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

***Abstract*—The growing complexity of modern computing systems increases vulnerability to evolving cyber threats. Recent breakthroughs in computer architecture security utilizes Hardware Performance Counters (HPCs) to access low-level application features, presenting a promising solution to the limitations of traditional software-based defenses. Specialized registers in microprocessors capture diverse hardware-related events, demonstrating efficacy in detecting malicious activities through application of Machine Learning (ML) algorithms. This survey offers a comprehensive analysis of recent advancements in the emerging field of intelligent malware detection based on hardware performance counters, 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 efficient ML-based security countermeasures based on microarchitectural features. This work serves as a helpful resource for researchers in hardware and systems security, offering insights into emerging developments and research directions in countering cyber-attacks at the hardware level using ML techniques.**

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

I. Introduction

Cybersecurity has become a paramount concern, with attackers increasingly exploiting software and hardware vulnerabilities to compromise information technology infrastructures. Recent advances reveal a surge in leveraging emerging hardware vulnerabilities for malicious activities, emphasizing the critical need for robust defenses against malware threats. Malware, a broad term encompassing malicious software, is designed to infiltrate computing systems without user consent, leading to unauthorized data access, file destruction, and other harmful actions. The escalating growth of information technology has amplified the severity of malware as a security threat.

Traditional software-based detection methods, relying on static signature analysis and continuous updates, face feasibility issues in the context of embedded systems due to limitations in computing and communication bandwidth. Additionally, the architecture-dependent nature of advanced analysis techniques further hinders the effective application of existing software-based detection methods to emerging embedded computing devices.

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. 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. Simultaneously, recent breakthroughs in Machine Learning (ML), fueled by the surge in data volume from high-performance computing systems, have yielded successful applications across various domains, especially in enhancing computer system security.

Recent studies in cybersecurity underscore that recognizing malicious activities at the processor hardware and architecture level is achievable through ML techniques. By classifying anomalies in low-level feature spaces, machine learning-based security countermeasures can effectively discern potential threats and respond proactively to evolving behaviors at runtime. These methods employ standard and advanced ML techniques on low-level features, such as microarchitectural events collected by Hardware Performance Counter (HPC) registers, to train classifiers for detecting malicious patterns in running programs. HPCs, specialized registers embedded in modern microprocessors, traditionally used for architectural performance analysis, have emerged as pivotal tools in securing hardware systems. Recent efforts have proposed leveraging HPC information to defend against both malware and microarchitectural security threats, providing a robust foundation for enhancing overall cybersecurity.

Ongoing research studies in intelligent malware detection at the hardware level have explored diverse computing platforms like mobile, embedded systems, IoT, and high-performance systems. State-of-the-art studies predominantly emphasize the development and application of machine learning techniques for evolving malware threats. Our work distinguishes itself by focusing on microarchitectural security studies, offering a unique analysis that delves into recent advancements on employing AI and machine learning techniques protecting the system agains malicious attacks. This article marks the first comprehensive exploration and survey of hardware-assisted malware techniques using machine learning. It addresses existing advances and challenges along with providing valuable insights for the future trajectory of this research. We anticipate that this review will lay the foundations and facilitate further studies, empowering the application of machine learning to combat the growing complexity of cyber threats at the hardware level of processors.

II. Malware Detection Techniques: Background and Taxonomy

In this section, we provide a brief overview of the classification of the existing malware detection methods. Malware detection methods are fundamentally categorized in different categories from different points of view. In this survey, we classified the existing malware detection techniques into two major categories including 1) Misuse-based malware detection, and 2) Anomaly-based malware detection methods [1], [2], [3].

1) Misuse-Based on Malware Detection: Misuse detection is a method for identifying computer security attacks by defining abnormal behavior as an initial reference for determining malicious software. In this approach, anything unknown is considered normal [4], [5], [6]. Signature-based malware detection, a subtype of misuse detection, identifies unique signatures and malicious patterns in malware programs [7], [8]. By profiling applications on computer systems, misuse techniques recognize malware based on predetermined execution signatures and patterns extracted from executed applications [2]. This category of malware detection includes signature-

based, heuristic-based, and cloud-based detection methods. Off-the-Shelf anti-virus software (e.g. Norton, MacAfee, etc.) are considered as traditional software malware detection methods that utilize the signatures of malicious pattern to detect the known malware attacks [9].

Misuse malware detection methods have a notable advantage in their simplicity of incorporating known attacks into the detection model. This is based on the assumption that abnormal behavior corresponds to easily defined malicious patterns, making it an easy-to-use method. However, a significant drawback of misuse detection is its inability to identify unknown security attacks [4], [2]. These methods are vulnerable to alterations that deviate from pre-identified malware signatures. While successful in detecting known attacks, signature-based techniques are incapable of detecting new (unseen) attacks lacking predefined malicious patterns.

2) Anomaly-Based Malware Detection: Anomaly-based or behavior-based malware detection contrasts with misuse detection by defining normal system behavior first and categorizing all other behavior as abnormal or malicious. This approach characterizes attacks based on deviations from predefined models [2] and has the potential to identify new, unknown malware [3], [2], [9], [1]. Anomaly-based detection relies on heuristics or rules, rather than patterns or signatures, aiming to detect any form of malicious activity outside normal system operation [10]. While antivirus solutions traditionally relied on signature-based methods, anomaly-based techniques are now deployed to proactively detect malicious applications, overcoming the limitations of signature-based analysis. Unlike signature-based methods that look for specific patterns, anomaly-based detection searches for certain instructions or rules within application behavior that are absent in normal programs [11], [12]. Thus, behavior-based malware detection methods, employ weight-based or rule-based systems to identify potentially malicious patterns not previously examined by the system [9], [13].

While anomaly-based detection is effective in recognizing unknown threats for real-time protection in antivirus tools, it comes with drawbacks. The scanning and analysis process is time-consuming, degrading computer system performance [3]. Additionally, manually constructing rules/patterns for recognizing unknown malware is error-prone, limiting the effectiveness of heuristic analysis [9]. Behavior-based detection can increase false positives, blocking benign programs and restricting normal computer system operations [13], [3].

