Intelligent Malware Detection based on Hardware Performance Counters: A Comprehensive Survey

Hossein Sayadi*, Zhangying He*, Hosein Mohammadi Makrani†, Houman Homayoun†

*Department of Computer Engineering and Computer Science, California State University, Long Beach, CA, USA
†Department of Electrical and Computer Engineering, University of California, Davis, CA, USA

***Abstract*—The growing complexity of contemporary computing systems heightens susceptibility to emerging cyber threats. Recent advancements in computer architecture security leverage Hardware Performance Counters (HPCs) registers to monitor applications behavior and access low-level features. The integration of Machine Learning (ML) techniques emerges as a promising solution, overcoming the performance limitations of conventional software-based defenses. Specialized HPC registers record varied hardware-related events, showcasing effectiveness in detecting malicious activities through the application of ML algorithms. This survey presents a comprehensive and comparative analysis of recent advancements in the emerging field of intelligent hardware-assisted malware detection, a topic that has garnered significant attention within the research community for the past decade. Additionally, it outlines current challenges and forecasts future research trends, offering insights for effective ML-based security countermeasures based on hardware performance counters.**

Keywords*—*Artificial Intelligence, Cybersecurity, Hardware Performance Counters, Hardware Security, Machine Learning, Malware Detection.

I. Introduction

Cybersecurity has become a paramount concern in computing systems, with attackers increasingly exploiting software and hardware vulnerabilities to compromise information technology infrastructures. Recent advances reveal a surge in leveraging emerging vulnerabilities for malicious activities, emphasizing the critical need for robust defenses against malware threats Malware, a broad term encompassing malicious software, is a piece of code developed by cyber-attackers to infiltrate computing systems without user consent, leading to unauthorized data access, file destruction, and other harmful actions [1], [2]. The escalating growth of information technology has amplified the severity of malware as a major security threat. Traditional software-based detection methods relying on static signature analysis face performance drawbacks such as static analysis limitations, inability to detect obfuscated attacks, and excessive computational overhead, particularly on resource-limited systems. These challenges stem from constraints in computing power and communication bandwidth within embedded system environments [3], [4], [5].

In addressing these challenges, the imperative is to develop effective and cost-efficient cybersecurity countermeasures, focusing on safeguarding user information and mitigating the impact of emerging cyber threats [6]. This involves a paradigm shift towards integrating security measures into the underlying hardware, establishing a bottom-up approach to fortify computing devices rather than treating security as an afterthought [7]. Simultaneously, recent breakthroughs in Artificial Intelligence (AI) and Machine Learning (ML), fueled by the increase in data volume across various computing systems, have yielded successful applications across various domains, especially in enhancing systems security [8], [9], [10].

Recent research underscores the importance of identifying malicious activities at the processor hardware and architecture level due to its speed, efficiency, and lower visibility to potential attacker exploits. To this aim, Hardware-Assisted Malware Detection (HMD) methods, specifically leveraging machine learning trained on Hardware Performance Counter (HPC) features, have emerged as a solution to the shortcomings of traditional software-based malware detection [11], [12], [13], [14], [15]. HPCs are specialized registers within a Performance Monitoring Unit (PMU) embedded in modern microprocessors that monitor the applications' hardware events (e.g., number of executed cycles, instructions, associated cache misses, etc.) [16], [17], [18], [19]. Furthermore, machine learning techniques have demonstrated efficacy in detecting and classifying these anomalies within low-level feature spaces. By leveraging machine learning, systems can effectively discern potential threats and proactively respond to evolving behaviors in real-time [7], [20].

Current research in intelligent malware detection at the hardware level spans various computing platforms such as embedded systems, Internet of Things (IoT), and high-performance systems. State-of-the-art HMD studies predominantly emphasize the development and application of standard and advanced machine learning techniques to counter evolving malware threats. This paper provides an in-depth analysis of hardware-assisted malware detection techniques, focusing on recent advancements in utilizing AI and machine learning for enhanced system protection against malicious attacks. Representing the first extensive exploration and survey of hardware-assisted malware techniques incorporating machine learning, the article discusses current breakthroughs and challenges, offering valuable insights for future research directions. We anticipate that this review will lay the foundations and facilitate further research, empowering the application of machine learning to combat the increasing complexity of cyber threats at the hardware level of processors, thereby enhancing the domain of hardware-assisted security.

The remainder of this paper is structured into distinct sections that delve into the key aspects of intelligent malware detection based on HPCs. Section II provides an exploration of the ML techniques employed for detecting anomalies in low-level feature spaces. Moving forward, Section III presents a thorough review and comparative analysis of the state-of-the-art studies in hardware-assisted malware detection. Next, Section IV describes open challenges and opportunities in this emerging research field. Lastly, Section V concludes this study.

TABLE I: Description of common malware types used in recent works

Type	Descriptions and Characteristics
Virus	Program that replicates and attaches itself to a legitimate program or document while relying on the host program to get activated. It causes the malicious pattern to spread and malfunction other executables and spreads from one computer to another.
Trojan	Program that appears legitimate, while performing malicious operations in the background to damage systems. Incapable of self-replication, Trojans rely on social engineering techniques to manipulate users into executing them.
Rootkit	Program that conceals its presence on a host system by modifying the operating system, ensuring privileged access while remaining undetected in the system's processes.
Backdoor	Program that exploits method of bypassing normal authentication procedures by avoiding the system's security mechanism. It can install itself as part of an exploit to take advantage of victim computing system's weaknesses or vulnerabilities.
Worms	Program that spreads autonomously across systems, often through networks, without relying on host file activation. Worms exploit operating system vulnerabilities to propagate invisibly and infect other systems.
Ransomware	Program that encrypts user or organizational data on a computer, demanding payment for access restoration. During this time, the files remain on the computer but are inaccessible. Once the payment is received, the ransomed data is unlocked and decrypted.
Spyware	Program that monitors and records user activities on infected systems, sending the collected data to a third party without user consent. It consumes significant system resources, affecting performance and speed, potentially causing permanent damage.
Blended Threat	A.k.a. hybrid malware that combines the features of two or more types of malware to build a more powerful and sophisticated attack [1]. It can cause harm to the infected system or network as they propagate, using multiple infection methods while exploiting the various systems' vulnerabilities.

II. Hardware-Assisted Malware Detection: Overview and Machine Learning Procedure

The escalating numbers and sophistication of malicious software infections affect individuals and organizations. These malware programs, with diverse functionalities, are designed for harmful purposes like remote control, data theft, unauthorized access, file destruction, and conducting Denial-of-Service attacks [1], [10]. In Table I, we present brief descriptions of common types of malware attacks that are widely analyzed in prior malware detection and classification researches.

Figure 1 provides an overview of process of machine learning-driven approaches designed for enhancing cybersecurity, specifically in the context of hardware-assisted malware detection. This process encompasses stages from application monitoring for HPC data profiling, feature engineering, and training ML-based detectors and online inference. The continuous learning of ML models through the analysis of low-level microarchitectural features aims to recognize and counteract malicious patterns. This proactive and intelligent approach safeguards the processor architecture from potential threats, encompassing not only malicious software but also extending to microarchitectural side-channel attacks [21], [22].

A. Feature Engineering: Determining Key Hardware Features

Developing effective ML-based hardware-assisted malware detectors begins with crucial steps like data collection and feature selection [7], [11], [14], [16]. In modern microprocessors, numerous microarchitectural events can be collected, but choosing relevant low-level features is essential to avoid computational complexity and delays associated with high-dimensional datasets. Specifically, identifying essential low-level microarchitectural features is crucial for hardware-assisted malware detection for several reasons: a) The abundance of microarchitectural events (e.g., 100+ in Intel Xeon) leads to high-dimensional data [14], b) Processing raw datasets involves computational complexity and induces delays [23], and c) Selecting pertinent microarchitectural events poses challenges in specifying non-trivial events for diverse malware classes [15]. This challenge is compounded by the limited availability of HPC registers in different processors, typically ranging from 2 to 8.

The issue of limited HPC registers, intricately connected to run-time malware detection, discusses a significant HMD challenge addressed in recent works [14], [15]. It involves pinpointing a minimal set of HPCs that precisely capture the characteristics of malicious attacks, thereby minimizing unnecessary computational overhead. This pursuit ensures the development of an efficient ML-based security countermeasure with minimal impact on system performance. Concerning the limitations of the underlying processor's architecture, especially in resource-constrained computing platforms like embedded systems and IoT devices with restricted HPC registers, efficient yet accurate run-time detection relies on critical feature selection. Recent HMD studies such as [14], [15], [19] have addressed effective run-time HMD, identifying the minimal set of essential performance counter events required for data collection in a single run.

