Enabling Micro AI for Securing Edge Devices at Hardware Level

Han Wang, *Member, IEEE*, Hossein Sayadi, *Member, IEEE*, Sai Manoj Pudukotai Dinakarrao, Avesta Sasan, *Senior Member, IEEE*, Setareh Rafatirad, *Member, IEEE*, and Houman Homayoun, *Senior Member, IEEE*

***Abstract*— Emerging embedded systems and Internet-of-Things (IoT) devices, which account for a wide range of applications are often highly resource-constrained that are challenging the software-based methods traditionally adopted for detecting and containing cyber-attacks (e.g., malware) in general-purpose computing systems. In addition to the complexity and cost (computing and storage), the software-based detection methods mainly rely on the static signature analysis of the running programs, requiring continuous software update which is not affordable for embedded systems and edge devices with limited computing and communication bandwidth. To address these challenges, this work proposes an accurate and cost-efficient micro AI enabled countermeasure for securing modern edge devices against emerging cyber-attacks, i.e., malware and Side-Channel Attacks (SCAs) at the hardware level by monitoring applications' Hardware Performance Counter (HPC) features. To realize a run-time ML-based solution that relies on limited available HPCs in modern edge processors, we first identify the most prominent HPC events for accurate attack detection with the aid of an effective feature selection method. Next, various standard machine learning classifiers are implemented for effective and accurate run-time hardware-assisted malware and side-channel attacks detection. They are compared and characterized in terms of detection accuracy, F-measure, robustness, latency, power consumption, and hardware overheads. Experimental results demonstrate that the J48 classifier achieves the highest detection rate (F-measure) for both malware and SCAs detection with 0.917 and 0.987, respectively, with relatively negligible latency and area overhead as compared to complex models making it a suitable algorithm for enabling an efficient hardware-assisted micro AI countermeasure in edge devices.**

***Index Terms*— Machine learning, micro AI, malware, side-channel attacks, edge devices, hardware performance counters.**

I. Introduction

RECENT advancements in digital electronics have enabled a widespread proliferation of embedded systems ranging from micro-sensors, cell phones, and PDAs to smart homes, healthcare, and military applications. The constant interaction between physical and cyber worlds has made security one of the major concerns in the design of embedded systems. Cybersecurity for the past decades has been in the front line of global attention as a critical threat to the information technology infrastructures [10], [11]. Attackers are increasingly motivated and enabled to compromise software and computing hardware infrastructure. Recent studies have shown that the attackers take advantage of emerging hardware vulnerabilities to compromise systems and deploy malicious activities [12]–[14]. Hence, the security of a computer system can be compromised at the hardware level through various types of attacks such as by executing malicious applications to infect the target host or deploying microarchitectural side-channel attacks (SCAs) [12]–[16] to infer confidential information.

Malware refers to any piece of software written to steal data, unauthorized data access, damaging devices, etc. Viruses, Trojans, Spyware, Rootkits and Ransomware are among the different types of malware [17]. Microarchitectural SCAs have also posed serious threats to the security of modern computing systems. Such attacks exploit side-channel vulnerabilities originating from fundamental performance-enhancing components such as cache memories. Many of the existing cyber-attacks on conventional computing platforms such as servers and desktops can be launched on the embedded systems due to their similarities in using general-purpose processors computing systems and their connectivity to the internet.

The significant growth of modern computing systems in embedded applications and IoT domains has further highlighted the severe impact of cybersecurity threats [5], [18]–[20]. Depending on the target application, these devices can produce a massive amount of data that needs to be handled, and the emerging edge computing paradigm is receiving a tremendous amount of interest to tackle this challenge. Edge devices could include many different computing platforms, such as IoT sensors, laptops, smartphones, security cameras, etc. Ensuring security in embedded systems and edge computing devices translates into several design challenges imposed by the unique features of these systems. There exist some important factors influencing the security vulnerability of embedded systems and IoTs, including the limited energy and resources available, the low computational capacity, and a significant number of computing nodes in the network. Hence, to keep on combating the increase in malicious cyber-

Manuscript received May 15, 2021; revised August 10, 2021 and October 1, 2021; accepted October 25, 2021. Date of publication November 9, 2021; date of current version December 13, 2021. This work was supported in part by the NSF Program under Award 1936836. This article was recommended by Guest Editor J.-S. Seo. *(Corresponding author: Han Wang.)*

Han Wang, Avesta Sasan, and Houman Homayoun are with the Department of Electrical and Computer Engineering, University of California at Davis, Davis, CA 95616 USA (e-mail: hjlwang@ucdavis.edu).

Hossein Sayadi is with the Department of Computer Engineering and Computer Science, California State University at Long Beach, Long Beach, CA 90840 USA.

Sai Manoj Pudukotai Dinakarrao is with the Department of Electrical and Computer Engineering, George Mason University, Fairfax, VA 22030 USA.

Setareh Rafatirad is with the Department of Computer Science, University of California at Davis, Davis, CA 95616 USA.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/JETCAS.2021.3126816.

Digital Object Identifier 10.1109/JETCAS.2021.3126816

TABLE I
COMPARISON OF RECENT HARDWARE-ASSISTED MALWARE AND SIDE-CHANNEL ATTACK DETECTION TECHNIQUES AND THEIR IMPLEMENTATION METHODS

Research	Architecture	Platform	Classification Model	Malware	SCAs	Feature Evaluation	Overhead Analysis (SW/HW)	Evaluation Metric	ACC	FP Rate
[1]	Intel	Linux	ocSVM	✓	✗	Fisher Score	–	ACC, F Score, AUC, ROC	99.5%	–
[2]	Intel	Windows	LR, NN	✓	✗	Expert Knowledge	Hardware	ACC, FP, ROC	94%	7%
[3]	Intel	Windows	LR, NN, EL	✓	✗	Expert Knowledge	Hardware	ACC, FP, ROC, AUC	86.7 %	–
[4]	Intel	Linux	SVM, ocSVM, NB, DT	✓	✗	Gain Ratio	Hardware	Confusion Matrix, ROC	99%	0.52%
[5]	Intel	Linux	BN, J48, JRip, MLP, OneR, RT, SGD, SMO, AB, BG	✓	✗	Correlation Attribute Evaluation	Hardware	ACC, AUC, ROC, ACC*AUC, Area	88%	–
[6]	Intel	Linux	DTW	✗	✓	Fisher Score	–	ACC, F Score, ROC	100%	20%
[7]	Intel	Linux	LDA, LR, and SVM	✗	✓	Expert Knowledge	–	ACC, FP, ROC	99.51%	0.40%
[8]	Intel	Linux	CPD	✗	✓	Relief Algorithm	–	ACC, FP	100%	4.5%
[9]	Intel	Linux	NN	✗	✓	–	–	ACC, F Score, FP	99%	–
This Work	**Intel, ARM**	**Linux, Android**	**NaiveBayes, MLP, J48, OneR, JRiP, SGD**	✓	✓	**Correlation Attribute Evaluation**	**Software and Hardware**	**ACC, AUC, ROC, Latency, FP, Power, Area**	**92.2%, 98.9%**	**4.6%, 0.8%**

SW: software, HW: hardware, Accuracy: ACC, Sensitivity: S, Specificity: C, K Nearest Neighbor: KNN, BayesNet: BN, NaiveBayes: NB, Logistic Regression: LR, AdaBoost: AB, Bagging: BG, Support Vector Machine: SVM, One Class SVM: ocSVM, Neural Netework: NN, Last Level Cache References: LLC, REPTree: RT, Decision Tree: DT, Random Forest: RF, Ensemble Learning: EL, Dynamic Time Warping: DTW, Linear Discriminant Analysis: LDA, Change Point Detection theory: CPD.

attacks, effective intelligent security countermeasures need to be proposed to protect the integrity and confidentiality of the authenticated users' information [20], [21].

Traditional software-based detection techniques have shown to be inefficient and imposed significant complexity and computational overheads on the system. For instance, signature-based detection and semantics-based solutions (e.g., off-the-shelf antivirus tools) are ineffective for resource-constrained embedded systems due to the limited available computing resources. In addition, such detection methods depend on the static signature analysis of executed applications that make them incapable of detecting complex unknown attacks. Recent advancements in microarchitectural security have demonstrated that malicious activities at the processor hardware level ranging from application-based malware to microarchitecture-level side-channel attacks can be accurately identified with the aid of standard Machine Learning (ML) algorithms [3], [5], [6], [19], [22]. In particular, Artificial Intelligence (AI) and ML, driven by a significant increase in the size of data from high-performance computing systems, have been widely adopted in various application domains, with hardware-assisted security being no exception.

ML-based security countermeasures at the hardware level apply the standard ML techniques on the low-level features such as microarchitectural events collected by Hardware Performance Counter (HPC) registers. HPCs are a set of special-purpose registers in the processing units to capture the trace of hardware events for a running application [23]. HPC registers have primarily been deployed to conduct architectural performance analysis and tuning. Recent works have proposed utilizing the HPCs information to secure the hardware systems against malware software and microarchitectural SCAs. In addition, the latency of detecting malicious code is negligible by order of magnitude with relatively low hardware costs [24].

