately detect the side-channel attacks. The motivations behind employing victim applications' HPCs traces are described as follows:

Unreliable Attackers' HPCs: Recent work [23] presents the problem of detecting attackers by classifying anomalies and benign applications based on HPC information with the aid of ML techniques. The work in [7] further points out the non-deterministic and over-counting problems of instructions associated with HPCs information, in which the attackers can intentionally modify instructions slightly and manipulate the counters. Such researches indicate that attackers' HPCs information could be easily manipulated, thus is not a reliable source for security enhancement of computer systems.

Side-Channel Attacks Design Principle: Current SCAs intentionally cause influence on victim applications' cache or branch predictor units by flushing/priming cache, mistraining branch predictors and then observing access time of the cache sets, which changes caching victims' data and microarchitectural behaviors of victim applications [22]. This also provides the opportunity of detecting SCAs by observing the alteration in microarchitectural behaviors.

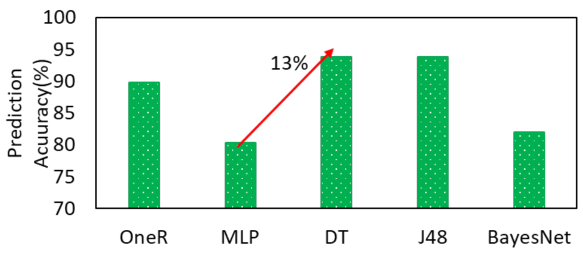


Fig. 1. Various ML classifiers prediction accuracy for RSA under Flush-Reload attack (dataset based on Section III-A)

B. *Comprehensive Analysis of Various ML Classifiers*

Our comprehensive comparison of different ML classifiers indicates that the prediction accuracy of SCA detection for different classifiers can vary significantly. For instance, AES under Flush Reload (AES-FR) attack is analyzed with five classification techniques including OneR, multilayer perceptron (MLP), DecisionTable (DT), J48, and BayesNet as shown in Figure 1. As shown, these ML classifiers have different SCA detection accuracy ranging from 80% to 93%. This highlights the importance of exploring the suitability of a wide range of machine learning algorithms for the purpose of SCA detection in order to identify the best ML classifiers achieving the highest possible detection accuracy.

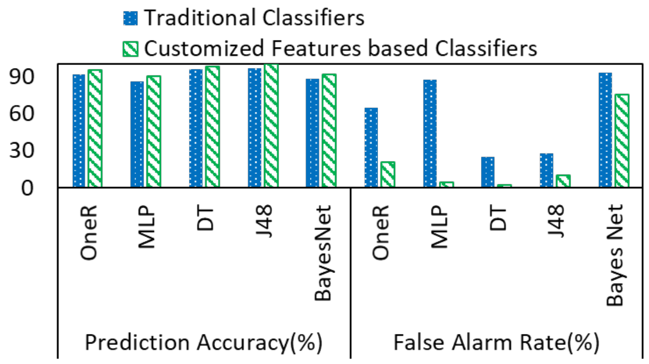


Fig. 2. Traditional and customized features based classifiers comparison (datasets collected based on Section III-A)

C. *Customized Features based ML Classifiers*

Prior works on SCA detection capture the sum of HPCs value for a certain time period as features and employ traditional ML classification methods to achieve a high prediction accuracy. However, as mentioned before, such methods could cause a high false alarm rate. As a result, in this work we proposed customized features based ML classifiers which extract more features, such as min, max, stdev, and sum of an interval to further boost the prediction accuracy and reduce false alarm rate. Prediction accuracy and false alarm rate of traditional and customized features based classifiers are plotted in Figure 2. It can be seen that customized features based ML classifiers outperform traditional methods by around 4% prediction accuracy. Furthermore, the false positive rate significantly drops from 87.2% to 4.7% for MLP when applying customized features based classifiers. Hence, for effective real-time SCA detection customized features based classifiers with more extracted HPCs features could enhance the detection accuracy while substantially reducing the false alarm rate.