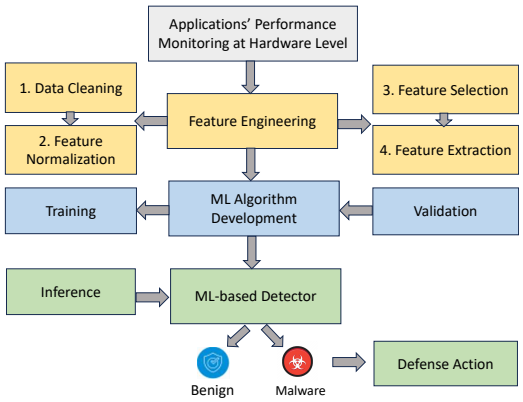

Fig. 1: General overview of hardware-assisted security countermeasures against malware using machine learning

Figure 1 demonstrates four major steps in the feature engineering process. (a) *Feature cleaning* involves analyzing the raw data to find empty entries, outliers, and any other abnormal data entries so that they can be removed from the ML process. It can also provide feedback for improving data collection. (b) *Feature normalization* is a critical step to scale the tabular data along its column or row values, preventing some data or features with large values from dominating the learning process. This technique is effective, particularly for ML algorithms that are sensitive to feature distance values. Common normalization techniques include L1/L2 normalization and MinMax normalization. (c) *Feature selection* includes feature importance analysis, feature correlation analysis, and selecting top features. This process is typically performed offline and tested effectively for the target ML model. (d) *Feature Extraction* is to extract the data entries with the top features to formulate a training dataset. In the online inference phase, it is to extract the online data to have the same top features and dimensionality as the training set for processing inference through ML detectors.

As depicted in Figure 1, the chosen HPC features are utilized to train individual ML-based detectors. The classifier endeavors to establish a correlation between the feature values and application behavior, aiming to predict the presence of malicious patterns (benign or attack type). Several feature selection techniques have played a prominent role in previous ML-based HMD efforts. These include techniques such as correlation attribute evaluation [14], [24], [25], [15], [26], [27], principal component analysis [15], [26], [28], gain ratio evaluation [16], [19], [29], and Fisher Score [30], [12].

B. ML Techniques for Malware Detection

Figure 2 illustrates various types of machine learning algorithms used in HMD techniques, where we briefly describe the

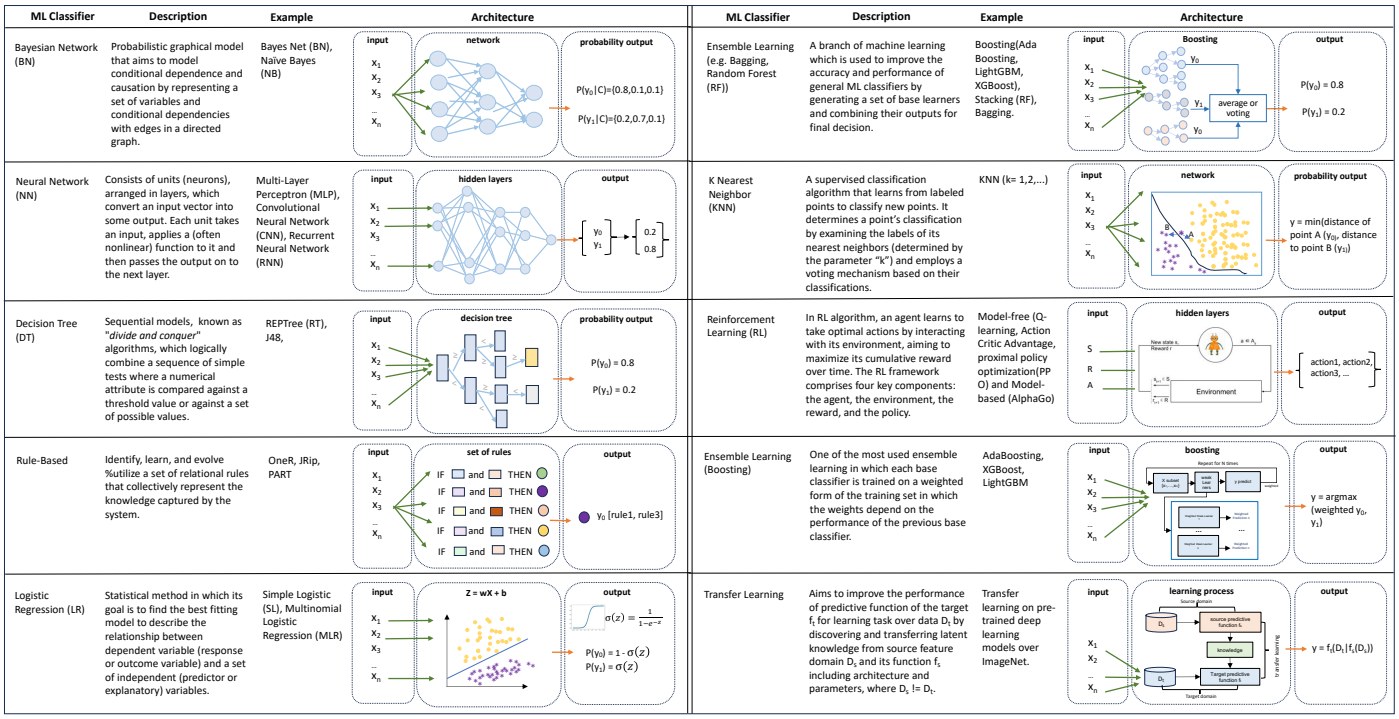

Fig. 2: Various types of machine learning algorithms used in HMD techniques, examples, and architectures

ML models, along with their examples and architectures. As depicted, various types of algorithms from classical rule-based and tree-based models to ensemble learning and advanced models like neural network, reinforcement learning and transfer learning have been deployed to address various challenges of hardware-assisted malware detection. Overall, to categorize unknown applications into either benign or malicious software using ML algorithms, the classification process can be divided into two stages including training and testing. First, we need to construct the classification model by training the ML classifiers using the extracted hardware data. The extracted features are then converted to vectors in the training set. Both the feature vectors and the class label of each sample (i.e., malicious or benign) are used as inputs for a classification algorithm.

By analyzing the training samples, the deployed ML algorithm constructs a classifier capable of detecting the patterns of malicious samples with some level of accuracy and performance detection. Next, during the testing stage, the vectors of the new file samples are first extracted using the same feature cleaning, normalization and extraction techniques as in the training phase. This unseen data then is fed to the trained classifier to examine the detection rate of the malware detection process. The classifier attempts to classify the new file samples based on the extracted feature vectors.

C. Performance Evaluation Metrics

Assessing the effectiveness of ML classifiers is a crucial phase in the implementation of robust malware detection techniques. Within the realms of AI and machine learning, various metrics are employed to evaluate the performance of an intelligent detection method. Table II provides a consolidated overview of the evaluation metrics utilized for performance analysis of ML-based malware detection techniques.

As observed, all metrics are calculated based on the counts of prediction correctness and incorrectness to each class, which are the counts of true positive, false positive, true negative, and false negative. These four counts form a confusion matrix, which is comprised of two dimensions namely "actual" and "predicted", and identical sets of "classes" in both dimensions. Each row of the confusion matrix represents the instances in a predicted class while each column represents the instances in an actual class (or vice versa) [31]. Various metrics can be employed to evaluate the performance of a model, depending on the system under consideration. While error rate and accuracy provide a general overview of model performance, they may not accurately represent performance in real-world network environments with imbalanced datasets. In such cases, where normal samples significantly outnumber abnormal ones, F-Measure emerges as a more comprehensive metric, accounting for both precision and recall, and proving resilient to class imbalance. F-Measure (F-score), is often favored for indicating the overall detection performance, offering a balanced perspective where precision and recall need to be traded off.

In binary malware detection, metrics such as True Positive Rate (TPR), True Negative Rate (TNR), False Positive Rate (FPR), and False Negative Rate (FNR) offer detailed insights into a model's prediction capabilities for malware and benign samples. TPR signifies the proportion of correctly identified malware among all predicted malware, crucial for assessing detection accuracy. Conversely, TNR measures the correct classification of benign samples, addressing false alarms. Precision gauges positive prediction accuracy, while recall evaluates positive prediction completeness. Receiver Operating Characteristic (ROC) and Area under the ROC Curve (AUC) provide a graphical representation of the model's performance, offering insights into trade-offs between True Positive (TP) and False Positive (FP). Additionally, AUC measures the entire area under the ROC, with higher values indicating better detection performance.