Nonetheless, when it comes to detecting malicious patterns at the hardware level in edge devices, the available underlying resources become a more critical point of concern. Compared to high-performance servers, edge devices host much fewer computation resources for processing heavy computations and complex workloads. In this paper, we have identified and addressed major challenges of ML applications for accurate and cost-efficient malware and side-channel attacks detection in edge devices that have been ignored in prior studies. In particular, limited computing power and resources in embedded systems and edge devices, as well as the small number of available hardware performance counter registers on the modern microprocessors chip (only 2-8 HPCs) that can be simultaneously monitored, have made accurate and cost-efficient run-time malware and side-channel attack detection in edge devices a challenging problem. Moreover, while a wide range of classification and anomaly detection techniques are developed by applying ML techniques, existing works in particular on malware and/or SCA detection have primarily focused on one or a few ML techniques for attacks detection and classification [6], [22]. Such an analysis leaves a void in terms of performance and overhead (latency, hardware implementation, power, etc.) of ML-based security countermeasures, as various ML classifiers yield different performance vs. overhead trade-offs in detecting various types of attacks. In fact, compared to high-performance servers, edge devices host fewer computation resources for processing heavy computations and complex workloads. Therefore, adopting complex learning classifiers (e.g., neural network) presented in prior works [6] could further result in congestion and racing up for CPU resources between malware/SCAs detector and other running applications. As a result, there is an urgent need for developing an accurate and cost-efficient micro AI enabled countermeasure to protect modern edge devices against emerging cyber-attacks.

In response to the discussed challenges, in this work we perform a comprehensive assessment of various machine learning-based countermeasures for accurate and cost-efficient malware and side-channel attacks detection using microarchi-

tectural features and propose a lightweight hardware-assisted micro AI enabled countermeasure against emerging malware and side-channels attacks for securing modern edge devices. The results of this research could help designers better realize and navigate the trade-offs between several design parameters offered by each learning algorithms to develop effective ML-based countermeasure against emerging cyber-attacks such as malware and side-channel attacks for modern edge devices, especially in resource-constrained systems. From Table I, we can conclude that our proposed work conducts an effective feature evaluation to analyze and select the most prominent HPC features for malware and microarchitectural side-channel attack detection. And it further performs a comprehensive analysis of various ML models for enabling an efficient and accurate micro AI-based solution to enhance the security of edge devices against emerging attacks. Additionally, our work implements the ML-based detectors in both software and hardware levels to present the suitability of our proposed micro AI for edge devices in terms of accuracy, latency, power, and area overheads. The main contributions of our work is summarized as follows:

- We profiled a large number of applications and built an extensive database of HPC samples collected from benign, android malware, and microarchitectural side-channel attacks executed. For this purpose, two widely used edge computing processors are used, including Google Pixel 4 with Arm processor for malware and Intel Core-I5 for side-channel attacks experiments.
- To eliminate the impact of limited HPCs in modern edge processors, the importance of HPC features is evaluated using an effective feature selection method and the most important HPC features are identified for effective malware and SCAs detection.
- We further explore the HPCs monitoring overhead when microarchitectural features are sampled at different intervals to determine the appropriate sampling interval for malware and SCAs detection.
- For a thorough analysis, various types of ML classifiers are implemented and precisely compared across different evaluation metrics including detection accuracy, F-measure, ROC curve analysis, computational latency, and hardware overhead to determine the most accurate and cost-efficient ML classifiers to enable on-device micro AI countermeasures for detecting signature of emerging cyber-attacks including malware and SCAs and securing modern edge devices at hardware level.

The remainder of this paper is organized as follows. The motivation and background are briefly described in section II. The proposed hardware-based malware detection framework and experimental setup details are discussed in Section III. Section IV presents the experimental results and provides a comprehensive analysis of different malware detectors across various metrics. Then, we present the state-of-the-art works on hardware-assisted malware detection (HMD) and SCAs detection in section V. Finally, Section VI presents the conclusion of this study.

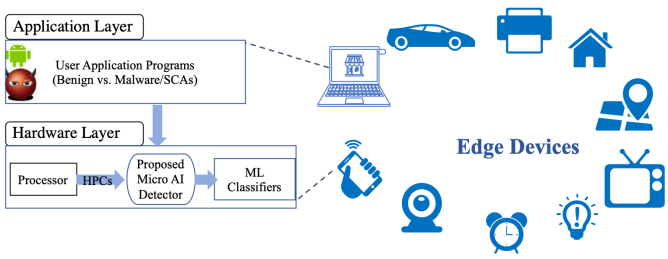


Fig. 1. Application of the proposed micro AI enabled countermeasure for securing edge devices at the hardware level.

II. Motivations and Background

A. Application of Proposed Micro AI Solution on Edge Devices

Figure 1 illustrates the applicability of the proposed micro AI countermeasure for securing edge devices at the hardware level against emerging malware and side-channel attacks. The proposed ML-based solution secures the system by estimating the type of running application (benign vs. attack) at run-time using limited HPC features collected at the hardware layer of edge processors. As shown in Figure 1, the network of embedded systems and IoT consists of a variety of edge devices, including smartphones, laptops, wearable devices, etc., that are connected via wired or wireless network [25]. Early edge devices are only in charge of collecting and sending data to a remote server for further analysis. With the development of hardware design and manufacturing technology, recent edge devices are also equipped with computation and analysis capability, enabling the computation near end users instead of sending data and workloads to central servers, which reduces network latency and response time.

Though benefits are brought by deploying more advanced edge devices, new security concerns are raised since more sensitive data, and critical applications reside in these devices. As shown, malware or SCAs can obfuscate as a user application and unconsciously execute users with benign applications. We design a micro AI detector that takes HPCs from processors as input features and sends them to ML classifiers to decide whether malware or SCAs is in the system to address the challenge. Prior detection approaches used in traditional computation platforms, like taint analysis [26], [27] and deep learning based detection [28], are not applicable for edge devices due to their high computation resource demand. Compared to traditional computation platforms, like desktops and servers, such edge devices are usually battery based or have limited energy sources due to the portability and low cost design consideration. This motivates the application of accurate and cost-efficient ML-based micro AI countermeasures for securing edge devices at the hardware level, which provides higher efficiency and less visibility to the potential exploit.

B. Emerging Cyber-Attacks in Edge Devices

In this work, we thoroughly analyze two emerging cyber-attacks, including malware and side-channel attacks (SCAs), that could compromise the security of edge devices at the hardware level.

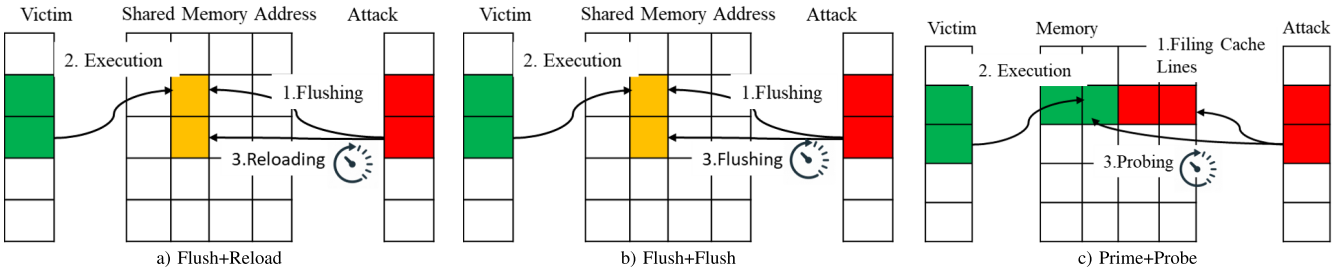


Fig. 2. Working principles of various emerging microarchitectural side-channel attacks, victim means benign applications that SCAs target to steal sensitive information from; Attack means SCAs.

1) Malware: Malicious software infections have plagued companies, organizations, and individual users for many years, and are significantly growing stealthier and increasing in numbers [29], [30]. Though varied in class and functionalities, malware primarily serves for harmful purposes such as providing a remote control for an attacker to use an infected machine, stealing sensitive information, unauthorized data access, destroying files, running intrusive programs on devices to perform Denial-of-Service (DoS) attack, and disrupting essential services to perform financial fraud. Malware authors spread malicious patterns and infect computing systems like devices and networks in a variety of ways. Their malicious programs can be delivered to a system physically through a USB drive or virtually via the Internet, often by automatically downloading malicious programs to users' systems without their approval or knowledge. Given the different malicious purposes and characteristics, malware can be classified into various categories.

2) 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 the exposure of new hardware vulnerabilities in the systems. This makes a unique opportunity for the attackers to exploit such vulnerabilities by deducing sensitive information, which results in microarchitectural side-channel attacks. We depict the flow of three important prominent SCAs in Figure 2. The "victim" means benign applications that SCAs target to steal sensitive information from, and "attack" refers to the side-channel attacks trying to disclose confidential information. In addition, the blocks mean the memory allocations of victims and attacks applications. The details of the three attacks are introduced following.

a) Flush+Reload: The researches in [31], [32] exploits the vulnerability of page de-duplication technique by monitoring the memory access lines in the shared pages. As demonstrated in Figure 2-(a), this attack targets the Last-Level Cache in the CPU and flushes out victim applications' data in the cache firstly and then waits for the victim application to execute. Lastly, the attacker tries to access the data after flushing the cache and measures the accessing time (latency). Shorter accessing time denotes that the victim application has accessed the data; otherwise, it has not been accessed.