Behavior-based malware detection relies on two key phases: a training phase, involving profiling applications to capture their behavior and implementing machine learning (ML) solutions based on the captured information; and a testing phase, where real malware and benign applications are compared with the profiled model from the training phase [14], [10]. These detection techniques predominantly leverage Artificial Intelligence (AI) and data mining methods to identify anomalies [14], [2]. In the feature analysis step, software and hardware-related features are extracted offline to capture file sample characteristics during training. Machine learning techniques then automatically classify file samples into ”benign” or ”malware” categories based on feature representations.

Anomaly-based malware detection employs various features, categorized into software-based and hardware-based approaches. Software features include permissions, network traffic, system calls, API, IPL, information flow, and covert channels. In recent years, there has been a shift toward hardware-based detection, utilizing information like memory access patterns, I/O interface, instruction execution, and low-level microarchitectural features. Hardware-based detectors offer real-time detection, efficient resource utilization, and resistance to infection, proving effective against emerging threats. Recent studies use ML algorithms for accurate and efficient detection of malicious attacks, leveraging low-level features captured from processors' HPCs.

III. Hardware-Assisted Malware Detection: Machine Learning Procedure

Figure 2 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 such as application monitoring for HPC data profiling, feature extraction and analysis, feature selection, and training/testing of the ML-based detector. The continuous learning of ML models through the analysis of low-level microarchitectural features aims to identify 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.

A. Feature Selection: Analysis of Key Features

Developing effective ML-based hardware-assisted malware detectors begins with crucial steps like data collection and feature selection [15], [16], [17], [18]. 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 due to several reasons: a) The abundance of microarchitectural events (e.g., 100+ in Intel Xeon) leads to high-dimensional data [19], b) Processing raw datasets involves computational complexity and induces delays [20], and c) The selection of the most relevant microarchitectural events varies across application classes, posing challenges in specifying non-trivial events for different malware classes [21]. In Figure 1, we illustrate the availability of HPC registers in different processors. As depicted, their numbers are constrained within the range of 2 to 8. The challenge, intricately connected to runtime malware detection, discusses a significant HMD challenge addressed in recent works [19], [21]. 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

Fig. 1: Number of HPC registers available in various processor types

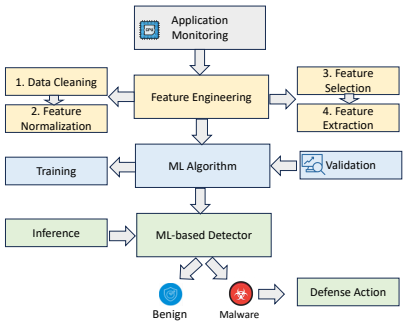

Fig. 2: General overview of hardware-assisted security countermeasures against malware using machine learning

HPC registers, efficient yet accurate runtime detection relies on critical feature selection. Recent HMD studies [19], [21] have addressed effective runtime hardware-assisted malware detection, identifying the minimal set of essential HPCs required for data collection in a single run.

As depicted in Figure 2, 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). There have been some feature selection techniques that are dominant in prior ML-based detection work including correlation attribute evaluation, principle component analysis, gain ratio evaluation, and Fisher Score. As a result, in this section each of these methods will be briefly discussed. Several feature selection techniques have played a prominent role in previous ML-based HMD efforts. These include techniques such as correlation attribute evaluation, principal component analysis, gain ratio evaluation, and Fisher Score. In the following section, a brief overview of each of these methods will be provided.

1) Correlation Attribute Evaluation (CAE): CAE is an algorithmic feature selection method that ranks features based on their importance and relevance to the target classified variable. It evaluates attribute importance by measuring the correlation (Pearson's) between the attribute and the corresponding class, calculated using the Pearson correlation coefficient ρ.

$$\rho(i) = \sum_{k=1}^{n} \frac{(x_{k,i} - \bar{x_i})(c_k - \bar{c})}{\sqrt{\sum_{k=1}^{n}(x_{k,i} - \bar{x_i})^2 \sum_{k=1}^{n}(c_k - \bar{c})^2}} \tag{1}$$

In this algorithm k is the total number of input values; $x_{(k,i)}$ is k^{th} value in input dataset for feature i; c_k is k^{th} value in output dataset. The mean of input data for feature i is denoted by $\bar{x_i}$, and that for the output data by $\bar{c}$. CAE finds correlation co-efficient for all captured features as per above equation and according to the ranking of ρ, top low-level features are selected for analysis and implementing the ML classifiers used for malware detection and classification. Correlation attribute evaluation is a key technique for feature selection in hardware-assisted malware detection, as seen in recent works [19], [22], [23], [21], [24], [25].

2) Principle Component Analysis (PCA): PCA is a statistical method for dimensionality reduction, transforming observations of correlated variables into a new feature space called Principal Components (PC) [26], [27], [28]. It identifies vital microarchitectural parameters by capturing data variation through uncorrelated PC dimensions, providing a linear combination of the original data [29], [30]. This reduction helps determine the most important features along different PC dimensions. It is essential to normalize data to the unit normal distribution to address sensitivity to the relative scaling of original variables [21].

3) Gain Ratio Evaluation (GRE): GRE is an extension of the Information Gain Ratio (IGR) feature selection method introduced in [31]. IGR aims to address the bias in Information Gain (IG) by normalizing it, considering how an attribute divides input samples. The Gain Ratio method is a normalized version of Information Gain, achieved by dividing the information gain by the entropy of the attribute with respect to the class, reducing bias. The entropy after partitioning samples according to a given feature A is denoted as SI(S, A). The IGR is computed using equations involving SI and IG.

$$SI(S, A) = -\sum_{v=1}^{|V|} \frac{|S_v|}{|S|} \times \log_2 \frac{|S_v|}{|S|} \tag{2}$$

$$IGR(S, A) = \frac{IG(S, A)}{SI(S, A)} \tag{3}$$

Gain Ratio evaluation addresses a limitation of the information gain approach, particularly its bias against attributes with a high number of distinct values. In the context of building a decision tree for hardware-assisted malware detection, this bias correction is crucial. Gain Ratio, commonly employed to determine the relevance of collected HPC features for malware detection, assists in making informed decisions about feature selection near the root of the tree. GRE and IGR are widely adopted feature selection method in recent HMD techniques [18], [32], [18], [33].