TABLE II: Performance metrics used for evaluating ML-based malware detection techniques

Evaluation Metric	Equation/Description
True Positive (TP)	Count of correct positive prediction (e.g., malware is predicted as malware).
False Positive (FP)	Count of incorrect positive prediction (e.g., benign is predicted as malware).
True Negative (TN)	Count of correct negative prediction (e.g., benign is predicted as benign).
False Negative (FN)	Count of incorrect negative prediction (e.g., malware is predicted as benign).
Specificity: True Negative Rate (TNR)	$TNR = TN/(TN + FP), TNR = 1 - FPR$, defined as the proportion of genuinely negative samples predicted as negative result among all negative samples.
False Negative Rate (FNR)	$FNR = FN/(FN + TP), FNR = 1 - TPR$, defined as the proportion of genuinely positive samples predicted as negative result among all positive samples.
False Positive Rate (FPR)	False alarm rate, defined as the proportion of genuinely negative samples predicted as positive results among all negative sample, $FPR = FP/(FP + TN), FPR = 1 - TNR$.
Recall/sensitivity: True Positive Rate (TPR)	$TPR = TP/(TP + FN), TPR = 1 - FNR$, defined as the proportion of genuinely positive samples predicted as positive results among total positive samples. It refers to the proportion of correctly identified positives (the rate of malware samples) by the model.
Precision (P)	$P = TP/(FP + TP)$, Defined as the ratio of true positive predictions to total predicted positives, indicating the confidence level of attack detection.
F-Measure (F1-Score)	$Fmeasure = 2 \times (P \times R)/(P + R)$, Represents a weighted average of precision (P) and recall (R). It offers a more comprehensive evaluation than accuracy, considering both precision and recall, and is robust for evaluating both balanced and imbalanced datasets.
Detection Accuracy (ACC)	$ACC = (TP + TN)/(TP + FP + TN + FN)$, The ratio of correctly classified samples to total samples. Accuracy is a suitable metric when the dataset is balanced.
Error Rate (ERR)	$ERR = (FP + FN)/(P + N)$, where P, N represents total positive and negative samples respectively. It is the number of all incorrect predictions divided by the total number of the dataset.
Receiver Operating Characteristic (ROC) Curve	Visually depicts the trade-offs between true positive rate and false positive rate, providing insights into detection performance with varying discrimination thresholds. Each prediction result corresponds to a point in the ROC space, and the upper-left corner (coordinate (0, 1)) signifies optimal detection, denoting 100% sensitivity and 100% specificity.
Area Under the Curve (AUC)	$AUC = \int_0^1 TPR(x)dx = \int_0^1 P(A > \tau(x))dx$, where τ is the thresholds on the decision function used to compute FPR and TPR. AUC quantifies the entire two-dimensional area beneath the ROC curve, spanning from (0,0) to (1,1).

D. Performance Monitoring Tools

Here, we review some widely used performance monitoring tools across various operating systems (Windows, Linux, and macOS) in previous studies. These tools are employed to monitor application behavior and collect hardware-related events, aiding in the analysis of application performance. These tools include Perf [32], Pin [33], PAPI [34], Intel VTune [35], and Intel PCM [36]. All these tools are available for Linux systems while only Intel VTune and Intel PCM can monitor HPCs in Windows and macOS systems. Perf, PAPI, and Pin demand some knowledge of command lines for users due to the lack of a GUI interface. Perf tool is a Linux-based low-level performance monitoring tool that can instrument CPU performance counters, tracepoints, kprobes, and uprobes (dynamic tracing). Its monitoring granularity scales as least as $10ms$ without customization. Pin tool collects various program's ISA-dependent features such as instruction mix, instruction-level parallelism, register traffic, and branch predictability to examine the application behavior [33].

Performance Application Programming Interface (PAPI) [37] provides a cross-platform interface for monitoring hardware performance counters on processors that are equipped with specific registers for hardware events. Intel has developed a licensed tool called Vtune [35] to discover and resolve performance bottlenecks in running programs for tweaking and debugging purposes. It offers a robust GUI interface and supports a wide range of profiling, including HPCs, call graphs, performance bottlenecks, and hotspot hunting, in comparison to the previous tools. Moreover, PCM [36] is the performance monitoring unit implemented in Intel's processors (e.g., Xeon, Atom, and Xeon Phi) to monitor performance and energy-related metrics in both Windows and Linux environments. Compared to Perf and PAPI tools, Intel PCM supports both core and uncore events monitoring in real-time.

III. State-of-the-Arts on Hardware-Assisted Malware Detection

In this section, we review the latest proposals on hardware-assisted malware detection using ML techniques. Figure 3 illustrates the yearly analysis of publications on this research topic, captured from Google Scholar which indicates a growing interest within the research community over the past decade. Furthermore, in Table III, we provide a comparative overview of prominent research on hardware-assisted malware detection. Space constraints limit the inclusion of all prior HMD research, thus, selected key studies are highlighted in this section. For the purpose of thorough and structured exploration, as described below we classify prior studies into several categories according to each work's specific focus and the challenges addressed.

A. General Hardware-Assisted Malware Detection

The study by Demme et al. [11] pioneered the exploration of HPCs for accurate malware detection via ML techniques. The research successfully demonstrated the efficacy of offline ML algorithms in pinpointing malicious software. Furthermore, it showcased the applicability of HPCs in detecting malware at the Linux OS level, including Linux rootkits and cache side-channel attacks on Intel and ARM processors. The study achieved notable detection performance results for Android malware by employing ML algorithms, such as Artificial Neural Network (ANN) and K-Nearest Neighbor (KNN). Prior to that, the study conducted in [38] utilized HPCs for both static and dynamic integrity checking of running programs. The authors employed a tool named Eurequa to identify malicious modifications in programs by detecting equations and hidden mathematical relationships among HPCs. While their approach did not involve ML, it showed the potential of HPCs for security applications with minimal run-time overhead.

Tang et al. [12] explored the feasibility of unsupervised learning with HPCs features to detect return-oriented programming (ROP) and buffer overflow attacks by identifying anomalies. The Fisher Score metric was employed for feature selection, distinguishing malicious code execution from non-malicious instances for each event and ranking them. The top 7 ranked features were then used to train one-class Support Vector Machine (oc-SVM) classifier, detecting deviations in program behavior indicative of potential malicious attacks. The study also compared performance across different sampling frequencies of HPCs.

The works by Wang et al. [56], [57] utilized HPCs information to detect rootkits that modify system calls through statistical methods. These approaches focused on counting

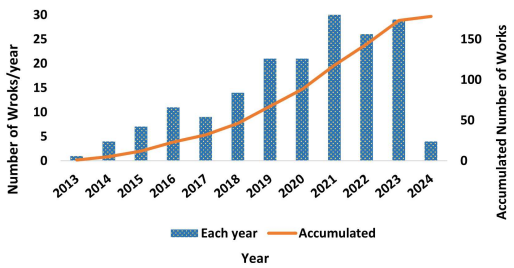

Fig. 3: Comprehensive yearly analysis of state-of-the-art researches on intelligent malware detection using hardware performance counters

TABLE III: Overview of recent research on hardware-assisted malware detection leveraging machine learning techniques

