IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received February 6, 2022, accepted February 24, 2022, date of publication March 10, 2022, date of current version March 18, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3157812

Performance Monitoring Counter Based Intelligent Malware Detection and Design Alternatives

JORDAN PATTEE, SHAFAYAT MOWLA ANIK, AND BYEONG KIL LEE, (Senior Member, IEEE)
Department of Electrical and Computer Engineering, University of Colorado Colorado Springs, Colorado Springs, CO 80918, USA

Corresponding author: Byeong Kil Lee (blee@uccs.edu)

This work was supported in part by UCCS, and in part by NVIDIA Corporation.

ABSTRACT Hardware solutions for malware detection are becoming increasingly important as software-based solutions can be easily compromised by intelligent malware. However, the cost of hardware solutions including design complexity and dynamic power consumption cannot be ignored. Many of the existing hardware solutions are based on statistical learning blocks with abnormal features of system calls, network traffics, or processor behaviors. Among those solutions, the performance of the learning techniques relies primarily on the quality of the training data. However, for the processor behavior-based solutions, only a few behavioral events can be monitored simultaneously due to the limited number of PMCs (Performance Monitoring Counters) in a processor. As a result, the quality and quantity of the data obtained from architectural features have become a critical issue for PMC-based malware detection. In this paper, to emphasize the importance of selecting architectural features for malware detection, the statistical differences between malware workloads and benign workloads were characterized based on the information from performance counters. Most malware can easily be detected with basic characteristics, but some malware types are statistically very similar to benign workloads which need to be handled more in-depth. Hence, we focus on multiple steps to investigate critical issues of PMC-based malware detection: (i) statistical characterization of malware; (ii) distribution-based feature selection; (iii) trade-off analysis of detection time and accuracy; and (iv) providing architectural design alternatives for hardware-based malware detection. Our results show that the existing number of performance counters is not enough to achieve the desired accuracy. For more accurate malware detection in real-time, we propose both accuracy improvement schemes (with additional PMCs, etc.) and hardware acceleration schemes. Both schemes provide accuracy improvement (5∼10%) and detection speedup (up to 10%) with the additional hardware cost (less than 1% of the chip complexity).

INDEX TERMS Hardware acceleration, machine learning, malware detection, workload characterization.

I. INTRODUCTION

As Internet technologies and smart devices are explosively growing, data is becoming more prevalent. Threat data has no exception. Research on computer security has dedicated a significant amount of effort to malware detection with multiple approaches, but automated analysis and detection of malware remain open issues. Software-based detection can remove harmful programs with a static signature-based detection mechanism. However, the detectors can be easily compromised as the usage of obfuscation techniques becomes more common in malware, which allows the malware to generate new patterns of signatures at runtime [1], [2]. Another issue of the static signature-based detectors is that they can also impact the performance of the host processor. For the past two decades, security has been a second or third consideration in computer systems design because priority has always been given to performance, power, and area (PPA). Consequently, in a performance-oriented architecture design, inherent security risks exist that are associated with architectural modules such as branch prediction, caches, instruction prefetching module, etc. These architecture-level vulnerabilities are difficult to remove due to the conflict of interests between system performance and security. In contrast, dedicated hardware towards security such as ARM TrustZone can be operated without burdening the host processor. However, the hardware

The associate editor coordinating the review of this manuscript and approving it for publication was Kostas Kolomvatsos.

still needs to share physical resources, which leads to the risk of side-channel information leakage [3]–[5]. Therefore, existing architecture-level solutions are usually not generic.

To address unsolved issues on malware detection, security providers recently focus on machine learning to improve security solutions [6]–[17]. However, there are still various issues that exist for applying machine learning to cybersecurity. For example, meaningful labeled datasets are not readily available, and the computational workload is too large to handle the big data.

Workload characterization is a very important step in designing processors or processor modules, and it can help to understand application behaviors on each architecture component. Characterized results are being used to design processors or hardware acceleration modules. In this paper, we focus on multiple steps to resolve critical issues of PMC-based malware detection including statistical workload characterization, statistical distribution based feature selection (feature tailoring), tradeoff analysis of detection time and accuracy, and architectural implications for hardware-based malware detection. Based on our experimental results and analysis, the existing number of performance counters is not enough to meet the desired accuracy in malware detection. For more accurate malware detection in real-time, we propose two architectural design alternatives: detection hardware with more performance monitoring counters and acceleration hardware with existing PMCs.