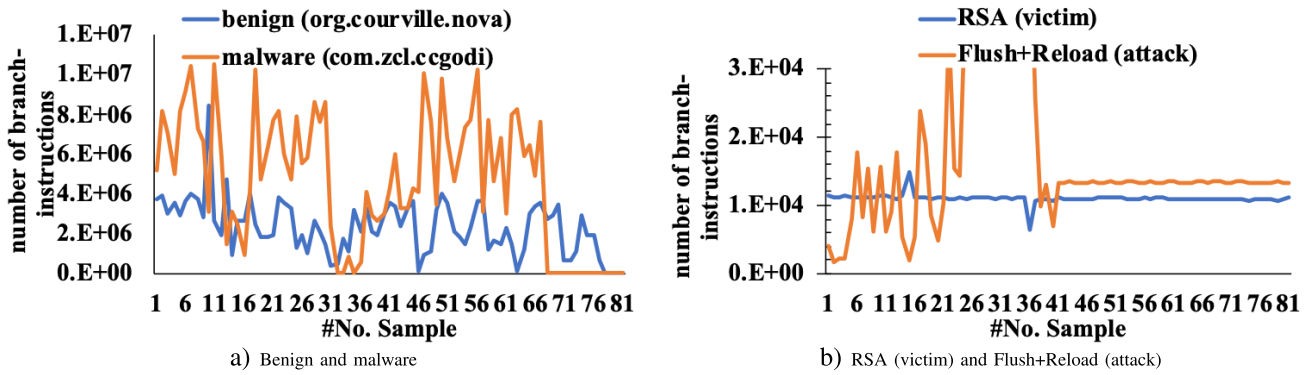
b) Flush+Flush: As shown in Figure 2-(b), Flush+Flush relies on the execution time of the flush instruction [14]. Unlike prior attacks, Flush-Flush does not make any memory accesses, nor does it rely on the data's access latency. The execution time of flush instruction depends on whether the data is 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 indicates that the corresponding data was brought to the cache and later accessed by the victim application.

c) Prime+Probe: Without the memory de-duplication restriction, Prime+Probe [13] could be applied to more systems. As presented in Figure 2-(c), it consists of two different stages: Prime and Probe. In the Prime stage, the attacker fills the cache line with memory blocks, i.e., "Priming" which can cause 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 ("Probing"). If the accessing time is long enough, it means the victim application has accessed the data; otherwise, the victim application does not access it.

C. Hardware Performance Counter Registers

Hardware Performance Counters (HPCs) are a set of special-purpose registers to record the occurrences of hardware events, including instructions, branches executed, and cache misses. They have been extensively used to predict the power estimation [33], performance tuning [34], debugging [35], and energy efficiency of computing systems [36]. They also help to enhance the systems' security by providing microarchitectural information of malware, side-channel attacks, and building detectors based on the events' information [3], [37], [38]. Both ARM and Intel x86 architecture provide the performance monitoring unit (PMU) interface to access HPCs. Since the Pentium Intel processors enable the PMU feature, the ARMv6 is the first PMU-enabled ARM processor and the afterward ARM11, Cortex-R, and Cortex-A cores also provide the functionality [23], [39]. Linux-based Perf [40], PAPI [41], and Intel VTune [42] are provided as tools to collect HPCs PCs or severs while Simpleperf [43] developed by Google Android team is for Android platform.

We present the number of $branch - instruction$ for each sampling data of benign and malware, and victim and SCAs in Figure 3 where sampling rate is 1000 per second, to further demonstrate the potential of leveraging hardware performance counters to enable HPCs-based micro AI for securing edge



a) Benign and malware

b) RSA (victim) and Flush+Reload (attack)

Fig. 3. Branch-instruction HPC traces between benign(victim) and attacks.

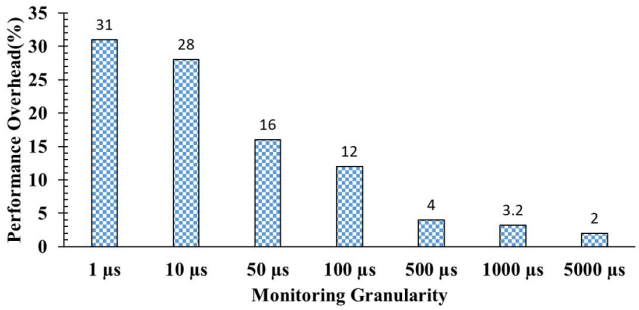


Fig. 4. Performance overhead with various monitoring granularity.

devices. Figure 3-(a) shows branch-instruction traces of malware and benign, indicating that the malware has higher branch-instruction than the benign application with totally a different and distinguishable trend. Similar observation can be found in Figure 3-(b) where Flush+Reload SCA shows higher branch-instruction events than RSA victim application. Both Figures 3-(a) and (b) demonstrate that there exists a clear difference between the microarchitectural behavior of benign and malware, and RSA (victim) and Flush+Reload (SCAs). This observation highlights the suitability of using HPCs data to distinguish the trace of malicious software from benign programs by applying effective machine learning algorithms.

D. HPCs Monitoring Overhead

For minimizing the overhead incurred by HPCs monitoring, we investigate the relation between sampling rate and overhead. As shown in Figure 4, 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 granularities. Execution time under no HPCs monitoring is used to obtain performance overhead percentage. It is observed that, generally, the smaller the monitoring granularity, the larger the performance overhead. When the monitoring scale is 1 μs, performance overhead is at its highest value reaching 30%. Due to the significant difference in monitoring overheads, it is vital to determine a proper level of monitoring granularity to balance the detection performance

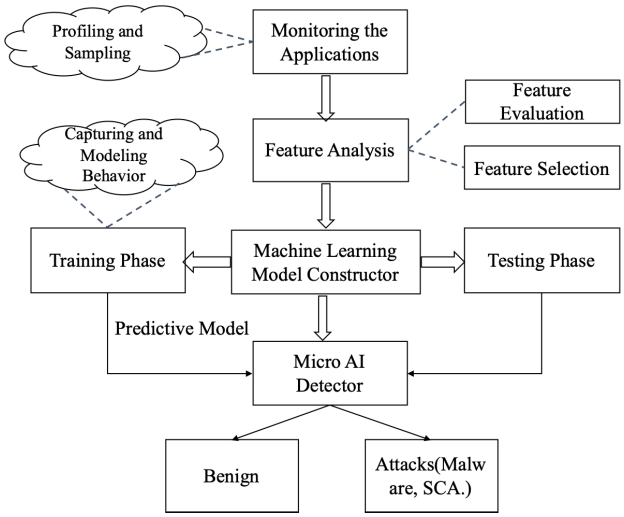


Fig. 5. Overview of the proposed machine learning-based countermeasure for edge devices.

and HPCs monitoring costs. In this work, we choose one millisecond as the sampling interval to ensure overhead is around 3%.

III. Proposed Micro AI-Based Countermeasure Against Malware and Side-Channel Attacks

In this section, we first present the ML-based countermeasures for protecting edge devices against malware and SCAs threats as shown in Figure 5. As established, the proposed approach is comprised of two main parts: off-line training and online deployment. For the training part: benign applications, malware, and SCAs are profiled by collecting HPC data with one millisecond sampling interval. And then, the HPCs data is used to train machine learning classifiers for building ML-based attacks detection models. A variety of classifiers are investigated to find the optimal one with high accuracy and low computation cost. The selected model will be deployed on edge devices for capturing attacks at run-time.

A. Data Collection

In this work, we leverage two platforms as examples of edge devices as shown in Table II. As introduced in Section 4, one millisecond is chosen as the sampling interval and 16 HPC

TABLE II
EXPERIMENT SETUP

Hardware	Processor	Operating system	Threats
Pixel 4	Qualcomm Snapdragon	Android 11.0	Malware
Desktop	4 cores	Ubuntu 16.0	SCAs

TABLE III
THE COLLECTED HPC FEATURES

HPC event	Description
branch-instructions	# branch instructions retired
branch-loads	# successful branches
branch-misses	# branches mispredicted
instructions	# instructions retired
bus-cycles	time to make a read/write between the cpu and memory
cache misses	# last level cache misses
cache-references	# last level cache references
l1-dcache-load-misses	# cache lines brought into L1 data cache
l1-dcache-loads	# retired memory load operations
l1-dcache-stores	# cache lines into L1 cache from DRAM
l1-icache-load-misses	# instruction misses in L1 instructions cache
llc store misses	# misses occurred in L3 cache during store operation
llc-load-misses	# cache lines brought into L3 cache from DRAM
llc-loads	# successful memory load operations in L3
iTLB loads	# load operations occurred in instruction TLB
iTLB-load-misses	# misses in instruction TLB during load operations

features are considered in this work for further analysis as listed in Table III. We execute each application (benign or malware) four times, and only 2 and 4 HPC events are selected to collect during each run from the tested ARM and Intel processors, respectively. In this way, we can monitor 16 HPC events after eight or four rounds of applications' execution. Each database introduced below is split into 70%-30% two parts where the 70% part of benign and malware/SCAs is used for training and the rest 30% is used for testing. The databases for Android and Linux are introduced in detail. We leverage Perf [40] and Simperperf [43] to collect HPCs for Linux and Android, respectively. Both of the tools can collect multiple events simultaneously, and the number of events to be collected concurrently depends on the design of architectures and the availability of HPC registers. For example, ARM Cortex-A5 processors provide two while ARM Cortex-A7 processors provide 4 HPC registers physically available on the processor chip to collect the hardware related events. We leverage these performance analysis tools to profile an extensive set of malicious (malware and side-channel attacks) and benign applications.