Research	Year	Platform	Machine Learning Model	Threat Type	Major Contribution and Challenge Addressed
[11]	2013	Android, Linux	KNN, NN, DT, RF	Malware	Utilized HPCs for ML-based malware detection at the OS level. The study revealed that performance counter data can effectively identify malware, and the ML model trained with HPCs demonstrates resilience to slight variations in malware programs. are robust to minor variations in malware programs.
[12]	2014	Linux	ocSVM	Malware	Demonstrated that applications' HPC features combined with unsupervised ML can detect deviations from malware programs. Their findings show the microarchitectural characteristics of benign and malware programs have different patterns.
[13]	2015	Windows	LR, ANN	Malware	Proposed Malware-Aware Processors (MAP) framework for real-time HMD. Their work proposed a two-level detection framework where the hardware classifier prioritizes the work of a more accurate but more expensive ML defense mechanism. They also explored integrating the MAP implementation with an open-source x86-compatible core.
[39]	2015	Windows	LR, NN, EL	Backdoor, PWS, Rogue, Trojan, Worm	Focused on per class malware detection using HPCs, compared the accuracy of specialized and general malware detectors, highlighting the effectiveness of specialization in HMD.
[16]	2017	Linux	SVM, ocSVM, NB, DT	Kernel Rootkits	Used synthetic rootkit traces of HPC features to train ML to detect kernel-level rootkit attacks. Their experimental results show that HPCs can be effective features for rootkit detection.
[24]	2017	Linux	OneR, MLP, BN, SMO, SGD, LR	Malware	Assessed various hardware-based malware detectors across different metrics of accuracy, accuracy/area, Power Delay Product (PDP), and latency.
[40]	2017	Windows	LR, MLP, DT, SVM	Adversarial malware	Proposed a resilient defense solution to reverse-engineering based adversarial attacks through retraining, randomized features and HPC event periods.
[14]	2018	Linux	BN, J48, JRip, MLP, OneR, RT, SGD, SMO, AB, BG	Malware	Broke the trade-off between the number of features used for developing effective malware detectors and the limited number of HPC registers available in today's microprocessors. Identified the top HPCs, implemented various MLs for HMD using different HPC numbers, and proposed ensemble solutions for run-time malware detection with only 2 or 4 HPCs.
[25]	2018	Linux	AB, LR, BN, MLP, J48, JRip	viruses, worms, rootkits, and trojans	Explored ensemble learning's effectiveness for HMD, comparing general and ensemble classifiers in terms of accuracy, robustness, performance, and hardware overhead.
[28]	2018	Windows	DT, RF, MLP, KNN, AB, NB	Malware	Questioned the effectiveness of HPC-based malware detection, revealing larger performance variations across different ML classifiers.
[26]	2018	Linux	BN, J48, JRip, MLP, OneR, RepTree, SMO	Ransomware	Presented a customized hardware malware detection and identification solution for embedded systems. Results show heavyweight classifiers (MLP, BayesNet, SMO) excel in average malware detection accuracy, while lightweight classifiers (JRip, OneR) demonstrate superior accuracy per unit area across all tested malware classes.
[15]	2019	Linux	J48, JRip, MLP, OneR, AB	Virus, Trojan, Rootkit, Backdoor	Introduced a specialized two-stage run-time HMD, called 2SMaRT. The first stage classifies applications into benign or specific malware classes (Virus, Rootkit, Backdoor, Trojan) using multiclass classification. In the second stage, 2SMaRT employs tailored machine learning models for each malware class, enhancing detection performance.
[41]	2019	Linux	LR, NN	Adversarial malware (Backdoor, Rootkit, Virus, Worm, Trojans)	Created adversarial attacks on HMD systems by injecting perturbations into HPC traces. Employed an adversarial sample predictor and then used reverse-engineered malware samples together with perturbed noise to evade malware detection systems.
[4]	2019	Linux	MLP, OneR, LR, JRip	IoT malware	Mitigated security risks in extensive IoT networks by introducing the HaRM framework. This framework utilizes a low computational overhead ML classifier (OneR), catering to the requirements of IoT devices while maintaining effective malware detection accuracy. Formulated an optimal control problem for malware confinement while maintaining network integrity.
[27]	2020	Linux	MLP, OneR, LR, J48, SVM, SGD	IoT malware	Proposed a two-stage framework with a lightweight malware detector in IoT network followed by a stochastic controller.
[42]	2020	Linux	J48, JRip, LR, KNN, BOFF, FCN	Stealthy Malware (Trojan, Rootkits, Backdoor, Blended)	Addressed the challenge of stealthy malware in hardware-assisted malware detection and proposed a specialized time series Fully Convolutional Neural Network (FCN) approach for accurate embedded malware detection using HPCs, in which standard ML techniques fail to detect.
[43]	2020	Xilinx Zynq SoC	RNN, Linear Regression	BASHLITE Botnet, PNScan Trojan, Mirai Botnet	Explored machine learning interpretability in HMD, establishing theoretical transparency. Demonstrated improved detection performance through explainable ML.
[44]	2020	Linux	oc SVM	Malware	Trained a one-class SVM with time series data in a multi-thread environment, considering per-thread and cross-thread features in embedded devices. This enables continuous monitoring of multi-thread HPC readings for run-time malware detection.
[17]	2021	Linux	Time series FCN, MLP, ResNet, MCDNN, JRip, J48, LR, KNN, BOPF	Stealthy Malware (Trojan, Rootkits, Back- door, Blended)	Proposed a lightweight time series FCN-based approach to accurately detect stealthy malware trace at run-time using the top HPC event, branch instructions. Presented a comparative analysis of diverse ML models for hardware-assisted stealthy malware detection.
[45]	2021	Windows	RF, LR	Malware	Quantified machine learning effectiveness with uncertainty in HMD techniques and introduced an uncertainty estimator to consider uncertain predictions when dealing with unseen malware.
[46]	2021	Linux	EL(AB), RF, DT, GNB, SGD, LR, ET	Zero-day malware, Worm, Virus, Botnet, Ransomware, Spyware, Adware, Trojan, Rootkit, and Backdoor	Addressed the challenge of zero-day malware in HMD techniques. Indicated that classical ML models proved effective for known malware but faced a high FPR in zero-day malware detection, leading to the proposal of AdaBoosting over RF for improved zero-day malware detection using HPCs.
[47]	2021	Windows, Linux	CNN, 1D-CNN, RF, DT, KNN	Trojan, Worm, Backdoor, Ransomware, Spyware, Virus	Addressed the challenge of real-time malware detection in cloud (IaaS) environment, extending the HMD solution to cloud.
[20]	2021	Linux	SGD, MLP, RepTree, OneR, Jrip	Worm, Virus, Trojan, and Backdoor	Proposed an online adaptive and cost-efficient decision maker using a rule-based JRip algorithm to select the most efficient ML models at run-time to be used as the online malware detector according to the users' preference and hardware overhead.
[29]	2022	Linux	CNN, TL, RF, DT, GNB, LR, ET, Ridge, KNN, SVM, BDT	Zero-day malware, Worm, Virus, Botnet, Ransomware, Spyware, Adware, Trojan, Rootkit, and Backdoor	Proposed Deep-HMD, a deep neural network and transfer learning-based approach for hardware image-based malware detection. Converted tabular HPCs malware and benign data into 2D images and applied transfer learning, demonstrating superior effectiveness in both known and zero-day malware detection.
[48]	2022	Linux	DT, NN	Adversarial malware	Presented a moving target defense mechanism to defend against adversarial attacks in HMD techniques by changing the numbers and set of HPCs and classifiers.
[49]	2022	Android	BN, SL, MLP, PART, SMO	Android malware	Proposed OptiEdge, an ML-guided hardware-assisted resource and timing estimation tool that can effectively reduce the design space exploration for edge devices' design through HLS optimization for MLs in on-device hardware-assisted cybersecurity in edge.
[50]	2022	Windows	LR, DT, SVM	Adversarial malware	It shows existing HMDs can be effectively reverse-engineered and subsequently evaded. They suggested that retraining over adversarial samples is not effective. As a result, they proposed uniform random switching among ML detectors at run-time to defend against effective reverse engineering attacks. Their approach showed an increased detection performance for both evasive and non-evasive malware.
[19]	2022	Linux	RL(UCB), EL, JRip, J48, LR, MLP, OneR, RepTree	Zero-day malware, Worm, Virus, Botnet, Ransomware, Trojan, Rootkit, Backdoor, etc.	First work that tackles major issues in adaptive and cost-aware zero-day malware detection using HPCs, considering desired performance metrics and available hardware resource; meantime propose a unified feature selection method based on heterogeneous feature fusion approach.
[51]	2023	Linux	RL(A2C), CNN, RF, DT, SGD, LR, Ridge, SVM, BDT	Worm, Virus, Botnet, Ransomware, Trojan, Rootkit, Backdoor, etc.	Proposed a hybrid and adaptive image-based framework for online hardware-assisted zero-day malware detection in IoMT devices using AI-enabled reinforcement learning approach. Assessed and compared various ML classifiers for zero-day malware detection using HPCs.
[52]	2023	Linux	SVM, RF, GBM, AB	Ransomware	Determined the optimal hardware features and time granularity for early ransomware detection.
[53]	2023	Linux	MLP, LR, DT	Adversarial malware (Backdoor, Rogue, Trojan, Worm)	Enhanced adversarial defense by adding stochastic noise through controlled undervolting in HMDs' computations during inference.
[54]	2023	Linux	Binary RF, LightGBM, XGBoost, DT, LR, MLP & multi-classifiers RF, DT, MLR, etc.	Adware, Botnet, Keylogger, Ransomware, Rootkit, Spyware, Trojan, Worm	Introduced a customized hardware monitoring framework and employed various MLs to enhance cybersecurity through a two-level malware detection and identification framework using HPCs in biomedical computing systems. Assessed and compared various binary and multiclass ML models for accurate and efficient hardware-assisted malware detection and classification in biomedical applications.
[55]	2024	Linux	RF, DT, LightGBM, XGBoost, GB, BDT, ET	Trojan, Botnet, Backdoor, Virus, Worm, Rootkit, Ransomware, Adware, Spyware	Presented a thorough evaluation of machine learning algorithms' reliability in hardware Intrusion Detection Systems (IDSs) considering factors of training data size, the number of HPCs used, and internal data separability (malware stealthiness). Further, introduced an model observer for an enhanced and reliable malware detection.