TABLE I
Attacks List

Victim	Attack	Source
RSA	L3 Flush Reload	Masitk [19]
	L1 Prime Probe	Masitk
AES	Flush Reload	Xlate [5]
	L3 Flush Flush	Xlate
victim_function	Spectre	Spectre [4]

III. Proposed Methodology

In this section, we first present details of the experimental setup and configurations. And then the proposed DREAL

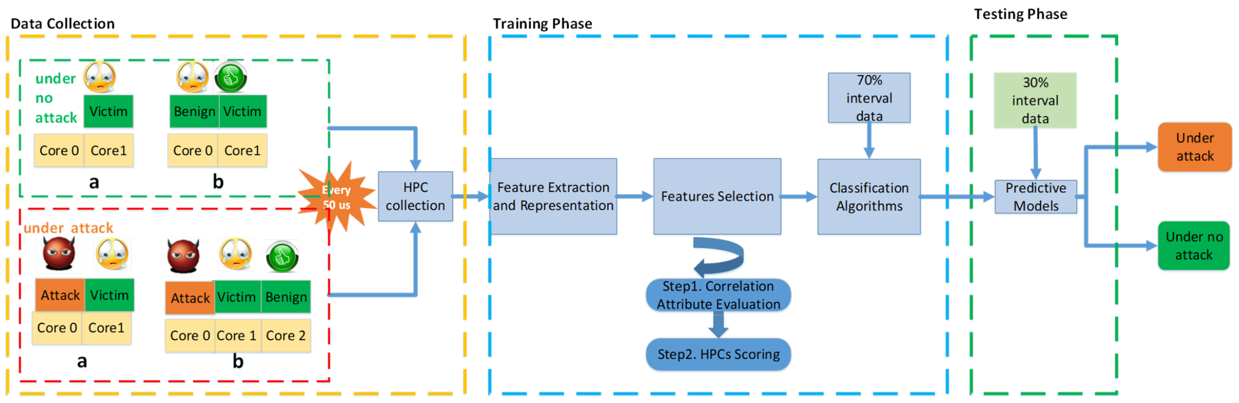


Fig. 3. Overview of DREAL, the proposed real-time SCAs detection methodology based on victim application HPCs

TABLE II
SELECTED MONITORING HPCS LIST

L1 HIT	L1 MISSES
L2 HIT	L2 MISSES
L3 HIT	L3 MISSES
All BRANCHES RETIRED	BRANCHES MISPREDICTED
BR_NONTAKEN_CONDITIONAL	BR_TAKEN_CONDITIONAL
TAKEN_INDIRECT_NEAR_CALL	UOPS_RETIRED.ALL
INST_RETIRED.ANY	DTLB_LOAD_MISSES
DTLB_STORE_MISSES	ITLB_MISSES

methodology shown in Figure 3 will be introduced. As shown, DREAL is comprised of three major steps including data collection, training phase, and testing phase. First, for feature extraction the ”under no attack” and ”under attack” HPC data will be collected within a) isolated scenario, and b) non-isolated scenario. The ”isolated” refers to the case that a computer only processes victim applications; whereas the ”non-isolated” denotes that a computer system processes victim applications on one core while benign applications are being executed on the rest of the cores. Then customized features are extracted and the data will be used to train various classifiers. The trained models will be employed in the testing phase and false alarm minimization technique further helps to reduce false alarm rate.

A. Data Collection

In this work, all experiments are conducted on an Intel I5-3470 desktop with 4 cores, 8GB DRAM, and three-level cache system. Victim applications and side-channel attacks are selected from Mastik [19] and Xlate [5]. Furthermore, MiBench [9] benchmark suit is used to represent benign applications. In this work, we propose using a customized tool to collect hardware performance counters based on model-specific registers (MSRs). The proposed customized monitoring tool collects HPCs per processor at microsecond scale with privileged access to avoid HPCs contamination from other processes addressing the overcounting challenges presented in a recent study [7]. Based on the behavior and functionality of studied SCAs, 16 HPCs are considered in this work for further analysis as listed in Table II.

Various hardware performance counters data are collected using the four available HPC registers in the experimented I5 processorat every 50 microseconds. Each pair of a VNA and VA executes for 50 times. Next, both VA and VNA HPC data are merged together to create the final dataset. Furthermore, Weka data mining tool is deployed for implementing the machine learning classifiers in which 70% of the randomized data is used for training the classifiers and the rest of 30% is used for testing evaluation.

B. Customized Features based classifiers

The proposed customized features based classification is comprised of three main steps: 1) feature extraction and representation; 2) HPCs selection due to a limited number of registers for effective real-time detection of the attacks; 3) training the ML classification algorithms.