4) Fisher Score (FS): Fisher Score (FS) is a widely used feature selection method aiming to measure discriminative power by minimizing redundancy and maximizing relevance to the target, such as class labels in classification [34]. In hardware-assisted malware detection, FS identifies top low-level features for malware detection using unsupervised ML techniques, as employed in [2]. FS provides a measure of average inter-class distance compared to intra-class distance for a given feature, helping select features that assign similar values to data samples in the same class and different values to samples from different classes [35], [36]. As a result, the FS for the i_th feature denoted as Si will be calculated as follows:

$$S_i = \frac{\sum_{k=1}^{K} n_j(\mu_i j - \mu_i)^2}{\sum_{k=1}^{K} n_j(\rho_i j)^2} \tag{4}$$

where $\mu_i j$ and $\rho_i j$ are the mean and the variance of the i_th feature in the j_th class respectively, n_j is the number of instances in the j_th class, and μ_i is the mean of the $i - th$ feature. Higher values of S_k imply that while members in the same class are closer together, the members belonging to different classes are further separated using the k_th feature. For the malware detection problem which is often a two-class problem including the positive class (malicious) or or negative class (benign), the FS method simplifies [36]. FS helps select a subset of significant features for distinguishing malicious patterns, but it does not consider the correlation and mutual dependency between features, which could be a limitation [2].

B. ML Techniques for Malware Detection

Data mining and machine learning techniques have shown to be effective in classifying anomalies to accurately detect the behavior of malicious applications. As mentioned before, Figure 2 illustrates the overview of malware detection process using such classification techniques.

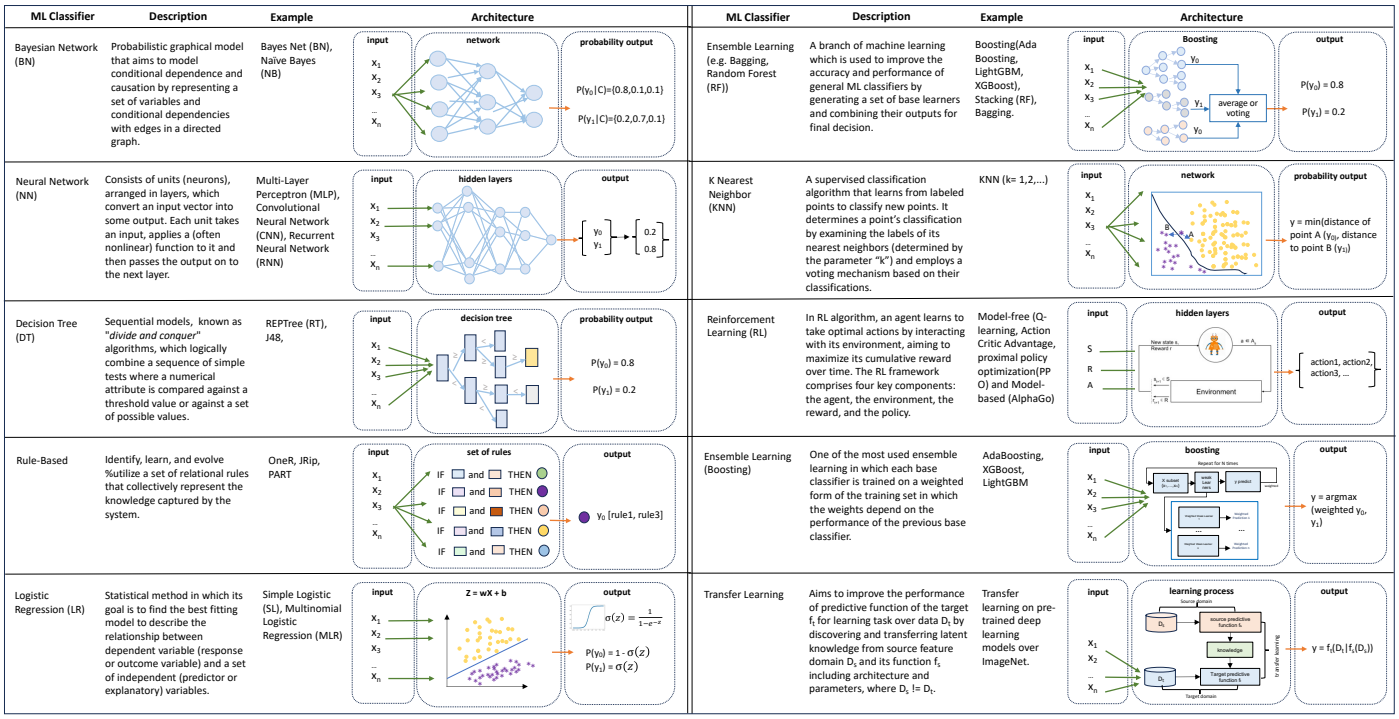

Fig. 3: Various branches of machine learning algorithms used in HMD techniques, examples, and architectures

To categorize the unknown applications into either benign or malicious software, 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 data (the HPCs information) for malware detection. 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 (e.g., Artificial Neural Network (ANN), Logistic Regression (LR), Decision Tree (DT), and Support Vector Machines (SVM), etc).

By analyzing the training file samples, the deployed ML classification 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, first the vectors of the new file samples are first extracted using the same feature extraction techniques as in the training phase. This completely unseen data then is fed to the trained classifier to examine the detection rate of malware detection process. The classifier attempts to classify the new file samples based on the extracted feature vectors. Figure 3 illustrates various branches of machine learning algorithms used in HMD techniques. We briefly describe the ML models, along with their examples and architectures, that have been used in intelligent hardware-assisted malware detection research.

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 ML and statistics, various metrics are employed to evaluate the performance of a detection method.Table I provides a consolidated overview of the evaluation metrics utilized for performance analysis of ML-based security countermeasures.