Related work: Basic motivation of this research starts from the intention to effectively use architectural profile information for malware detection. The main purpose of PMCs is to profile and tune the system performance at the architectural level [18]–[20]. Recently, PMCs are widely used in various domains including system power estimation, firmware modification, and malware detection [3], [19]. One of the primary drawbacks of using PMCs is the limited number of monitoring counters in a processor. Based on our investigation, more profile data from the performance counters can provide more accurate detection results. Recently, machine learning techniques have been used for classifying malware [13], [20]–[25] with multiple types of data including performance counter information. Garcia-Serrano *et al.* [21] discuss the feasibility of unsupervised learning to detect attacks. Conversely, Zhou *et al.* [26] claim incapability and difficulty of malware detection with the hardware performance counters in terms of detection accuracy. Our research focuses on improving the detection accuracy; as well as latency by adding additional hardware modules. Based on our previous research [27], we perform more characterizations on benign malware applications' profiles from PMC events. Also, we design the hardware architecture to improve the accuracy and detection latency by adding more PMC modules and the hardware module for the detection.

The rest of the paper is organized as follows. In Section II, we describe a statistical characterization of malware workloads from data collection to feature tailoring. The proposed malware detection is described in Section III, which includes details about statistical distribution based detection, supervised learning framework, etc. In Section IV, evaluation results are explained and compared with multiple approaches, and accuracy issues are also discussed. Implications for hardware design to improve the performance are provided in Section V, and we conclude with section VI.

II. STATISTICAL CHARACTERIZATION OF MALWARE

For the characterization of malware, PMCs are used to collect the data from microprocessors. Due to cost and area issues, processors have only a limited number of counters (registers), and only a few processor behavioral events can be simultaneously captured. In our data collection procedure, four architectural events from four PMCs are collected at the same time. Recent microprocessors tend to have more PMCs with registers for multiple purposes [18], [19].

A. DATA COLLECTION FROM PMCS

We use *perf* tool of Ubuntu 18.04 on the Intel Xeon processors (Skylake microarchitecture) to capture the behaviors of microarchitectural features. Both 20 benign samples and 20 malware samples are used for collecting architectural information and characterizing each workload from the architectural point of view. Each malware sample includes a combined 10 profiles of the same category of malware. Therefore, 200 benign and malware profiles were used for our experiments, respectively. Each profile is captured for 30 minutes of processor behaviors of a malware application. We assume that 30 minutes is enough time to statistically characterize differences between malware and benign applications. We collect malware applications from multiple sources including Virus Total [28] and Virus Sign [29]. The majority of the malicious samples comprised of Linux ELFs. The distribution of malware types used in our experiments is Trojans (40%), spyware (20%), adware (15%), worms (15%), and keyloggers (10%). Some types of malware including rootkits and ransomware were excluded in our experiment due to the lack of sources. For benign samples, we monitor the behaviors of Ubuntu applications including media player, text editor, photo editor, package manager, Firefox [33], rhythm box [34], etc. In addition, several shell scripts which include multiple benign applications are also monitored. To avoid any contamination or infection from malware under the experiment, data collection is performed on isolated Linux containers (LXCs). LXCs are chosen over virtualization through a virtual machine because containers provide the isolated systems on the host OS; instead of emulating the hardware. Among perf attributes, we capture 40 hardware events – 4 events as a group. The 40 events are based on two types of events which are HARDWARE and HW_CACHE as shown in Table 1. We make four events as a group since the processor we use for data collection has only 4 counters.

Some malware profiles have all-zero counts for some periods from the perf monitoring. We assume those malware instances are hibernating and could be active at a specific event or time, so those applications should be included in

TABLE 1. Perf events used for characterization.

Type	Event
PERF_TYPE_ HARDWARE	CPU Cycles, INSTRUCTIONS, BUS Cycles, CACHE References, CACHE Misses, BRANCH Instructions, BRANCH Misses
PERF_TYPE_ HW_CACHE	L1D Prefetch Accesses, L1D Read Accesses, L1D Read Misses, L1D Write Accesses, L1D Write Misses, L1D Prefetch Misses, L1I Prefetch Accesses, L1I Read Accesses, L1I Read Misses, L1I Write Accesses, L1I Write Misses, L1I Prefetch Misses, LL Prefetch Accesses, LL Read Accesses, LL Read Misses, LL Write Accesses, LL Write Misses, LL Prefetch Misses, DTLB Read Accesses, DTLB Read Misses, DTLB Prefetch Accesses, DTLB Write Accesses, DTLB Write Misses, DTLB Prefetch Misses, ITLB Prefetch Accesses, ITLB Read Accesses, ITLB Read Misses, ITLB Write Accesses, ITLB Write Misses, BPU Read Accesses, BPU Read Misses, BPU Write Accesses, BPU Write Misses

the experiments. For each experiment, we collect the PMC information for five hours for each malware application and benign application.