1) Feature Vector Extraction and Representation: For proposed classification, time-series data will be transformed from a time sub-sequence to a vector of features. In the first step of the transformation process, the raw data will be received from the monitoring module. The time-series sub-sequences’ properties will be extracted which include statistics of distribution values (e.g., max value, min value, sum value, Gaussianity), stationarity (e.g.,StatAv) and etc. In order to effectively determine the most prominent features for the purpose of real-time SCA detection, we deployed Greedy Forward Selection algorithm [3], [8] and we found that max, and stdev contribute more to assisting in distinguishing the difference between ”under no attack” and ”under attack” conditions. Hence, the input for each transformation is $T = (t1, t2, ..., tm)$ where tm is a vector of HPCs values. Furthermore, the outputs of transformation are a vector of actual HPC values i.e., L1HIT sum, L1 HIT max, L1 HIT min, L1 HIT stdev, ..., ITLB_MISSES sum, ITLB_MISSES max, ITLB_MISSES min, ITLB_MISSES stdev.

2) HPCs Selection: Given that there exists a limited number of HPCs available (only 4 HPCs) to be collected one time simultaneously [14]–[16], it is necessary to identify the most important HPCs for classifying the VA and VNA conditions for different types of SCAs. For HPCs reduction, we employ Correlation Attribute Evaluation ($CorrelationAttributeEval$ in Weka) with its default settings to calculate the Pearson correlation between attributes (HPC features) and class (VA and VNA conditions). And then the sum score of each HPC features (min, max, stdev, and sum in this work) will be calculated.

C. ML Classifiers Implementation

For the purpose of a thorough analysis of various types of ML classifiers, OneR, MLP (multilayer perceptron), DT (decision table), J48, and BayesNet ML algorithms are deployed as our final classification models. The rationale for selecting these machine learning models are: firstly, they are from different branches of ML: regression, neural network, decision tree, and rule-based techniques covering a diverse range of learning algorithms which are inclusive to model both linear and nonlinear problems; secondly, the prediction model produced by these learning algorithms can be a binary classification model which is compatible with the SCA detection problem in our work. As mentioned before, only four HPCs can be collected for most processors once due to a limited number of registers for storing them. For this purpose, various number of HPCs from 16 to 4 (16, 12, 8 and 4) 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 (only 4) for effective real-time SCA detection in DREAL method.

IV. Results Evaluation

In this section, we evaluate the effectiveness of the detection approach and compare it in terms of various evaluation metrics including detection accuracy, false alarm rate, and performance overhead of proposed customized features based classifiers over traditional and time-series classifiers.

A. Classifiers Comparison

In order to present the effectiveness of using customized features, ML classifiers are fed with only sum value of HPCs named as traditional classifiers. The implemented ML classification algorithms are OneR, multi-layer perceptron (MLP), DecisionTable (DT), J48, and BayesNet that are covering a diverse range of machine learning techniques.

Prediction Accuracy: Figure 4-(a) presents the SCA detection accuracy with a varied number of HPCs for the proposed detector (customized features based classifiers) and existing works (traditional and time-series classifiers using different techniques) [6], [21]. As shown, the proposed and traditional classifiers achieve above 80% prediction accuracy despite utilizing less number of HPCs, which makes them formidable candidates to consider for real-time SCA detection. Figure 4-(b) zooms in the comparison between proposed and traditional classifiers. It can be seen that the method by using customized features based ML classifiers is able to further boost prediction accuracy, ranging from 2% to 6%.

False Alarm Rate: As discussed, despite high detection accuracy, one of the major challenges associated with detection is the false alarms in which we evaluate the false alarm rate for different techniques below. Figure 5 depicts the false alarm rate with proposed and existing techniques when utilizing a varied number of HPCs for SCA detection. The false alarm rates produced by traditional classifiers based SCA detection is significantly high, 57% on an average across all ML techniques and HPC values. This is due to the fact that traditional methods are biased to "under attack". However, the proposed technique with using customized features employs more features that aid the ML classifiers to predict "under attack" scenario with higher confidence and accuracy. Taking MLP-based SCA detector as an example, the proposed customized classifier can decrease false alarm rate from 87% (obtained when utilizing traditional classifier) to 4.7%, though the detection accuracy is similar.