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) [37]. 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). AUC measures the entire area under the ROC, with higher values indicating better detection performance.

D. Performance Monitoring Tools

In this section, we introduce some most prevalent performance monitoring tools under different operating systems (Windows, Linux, and macOS systems) used in prior works

TABLE I: Evaluation metrics used for analyzing ML-based security countermeasures

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).

to monitor applications behavior and collect hardware-related events that assist in application performance analysis and tuning. In order to monitor applications behavior and collect hardware-related events that assist in application performance analysis and tuning various performance monitoring tools have been used in prior works. These tools include Perf [38], Pin [39], PAPI [40], Intel VTune [41], and Intel PCM [42]. All these tools are available for Linux systems while only Intel VTune and Intel PCM are able to monitor HPCs in Windows and macOS systems. Perf, PAPI, and Pin demand some knowledge of command lines for users due to the lack of 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) [43]. 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, branch predictability, etc. to examine the applications behavior [39].

Performance Application Programming Interface (PAPI) [44] provides a cross-platform interface for monitoring hardware performance counters on processors that are equipped with specific registers for hardware events. To help with discovering and resolving performance bottlenecks in running programs for tweaking and debugging purposes, Intel has developed a licensed-based tool called Vtune [41]. It can record and show performance-related information. 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 tool. Last but not least, PCM [42], [45] is the performance monitoring units (PMU) implemented in Intel's processors (e.g., Xeon, Atom, and Xeon Phi) that help 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.

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

In this section, we survey the latest proposals on hardware-assisted malware detection using ML techniques. Figure 4 illustrates the yearly analysis of publications on hardware-assisted malware detection techniques, captured from Google Scholar. For the purpose of thorough and structured exploration, we classify prior studies into several categories according to each work's specific focus and the challenges addressed.

A. General Hardware Malware Detection

The study by Demme et al. [15] pioneered the exploration of HPCs for accurate malware detection via ML techniques. The research successfully demonstrated the efficacy of offline

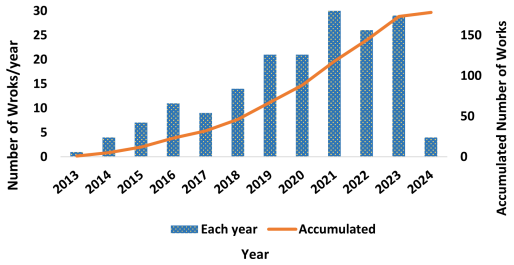

Fig. 4: Comprehensive yearly analysis of state-of-the-art researches on intelligent malware detection using hardware performance counters

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 [46] 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, but it showed the potential of HPCs for security applications with minimal runtime overhead.

Tang et al. [2] 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. [70], [71] utilize HPCs information to detect rootkits that modify system calls through statistical methods. These approaches focus on counting 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 employ 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 [72] presented HPCMalHunter, an anomaly-based malware detection technique utilizing machine learning classifiers at the hardware level. This framework, termed a behavioral online malware detector, predicts malware

TABLE II: Overview of recent research on hardware-assisted malware detection with machine learning techniques