B. STATISTICAL CHARACTERIZATION AND FEATURE TAILORING

Based on the data collected from the performance monitoring counters, we observe some features to differentiate malware and benign samples. One of the features is the sum for each hardware event over the 30-minute profiling period. The magnitude and frequency of the PMC access for the malicious and benign profiles can be distinguishable characteristics based on our observation. The executable malware has single counter magnitudes up to 100x smaller than benign samples' profiles. However, there is not a clearly defined decision boundary for the two classes: resulting in some overlap. This decision can be made with statistical criteria and the help from machine learning with well-labeled data. Figure 1 shows the significant difference in PMC measurements between the two for the number of cache references. The average numbers are also showing the differences in both cases. Average cache references in benign applications are almost 90 times. Based on our observation, the frequency and magnitude of the access values can be used as unique criteria that separate malware profiles from benign profiles.

Figure 2 shows the comparison of benign and malware samples in terms of sum for each hardware event. The ratios between benign and malware are ranging from 30x to 100x. The sum of events can be used to detect malware, but only considering the sum can skew the results because the performance features from malware datasets are irregularly distributed, and numerous malware samples have zero counts for most of the sampling time. Therefore, we determine that the sparseness of the events monitored from the PMCs can be one of the characteristics associated with malware. The sparseness of the events, as another characteristic, can be obtained from the data, where the numbers from the sum of events are divided by the sum of non-zero events per sample – we refer to the feature as *effective sum*. The ratios between benign and malware of the effective sum are ranging from 1x to 66x. The ratio of 1x indicates that some malware types have very similar behaviors to benign applications based on architectural profiling. We need to have more analytic criteria to differentiate the similarity of the effective sum between malware and benign applications' profile.

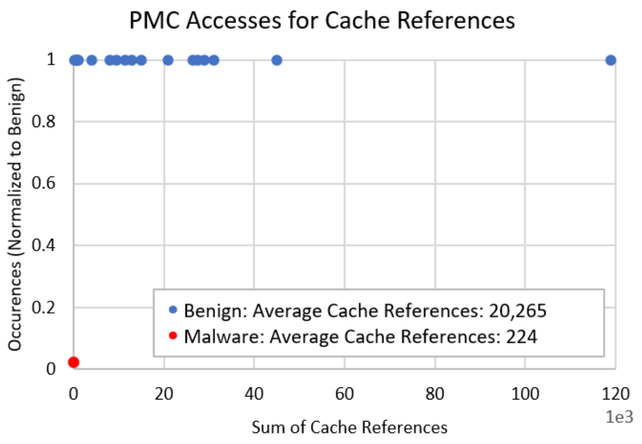


FIGURE 1. PMC accesses for cache reference.

Based on more in-depth analysis and observation, we come up with a metric called *Degree of Distribution* (DoD) as one of the differentiation criteria between malware and benign. Mean and standard deviation values are used to get the Degree of Distribution of the sum and the DoD of the effective sum as shown in equation (1). If the standard deviation is 0, the DoD value will be 1. In case the standard deviation is increased, DoD values will be less than 1. For a group of malware, DoD values will be relatively small due to the intermittent events.

Given two datasets – sum and effective sum, DoD values are extracted as shown in Figure 3. For each PMC event, we extract the average DoD value from 20 malware and 20 benign samples, respectively. Figure 3 shows the characterization results of each PMC event to the DoD. In the case of the sum datasets, the two graph lines are almost flat which reveals that there are no unique features between the malware and benign profiles. However, for the effective sum datasets, distinct features can be observed between benign and malware applications – especially for 6 performance events (marked with a red circle) including L1 data events and L1 instruction prefetch events among 40 PMC events. We use these 6 distinguishing performance events as the selected features for supervised learning.