1) Android Database: As for the malware, we selected 100 Android malware randomly from VirusTotal [44] and 100 Linux malware. We downloaded the top 200 most popular free apps in Google Play by the end of Feb 2021 as our target programs to form benign applications. Then, we employ the apps with VirusTotal [44] to confirm if the apps are benign. Lastly, the top 100 apps which are confirmed as benign are selected as the benign apps dataset.

2) Linux Database: MiBench [45] benchmark suite is used to represent benign applications with 200 benign applications are selected. For SCAs, we use attacks from Mastik [31] including Flush+Reload, Flush+Flush, L1 Prime+Probe, and L3 Prime+Probe. And each type of SCAs is monitored 50 times to balance the benign and SCAs samples.

TABLE IV
THE COLLECTED HPC FEATURES AND THEIR RANKING

Ranking_Malware	HPC Name	Ranking_SCAs	HPC Name
1	L1-dcache-stores	1	L1-dcache-loads
2	branch-instructions	2	instruction
3	L1-dcache-loads	3	branch-instructions
4	iTLB-load	4	branch-misses
5	iTLB-load-misses	5	cache misses
6	L1-icache-load-misses	6	L1-dcache-store-misses
7	L1-dcache-store-misses	7	L1-dcache-load-misses
8	branch-misses	8	iTLB-load-misses

B. Feature Selection

Detecting malware and side-channel attacks at the hardware level using ML models requires representing programs at low-level features, leading to high-dimensional data processing and increasing computational overheads and complexity. Furthermore, incorporating irrelevant features would lead to lower accuracy and performance for the classifiers. Hence, it is crucial to perform an effective feature reduction of collected data to alleviate unnecessary computational overheads and determine the most prominent low-level features [46], [47]. To detect the attacks in real-time with minimal overhead, we intend to identify a minimal set of critical HPCs that are feasible to collect even on low-end processors with a small number of HPCs in a single run. Therefore, a subset of HPC features is selected, representing the most important features for classification. The selected features are then supplied to each ML-based attacks detector. The detector attempts to find a correlation between the feature values and the application behavior to predict the attacks.

Given the limited number of HPCs available in modern microprocessors (e.g., only 4 HPCs on tested Intel I5-3470) to be collected at one time simultaneously, it is necessary to identify the most important ones among all available HPCs listed in Table III for classifying the attacks and benign applications conditions for different types of attacks [5]. 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 (attacks and benign applications conditions). Correlation attribute evaluation algorithm calculates the Pearson correlation coefficient between each attribute and class, as given below:

$$\rho(i) = \frac{cov(Z_i, C)}{\sqrt{var(Z_i)\ var(C)}} \qquad i = 1, \ldots, 16 \tag{1}$$

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. "malware/SCAs" or "benign/victmi" 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. After the evaluation and reduction, the top eight HPCs for malware and SCAs detection are listed in Table IV.

C. Training and Testing ML-Based Micro AI Classifiers

Table V describes the ML classifiers evaluated in this work that is selected from seven different categories. These ML

TABLE V
EVALUATED ML CLASSIFIERS FOR ATTACKS DETECTION

ML Category	Notation	Selected Classifier
Bayesian Network	Algorithms that use Bayes Theorem in some core way, like Naive Bayes.	NaiveBayes
Neural Network	Series of algorithms that attempt to recognize and mimic the human brain operations.	MultilayerPerceptron (MLP)
Support Vector Machine	Linear model for classification and regression problems.	SGD
Rules	Algorithms that use rules, like One Rule.	JRiP, OneR
Trees	Algorithms that use decision trees, like Random Forest.	J48

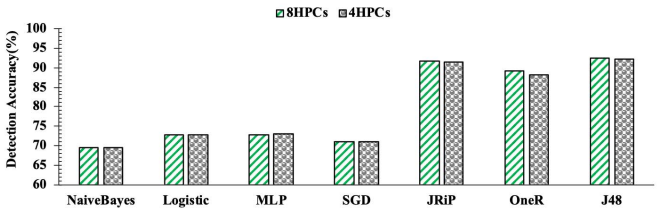


Fig. 6. **Malware** detection accuracy of different classifiers with various HPCs.

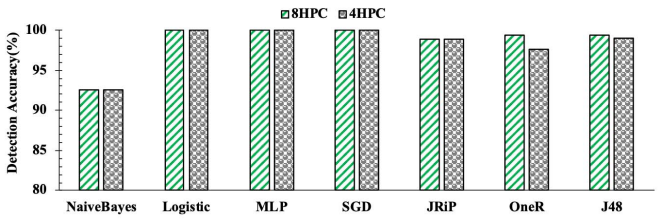


Fig. 7. **SCAs** detection accuracy of different classifiers with various HPCs.

classifiers include NaiveBayes, Logistic, MultiLayerPerceptron (MLP), SGD, JRip, OneR, J48. The rationale for selecting these machine learning models is that they are from different branches of ML, including Bayesian network-based, neural network, support vector machine, lazy learning-based, rule-based, and tree-based techniques covering a diverse range of learning algorithms which are inclusive of modeling both linear and nonlinear problems. In addition, the prediction model produced by these learning algorithms can be a binary classification model which is compatible with the attacks detection problem in our work. Furthermore, the Weka data mining tool is deployed for implementing the machine learning classifiers. A standard 70%-30% dataset split for training and testing is followed to validate each of the utilized ML classifiers. Next, 70% of the randomized data is used for training the classifiers for the percentage split testing, and the rest of 30% is used for testing evaluation.

IV. EXPERIMENTAL RESULTS AND EVALUATION

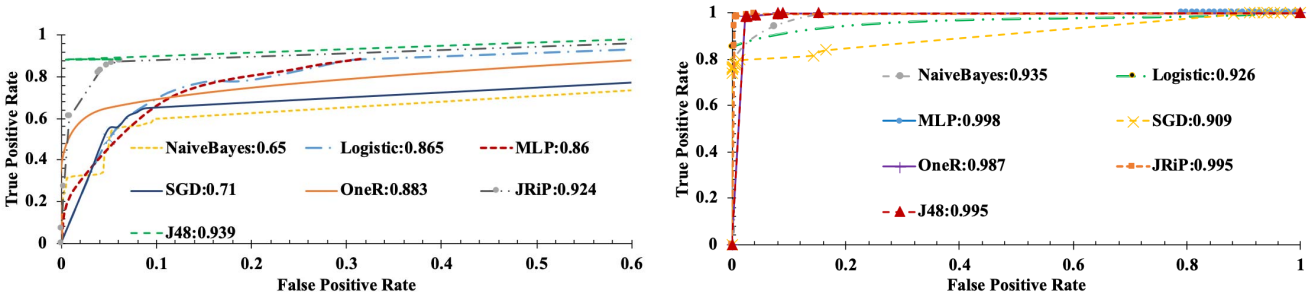
To comprehensively analyze the effectiveness of ML-based micro AI solutions for securing edge devices against emerging cyber-attacks, in this section, we present the evaluation and comparison results across different evaluation metrics, including detection accuracy, false positive rate, F-measure (F-score), robustness and efficiency, hardware overhead, and cost.

A. Detection Accuracy

Detection accuracy in machine learning-based security countermeasures is defined as the percentage of correctly classified samples. To select the most effective classification model and evaluate the influence of different number of HPCs used (8 vs. 4 features), in Figure 6 and Figure 7, we demonstrate the observed detection accuracy against malware and SCAs respectively across seven implemented ML classifiers trained and tested with 8 and 4 HPCs. In particular, we focus on analyzing the effectiveness of proposed ML-based micro AI countermeasures by considering 8 and 4 HPC features, since most modern microprocessors deployed in edge platforms are equipped with 8 to 4 HPC registers physically available on the chip to monitor the applications' low-level behavior. It can be observed that reducing the number of HPC features from 8 to 4 has a relatively minor impact on the accuracy of most ML classifiers investigated for both malware and SCAs detection tasks. Therefore, to accordingly address the challenge of accurate run-time attacks detection, we adopt the 4 most prominent HPC features to make the proposed micro AI model applicable to most resource-limited edge platform architectures for both malware and SCAs detection. As the results show, JRip and J48 models are the two classifiers achieving higher detection accuracy $> 90\%$ for malware and benign classification. By comparison, all classifiers can achieve a high accuracy $> 90\%$ except NaiveBayes, highlighting that due to the higher complexity and variance, malware attacks require even a more careful analysis and selection of suitable HPCs and micro AI model to accurately distinguish the malicious patterns from normal traces. We also observe that SCAs detection accuracy are higher than malware detection accuracy with the same number HPCs and classification mode. Based on prior literature [3], [6], we can conclude that detecting malware is generally a more difficult task than microarchitectural side-channel attacks, where [6] for SCAs gives close 100% accuracy while malware gives 85% accuracy in [3]. Malicious software attacks have continued to evolve in quantity and sophistication during the past decade. Due to the ever-increasing complexity of malware attacks and the financial motivations of attackers, malware trends are even recently shifting towards stealthy attacks by embedding the malicious code inside the benign application for harmful purposes that could simply bypass the standard detection mechanisms.