Research	Year	Platform	Classification Model	Threat Type	Major Contribution and Challenge Addressed
[15]	2013	Android, Linux	KNN, NN, DT, RF	Malware	It was the first work to use HPCs in ML-based malware detection on OS level, their findings show that data from performance counters can be used to identify malware and that the trained ML model using HPCs are are robust to minor variations in malware programs.
[47]	2014	Linux	ocSVM	Malware	It demonstrated that HPC features combined with unsupervised machine learning using benign program executions can detect deviations from malware programs. Their findings show the microarchitectural characteristics of benign and malware programs have different patterns, while benign are noisy, and the deviations exhibited by malware are minute.
[48]	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, synthesizing the resulting design to run on an FPGA.
[49]	2015	Windows	LR, NN, EL	Backdoor, PWS, Rogue, Trojan, Worm	Focus on per class malware detection using HPCs, compared the accuracy of specialized and general malware detectors, highlighting the effectiveness of specialization in HMD.
[50]	2016	ARM	ocSVM	Firmware malware	Introduced as a cost-efficient technique for identifying malicious changes in embedded control system firmware using MLs.
[18]	2017	Linux	SVM, ocSVM, NB, DT	Kernel Rootkits	utilizing the 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.
[22]	2017	Linux	OneR, MLP, BN, SMO, SGD, LR	Malware	Assessed detectors based on accuracy, accuracy/area, Power Delay Product (PDP), and latency.
[51]	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.
[19]	2018	Linux	BN, J48, JRip, MLP, OneR, RT, SGD, SMO, AB, BG	Malware	It was the first paper to break the trade-off of more HPCs used for training effective MLs than the limited number of HPC registers available in today's microprocessors. Their work shows using ensemble machine learning solutions can be effective in training runtime malware detectors with only 2 or 4 HPCs which makes online malware detection more available.
[23]	2018	Linux	AdaBoost on five ML classifiers	Malware	Explored ensemble learning's effectiveness, comparing general and ensemble classifiers in terms of accuracy, robustness, performance, and hardware overhead.
[52]	2018	Windows	DT, RF, MLP, KNN, AB, NB	Malware	Questioned the effectiveness of HPC based method, revealing larger performance variations across different MLs.
[24]	2018	Linux	MLs	Ess Malware.	Presenting a customized ML-based HMD and identification solution for embedded systems.
[21]	2019	Linux	J48, JRip, MLP, OneR, AB	Virus, Trojan, Rootkits, Backdoor	A two-stage ML-based approach for specialized runtime malware identification and detection.
[53]	2019	Linux	Adversarial malware (Backdoor, Rootkit, Virus, Worm, Trojans)	LR, NN	It created adversarial attacks on HMD systems by injecting perturbations into HPC traces. They first employ an adversarial sample predictor to predict the number of HPC features used on the target system, then use reverse-engineered malware samples together with perturbed noise to compromise victim systems. Their results show such adversarial attacks can be effective in evading malware detections.
[54]	2019	Linux	MLP, OneR, LR, JRip	IoT malware	It addressed security risks in large-scale IoT networks. They proposed HaRMemploys with low computational overhead ML classifier (OneR) suitable to the needs of IoT devices with good malware detection accuracy. They developed a framework for combining the models of malware spreading processes on networks explicitly with their direct adverse effects on network performance and formulate an optimal control problem for malware confinement while maintaining network integrity. The experiment results show this method can be used to generate imperfect estimates of infection state in IoT network.
[25]	2020	Linux	MLP, OneR, LR, J48, SVM, SGD	IoT malware	proposing a two-staged framework with a lightweight malware detector followed by a stochastic controller.
[55]	2020	Linux	J48, JRip, LR, KNN, BOFF, FCN	Stealthy Malware (Trojan, Rootkits, Backdoor, Blended)	First to address the challenge of stealthy malware using HPCs, proposed a specialized time series ML approach for stealthy malware detection.
[56]	2020	Xilinx Zynq7000 SoC	RNN, Linear Regression	Malware	It explores the interpretability of ML HMD. They first theoretically establish that the proposed method can provide an interpretable explanation of classification results to address the challenge of transparency. Then they show through explainable ML, detection performance can be improved. Their results show promising detection performance.
[57]	2020	Linux	oc SVM,	Malware	They use time series benign data only to train one class SVM in a multi-thread environment, which considers per-thread and cross-thread features in embedded devices. Their approach enables continuous monitoring of time series of multi-thread HPC readings for run-time malware detection.
[58]	2021	Linux	time series FCN, MLP,ResNet, MCDNN, JRip, J48, LR, KNN, BOPF	Stealthy Malware (Trojan, Rootkits, Back- door, Blended)	Proposed StealthMiner, a lightweight time series Fully Convolutional Neural Network (FCN) to accurately detect stealthy malware trace at run-time using branch instructions.
[59]	2021	Linux	RF	Malware	Quantifying ML with uncertainty, introduced an uncertainty estimator to consider undertain predictions when handling zero-day malware.
[60]	2021	Linux	EL(AB), RF, DT, GNB, SGD, LR	Zero-day malware	Suggest classical MLs were found effective for known malware but suffered a high FPR in zero-day malware detection, proposed Ada Boosting over RandomForest for zero-day malware detection.
[61]	2021		CNN	Cloud malware	Real-time malware detection in cloud (IaaS) environment.
[33]	2022	Linux	CNN, TL, RF,DT,	Zero-day malware	Image-based transfer learning on tabular HPC data proved to be effective in detecting zero-day malware.
[62]	2022	Linux	DT, NN	Adversarial malware	Moving target defense of adversarial defense by changing the numbers and set of HPCs and classifiers.
[63]	2022	Android	BN,SL,MLP,PART,SMO	Malware	Propose 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.
[64]	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.
[32]	2022	Linux	RL(UCB), EL,JRip, J48, LR, MLP, OneR, RepTree	Zero-day malware	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.
[65]	2023	Linux	RL(A2C), CNN, MLs	IoMT Malware	Proposed a hybrid and adaptive image-based framework for online hardware-assisted zero-day malware detection.
[66]	2023	Linux	SVM, RF, GBM, AB	Ransomware	Determining the optimal hardware features and time granularity for early ransomware detection.
[67]	2023	Linux	MLP, LR, DT	Adversarial malware (backdoors, rogues, password stealers, trojans, and worms)	Enhance adversarial defense by adding stochastic noise (through controlled undervolting) in HMDs' computations during inference.
[68]	2023	Linux	Various binary & multi-classifiers.	BioMedical malware	Introduces a tailored hardware monitoring framework and employs MLs to enhance cybersecurity in biomedical computing systems.
[69]	2024	Linux	Various binary & multi-classifiers.	Malware	A thorough evaluation of MLs' reliability considering factors of training data size, the number of HPCs used, and internal data separability (malware stealthiness); introduced model observer to enhance reliable intrusion detection during inference.

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, Fully Convolutional Network: FCN, TL: Transfer Learning, Reinforcement Learning: RL, Actor Critic Advantage: A2C, Upper Confidence Bound: UCB, Embedded Systems: ESs, High Level Synthesize: HLS, SimpleLogic: SL, Sequential Minimal Optimization: SMO.

presence with high accuracy using a Support Vector Machine (SVM) classifier. Initially, the detector gathers a set of hardware performance counter events concurrently from the running application. In the subsequent step, the authors employ a Singular Value Decomposition (SVD)-based feature reduction technique to identify the most significant HPC events.

In studies [48], [73], 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 [74], 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. [48], 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 [18] focuses on utilizing ML algorithms on synthetic traces of HPC features to detect kernel-level rootkit attacks. To reduce features, they employ the Gain Ratio technique from the WEKA toolkit [75], achieving high accuracy in detecting synthetic rootkits. The study collects 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 [22], 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. [19], [23] proposed ensemble machine learning solutions for effective runtime malware detection using low-level microarchitectural features. To optimize runtime detection with limited hardware performance counters, the authors employed systematic feature reduction. 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 [23], the authors explored ensemble learning's effectiveness with only 8 HPC features, equivalent to the available number of HPC registers. They applied AdaBoost on five ML classifiers, comparing general and ensemble classifiers in terms of accuracy, robustness, performance, and hardware overhead.

The research in [19], the challenge of limited HPC registers for runtime 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 runtime malware detection.

Furthermore, the work in [21] proposed a two-stage machine learning-based approach for specialized runtime 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.

In [76], unlike prior HMD research, the suitability of low-level microarchitectural features for distinguishing malware from benign applications is questioned. The authors argue that there is no inherent relationship between low-level microarchitectural features and high-level application behavior. They contend that positive results in previous works stem from optimistic assumptions, presenting their best result with an F1-score of 80.78%. However, they conduct 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.

The research in [77], presents 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. DWT serves as a tool to reduce the dimensionality and noise in measurement traces. In work [66], the focus is on determining the optimal hardware features and time granularity for early ransomware detection. The study examines HPC counter statistics gathered at intervals of 100ms, 500ms, and five seconds. They train several classical ML models to compare configurations and find that capturing 5 HPC registers every 100ms for the first 3 seconds of payload execution achieves the best results with AdaBoost classifier, achieving an accuracy above 90%.