III. MALWARE DETECTION BASED ON STATISTICAL CHARACTERIZATION

Generally, hardware-based malware detection has some advantages: it can provide a capability for dynamic mechanisms without relying on static signatures, and hardware-based detection also delivers faster processing time. However, one of the disadvantages is the cost of architectural resources (e.g., additional registers and logic). Modern processors provide a few special registers and hardware modules for

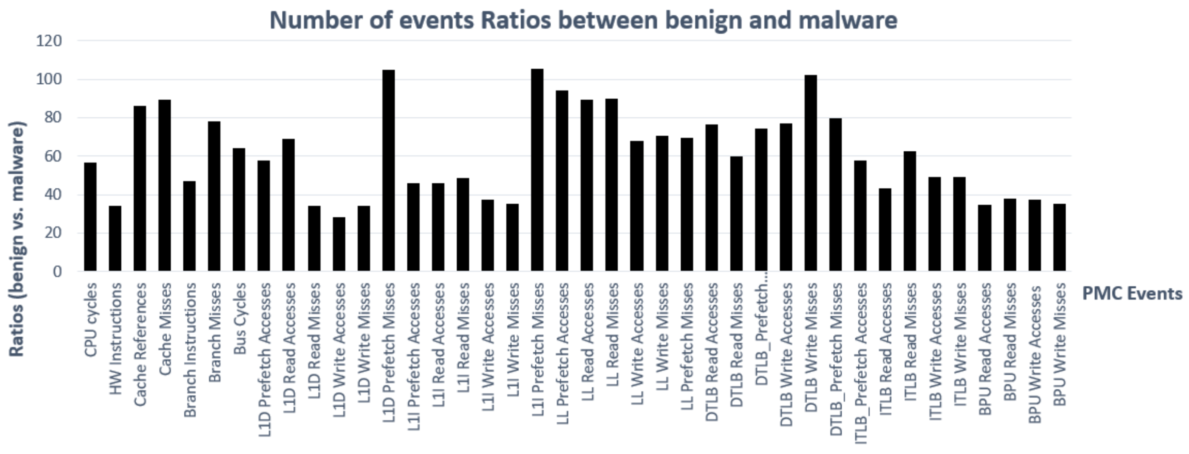


FIGURE 2. The number of events ratios between benign and malware.

performance monitoring and performance tuning, but that is not enough to capture various architectural events if they are used for other purposes such as malware detection rather than performance tuning. Based on our observation, PMC-based malware detection can be useful if we properly use statistical characterized information and machine learning mechanism to fill some potential gaps, since malware does have some unique characteristics in terms of workload behavior. In this paper, we use a statistical characteristic feature – DoD - based on performance counters in one of our experiments for malware detection.

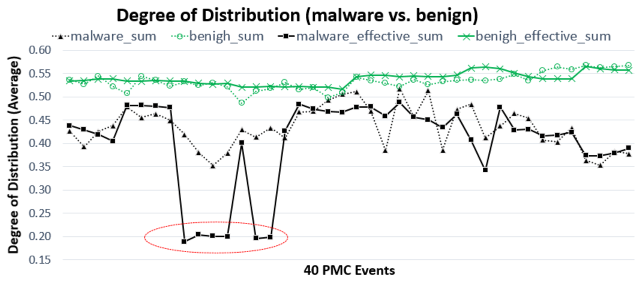


FIGURE 3. Characterization of each PMC event to the Degree of distribution (average value) of 20 malware and 20 benign samples – sum and effective sum.

Figure 4 shows the comparison of malware classification with 6 events (tailored) and the case with 40 events (full). For each sample, we extract the average DoD value from 40 PMC events. The threshold lines (red-dotted line) for malware detection are based on the DoD values (average) for each sample, where the case with 40 events has a slightly better threshold line than the case with only 6 events but the results are comparable. The DoD values can be directly used for malware detection with an appropriate threshold value, but there will surely be exceptions, and using a static number (e.g., threshold value) is not a good idea for the detection mechanism. In our research, we combine the statistical information (DoD) with a supervised learning approach for binary classification to improve the detection accuracy with a smaller number of events.