B. False Positive Rate

Table VI presents the false positive rate of all ML classifiers investigated in this work for malware and SCAs detection with 4 HPC features. For malware detection, we observe that



a) malware detection b) SCAs detection

Fig. 8. ROC Curve and AUC values comparison across various classifiers.

TABLE VI
FALSE POSITIVE RATE OF VARIOUS CLASSIFIERS WITH 4 HPCS

Attacks	NaiveBayes	Logistic	MLP	SGD	JRiP	OneR	J48
Malware(%)	56.3	49.1	48.8	53	13.1	16.1	12.7
SCAs	5.6	14.9	5.3	23.8	0.9	1.3	1.3

TABLE VII
F-MEASURE OF VARIOUS CLASSIFIERS WITH 4 HPCS

Attacks	NaiveBayes	Logistic	MLP	SGD	JRiP	OneR	J48
Malware	0.685	0.696	0.698	0.671	0.91	0.879	0.917
SCAs	0.929	0.919	0.97	0.876	0.989	0.981	0.987

JRiP, OneR, and J48 achieve lower FPR than the rest four classifiers with J48 delivering the lowest false positive rate of 12.7%. In addition, the proposed SCAs detection using various classifiers gives a much lower false positive rate as compared to malware detection where JRiP, OneR, and J48 achieve low false positive rates, 0.9%, 1.3% and 1.3%, respectively.

C. F-Measure

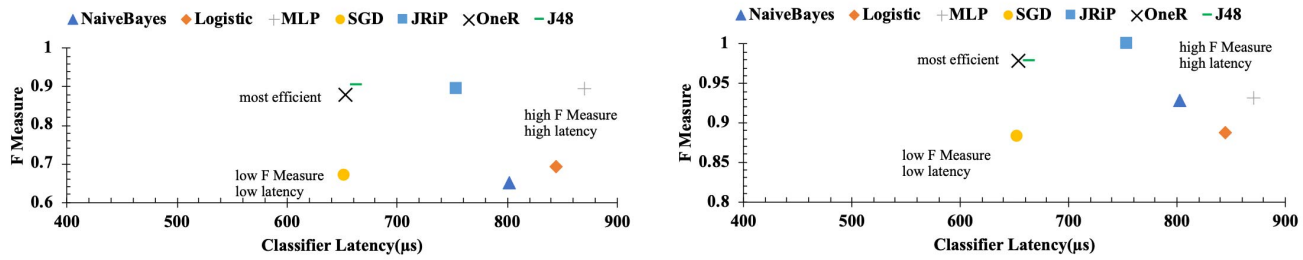
F-measure (F-score) is interpreted as a weighted average of the precision (p) and recall (r) which is formulated as $\frac{2\times(p\times r)}{p+r}$. The precision is the proportion of the sum of true positives versus the sum of positive instances, and the recall is the proportion of instances that are predicted positive of all the positive instances. F-measure can be considered as a more comprehensive evaluation metric over accuracy (percentage of correctly classified samples) since it accounts for both the precision and the recall values. More importantly, F-measure is also resilient to class imbalance in the dataset, which is the case in our experiments. Table VII presents the F-measure results of all implemented ML classifiers for malware and SCAs detection based on the top 4 HPC features. As seen, for malware detection, J48 model gives the highest F-measure value, 0.917, followed by JRiP, having a similar trend as accuracy. For the SCAs part, JRip gives the highest F-measure with 0.989 followed by J48. Hence, it can be concluded that J48 and JRiP are the two classifiers capable of achieving a high F-measure for both malware and SCAs detection.

D. Area Under the ROC Curve (AUC)

Receiver Operating Characteristics (ROC) Curve plots the fraction of true positives rate versus the fraction of false positives for a binary classifier. The best possible classifier would thus yield a point in the upper left corner or coordinate (0,1) of the ROC space, representing 0% false positives and 100% true positives. The Area under the ROC Curve (AUC) metric corresponds to the probability of correctly identifying "under attack" and "under no attack" and robustness is referred to how well the classifier distinguishes between the two classes, for all possible threshold values. Higher AUC value represents better robustness for the ML-based micro AI countermeasure. Due to space limitation and given that NaiveBayes, Logistic, MLP, and SGD have resulted in lower detection accuracy for malware detection, their ROC curve and AUC values are not presented in Figure 8-(a). Similar to the F-measure rate, we notice in Figure 8-(a) that the OneR algorithm performs the worst in terms of ROC Curve for malware detection having the biggest distance to point (0,1) among the three other ML models. In comparison, the J48 classifier is closer to the coordinate (0,1) with a 0.939 AUC value, indicating a higher true positive rate and less false positive rate compared with the other seven ML classifiers evaluated in this work. Moreover, for SCAs detection, the ROC curve and AUC values of all seven classifiers are presented in Figure 8-(b). The results demonstrate that most classifiers with 4 HPCs are able to give a high AUC value and among them, JRiP, MLP, and J48, can yield above 0.99 AUC value.

E. Software Implementation and Efficiency Analysis

The software implementation and efficiency analysis are discussed in the section. As presented in Table VIII, the seven classifiers discussed in prior sections are implemented at the Linux kernel level on the Intel I5 processor to evaluate the incurred software overheads. The overhead includes the time spent on reading the HPCs and the time spent on executing the classifiers. Since Intel processors are equipped with Turbo Boost technology, time measurement may suffer from time measurement error. Hence, we disabled Turbo Boost and set CPU to governor mode with 1200MHz operating frequency to avoid possible measurement errors. Performance counter reading overhead is negligible in kernel space when monitoring a single core but overall system overhead increases for monitoring processes running on multiple cores, which may go up to as high as 20% depending on the frequency of events that are being sampled. The results for software implementation overhead of these classifiers show them to be



a) malware detection

b) SCAs detection

Fig. 9. Efficiency comparison across various classifiers with 4 HPCs.

TABLE VIII
ML CLASSIFIER EXECUTION OVERHEAD

Classifier	Latency(ms)	Classifier	Latency(ms)
SGD	0.652	NaiveBayes	0.802
OneR	0.653	MLP	0.87
Logistic	0.844	JRIP	0.653
J48	0.663		

slow with the execution time in the range of milliseconds, which is the order of magnitude higher than the latency needed to capture malware at run-time. It is important to note that several studied malware have execution time in the range of milliseconds or less, requiring fast detection to prevent them from corrupting the system.

Furthermore, we demonstrate the efficiency, i.e., F-measure vs. latency, of all the seven explored classifiers against malware and SCAs in Figure 9-(a) and Figure (b). A classifier with a higher ratio is considered a more efficient detector than the classifier with a lower ratio. As shown in Figure 9-(a) and (b), a clear trade-off is seen between detection rate and latency achievable for hardware-assisted malware and SCAs detection. For malware, J48 and OneR give high F-measure with 0.917 and 0.879 respectively and incur the least computation overhead compared to others. Though JRiP giving higher F-measure than OneR, it incurs over 100 *us* computation overhead. Similarly, ML classifiers such as MLP for SCAs achieve a high F-measure rate and higher computational overhead. The techniques such as SGD and Logistic show relatively more minor timing costs with low SCAs F-measure. While J48 and OneR having high F-measure with above 0.98 F-measure are more suitable for highly resource-constrained embedded systems due to the smallest computational overhead. Clearly, the results show trade-offs between F-measure and latency. Therefore, it is crucial to compare ML classifiers for effective SCAs detection by considering all these parameters.

F. Hardware Implementation

As discussed, the software implementation of ML classifiers for malware and side-channel attacks detection is slow in the range of hundreds of microseconds which is an order of magnitude higher than the latency required to capture the malware at run-time, making it less a practical and effective solution for effective run-time detection of attacks. In this work, we implement the ML models on an FPGA to present the reduced latency and higher efficiency of hardware implementations compared to the software implementations. FPGA is a target in our study, as few modern microprocessors have provided access to on-chip FPGA units for programmable logic implementation. In recent years, SoC FPGA has been widely studied in both industry and academia [48], [49], which integrates both CPU processor and FPGA into one device. Such arrangement makes it feasible to implement reprogrammable low-level attack (e.g., malware and side-channel attacks) detection logic of the adopted ML models which can detect the malicious pattern by reading the CPU hardware performance counters through the shared memory bus. Under such design, the communication latency is relatively minimal as compared to the execution time of detecting malicious and benign programs and can be ignored. Hence, the total latency considered in our work is the time spent for ML-based detectors on the target FPGA for detecting the attacks and the communication latency was considered negligible as compared to hardware computational latency of the ML-based detectors. To address this challenge, in this work, we develop a hardware implementation of the adopted machine learning classifiers to analyze the efficiency of micro AI solutions for securing edge devices. To this aim, we use Xilinx Vivado Design suite to synthesize ML classifiers for Xilinx Virtex 7 FPGA. Latency and power estimation are collected at 10ns clock cycle time. The accuracy of ML classifiers is based on data collected at 10ms intervals using perf.