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

Stealthy attacks involve concealing malicious code within benign applications, making detection more challenging [78]. 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 [55], [79] 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 cyberattacks 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 introduce StealthMiner, a specialized time series ML approach based on the Fully Convolutional Network (FCN), aiming to detect stealthy malware, embedded within benign traces, at runtime using the time series branch instructions feature, the most prominent hardware event.

The study in [59] proposes an ensemble-based (bagging) approach for quantifying uncertainty in predictions made by ML models in HMD techniques. The study introduces an uncertainty estimator, showing that considering uncertain predictions enables ML models to handle zero-day malware. Furthermore, the work in [80] employs a power grid case study to demonstrate that HPC effectively detect 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 have 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 [81]. In addressing the challenge of zero-day malware detection, He et al. [60] conducted experiments using machine learning for detecting 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 Ada-boosting over the robust Random Forest, they achieved a 4% improvement in F1-score for zero-day malware detection. Additionally, in a recent study [33], a two-staged zero-day malware detection method based on deep transfer learning is 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 [33] involves converting four HPC features into two-dimensional images and applying transfer learning with a pre-trained ResNet model on ImageNet to enhance learning hidden patterns of zero-day malware. Results demonstrates the effectiveness of transfer learning for HMD, addressing an open question in current research.

Furthermore, He et al. [32] is the first work that tackles major issues in adaptive and cost-aware zero-day malware detection using low-level hardware events. They propose a unified feature selection method based on heterogeneous feature fusion to determine prominent HPC events for on-device HMD. Additionally, the authors introduce 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 utilizes 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 demonstrates Reinforced-HMD's accuracy and robustness, achieving a 96% F1-score and AUC metrics.

In a recent study [69], the authors focus on the reliability analysis of hardware-oriented Intrusion Detection Systems (IDSs). While ensuring the dependability of ML models' decisions is crucial, it has been largely overlooked in previous studies. This work conducts 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 runtime, 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 [82], [83], [84]. 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 evolving adversarial attacks, especially those targeting hardware features, is a critical challenge. In the context of HMD, which relies on microarchitectural events captured via HPCs, [53] demonstrated an adversarial attack on HMD systems. This attack involves 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 and side-channel attack detectors.

In [51], the authors propose a resilient solution to defend ML-based HMD against reverse engineering. They highlight HMD's vulnerability to adversarial attacks and emphasize that retraining on adversarial malware datasets is ineffective. To address this, they suggest constructing detectors with randomized features and Hardware Performance Counter (HPC) collection periods, switching them stochastically to thwart attackers' predictions. In work [62], a Moving Target Defense (MTD) technique is proposed for adversarial attacks on HMD. MTD dynamically changes the number and set of performance counters and the classifier, confusing attackers. It uses random selection of 4 features and 2 ML models (Decision Tree, Neural Network) at runtime, successfully defending against adversarial attacks without performance degradation. However, its effectiveness against non-adversarial malware attacks remains untested. Similarly, in [64], the authors demonstrate the effectiveness of reverse-engineering ML models while highlighting the limitations of retraining for runtime attacks. They propose 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. [67] address the adversarial attacks through reverse engineering. They propose Stochastic-HMDs, which introduce stochastic noises into the computations of model inference to defend reverse engineering based adversarial attacks. They manipulate 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. [24] extends 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 identifies 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 addresses the constraints of embedded systems by leveraging HPC features. It employs various ML classifiers to detect and classify different malware classes at runtime, using four crucial HPC features: branch instructions, cache references, branch misses, and node-stores. This approach aims to achieve accurate and effective runtime malware detection despite the limited computing power and resources in embedded systems.

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 [54], the authors address security risks in large-scale IoT networks, emphasizing the challenges of malware propagation. Traditional approaches fall short, leading to the introduction of HaRM, a runtime 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 [25] the authors expand their prior work exploring network effects, proposing a two-staged 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 al-

terations for malware detection, proving effective in detecting firmware modifications. In the study presented in [**?**], ConFirm is introduced as a cost-efficient technique for identifying malicious changes in embedded control system firmware. The approach involves measuring low-level hardware events captured by HPC registers during firmware execution. The evaluation encompasses 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 [50] to automatically extract relations between different hardware features.

The study described in [56] explores the interpretability of HMD techniques by introducing a framework that employs explainable machine learning. This framework enhances the explainablity of classification results, making them more accessible and understandable for human analysis. They collect time series HPC data from Xilinx Zynq7000 SoC ZC702 evluation board, train an RNN, and use the model's output to train a linear regression model for feature contribution factors. This interprets which HPC features contribute to malicious attacks and when they occur.

Moreover, in study [57], a one-class SVM is built using only benign data to classify normal and malware on programmable logic controllers (PLCs). PLCs often have a mix of timing-based and event-triggered components in a multi-threaded environment. The approach monitors 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 [85] explores research landscape utilizing performance counters and ML for IoT device security. The study delves 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 [65], 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 selects the best Deep Neural Network (DNN) detector at runtime from a pool of continuously trained models, customized for each device. Tabular hardware-based data are converted into small-size images using transfer learning, enhancing 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 runtime. This decision-making process ensures highly accurate zero-day malware detection using a limited number of hardware events, leading to high malware detection performance.

Tian et al. [61] address 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 captures control flow information outside the target VM. They convert this information into two-dimensional color images and employ a CNN-based method for malware detection. This represents the sole effort in cloud-based malware detection utilizing HPCs. Although its performance may not reach state-of-the-art levels, it underscores potential avenues for future research in cloud security.

The work in [63] proposes accelerated ML for efficient on-device HMD. They propose 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. The study in [68] extends the applicability of HMDs to healthcare systems, particularly for biomedical computing systems. The work introduces a tailored hardware monitoring framework and employs ML to improve the accuracy and efficiency of malware detection and classification using real-time data from biomedical processors' hardware events. Experimental results highlight the effectiveness of the XGBoost model, achieving a 95% detection rate in F-measure and accuracy with efficient resource utilization and low inference latency using only four hardware events. biomedical processors' hardware events. Their experimental results demonstrate that the XGBoost model, using 4 hardware events, excels in malware detection, boasting 95% detection rate in F measure and accuracy, efficient resource utilization, and low inference latency.