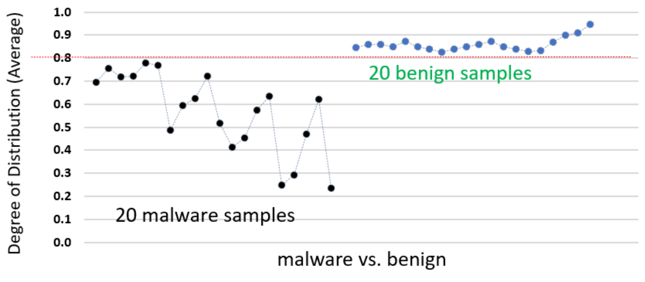
A. SUPERVISED LEARNING BASED MALWARE DETECTION WITH PERFORMANCE MONITORING COUNTER INFORMATION

1) FEATURE SELECTION (TAILORING) FOR MACHINE LEARNING

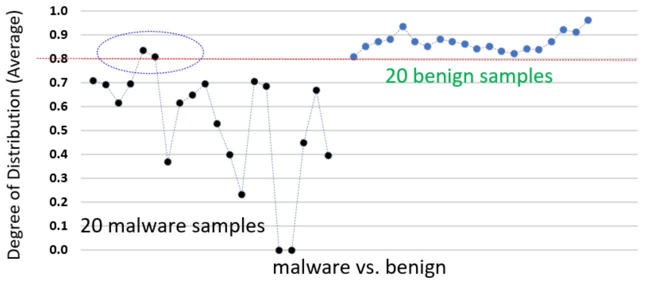
As a pre-processing strategy of machine learning, feature selection is very important, and will determine the quality of results and processing time [30], [31]. The proposed features based on statistical distribution are applied to the machine learning framework and the results are compared to the results from the features based on attribute evaluation. There are many attribute evaluators such as correlation, gain-ratio, info-gain, or oneR attribute evaluator that are available with a tool called *Weka* [32]. Weka is a collection of machine learning algorithms for data mining tasks, which provides multiple tools for data preprocessing, classification, regression, clustering, association rules mining, and visualization. The attribute evaluators have very different rankings for the 40 features, so the top 10 features from multiple evaluators are initially trained and tested using machine learning classifiers in our experiments. The attribute evaluator that yielded the best classification results is the cfsSubsetEval (Correlation-based Feature Subset Selection). The top 6 features from the cfsSubsetEval were then selected for further classification training/testing with different options and compared to the proposed DoD-based features.

2) BINARY CLASSIFICATION

A malware detection scheme is a binary classification: malware or not. There are many classifiers for binary classification [35]. For this experiment, we use 10 classifiers that include Bayes network, logistic classification, multi-layer classification, OneR, decision trees, JRIP, Bagging, AdaBoostM1, KStar classification, and random forest [32]. Figure 5 shows the overview of the complete learning framework from prepossessing to classification. Data sets are split into 3 different methods – standard, 3-fold, and 5-fold cross-validation. The standard dataset split uses 70% of the samples



(a) Malware classification with 40 PMC events



(b) Malware classification with 6 PMC events

FIGURE 4. Malware classification with 6 events [tailored] is comparable to the case with 40 events [full] (some exceptions are observed in the case with 6 events).

for training and 30% of the samples for testing data [36]. N-fold split means that the first 1/N portion of the dataset is used for testing and next 1/N portion of the dataset is used, and so on. Therefore, every data point will be in the testing set once, and in the training set N-1 times. Cross-validation which is the N-fold split method provides more training and testing cases and can reduce overfitting and underfitting [37].

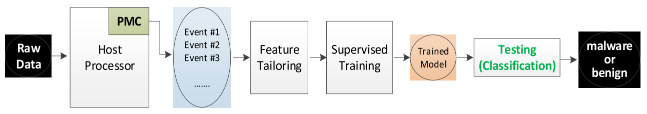


FIGURE 5. Overview of the complete learning framework from prepossessing to classification.

IV. EVALUATION

A. COMPARISON OF FEATURE TAILORING METHODS

The proposed DoD-based features are compared to the features selected from the feature tailoring method based on the attribute evaluation in Weka. Table 2 shows the two lists of features, where only one event is common. DoD-based features are all L1 cache events – 4 data cache events and 2 instruction cache events. On the other hand, the attribute evaluation-based features include representative architectural events such as cache, branch, and bus cycle. Based on the extracted features, we see that malware characteristics are closely related to data read and write to import malicious data. The two lists of features are used for binary classification through supervised learning with 3 different training and testing frameworks.

B. DISCUSSION ON ACCURACY

Table 3 shows the malware detection results from supervised learning using the proposed feature tailoring methods. Detection will be based on the process IDs. All processes are monitored with the information from the performance monitoring counters. Five accuracy metrics were used for accuracy comparison, including false positive, true negative, f-measure, AUC-ROC (Area Under Curve - Receiver Operating Characteristic), and AUC-PRC (Area Under Curve - associated Precision/ReCall) [38]–[40].