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, Bag-of-Pattern-Features: BOFF, Ridge Classifier: Ridge, BaggedDT: BDT, ExtraTree: ET, Multinomial Logistic Regression: MLR, Fully Convolutional Network: FCN, TL: Transfer Learning, Reinforcement Learning: RL, Actor Critic Advantage: A2C, Upper Confidence Bound: UCB, High Level Synthesize: HLS, SimpleLogic: SL, Sequential Minimal Optimization: SMO.

hardware events during each system call execution in a guest Virtual Machine, enabling the identification of modifications to kernel control flow. Despite their effectiveness, these works employed complex detection architectures that do not rely on machine learning and data mining solutions. Such architectures may not be suitable for implementation in resource-constrained embedded and IoT devices.

The research in [58] presented HPCMalHunter, an anomaly-based malware detection technique utilizing machine learning classifiers at the hardware level. This framework predicted malware presence with high accuracy using a Support Vector Machine (SVM) classifier. Initially, the detector gathered a set of HPC events concurrently from the running application. In the subsequent step, the authors employed a Singular Value Decomposition (SVD)-based feature reduction technique to identify the most significant HPC events.

In studies [13], [59], the authors introduced the MAP framework for real-time hardware-assisted malware detection. They explored sub-semantic features in the low-level microarchitectural space, including executed instruction features, memory address pattern features, and architectural event features. Their approach utilized Logistic Regression and

Artificial Neural Network classifiers for malware detection, requiring changes to the microprocessor pipeline for real-time implementation. The authors discussed estimated latency and area utilization of the proposed algorithm implementations. In [39], the focus is on per-class malware detection using hardware performance counter information. The authors developed ML-based specialized detectors trained for individual malware classes, predominantly employing logistic regression and neural network classifiers. Utilizing the same features as Ozsoy et al. [13], they compared the accuracy of specialized and general malware detectors, highlighting the effectiveness of specialization in hardware-assisted malware detection. They further enhanced accuracy through specialized ensemble learning, combining LR and NN classifiers.

Singh et al.'s work [16] focused on utilizing ML algorithms on synthetic traces of HPC features to detect kernel-level rootkit attacks. To reduce features, they employed the Gain Ratio technique from the WEKA toolkit [60], achieving high accuracy in detecting synthetic rootkits. The study collected HPC samples only at the program's end and trains ML classifiers using these HPCs to detect and classify rootkits. Notably, rootkits utilizing direct kernel object manipulation (DKOM) have minimal impact on HPCs, posing a challenge for simple HPC-based detection. In [24], various ML classifiers for malware detection were evaluated, ranging from simple OneR to complex MLP. The study assessed detectors based on accuracy, accuracy/area, Power Delay Product (PDP), and latency. While complex classifiers achieved close to 90% accuracy, their implementation overheads led to inferior performance in PDP, accuracy/area, and latency compared to simpler alternatives. The OneR algorithm emerged as the most cost-effective, with over 80% accuracy and fast execution (less than 10ns), achieving the highest accuracy per logic area while primarily relying on a single branch-instruction feature.

Sayadi et al. [14], [25] proposed ensemble machine learning solutions for effective run-time malware detection using low-level microarchitectural features. To optimize run-time detection with limited hardware performance counters, the authors employed systematic feature selection. They used the Correlation Attribute Evaluation technique to select top events by calculating Pearson's correlation coefficient between HPC features and determining the most significant ones.

In particular, the research in [14], the challenge of limited HPC registers for run-time malware detection was addressed by focusing on specialized ML techniques trained with a small number of HPC features (2-4). The study highlighted that across various ML models the accuracy of hardware-based malware detection decreases with the number of HPCs used. To enhance performance, ensemble learning techniques were proposed, eliminating the need to run an application multiple times. They implemented eight robust ML models and two ensemble learning classifiers (Adaboost and Bagging), and compared them in terms of detection accuracy, robustness, performance (accuracy×robustness) and hardware overhead. Results showed that the proposed ensemble learning malware detection with just 2 HPCs outperformed standard classifiers with 8 HPCs by up to 17%, matching the robustness and performance of standard ML-based detectors with 16 HPCs while using only 4 HPCs and enabling effective run-time hardware-assisted malware detection.

The work in [15] proposed, 2SMaRT, a two-stage machine learning-based approach for specialized 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 the performance of malware detection. The experimental results indicated that 2SMaRT using ensemble technique with just 4HPCs outperforms state-of-the-art classifiers with 8HPCs by up to 31.25% in terms of detection performance, on average across different classes of malware.

The research in [61], presented a hardware-level malware detection framework named Akoman that utilizes Discrete Wavelet Transform (DWT) and behavioral signatures derived from hardware events to determine the behavior of running programs. For each known malware type, two signatures are generated by collecting four hardware event traces from the executions of malware samples belonging to that family. The first signature, obtained through SVD, is used for fast initial matching, while the second, obtained through discrete wavelet transform, is employed for precise final matching. The work in [52] focused on determining the optimal hardware features and time granularity for early ransomware detection. The study examined HPC counter statistics gathered at intervals of 100ms, 500ms, and five seconds. The authors trained several classical ML models to compare configurations and found that capturing 5 HPC registers every 100ms for the first 3 seconds of payload execution achieves the best results with the AdaBoost classifier, achieving above 90% accuracy.

In [28], unlike other HMD research, the suitability of low-level microarchitectural features for distinguishing malware from benign applications is questioned. The authors argued that there is no inherent relationship between low-level microarchitectural features and high-level application behavior. They contended that positive results in previous works stem from optimistic assumptions, presenting their best result with an F1-score of 80.78%. However, they conducted a 10-fold cross-validation of HPC-based malware detection, revealing larger performance variations across different machine learning classifiers. Similar variations are noted in previous works employing different machine learning classifiers.

B. Addressing Advanced Threats: Stealthy and Zero-Day Malware Detection

Stealthy attacks involve concealing malicious code within benign applications, making detection more challenging [62]. Prior hardware-assisted malware detection approaches often assume malware as a separate thread, overlooking scenarios where malware is embedded in benign applications. This embedded malware, a form of stealthy threat, remains undetected by commercial antivirus software.

The works in [42], [17] were the first HMD efforts that addressed this research gap by tackling the challenge of detecting embedded malware using hardware features. Embedded malware involves stealthy cyber-attacks where malicious code hides within benign applications, eluding traditional detection methods. In HMD methods, directly inputting HPC data into ML can lead to contamination, as malicious code within benign applications combines with HPC features. Addressing this issue, the authors introduced StealthMiner, a specialized time series ML approach based on the Fully Convolutional

Network (FCN), aiming to detect stealthy malware, embedded within benign traces, at run-time using the time series branch instructions feature, the most prominent hardware event.

The study in [45] proposed an ensemble-based (bagging) approach for quantifying uncertainty in predictions made by ML models in HMD techniques. The study introduced an uncertainty estimator, showing that considering uncertain predictions enables ML models to handle zero-day malware. Furthermore, the work in [63] employed a power grid case study to demonstrate that HPC effectively detects stealthy rootkits in an 8-grid power system, offering a landscape review of HPC's role in malware detection.

While showing promise for known malware detection, accurately identifying zero-day malware has been overlooked in prior HMD works. Zero-day attack is a type of serious cybersecurity threat that exploits software security vulnerabilities that are undocumented (unknown) in the training database of the detection mechanism. Zero-day attacks exploit potentially serious software security vulnerabilities that are undocumented (unknown) in the database of the detection mechanism [64], [65]. In addressing the challenge of zero-day malware detection, He et al. [46] conducted experiments using machine learning to detect known and zero-day malware. Classical MLs were found effective for known malware types but suffered from a high false positive rate in zero-day malware detection. By applying AdaBoosting over RF model, they achieved a 4% improvement in the F1-score for zero-day malware detection.