Therefore, when it comes to adopting effective AI/ML models for hardware implementation in edge devices, the accuracy of an algorithm is not the only parameter in decision-making. This motivates the analysis of effective micro AI solutions for on-chip intelligence. In these methods, the design area and response time (latency) overheads of the ML classifiers also play a crucial role in selecting the cost-efficient hardware solution, particularly in edge devices considered emerging resource-limited computing systems. While complex algorithms such as MLP and Logistic could provide higher detection accuracy, they would add considerable hardware overhead and implementation cost to the design. In addition, given their complexity, they can be slow in detecting malware. All these factors would make such heavyweight ML models less suitable for micro AI-based countermeasures for on-chip intelligence in edge devices.

Table IX presents the hardware implementation costs for various ML classifiers used for hardware-assisted malware and side-channel attacks using 8 and 4 most important HPCs.

TABLE IX
HARDWARE IMPLEMENTATION RESULTS OF ML-BASED MICRO AI COUNTERMEASURES

Number of Features	8 HPC			4 HPC		
Classifier	Latency(cycles@10ns)	Power(W)	Area(LUTs+FFs+DSPs)	Latency(cycles@10ns)	Power(W)	Area(LUTs+FFs+DSPs)
NaiveBayes	233	1.34	58177	90	1.12	56633
Logistic	68	0.63	13041	59	0.54	11815
JRIP	4	0.436	1504	2	0.35	156
J48	9	0.436	1801	3	0.34	584
SGD	34	0.444	2556	22	0.36	2466
OneR	1	0.324	1258	1	0.18	292
MLP	302	1.03	36252	102	0.84	25667

The latency unit is represented by the number of clock cycles (cycles @10 ns) required to classify the input vector. The unit for power consumption is Watt.

As seen, the NaiveBayes and MLP models, as expected, result in a significant area, power consumption, and latency overhead, as compared to other ML methods across both 8 and 4 HPC-based detectors. Moreover, JRip classifier delivers the most efficient implementation costs as compared to other algorithms. This highlights the cost-efficiency of JRip algorithm, particularly using 4 HPC features, for building an efficient micro AI countermeasure and on-device intelligence against emerging malware and SCAs. Clearly, the results show some trade-offs between accuracy, latency, and area overhead. Therefore, it is important to compare classifiers by taking all of these parameters into consideration.

V. RELATED WORK

A. ML for Hardware-Assisted Malware Detection

Demme *et al.* [19] was the first study to examine the effectiveness of hardware performance counter information for accurate malware detection. The authors proposed the idea of using hardware performance counter data to accurately detect malicious behavior patterns using machine learning techniques primarily on mobile operating systems such as Android. The paper successfully demonstrated the effectiveness of offline machine learning algorithms in identifying malicious software. In addition, it illustrated the suitability of employing HPC information in detecting malware at the Linux OS level such as Linux rootkits and cache side-channel attacks on Intel and ARM processors. It exhibited high detection performance results for Android malware by applying complex ML algorithms, namely Artificial Neural Network (ANN) and K-Nearest Neighbor (KNN).

In a different study, Tang *et al.* [1] further discussed the feasibility of unsupervised learning that employs low-level HPCs features for detecting return-oriented programming (ROP) and buffer overflow attacks by finding anomalies in hardware performance counter information. For feature selection, this study used the Fisher Score metric to identify the top 7 low level features for malware detection. These reduced features are then used to train unsupervised machine learning methods for detecting deviations in program behavior that occur due to a potential malicious attack. The work further provides a comparison of performance using different sampling frequencies of the HPCs. Though an effective detection model provided by the work, the work did not analyze the latency and hardware implementation analysis of the adopted method. However, the limited resources of edge devices demand lower cost as well efficiency.

The work in [2] used sub-semantic features to detect malware using Logistic Regression (LR) and ANN algorithms. Moreover, they suggested changes in microprocessor pipeline to detect malware in truly real-time nature which increases the overhead and complexity. They explored sub-semantic features in the low-level features space by evaluating three types of features including 1) features based on executed instructions that include frequency of instruction categories, frequency of opcodes with the largest difference, existence of categories, and existence of opcodes 2) features based on memory address patterns consist of frequency of memory address distance histogram and memory address distance histogram mix, and 3) features based on architectural events that include frequency of memory read/writes, taken and immediate branches, and unaligned memory accesses. The HMD study in [3] used the same feature set, applied logistic regression to classify malware into different types, and trained a specialized classifier to detect each class. They further used specialized ensemble learning to improve the accuracy of malware detectors.

Singh *et al.* [4] is a recent work on HMD that deploys machine learning algorithms applied on synthetic traces of HPC features for detection of kernel-level rootkit attacks. For feature reduction, they process the application traces using the Gain Ratio feature selection technique from the WEKA machine learning toolkit to determine which features are the most prominent for each rootkit. The authors achieve high prediction accuracy in detecting five self-developed synthetic rootkits models. Nevertheless, while important, this work only focused on detecting kernel rootkit attacks using a limited set of synthesis datasets.

The research in [5] proposed ensemble learning techniques to facilitate run-time hardware-assisted malware detection and improved the performance of HMD by accounting for the impact of reducing the number of HPC features on the performance of malware detectors. In addition, recent work in [46] proposed a two-stage machine learning-based approach for run-time malware detection in which in the first level classifies applications using a multiclass classification technique into either benign or one of the malware classes (Virus, Rootkit, Backdoor, and Trojan). In the second level, to have a high detection performance, the authors deploy a machine learning model that works best for each class of malware and further apply effective ensemble learning to enhance malware detection performance.

B. ML for Microarchitectural Side-Channel Attack Detection

Chiappetta *et al.* [22] uses HPCs and then apply and compare three different detecting attacks: finding a correlation between victims and attacks; building supervised machine learning models based on HPCs from victims and attacks; detecting anomalies by validating attack HPCs as samples and other processes as outliers. This work can achieve real-time cache-based SCAs. This work takes Flush+Reload as an example and evaluates the detection effectiveness in terms of F Score, confidence.

Unlike the prior work, CloudRadar [6] proposes a detection system for a cloud-based environment targeting Flush+Reload, Prime+Probe. It first monitors the HPCs of cryptographic applications and trains a dynamic time warping machine learning model which identifies if an application is a cryptographic application under victim virtual machine. If it is, the HPCs from untrusted VM would be collected and correlated with the protected VM's sensitive operations. According to experimental results, the proposed detection system shows a high true positive (100%) and close to 0 false positive when window size increases to 5, meaning 5.1 milliseconds.

In addition, the work in [50] proposes a multi-layer detection approach targeting identifying the existence of the abnormal process (the attack) and outputting the possible type (like cache-based or branch prediction based) of side-channel-attack for the abnormal process. To this aim, it profiles processes with low-level hardware events using Linux-based perf event API. It then analyzes the HPCs with One-Class Support Vector Machine (OC-SVM) to identify if the process is normal or abnormal. Once it is identified as abnormal, the HPCs would be sent to the pre-trained classifiers (Adaboost, Random Forest, Naive Bayes and SVM) and gives the category of the process, like cache-based, branch-based, etc. The detection system correlates the anomaly process and the encryption applications with the Fast Dynamic Time Warping (fast-DTW) algorithm. If the similarity is larger than a threshold, the process will finally be identified as a side-channel attack with a specific type.

The work in [51] adopts an unsupervised learning anomaly detection (Gaussian anomaly detection) to detect potential malicious VMs with side-channel attacks in IaaS (infrastructure as a service) platform. The HPCs monitoring part leverages Intel Cache Monitoring Technology (CMT) to profile the microarchitectural behaviors. It employs Prime+Probe as a case study, and the effectiveness is evaluated in terms of accuracy, F score, and roc curve. The experimental results show that the proposed detection works very well when system does not have a compute-intensive workload (CIW) while it gives false positives when CIW is present.

To cope with the newly side-channel attack of Spectre [52], the work in [9] trains a neural network with the hardware performance counters to identify a Spectre attack. The whole dataset is split as 90%-10% for training and testing respectively, and the experiment shows over 99% accuracy and 0.972 F score. The high detection accuracy indicates that the cache patterns of Spectre are very distinctive. Compared to prior works, this work also evaluates the performance overhead caused by HPCs monitoring and finds that the used sampling rate 100 milliseconds in this work only causes 4-5% performance overhead of the machine it was running on. One drawback of this work is that it only uses one Spectre variant while different side-channel attacks have very different cache patterns. Since the trained detection model has not met such a pattern before, the proposed detection might not be able to detect.

Compared to prior works profiling both victim and attack applications, [7], [53] monitors the hardware performance counters of victim application only (like RSA and AES) under SCAs and normal scenario. Two state-of-the-art SCAs (Flush+Reload and Flush+Flush) are included in this work. PAPI-based HPCs monitoring tool is leveraged and roughly 1 million samples are collected where a K-fold cross-validation is conducted to evaluate the effectiveness of the proposed detection system. To simulate the realistic workload of computer systems, accuracy for no load, average load, and full load is investigated with three different types of classifiers: Linear Discriminant Analysis (LDA), Logistic Regression (LR) and Support Vector Machine (SVM). The experiment shows that the detection accuracy for Flush+Reload is 99.51%, 99.50% and 99.44% under no, average and full load conditions respectively within 1% completion of a RSA encryption operation. For Flush+Flush, the detection accuracy is 99.97%, 98.74% and 95.20%detection accuracy for no load, average load, and full load conditions, respectively, at the cost of 12.5% completion of an attack on AES encryption round.