Classifiers Performance Overhead: In order to perform real-time detection, the latency of SCA detection needs to be minimal (in the range of a few hundred of μs, as the SCAs need dozens of ms to execute). As shown in Table III, traditional classifiers and proposed customized features based classifiers have much lower overhead and no more than 300 μs. It can also be observed that proposed detection using customized features based classifiers have only a few microseconds performance overhead compared to traditional classifiers. This is because while both of the methods adopt the same algorithms, the proposed customized features based classifiers have fewer dimensions (min, max, stdev of HPCs) which leads the classifiers to detect the SCA behavior with much lower performance overhead.

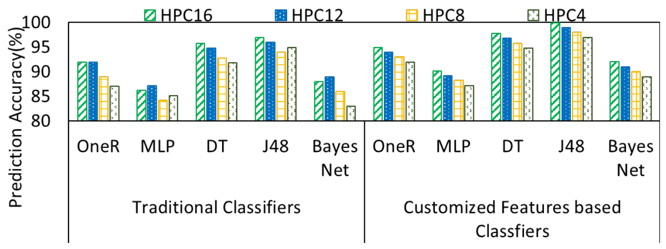
V. Relevant Works

In this section, we discuss the latest studies on side-channel attack detection. Recent work [18] uses cache latency to build cache occupancy of victims and attacks. Based on the cache occupancy relation of the two processes, SCAs can be deduced. However, some attacks rely on "flush" instruction to identify the position of victim and do not have cache occupancy including Flush Flush and Spectre. The work in [13] proposes to analyze the executable binary file and detect if there is any potential risk for Prime Probe attacks. It shows 100% attacks detection with 7.4% false positive rate. However, it only evaluates Prime Probe attack with timing analysis at the granularity of minute. Such works are offline and not able to provide effective real-time protection for the computer system.

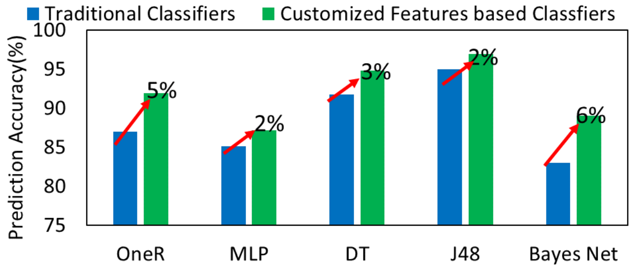
Prior detection work [1] monitors HPCs trace of both victim and attack processes and compare the effectiveness of three ML classifiers: neural network, decision tree C4.5, and K nearest neighbors. [12] proposes a detection system containing one analytic server and one or more monitored computing devices to detect SCAs including Spectre and Meltdown. The analytic server receives HPCs data from monitored devices and identifies suspicious core activity. Once detected, application level monitor will be deployed on the computing devices and take corrective actions as soon as finding suspicious application activity. However, these two works can be bypassed when SCAs manipulate HPCs values according to HPC monitoring challenges discussed in recent work [7].

VI. Conclusion

In this work, we propose DREAL, a machine learning-based methodology for detecting SCAs at real-time using the processor's Hardware Performance Counters (HPCs) information. The proposed SCA detection methodology first solves



a)



b)

Fig. 4. Prediction accuracy comparison: a) prediction accuracy of proposed customized features based classifiers and the rest two type classifiers; b) zooming in prediction accuracy of traditional and customized features based classifiers

TABLE III
LATENCY COMPARISON BETWEEN TRADITIONAL AND CUSTOMIZED FEATURES

	Traditional-based classifiers					Customized features based classifiers				
Models	OneR	MLP	DecisionTree	J48	BayesNet	OneR	MLP	DecisionTree	J48	BayesNet
Overhead (us)	212	231	252	235	228	226	241	267	248	232