In a recent research [29], a multi-stage zero-day malware detection method based on deep transfer learning was developed. Investigating the feasibility of transfer learning across different domains and applications is crucial. Training models on diverse datasets from various domains may lead to more robust and generalized malware detection capabilities. The proposed method in this study involved converting four HPC features into two-dimensional images and applying transfer learning with a pre-trained ResNet model on ImageNet to enhance learning of hidden patterns of zero-day malware. Results demonstrated the effectiveness of deep neural network and transfer learning for HMD, addressing the open challenge of zero-day malware detection in current research.

Furthermore, He et al. [19] is the first work that tackles major issues in adaptive and cost-aware zero-day malware detection using low-level hardware events. They proposed a unified feature selection method based on heterogeneous feature fusion to determine prominent HPC events for on-device HMD. Additionally, the authors introduced Reinforced-HMD, a novel reinforcement learning-based framework designed for adaptive and cost-aware unknown malware detection, focusing on desired performance metrics and available hardware resources. The framework utilized six classical and two reinforcement learning algorithms, including the Upper Confidence Bound (UCB) approach, and undergoes thorough efficiency analysis for detecting unknown malware using HPC events. Their analysis demonstrated Reinforced-HMD's accuracy and robustness, achieving a 96% F1-score and AUC metrics.

In a recent study [55], the authors focused on the reliability analysis of hardware-oriented Intrusion Detection Systems (IDSs). While ensuring the dependability of ML models' decisions is crucial, it has been overlooked in previous studies. This work conducted a thorough evaluation of ML algorithms in IDSs, considering factors such as training data size, the number of hardware events used, and internal data separability (malware stealthiness). To enhance reliable intrusion detection, an effective model observer module is integrated during ML inference to assess prediction reliability at run-time, determining the ML model's confidence.

C. Impact of Adversarial Attacks on HMD Techniques

While artificial intelligence, in particular machine learning, has been widely embraced to enhance security countermeasures, recent research has uncovered new security challenges, notably adversarial attacks [66], [67], [68]. Despite ML classifiers demonstrating resilience against random noises, vulnerabilities have emerged, allowing adversaries to manipulate outcomes by adding specially crafted perturbations to input data. As ML models become integral for malware detection, adversaries may employ dynamic strategies to evade detection. Investigating robustness against adversarial attacks, especially those targeting hardware features, is a critical challenge.

In the context of HMD, which relies on microarchitectural events captured via HPCs, [41] demonstrated an adversarial attack on malware detection systems. This attack involved injecting perturbations into HPC traces using an adversarial sample generator application. Addressing these vulnerabilities presents a new avenue for future research in developing adversary-resilient ML-based malware attack detectors. Das et al. [18] explore the accurate measurement of events using performance counters, uncovering challenges of dealing with performance counters data in security applications. They highlight the impact of HPC intricacies on exploit prevention and malware detection, while showing the potential manipulation of HPCs by adversaries to bypass security defenses.

In [40], the authors proposed a resilient solution to defend ML-based HMD against reverse engineering. They highlighted HMD's vulnerability to adversarial attacks and emphasize that retraining on adversarial malware datasets is ineffective. To address this, they suggested constructing detectors with randomized features and HPCs collection periods, switching them stochastically to thwart attackers' predictions. In another study [48], a Moving Target Defense (MTD) technique is proposed for adversarial attacks on HMD. MTD dynamically changed the number and set of performance counters and the classifier, confusing attackers. It used random selection of 4 features and 2 ML models (Decision Tree, Neural Network) at run-time, successfully defending against adversarial attacks without performance degradation. However, its effectiveness against non-adversarial malware attacks remained untested.

In [50], the authors demonstrated the effectiveness of reverse-engineering ML models while highlighting the limitations of retraining for run-time attacks. They proposed a strategy of uniform random switching among ML detectors to enhance defense against reverse engineering, akin to the concept of moving target defense. Similarly, Islam et al. [53] addressed the adversarial attacks through reverse engineering. They proposed Stochastic-HMDs, which involved introducing stochastic noises into the computations of model inference to defend reverse engineering based adversarial attacks. They manipulated the stochastic noise through controlled undervolting by scaling the supply voltage below nominal level to add noises in the HMDs' computations during inference.

D. Securing Edge and Beyond: Malware Detection in Embedded Systems, IoT, and Cloud

Sayadi et al. [26] extended the concept of malware detection using microarchitectural events to embedded systems,

presenting a customized ML-based hardware-assisted malware detection and identification solution for these resource-constrained devices. The work identified challenges in effective malware detection for embedded devices, emphasizing the limitations of conventional software-based methods in these systems. Due to the low overhead of hardware monitoring, their deployment is seen as a promising solution. The proposed lightweight HMD addressed the constraints of embedded systems by leveraging HPC features. It employed various ML classifiers to detect and classify different malware classes at run-time, using four crucial HPC features: branch instructions, cache references, branch misses, and node-stores. This approach aimed to achieve accurate and effective run-time malware detection despite the limited computing power and resources in embedded systems. Results showed heavyweight classifiers (MLP, BayesNet, SMO) excelled in average malware detection accuracy, while lightweight classifiers (JRip, OneR) demonstrated superior accuracy per unit, with MLP being the least effective.

Exploring synergies between hardware-assisted detection and network-level detection techniques can improve overall system security. Integrating insights from both levels can enhance the ability to detect sophisticated malware threats. Aligned with this idea, in [4], the authors addressed security risks in large-scale IoT networks, emphasizing the challenges of malware propagation. Traditional approaches fall short, leading to the introduction of HaRM, a run-time malware detector achieving rapid 92.21% accuracy within 10ns. A stochastic model predictive controller confines malware propagation in real-time, ensuring uncompromised network performance. Further, in [27] the authors expanded their prior work exploring network effects, proposing a two-stage framework with a lightweight malware detector followed by a stochastic controller, outperforming existing solutions by achieving nearly 200% higher network throughput on IoT devices.

The utilization of HPCs extends beyond control-flow alterations for malware detection, proving effective in detecting firmware modifications. In the study presented in [69], ConFirm is introduced as a cost-efficient technique for identifying malicious changes in embedded control system firmware. The approach involved measuring low-level hardware events captured by HPC registers during firmware execution. The evaluation encompassed various firmware types on ARM- and PowerPC-based embedded processors, assessing detection capability and performance overhead. Furthermore, the proposal is extended to handle more complex control flows, introducing a machine learning-based classifier in [70] to automatically extract relations between different hardware features.

The study presented in [43] explored the interpretability of HMD techniques by introducing a framework that employed explainable machine learning. This framework enhanced the explainablity of classification results, making them more accessible and understandable for human analysis. They collected time series HPC data from Xilinx Zynq7000 SoC evaluation board, trained an RNN, and employed the model's output to train a linear regression model for feature contribution factors. This interpreted which HPC features contribute to malicious attacks and when they occur.

Moreover, in research [44], a one-class SVM is built using only benign data to classify normal and malware on programmable logic controllers (PLCs). Their approach monitored real-time HPCs with an outside-of-the-process approach, collecting separate HPCs from each thread. Per-thread and cross-thread features are extracted, with the former modeling activity patterns within threads and the latter modeling temporal relationships among activity patterns between threads. The work in [71] explored utilizing HPCs and ML for IoT device security. The study delved into topics including authentication, access control, secure offloading, and malware detection schemes, assessing their benefits, drawbacks, and potential for safeguarding IoT infrastructure on both the edge and the cloud, along with individual IoT devices.

In [51], a hybrid and adaptive image-based framework for online hardware-assisted zero-day malware detection in the Internet of Medical Things (IoMT) is proposed. The method based upon Deep Reinforcement Learning (DRL) dynamically selected the best Deep Neural Network (DNN) detector at run-time from a pool of continuously trained models, customized for each device. Tabular hardware-based data are converted into small-size images then using transfer learning to enhance model performance for unknown malware detection. A DRL agent, consisting of two Multi-Layer Perceptrons (MLPs) functioning as an Actor and a Critic, is trained to dynamically select the optimal DNN model at run-time. This decision-making process ensured highly accurate zero-day malware detection using a limited number of hardware events, leading to high malware detection performance.

Tian et al. [47] addressed real-time security challenges in virtual machines in the Infrastructure as a Service (IaaS) cloud environment. Using Lamport's ring buffer algorithm, they implement concurrent real-time control flow collection and security checks. Intel Processor Trace (PT)'s Virtual Machine Introspection captured control flow information outside the target VM. They converted this information into two-dimensional color images and employ a CNN-based method for malware detection. This represented the sole effort in cloud-based malware detection utilizing HPCs. Although its performance may not reach state-of-the-art levels, it underscored potential avenues for future research in cloud security.