Similarly, CacheSheild [8] also profiles victim under attacks, under normal conditions to identify the existence of SCAs. While, this work adopts a change point detection algorithm (CPD) which is mostly used to construct the commonly known quick detection models. Three classical side-channel attacks: Flush+Reload, Flush+Flush, and Prime+Probe are the detection target of this work. L3_Misses is selected based on the understanding of SCAs, and evaluation is conducted on both KVM and VMware platforms. To investigate the influence of other running workloads, Yahoo Cloud Serving Benchmark, Video Streaming, and Randmem Benchmark are leveraged. The detection speed ranges from 4 ms to 13 ms, while the false positive rate ranges from 0.1% (AES with VMware) to 24.4% (RSA with KVM running together with Randmem Benchmark).

VI. Conclusion

The ever-increasing complexity of modern computing systems especially embedded systems and Internet-of-Things (IoT) devices, has led to the growth of security vulnerabilities, making such systems appealing targets for increasingly sophisticated cyber-attacks. Recent microarchitectural security countermeasures by employing applications' low-level features collected from Hardware Performance Counters (HPCs) registers have emerged as a promising solution to address the inefficiency of traditional software-based methods. This work proposes an accurate and cost-efficient micro AI enabled countermeasure for securing emerging edge devices against emerging malware and side-channels attacks using processors' HPCs data. To this aim, we comprehensively explore the suitability of applying various types of machine learning

classifiers for hardware-assisted malware and side-channel attacks detection by precisely comparing them in terms of detection accuracy, F-measure, AUC metric, latency, power consumption, and hardware overheads. We examined the proposed ML-based micro AI countermeasure on two different widely used processors in edge computing (Intel and Arm). Given the implementation cost of on-chip HPCs and their limited availability and accuracy, the results of this research will help the designers in making effective architectural decisions on the number and types of HPCs needed to implement in future architectures and to better realize the challenges of leveraging micro AI solutions at the hardware level to most effectively improve the performance of ML classifiers for detecting the malicious software.

REFERENCES

[1] A. Tang, S. Sethumadhavan, and S. J. Stolfo, "Unsupervised anomaly-based malware detection using hardware features," in *Proc. RAID*. Cham, Switzerland: Springer, 2014, pp. 109–129.

[2] M. Ozsoy, C. Donovick, I. Gorelik, N. Abu-Ghazaleh, and D. Ponomarev, "Malware-aware processors: A framework for efficient online malware detection," in *Proc. IEEE 21st Int. Symp. High Perform. Comput. Archit. (HPCA)*, Feb. 2015, pp. 651–661.

[3] K. N. Khasawneh, M. Ozsoy, C. Donovick, N. Abu-Ghazaleh, and D. Ponomarev, "Ensemble learning for low-level hardware-supported malware detection," in *Proc. Int. Symp. Recent Adv. Intrusion Detection*. Cham, Switzerland: Springer, 2015, pp. 3–25.

[4] B. Singh, D. Evtyushkin, J. Elwell, R. Riley, and I. Cervesato, "On the detection of kernel-level rootkits using hardware performance counters," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, 2017, pp. 483–493.

[5] H. Sayadi, N. Patel, P. D. S. Manoj, A. Sasan, S. Rafatirad, and H. Homayoun, "Ensemble learning for effective run-time hardware-based malware detection: A comprehensive analysis and classification," in *Proc. 55th ACM/ESDA/IEEE Design Automat. Conf. (DAC)*, Jun. 2018, pp. 1–6.

[6] T. Zhang, Y. Zhang, and R. B. Lee, "CloudRadar: A real-time side-channel attack detection system in clouds," in *Proc. Int. Symp. Res. Attacks, Intrusions, Defenses*. Cham, Switzerland: Springer, 2016, pp. 118–140.

[7] 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 *Proc. 7th Int. Workshop Hardw. Archit. Support Secur. Privacy*, Jun. 2018, pp. 1–8.

[8] S. Briongos, G. Irazoqui, P. Malagón, and T. Eisenbarth, "CacheShield: Detecting cache attacks through self-observation," in *Proc. 8th ACM Conf. Data Appl. Secur. Privacy*, Mar. 2018, pp. 224–235.

[9] J. Depoix and P. Altmeyer, "Detecting spectre attacks by identifying cache side-channel attacks using machine learning," *Adv. Microkernel Oper. Syst.*, vol. 75, pp. 1–11, Aug. 2018.

[10] *Smartphone Market Share*. Accessed: Jul. 29, 2021. [Online]. Available: https://www.idc.com/promo/smartphone-market-share/os

[11] (Apr. 2021). *Mcafee Labs Threats Report*. [Online]. Available: https://www.mcafee.com/enterprise/en-us/assets/reports/rp-quarterly-threats-Apr.-2021.pdf

[12] Y. Yarom and K. Falkner, "FLUSH+RELOAD: A high resolution, low noise, L3 cache side-channel attack," in *Proc. USENIX Secur. Symp.*, vol. 1, 2014, pp. 719–732.

[13] F. Liu, Y. Yarom, Q. Ge, G. Heiser, and R. B. Lee, "Last-level cache side-channel attacks are practical," in *Proc. IEEE Symp. Secur. Privacy*, May 2015, pp. 605–622.

[14] D. Gruss, C. Maurice, K. Wagner, and S. Mangard, "Flush+Flush: A fast and stealthy cache attack," in *Proc. Int. Conf. Detection Intrusions Malware, Vulnerability Assessment*. Cham, Switzerland: Springer, 2016, pp. 279–299.

[15] Y. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Cross-tenant side-channel attacks in PaaS clouds," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2014, pp. 990–1003.

[16] G. Irazoqui, T. Eisenbarth, and B. Sunar, "S$A: A shared cache attack that works across cores and defies VM sandboxing- and its application to AES," in *Proc. IEEE Symp. Secur. Privacy*. May 2015, pp. 591–604.

[17] G. Jacob, H. Debar, and E. Filiol, "Behavioral detection of malware: From a survey towards an established taxonomy," *J. Comput. Virol.*, vol. 4, no. 3, pp. 251–266, Aug. 2008.

[18] M. Miettinen, S. Marchal, I. Hafeez, N. Asokan, A.-R. Sadeghi, and S. Tarkoma, "IoT SENTINEL: Automated device-type identification for security enforcement in IoT," in *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jun. 2017, pp. 2177–2184.

[19] J. Demme *et al.*, "On the feasibility of online malware detection with performance counters," *ACM SIGARCH Comput. Archit. News*, vol. 41, no. 3, pp. 559–570, 2013.

[20] A. Mosenia and N. K. Jha, "A comprehensive study of security of Internet-of-Things," *IEEE Trans. Emerg. Topics Comput.*, vol. 5, no. 4, pp. 586–602, Dec. 2017.

[21] A. Mudgerikar, P. Sharma, and E. Bertino, "Edge-based intrusion detection for IoT devices," *ACM Trans. Manage. Inf. Syst.*, vol. 11, no. 4, pp. 1–21, Dec. 2020.

[22] 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.

[23] *IA-32 Architectures Software Developer's Manual, Volume 3B: System Programming Guide*, Intel Corp., Austin, TX, USA, May 2007, p. 64, vol. 1.

[24] N. Patel, A. Sasan, and H. Homayoun, "Analyzing hardware based malware detectors," in *Proc. ACM/EDAC/IEEE 54th Annu. Design Automat. Conf.*, Jun. 2017, pp. 1–6.

[25] Z. Lv, "Security of Internet of Things edge devices," *Softw., Pract. Exper.*, vol. 51, no. 12, pp. 2446–2456, Dec. 2021.

[26] W. Enck *et al.*, "Taintdroid: An information-flow tracking system for realtime privacy monitoring on smartphones," *ACM Trans. Comput. Syst.*, vol. 32, no. 2, pp. 1–29, 2014.

[27] V. G. Shankar, G. Somani, M. S. Gaur, V. Laxmi, and M. Conti, "AndroTaint: An efficient Android malware detection framework using dynamic taint analysis," in *Proc. ISEA Asia Secur. Privacy (ISEASP)*, Jan. 2017, pp. 1–13.

[28] N. McLaughlin *et al.*, "Deep Android malware detection," in *Proc. 7th ACM Conf. Data Appl. Secur. Privacy*, 2017, pp. 301–308.

[29] E. M. Rudd, A. Rozsa, M. Günther, and T. E. Boult, "A survey of stealth malware attacks, mitigation measures, and steps toward autonomous open world solutions," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 2, pp. 1145–1172, 2nd Quart., 2017.

[30] S. J. Stolfo, K. Wang, and W.-J. Li, "Towards stealthy malware detection," in *Malware Detection*. Boston, MA, USA: Springer, 2007, pp. 231–249.