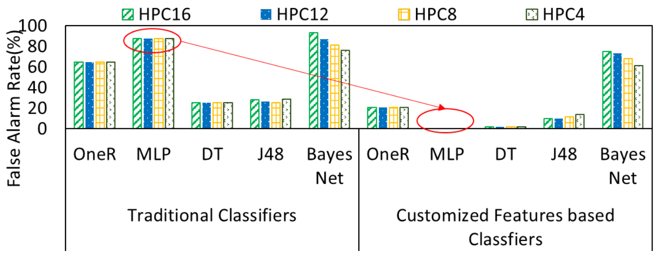


Fig. 5. False alarm rate comparison

the challenge of the lack of attacks' HPCs data by analyzing the difference between Victim under Attack (VA) and Victim Under No Attack (VNA) conditions. Next, it uses HPCs importance evaluation with Correlation Attribute Evaluation algorithm to identify the most prominent HPC features suitable for real-time SCA detection. Furthermore, DREAL presents customized features based machine learning classifiers to achieve accurate real-time SCA detection based on only victim applications' HPCs with optimal false alarm rate. Compared to state-of-the-art solutions, the proposed machine learning-based approach shows higher detection accuracy and robustness with lower false alarm rate, achieving 100% attack detection rate with 0% false alarm rate.

REFERENCES

[1] ALLAF, Z., ADDA, M., AND GEGOV, A. A comparison study on flush+ reload and prime+ probe attacks on aes using machine learning approaches. In *UK Workshop on Computational Intelligence* (2017), Springer, pp. 203–213.
[2] BRIONGOS, S., ET AL. Cacheshield: Detecting cache attacks through self-observation. In *ACM Conference on Data and Application Security and Privacy* (2018), ACM.
[3] CARUANA, R., AND FREITAG, D. Greedy attribute selection. In *Machine Learning Proceedings 1994*. Elsevier, 1994, pp. 28–36.
[4] CHIAPPETTA, M., SAVAS, E., AND YILMAZ, C. Spectre attack: https://github.com/eugnis/spectre-attack.git.
[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] DAS, S., ET AL. Sok: The challenges, pitfalls, and perils of using hardware performance counters for security.
[8] FULCHER, B. D., AND JONES, N. S. Highly comparative feature-based time-series classification. *IEEE Transactions on Knowledge and Data Engineering* (2014).
[9] GUTHAUS, M. R., RINGENBERG, J. S., ERNST, D., AUSTIN, T. M., MUDGE, T., AND BROWN, R. B. Mibench: A free, commercially representative embedded benchmark suite. In *Proceedings of the fourth WWC* (2001), IEEE, pp. 3–14.
[10] KOCHER, P., GENKIN, D., GRUSS, D., HAAS, W., HAMBURG, M., LIPP, M., MANGARD, S., PRESCHER, T., SCHWARZ, M., AND YAROM, Y. Spectre attacks: Exploiting speculative execution. *arXiv preprint arXiv:1801.01203* (2018).
[11] NARUDIN, F. A., ET AL. Evaluation of machine learning classifiers for mobile malware detection. *Soft Computing* (2016), 343–357.
[12] 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).
[13] SABBAGH, M., ET AL. Scadet: A side-channel attack detection tool for tracking prime-probe. In *2018 ICCAD* (2018), IEEE.
[14] SAYADI, H., ET AL. Customized machine learning-based hardware-assisted malware detection in embedded devices. In *2018 TrustCom/BigDataSE* (2018), IEEE.
[15] SAYADI, H., ET AL. Ensemble learning for effective run-time hardware-based malware detection: A comprehensive analysis and classification. In *2018 DAC* (2018), IEEE.
[16] SAYADI, H., ET AL. 2smart: A two-stage machine learning-based approach for run-time specialized hardware-assisted malware detection. In *2019 DATE* (2019), IEEE, pp. 728–733.
[17] WANG, H., ET AL. Mitigating cache-based side-channel attacks through randomization: A comprehensive system and architecture level analysis. In *2020 DATE* (2020), IEEE.
[18] YAO, F., FANG, H., DOROSLOVACKI, M., AND VENKATARAMANI, G. Towards a better indicator for cache timing channels. *arXiv preprint arXiv:1902.04711* (2019).
[19] YAROM, Y. Mastik: A micro-architectural side-channel toolkit. *Retrieved from School of Computer Science Adelaide: http://cs. adelaide. edu. au/˜ yval/Mastik* (2016).
[20] 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.
[21] ZHANG, T., ET AL. Cloudradar: A real-time side-channel attack detection system in clouds. In *RAID* (2016), Springer.
[22] ZHANG, T., AND LEE, R. B. Secure cache modeling for measuring side-channel leakage. *Technical Report, Princeton University* (2014).
[23] ZHOU, B., GUPTA, A., ET AL. Hardware performance counters can detect malware: Myth or fact? In *Proceedings of the 2018 on Asia CCS* (2018), ACM, pp. 457–468.