V. 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 must 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) Challenge of HPC Information for Security Implications and Cross-Architecture Compatibility: The growing number of microarchitectural events in modern processors, surpassing the growth of physical HPC registers, demands effective feature reduction techniques and machine learning algorithms. Ambiguous documentation and indefinite inference of low-level features pose challenges, requiring expert-level understanding for accurate execution and data capture. Complexity variations across architectures make consistent hardware performance counter-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 methods that effectively discern malicious activity

while safeguarding sensitive user information. This entails exploring innovative techniques and frameworks for privacy prioritization, ensuring ethical and regulatory alignment. This challenge is vital for instilling user trust and compliance. 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. Minimizing the energy overhead introduced by detection processes is a key direction for future research. Moreover, the exploration of dedicated hardware accelerators and co-processors designed specifically for malware detection in resource-constrained environments offers a potential solution to offload computational burdens and conserve energy resources.

VI. Conclusion

Hardware performance counter registers are hardware units that are designed to count low-level micro-architectural events in modern microporcessors. Many research proposals have investigated the usage of machine learning techniques for malware detection using Hardware performance counters profiles. This survey attempted to provide a comprehensive overview and comparative analysis of the state-of-the-art studies in the field of hardware-based malware detection techniques using artificial intelligence and machine learning techniques. Furthermore, based on the current research challenges and pitfalls of hardware-assisted malware detection, this work forecasts the future research trends by providing insights into future directions of efficient and enhanced hardware-assisted malware detection techniques.

References

[1] H. Yin and et al., "Panorama: Capturing system-wide information flow for malware detection and analysis," in *Proceedings of the 14th ACM Conference on Computer and Communications Security*, ser. CCS '07. ACM, 2007, pp. 116–127.

[2] A. Tang and et al., "Unsupervised anomaly-based malware detection using hardware features," in *RAID'14*, 2014.

[3] Z. Bazrafshan and et al., "A survey on heuristic malware detection techniques," in *The 5th Conference on Information and Knowledge Technology*, May 2013, pp. 113–120.

[4] R. Cathey and et al., "Misuse detection for information retrieval systems," in *Proceedings of the twelfth international conference on Information and knowledge management*. ACM, 2003, pp. 183–190.

[5] "Misuse," in *https://en.wikipedia.org/wiki/Misuse detection*.

[6] P. Helman and et al., "Foundations of intrusion detection (computer security)," in *Computer Security Foundations Workshop V, 1992. Proceedings.* IEEE, 1992, pp. 114–120.

[7] S. A. Hofmeyr and et al., "Intrusion detection using sequences of system calls," *Journal of Computer Security.*, vol. 6, no. 3, pp. 151–180, Aug. 1998.

[8] A. Somayaji and S. Forrest, "Automated response using system-call delays," in *Proceedings of the 9th Conference on USENIX Security Symposium - Volume 9*, ser. SSYM'00. Berkeley, CA, USA: USENIX Association, 2000, pp. 14–14.

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

[10] "Anomaly based intrusion detection," in *https://en.wikipedia.org/wiki/Anomaly based intrusion detection system*.

[11] M. Polychronakis and et al., "Comprehensive shellcode detection using runtime heuristics," in *Proceedings of the 26th Annual Computer Security Applications Conference*, ser. ACSAC '10. ACM, 2010, pp. 287–296.

[12] M. Polychronakis and et al., "Emulation-based detection of non-self-contained polymorphic shellcode," in *Proceedings of the 10th International Conference on Recent Advances in Intrusion Detection*, ser. RAID'07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 87–106.

[13] M. Schmall, "Heuristic techniques in av solutions: An overview," in *https://www.symantec.com/connect/articles/heuristic-techniques-av-solutions-overview*, 2002.

[14] I. Khalkhali and et al., "Host-based web anomaly intrusion detection system, an artificial immune system approach," 2011.

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

[16] H. Sayadi and et al., "Towards ai-enabled hardware security: Challenges and opportunities," in *2022 IEEE 28th International Symposium on On-Line Testing and Robust System Design (IOLTS)*. IEEE, 2022, pp. 1–10.

[17] H. Sayadi and et al., "Recent advancements in microarchitectural security: Review of machine learning countermeasures," in *MWSCAS'20*, 2020, pp. 949–952.

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

[19] H. Sayadi and et al., "Ensemble learning for effective run-time hardware-based malware detection: A comprehensive analysis and classification," in *Design Automation Conference*, 2018.

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

[21] H. Sayadi and et al., "2smart: A two-stage machine learning-based approach for run-time specialized hardware-assisted malware detection," in *Design, Automation Test in Europe Conference Exhibition (DATE'19)*, March 2019, pp. 728–733.

[22] N. Patel and et al., "Analyzing hardware based malware detectors," in *Proceedings of the 54th Annual Design Automation Conference 2017*, ser. DAC '17. ACM, 2017, pp. 25:1–25:6.

[23] H. Sayadi and et al., "Comprehensive assessment of run-time hardware-supported malware detection using general and ensemble learning," in *ACM International Conference on Computing Frontiers, CF*, 2018.

[24] H. Sayadi and et al., "Customized machine learning-based hardware-assisted malware detection in embedded devices," *The 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications (IEEE TrustCom-18)*, 2018.

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

[26] M. Malik and et al., "Co-locating and concurrent fine-tuning mapreduce applications on microservers for energy efficiency," in *Workload Characterization (IISWC), 2017 IEEE International Symposium on*. IEEE, 2017, pp. 22–31.

[27] "Trojan horse (computing)," in *https://en.wikipedia.org/wiki/Principal component analysis*.

[28] A. E. Serpen G., "Host-based misuse intrusion detection using pca feature extraction and knn classification algorithms," in *Intelligent Data Analysis*, 2018.