The work in [49] proposed accelerated MLs for efficient on-device HMD. The authors proposed OptiEdge, an ML-guided hardware-assisted resource and timing estimation tool that can effectively reduce the design space exploration for edge devices' design through effective High Level Synthesize (HLS) optimization techniques for different ML algorithms. Furthermore, the recent research in [54] extends the applicability of HMDs to biomedical computing systems. The work introduced a tailored hardware monitoring framework and employed ML algorithms to enhance the accuracy and efficiency of malware detection and classification using real-time data from biomedical processors' hardware events. The reported results highlighted the effectiveness of the XGBoost model, achieving a 95% detection rate in F-measure and accuracy with efficient resource utilization and low inference latency.

IV. Research Challenges and Opportunities

1) Architectural Reasoning of HPCs for Malware Detection: While hardware performance counter registers have been extensively utilized for enhancing security, the lack of detailed architectural analysis poses a challenge. Obtaining a deep understanding of the interactions between microarchitectural features and malware behavior is crucial for effective security analysis. Current practices often involve extracting features

without comprehensive explanations of their relevance to malware traits. A common approach includes feature selection, but the connection between selected features and achieved results remains largely unexplored. Researchers need to emphasize providing clear explanations for the chosen features in machine learning-based malware detection, ensuring a more comprehensive and interpretable analysis of security implications.

2) HPCs Validation for Security and Cross-Architecture Compatibility: Lack of solid documentation and indefinite inference of HPC features pose challenges, requiring further support and understanding for accurate performance monitoring. This is because security analysis was not the initial purpose for designing such registers in modern microprocessors. Complexity variations across architectures make consistent HPC-assisted information extraction challenging. Researchers should validate findings on diverse microprocessor architectures and provide detailed documentation of performance counter configurations for enhanced reproducibility and credibility of security-related works. Ensuring compatibility and effectiveness across different processor architectures is crucial, as microarchitectural events may vary, necessitating adaptability for widespread applicability. Addressing these challenges will contribute to the robustness and reliability of hardware-assisted malware detection techniques.

3) Privacy-Preserving Malware Detection: Balancing accurate malware detection and identification with user privacy has become a growing concern. Future research should prioritize developing privacy-preserving methods that effectively recognize malicious activity while protecting users' sensitive information. This entails exploring innovative techniques for privacy prioritization, ensuring ethical and regulatory alignment. In networked and distributed systems, privacy is a paramount concern due to device interconnectivity. Traditional centralized malware detection raises data privacy concerns. Exploring solutions such as decentralized models leveraging edge computing, blockchain technology, and incorporating differential privacy for peer-to-peer collaboration present a potential to address these privacy challenges.

4) Energy-Efficient Malware Detection: Addressing energy constraints of resource-limited computing systems (e.g. mobile platforms, embedded systems, and biomedical devices) is a crucial focus for advancing HMD methods. The challenge lies in minimizing the energy overhead associated with detection processes to ensure optimal system performance. A promising avenue involves the integration of Tiny Machine Learning (TinyML) techniques, which specialize in deploying lightweight ML models tailored for devices with limited computational capabilities. This approach aims to strike a balance between accuracy and computational efficiency, optimizing the use of hardware features for effective malware detection. In addition, investigating dedicated hardware accelerators and co-processors tailored for on-core malware detection in resource-constrained environments presents a potential solution to offload computational burdens and conserve energy resources, enabling accurate yet efficient malware detection.

V. Conclusion

Hardware performance counter registers serve as dedicated hardware units for tracking applications' performance-related events in modern microprocessors. This paper extensively explored recent advancements in application of machine learning techniques for malware detection based on hardware performance counters profiles, known as Hardware-Assisted Malware Detection (HMD). Additionally, it provides insights into current challenges and future directions for the development of more efficient and advanced intelligent malware detection techniques leveraging hardware performance counters. This work serves as a valuable resource for cybersecurity researchers, offering insights into countering cyber-attacks at the hardware level using machine learning techniques.

VI. Acknowledgment

This work is supported by the National Science Foundation under Award No. 2139034.

References

[1] Y. Ye *et al.*, "A survey on malware detection using data mining techniques," *ACM Computing Surveys*, vol. 50, no. 3, pp. 1–40, 2017.

[2] Y. Li and Q. Liu, "A comprehensive review study of cyber-attacks and cyber security; emerging trends and recent developments," *Energy Reports*, vol. 7, pp. 8176–8186, 2021.

[3] A. Mosenia and N. K. Jha, "A comprehensive study of security of internet-of-things," *IEEE Transactions on Emerging Topics in Computing*, vol. 5, no. 4, pp. 586–602, Oct 2017.

[4] S. M. P. Dinakarrao *et al.*, "Lightweight node-level malware detection and network-level malware confinement in iot networks," in *DATE'19*, March 2019, pp. 776–781.

[5] V. Adat and B. B. Gupta, "Security in internet of things: issues, challenges, taxonomy, and architecture," *Telecommunication Systems*, vol. 67, pp. 423–441, 2018.

[6] P. Nespoli *et al.*, "Optimal countermeasures selection against cyber attacks: A comprehensive survey on reaction frameworks," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 2, pp. 1361–1396, 2017.

[7] H. Sayadi *et al.*, "Towards ai-enabled hardware security: Challenges and opportunities," in *IOLTS*, 2022, pp. 1–10.

[8] N. Kaloudi and J. Li, "The ai-based cyber threat landscape: A survey," *ACM Comput. Surv.*, vol. 53, no. 1, feb 2020.

[9] H. Sayadi *et al.*, "Recent advancements in microarchitectural security: Review of machine learning countermeasures," in *MWSCAS*, 2020, pp. 949–952.

[10] D. Gibert *et al.*, "The rise of machine learning for detection and classification of malware: Research developments, trends and challenges," *J. of Network and Computer Applications*, vol. 153, p. 102526, 2020.

[11] J. Demme *et al.*, "On the feasibility of online malware detection with performance counters," in *ISCA'13*. ACM, 2013, pp. 559–570.

[12] A. Tang *et al.*, "Unsupervised anomaly-based malware detection using hardware features," in *RAID'14*. Springer, 2014, pp. 109–129.

[13] M. Ozsoy *et al.*, "Malware-aware processors: A framework for efficient online malware detection," in *HPCA'15*, 2015.

[14] H. Sayadi *et al.*, "Ensemble learning for effective run-time hardware-based malware detection: A comprehensive analysis and classification," in *Design Automation Conference (DAC'18)*, 2018, pp. 1–6.

[15] H. Sayadi *et al.*, "2smart: A two-stage machine learning-based approach for run-time specialized hardware-assisted malware detection," in *DATE'19*, March 2019, pp. 728–733.

[16] S. Baljit *et al.*, "On the detection of kernel-level rootkits using hardware performance counters," in *ACM AsiaCCS'17*, 2017, pp. 483–493.

[17] H. Sayadi *et al.*, "Towards accurate run-time hardware-assisted stealthy malware detection: a lightweight, yet effective time series cnn-based approach," *Cryptography*, vol. 5, no. 4, p. 28, 2021.

[18] S. Das *et al.*, "Sok: The challenges, pitfalls, and perils of using hardware performance counters for security," in *IEEE Security and Privacy (SP)*, 2019, pp. 20–38.

[19] Z. He *et al.*, "Breakthrough to adaptive and cost-aware hardware-assisted zero-day malware detection: A reinforcement learning-based approach," in *Internatonal Conference on Computer Design (ICCD'22)*, 2022, pp. 231–238.

[20] Y. Gao *et al.*, "Adaptive-hmd: Accurate and cost-efficient machine learning-driven malware detection using microarchitectural events," in *IOLTS'2021*. IEEE, 2021, pp. 1–7.

[21] H. Wang *et al.*, "Hybrid-shield: Accurate and efficient cross-layer countermeasure for run-time detection and mitigation of cache-based side-channel attacks," in *ICCAD'20*, 2020, pp. 1–9.

[22] H. Wang *et al.*, "Phased-guard: Multi-phase machine learning framework for detection and identification of zero-day microarchitectural side-channel attacks," in *ICCD'20*. IEEE, 2020, pp. 648–655.

[23] H. Liu *et al.*, *Feature selection for knowledge discovery and data mining*. Springer Science & Business Media, 2012, vol. 454.