[31] Y. Yarom. (2016). *Mastik: A Micro-Architectural Side-Channel Toolkit*. [Online]. Available: https://cs.adelaide.edu.au/~yval/Mastik/

[32] M. Chiappetta, E. Savas, and C. Yilmaz. *Xlate*. Accessed: Aug. 12, 2021. [Online]. Available: https://www.vusec.net/projects/xlate/

[33] M. Y. Lim, A. Porterfield, and R. Fowler, "SoftPower: Fine-grain power estimations using performance counters," in *Proc. 19th ACM Int. Symp. High Perform. Distrib. Comput.*, 2010, pp. 308–311.

[34] H. Wang, S. Rafatirad, and H. Homayoun, "A+ tuning: Architecture+application auto-tuning for in-memory data-processing frameworks," in *Proc. IEEE 25th Int. Conf. Parallel Distrib. Syst. (ICPADS)*, Dec. 2019, pp. 163–166.

[35] R. O'Callahan, C. Jones, N. Froyd, K. Huey, A. Noll, and N. Partush, "Engineering record and replay for deployability," in *Proc. USENIX Annu. Tech. Conf.*, 2017, pp. 377–389.

[36] G. L. T. Chetsa, L. Lefèvre, J. M. Pierson, P. Stolf, and G. Da Costa, "Exploiting performance counters to predict and improve energy performance of HPC systems," *Future Gener. Comput. Syst.*, vol. 36, pp. 287–298, Jul. 2014.

[37] H. Wang, H. Sayadi, A. Sasan, S. Rafatirad, and H. Homayoun, "Hybrid-shield: Accurate and efficient cross-layer countermeasure for run-time detection and mitigation of cache-based side-channel attacks," in *Proc. 39th Int. Conf. Comput.-Aided Design*, Nov. 2020, pp. 1–9.

[38] H. Wang, H. Sayadi, A. Sasan, S. Rafatirad, and H. Homayoun, "Hybridg: Hybrid dynamic time warping and Gaussian distribution model for detecting emerging zero-day microarchitectural side-channel attacks," in *Proc. 19th IEEE Int. Conf. Mach. Learn. Appl. (ICMLA)*, Dec. 2020, pp. 604–611.

[39] *Arm Cortex-A7*. Accessed: Jul. 23, 2021. [Online]. Available: https://developer.arm.com/documentation/ddi0464/d/Performance-Monitoring-Unit/About-the-Performance-Monitoring-Unit

[40] (Feb. 20, 2019). *Linux Profiling With Performance Counters*. [Online]. Available: https://perf.wiki.kernel.org/index.php/Main_Page

[41] P. J. Mucci, S. Browne, C. Deane, and G. Ho, "PAPI: A portable interface to hardware performance counters," in *Proc. Dept. Defense HPCMP Users Group Conf.*, vol. 710, 1999, pp. 1–8.

[42] J. Reinders, *VTune Performance Analyzer Essentials*. Hillsboro, OR, USA: Intel Press, 2005.

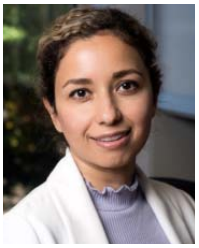
[43] *Simpleperf*. Accessed: Jul. 20, 2021. [Online]. Available: https://android.googlesource.com/platform/system/extras/+/master/simpleperf
[44] (2018). *Virustotal Intelligence Service*. Accessed: Feb. 20, 2019. [Online]. Available: https://www.virustotal.com/gui/home/upload
[45] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown, "MiBench: A free, commercially representative embedded benchmark suite," in *Proc. 4th Annu. IEEE Int. Workshop Workload Characterization (WWC)*, Dec. 2001, pp. 3–14.
[46] H. Sayadi *et al.*, "2SMaRT: A two-stage machine learning-based approach for run-time specialized hardware-assisted malware detection," in *Proc. Design, Automat. Test Eur. Conf. Exhib. (DATE)*, Mar. 2019, pp. 728–733.
[47] H. Wang, H. Sayadi, A. Sasan, S. Rafatirad, and H. Homayoun, "SCARF: Detecting side-channel attacks at real-time using low-level hardware features," in *Proc. IEEE 26th Int. Symp. On-Line Test. Robust Syst. Design (IOLTS)*, Jul. 2020, pp. 1–6.
[48] VentureBeat. *Xilinx Will Use Arm Cores in FPGA Chips*. Accessed: Oct. 18, 2021. [Online]. Available: https://venturebeat.com/2018/10/01/xilinx-will-use-arm-cores-in-fpga chips/
[49] I. K. Ganusov, M. A. Iyer, N. Cheng, and A. Meisler, "Agilex generation of Intel FPGAs," in *Proc. IEEE Hot Chips Symp. (HCS)*, Aug. 2020, pp. 1–26.
[50] M. Alam, S. Bhattacharya, D. Mukhopadhyay, and S. Bhattacharya, "Performance counters to rescue: A machine learning based safeguard against micro-architectural side-channel-attacks," Cryptol. ePrint Arch., Tech. Rep. 2017/564, Jul. 2017, p. 564.
[51] 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 *Proc. 3rd Int. Conf. Fog Mobile Edge Comput. (FMEC)*, 2018, pp. 7–12.
[52] P. Kocher *et al.*, "Spectre attacks: Exploiting speculative execution," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2019, pp. 1–19.
[53] M. Mushtaq *et al.*, "WHISPER: A tool for run-time detection of side-channel attacks," *IEEE Access*, vol. 8, pp. 83871–83900, 2020.

Han Wang (Member, IEEE) received the master's degree from George Mason University in 2019. She is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, University of California at Davis, advised by Dr. Houman Homayoun. She is also a member of Noyce Research Project for the security and privacy of AI-enabled IoT eco-systems. During her master's degree, she worked on scheduling and tuning of big data applications for datacenters, where she applied state-of-the-art machine learning algorithms to predict the optimal settings to boost performance and energy efficiency. Her research is on developing machine learning algorithms for effectively detecting microarchitectural attacks and proposing light-weight defense mechanisms to protect computer system from attacks exploiting hardware vulnerability. She was a recipient of Young Newton Student Fellowship in 2018.

Hossein Sayadi (Member, IEEE) received the Ph.D. degree in electrical and computer engineering from George Mason University in 2019, where he worked on developing effective and complexity-aware machine learning-based solutions for predicting the behavior of emerging applications to enhance the security and energy-efficiency of modern computer systems in various applications. He is currently an Assistant Professor with the Department of Computer Engineering and Computer Science, California State University at Long Beach. His research interests mainly lie in several areas of computer engineering and computer science, including computer systems cybersecurity, machine learning, hardware security, and computer architecture. He was a two-time recipient of the Provost Doctoral Fellowship from George Mason University.

Sai Manoj Pudukotai Dinakarrao received the Ph.D. degree in electrical engineering from Nanyang Technological University, Singapore, in 2015. From 2015 to 2017, he worked as a Post-Doctoral Researcher at TU Wien. He is currently an Assistant Professor of electrical and computer engineering. His research interests include hardware security, adversarial machine learning, the IoT networks, deep learning in resource constrained environments, accelerator design, algorithms, and design of self-aware many-core microprocessors. His students won the Xilinx Open Hardware Contest 2017 (Student Category) and he was a recipient of Young Research Fellow Award in Design Automation Conference (DAC) 2013.

Avesta Sasan (Senior Member, IEEE) received the Ph.D. degree in electrical and computer engineering from the University of California at Irvine, in 2010. In 2014, he was recruited by Qualcomm Office of VLSI Technology, working on the physical design and implementation of ARM processors, and serving as a Physical Designer, a Timing Signoff Specialist, and the Leader of signal and power integrity signoff for this team. He joined George Mason University in 2016. He is currently an Associate Professor with the Department of Electrical and Computer Engineering, University of California at Davis. He is also serving as an Associate Professor for the Department of Electrical and Computer Engineering, University of California at Davis. His research spans low power design and methodology, hardware security, accelerated computing, approximate computing, near threshold computing, neuromorphic computing, and the Internet of Things (IoT) solutions.



Setareh Rafatirad (Member, IEEE) received the Ph.D. degree from the Department of Information and Computer Science, University of California at Irvine, in 2012. She was an Associate Term Professor with the Department of Information Sciences and Technology, George Mason University. She is currently an Assistant Professor of teaching with the Department of Computer Science, University of California at Davis. She has published on a variety of topics related to big data, and served on the panel of scientific boards. Her research focuses on data analytics, data mining, knowledge discovery and knowledge representation, the IoT security, and applied machine learning. She received the ICDM 2019 Best Paper Award and was nominated for the ICCAD 2019 Best Paper Award.

Houman Homayoun (Senior Member, IEEE) received the Ph.D. degree from the Department of Computer Science, University of California at Irvine, Irvine, CA, USA, in 2010. He is currently an Associate Professor with the Department of Electrical and Computer Engineering, University of California at Davis, Davis. He leads the ASSEC Research Lab and is the Director of the I/UCRC Center for Hardware and Embedded System Security and Trust (CHEST) supported by NSF and the Air Force Research Laboratory (AFRL). His current research interests include hardware security, the Internet of Things (IoT), the design of next generation heterogeneous multicore accelerator for big data processing, and logical vanishable design to enhance hardware security. His current research interests include hardware security, the Internet of Things (IoT), the design of next generation heterogeneous multicore accelerator for big data processing, and logical vanishable design to enhance hardware security.