[29] S. Wold and et al., "Principal component analysis," *Chemometrics and Intelligent Laboratory Systems*, vol. 2, no. 1, pp. 37 – 52, 1987,

proceedings of the Multivariate Statistical Workshop for Geologists and Geochemists.

[30] C. O. S. Sorzano and et al., "A survey of dimensionality reduction techniques," *ArXiv e-prints*, Mar. 2014.

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

[32] Z. He and et al., "Breakthrough to adaptive and cost-aware hardware-assisted zero-day malware detection: A reinforcement learning-based approach," in *2022 IEEE 40th International Conference on Computer Design (ICCD)*, 2022, pp. 231–238.

[33] Z. He and et al., "Deep neural network and transfer learning for accurate hardware-based zero-day malware detection," in *Proceedings of the Great Lakes Symposium on VLSI 2022*, ser. GLSVLSI '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 27–32.

[34] R. Moskovitch and et al., "Unknown malcode detection using opcode representation," in *Intelligence and Security Informatics*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 204–215.

[35] R. Duda and et al., *Pattern Classification*. John Wiley & Sons, 2001.

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

[37] D. M. Powers, "Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation," 2011.

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

[39] V. J. Reddi and et al., "Pin: a binary instrumentation tool for computer architecture research and education," in *Proceedings of the 2004 workshop on Computer architecture education: held in conjunction with the 31st International Symposium on Computer Architecture*, 2004, pp. 22–es.

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

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

[42] T. Willhalm *et al.*, "Intel performance counter monitor-a better way to measure cpu utilization," *Dosegljivo: https://software. intel. com/en-us/articles/intelperformance-countermonitor-a-better-way-to-measure-cpu-utilization.[Dostopano: September 2014]*, 2012.

[43] *https://perf.wiki.kernel.org/index.php*, last accessed: 20-Feb-2019.

[44] 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.

[45] 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*, 2017.

[46] C. Malone and et al., "Are hardware performance counters a cost effective way for integrity checking of programs," in *Proceedings of the Sixth ACM Workshop on Scalable Trusted Computing*, ser. STC'11. New York, NY, USA: Association for Computing Machinery, 2011, p. 71–76.

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

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

[49] K. N. Khasawneh and et al., "Ensemble learning for low-level hardware-supported malware detection," in *RAID'15*, 2015, pp. 3–25.

[50] X. Wang and 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.

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

[52] B. Zhou and et al., "Hardware performance counters can detect malware: Myth or fact?" in *ASIACCS'18*, 2018, pp. 457–468.

[53] S. M. P. Dinakarrao and et al., "Adversarial attack on microarchitectural events based malware detectors," in *Proceedings of the 56th Annual Design Automation Conference 2019*, 2019, pp. 1–6.

[54] S. M. P. Dinakarrao and et al., "Lightweight node-level malware detection and network-level malware confinement in iot networks," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2019, pp. 776–781.

[55] H. Sayadi and 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.

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

[57] P. Krishnamurthy and et al., "Anomaly detection in real-time multi-threaded processes using hardware performance counters," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 666–680, 2020.

[58] H. Sayadi and 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, 2021.

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

[60] Z. He and et al., "When machine learning meets hardware cybersecurity: Delving into accurate zero-day malware detection," in *2021 22nd International Symposium on Quality Electronic Design (ISQED)*, 2021, pp. 85–90.

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

[62] A. P. Kuruvila and et al., "Defending hardware-based malware detectors against adversarial attacks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 9, pp. 1727–1739, 2021.

[63] H. Mohammadi Makrani and et al., "Accelerated machine learning for on-device hardware-assisted cybersecurity in edge platforms," in *2022 23rd International Symposium on Quality Electronic Design (ISQED)*, 2022, pp. 77–83.

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

[65] Z. He and H. Sayadi, "Image-based zero-day malware detection in iomt devices: A hybrid ai-enabled method," in *2023 24th International Symposium on Quality Electronic Design (ISQED)*, 2023, pp. 1–8.

[66] M. A. Putrevu and et al., "Early detection of ransomware activity based on hardware performance counters," in *Proceedings of the 2023 Australasian Computer Science Week*, ser. ACSW '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 10–17.

[67] M. S. Islam and et al., "Stochastic-hmds: Adversarial-resilient hardware malware detectors via undervolting," in *2023 60th ACM/IEEE Design Automation Conference (DAC)*, 2023, pp. 1–6.

[68] H. Sayadi and et al., "Cyber-immunity at the core: Securing biomedical devices through hardware-level machine learning defense," in *2023 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, 2023, pp. 1–5.

[69] H. Sayadi and et al., "Redefining trust: Assessing reliability of machine learning algorithms in intrusion detection systems," in *2024 IEEE International Symposium of Circuits and Systems (ISCAS)*, 2024.

[70] X. Wang and R. Karri, "Numchecker: Detecting kernel control-flow modifying rootkits by using hardware performance counters," in *Design Automation Conference (DAC), 2013 50th ACM/EDAC/IEEE*. IEEE, 2013, pp. 1–7.

[71] X. Wang and R. Karri, "Reusing hardware performance counters to detect and identify kernel control-flow modifying rootkits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 3, pp. 485–498, 2016.

[72] M. B. Bahador and et al., "Hpcmalhunter: Behavioral malware detection using hardware performance counters and singular value decomposition," in *2014 4th International Conference on Computer and Knowledge Engineering (ICCKE)*, Oct 2014, pp. 703–708.

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

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

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

[76] B. Zhou and 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.

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

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

[79] H. Sayadi and 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.

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

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

[82] O. Suciu and et al., "Exploring adversarial examples in malware detection," in *2019 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2019, pp. 8–14.

[83] N. Papernot *et al.*, "The limitations of deep learning in adversarial settings," in *2016 IEEE European Symposium on Security and Privacy (EuroS P)*, 2016, pp. 372–387.

[84] Y. Liu *et al.*, "Delving into transferable adversarial examples and black-box attacks," in *5th International Conference on Learning Representations, ICLR 2017,Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[85] 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.