[24] N. Patel *et al.*, "Analyzing hardware based malware detectors," in *DAC'17*, ser. DAC '17. ACM, 2017, pp. 25:1–25:6.

[25] H. Sayadi *et al.*, "Comprehensive assessment of run-time hardware-supported malware detection using general and ensemble learning," in *CF'18*, 2018.

[26] H. Sayadi *et al.*, "Customized machine learning-based hardware-assisted malware detection in embedded devices," *IEEE TrustCom'18*, 2018.

[27] S. M. P. Dinakarrao *et al.*, "Cognitive and scalable technique for securing iot networks against malware epidemics," *IEEE Access*, vol. 8, pp. 138 508–138 528, 2020.

[28] B. Zhou *et al.*, "Hardware performance counters can detect malware: Myth or fact?" in *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*. ACM, 2018, pp. 457–468.

[29] Z. He *et al.*, "Deep neural network and transfer learning for accurate hardware-based zero-day malware detection," in *GLVLSI'22*. NY, USA: ACM, 2022, p. 27–32.

[30] P. Yan and Z. Yan, "A survey on dynamic mobile malware detection," *Software Quality Journal*, May 2017.

[31] D. Powers, "Evaluation: From precision, recall and f-measure to roc, informedness, markedness & correlation," *Journal of Machine Learning Technologies*, vol. 2, no. 1, pp. 37–63, 2011.

[32] "Intel performance monitoring unit," in *https://software.intel.com/en-us/articles/intel-performance-counter-monitor*.

[33] V. J. Reddi *et al.*, "Pin: a binary instrumentation tool for computer architecture research and education," in *in workshop on Computer architecture education: held in conjunction with ISCA*, 2004, pp. 22–es.

[34] P. J. Mucci *et al.*, "Papi: A portable interface to hardware performance counters," in *Proceedings of the department of defense HPCMP users group conference*, vol. 710, 1999.

[35] J. Reinders, "Vtune performance analyzer essentials," *Intel Press*, 2005.

[36] T. Willhalm *et al.*, "Intel® performance counter monitor - a better way to measure cpu utilization," *https://software.intel.com/content/www/us/en/develop/articles/intel-performance-counter-monitor.html*, 2022.

[37] K. London *et al.*, "The papi cross-platform interface to hardware performance counters," in *Department of Defense Users' Group Conference Proceedings*, 2001, pp. 18–21.

[38] C. Malone *et al.*, "Are hardware performance counters a cost effective way for integrity checking of programs," in *ACM Workshop on Scalable Trusted Computing*, ser. STC'11. ACM, 2011, p. 71–76.

[39] K. N. Kh. *et al.*, "Ensemble learning for low-level hardware-supported malware detection," in *RAID'15*, 2015.

[40] K. N. Khasawneh *et al.*, "Rhmd: Evasion-resilient hardware malware detectors," in *2017 50th MICRO*, 2017, pp. 315–327.

[41] S. M. P. Dinakarrao *et al.*, "Adversarial attack on microarchitectural events based malware detectors," in *DAC'19*, 2019, pp. 1–6.

[42] H. Sayadi *et al.*, "Stealthminer: Specialized time series machine learning for run-time stealthy malware detection based on microarchitectural features," in *GLSVLSI'20*, 2020, p. 175–180.

[43] Z. Pan *et al.*, "Hardware-assisted malware detection using explainable machine learning," in *ICCD'20*, 2020, pp. 663–666.

[44] P. Krishnamurthy *et al.*, "Anomaly detection in real-time multi-threaded processes using hardware performance counters," *IEEE Trans. Inf. Forensics Secur*, vol. 15, pp. 666–680, 2020.

[45] H. Kumar *et al.*, "Towards improving the trustworthiness of hardware based malware detector using online uncertainty estimation," in *DAC'21*. IEEE Press, 2021, p. 961–966.

[46] Z. He *et al.*, "When machine learning meets hardware cybersecurity: Delving into accurate zero-day malware detection," in *ISQED'21*, 2021, pp. 85–90.

[47] D. Tian *et al.*, "Mdchd: A novel malware detection method in cloud using hardware trace and deep learning," *Computer Networks*, vol. 198, p. 108394, 2021.

[48] A. P. Kuruvila *et al.*, "Defending hardware-based malware detectors against adversarial attacks," *IEEE TCAD*, vol. 40, no. 9, pp. 1727–1739, 2021.

[49] H. Mohammadi Makrani *et al.*, "Accelerated machine learning for on-device hardware-assisted cybersecurity in edge platforms," in *ISQED'22*, 2022, pp. 77–83.

[50] M. S. Islam *et al.*, "Efficient hardware malware detectors that are resilient to adversarial evasion," vol. 71, no. 11, pp. 2872–2887, 2022.

[51] Z. He and H. Sayadi, "Image-based zero-day malware detection in iomt devices: A hybrid ai-enabled method," in *ISQED'23*, 2023, pp. 1–8.

[52] M. A. Putrevu *et al.*, "Early detection of ransomware activity based on hardware performance counters," in *ACSW'23*, ser. ACSW '23. New York, NY, USA: ACM, 2023, p. 10–17.

[53] M. S. Islam *et al.*, "Stochastic-hmds: Adversarial-resilient hardware malware detectors via undervolting," in *DAC'23*, 2023, pp. 1–6.

[54] H. Sayadi *et al.*, "Cyber-immunity at the core: Securing biomedical devices through hardware-level machine learning defense," in *BioCAS'23*, 2023, pp. 1–5.

[55] H. Sayadi *et al.*, "Redefining trust: Assessing reliability of machine learning algorithms in intrusion detection systems," in *ISCAS'24*, 2024.

[56] X. Wang and R. Karri, "Numchecker: Detecting kernel control-flow modifying rootkits by using hardware performance counters," in *DAC'13*. IEEE, 2013, pp. 1–7.

[57] X. Wang and R. Karri, "Reusing hardware performance counters to detect and identify kernel control-flow modifying rootkits," *IEEE TCAD*, vol. 35, no. 3, pp. 485–498, 2016.

[58] M. B. Bahador *et al.*, "Hpcmalhunter: Behavioral malware detection using hardware performance counters and singular value decomposition," in *ICCKE*, Oct 2014, pp. 703–708.

[59] M. Ozsoy *et al.*, "Hardware-based malware detection using low-level architectural features," *IEEE Transactions on Computers*, vol. 65, no. 11, pp. 3332–3344, Nov 2016.

[60] M. Hall *et al.*, "The WEKA data mining software: An update," *SIGKDD Explor. Newsl.*, vol. 11, no. 1, pp. 10–18, Nov 2009.

[61] N. Alizadeh and M. Abadi, "Akoman: Hardware-level malware detection using discrete wavelet transform," in *2018 IEEE International Conference on Smart Computing (SMARTCOMP)*, 2018, pp. 476–481.

[62] S. J. Stolfo *et al.*, "Towards stealthy malware detection," in *Malware Detection*. Boston, MA: Springer US, 2007, pp. 231–249.

[63] C. Konstantinou *et al.*, "Hpc-based malware detectors actually work: Transition to practice after a decade of research," *IEEE Design and Test*, vol. 39, no. 4, pp. 23–32, 2022.

[64] L. Bilge and T. Dumitraș, "Before we knew it: An empirical study of zero-day attacks in the real world," in *CCS'12*. ACM, 2012, p. 833–844.

[65] J.-Y. Kim *et al.*, "Zero-day malware detection using transferred generative adversarial networks based on deep autoencoders," *Information Sciences*, vol. 460, pp. 83–102, 2018.

[66] O. Suciu *et al.*, "Exploring adversarial examples in malware detection," in *SPW'19*. IEEE, 2019, pp. 8–14.

[67] N. Papernot *et al.*, "The limitations of deep learning in adversarial settings," in *EuroSP'16*, 2016, pp. 372–387.

[68] Y. Liu *et al.*, "Delving into transferable adversarial examples and black-box attacks," in *ICLR'2017*. OpenReview.net, 2017.

[69] X. Wang *et al.*, "Confirm: Detecting firmware modifications in embedded systems using hardware performance counters," in *ICCAD'15*.

[70] X. Wang *et al.*, "Malicious firmware detection with hardware performance counters," *IEEE Transactions on Multi-Scale Computing Systems*, vol. 2, no. 3, pp. 160–173, July 2016.

[71] G. Kornaros, "Hardware-assisted machine learning in resource-constrained iot environments for security: Review and future prospective," *IEEE Access*, vol. 10, pp. 58 603–58 622, 2022.