TABLE 2. Feature comparison from two tailoring methods.

Degree of Distribution (DoD)	Attribute Evaluation
L1D Read Accesses	Cache References
L1D Read Misses	Cache Misses
L1D Write Accesses	Branch Instructions
L1D Write Misses	Branch Misses
L1I Prefetch Accesses	Bus Cycles
L1I Read Accesses	L1D Read Accesses

As shown in Table 3, six DoD-based features show better accuracy overall, compared to six attribute-based features. Among the tailoring with 3 different datasets, '6-DoD-standard' shows the best accuracy in all accuracy metrics. Therefore, the degree of distribution (DoD) can differentiate malware from benign samples and can also provide highly accurate malware detection through the machine learning framework.

TABLE 3. Accuracy comparison.

Category	False Positive	True Negative	F-measure	AUC (ROC)	AUC (PRC)
6-attrib-standard	0.44	0.56	0.93	0.85	0.92
6-attrib-3-fold	0.46	0.54	0.93	0.85	0.94
6-attrib-5-fold	0.42	0.58	0.94	0.86	0.98
6-DoD-standard	**0.15**	**0.85**	**0.97**	**0.99**	**1.00**
6-DoD-3-fold	0.26	0.74	0.95	0.96	0.98
6-DoD-5-fold	0.25	0.75	0.96	0.97	**1.00**

AUC (ROC): Area Under Curve - ROC (Receiver Operating Characteristic)
AUC (PRC): Area Under Curve - PRC (associated Precision/ReCall)

The proposed malware detection method is based on hardware components' activities, therefore malware types including previously unseen malware samples will not affect the detection accuracy. In addition to testing our scheme with cross-validation, we use the data augmentation scheme to generate the trace profile of malware variants by changing the interval of the activities, combining multiple malware profiles, etc. The proposed method can efficiently detect newly generated malware variants within a 5% error rate.

C. TRADEOFF ANALYSIS: DETECTION TIME vs. ACCURACY

Generally, malware is active only for a very short period, and some malware hibernates until a specific event occurs. Based on our analysis, more accurate results can be achieved if we have more microarchitectural information from more performance monitoring counters simultaneously. However, most microprocessors have a small number of performance counters (e.g., 4∼8) running at the same time, which means that some behavior events can be missed when sampling processor behaviors. Accuracy for a detection algorithm is very important, but effective (dynamic) accuracy will be worsened if we cannot get proper datasets because of the dormant nature of malware and the sampling period. Therefore, an additional hardware module to extract statistical information with additional PMC registers is required to collect more profile information simultaneously and to promptly extract meaningful statistical information. With more PMC registers, more events such as branch behaviors and TLB behaviors can be used as classification features that can improve the performance in terms of accuracy.

Based on our experiments and analysis, the detection rate is 10∼20ms and classification accuracy is 90∼97%. The detection rate depends on the sampling rate for capturing profile information, and the classification accuracy depends on the number of PMC registers. By adding more PMC registers, classification accuracy is improved (5∼10%), but the detection rate shows very limited improvement (up to 10%) due to calculating more information, even with hardware acceleration. The accuracy improvement (95∼99%) provides more confidence in detection.

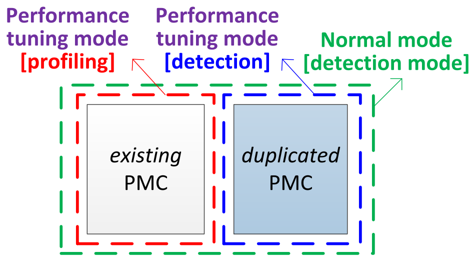
V. IMPLICATIONS FOR HARDWARE DESIGN FOR PERFORMANCE IMPROVEMENT

To improve the accuracy of malware detection, more performance features are required. But there are not enough PMC registers in modern microprocessors to monitor a large number of profiling events. However, adding more registers to microprocessors needs more manufacturing costs and operational costs. As a compromised way, an additional set of PMCs should be logically combined with existing counters since existing counters are not always actively used. Alternatively, a large set of profiling events can be captured with the shorter sampling time with existing PMCs without adding more PMC registers. Detailed schemes are described in the following subsections.

A. PMCS vs. ACCURACY

1) WITH ADDITIONAL PMCs

For large-scale systems, it is meaningful to add more hardware resources to existing processors to provide a more secure computing environment. Generally, most microprocessors already have PMC registers for performance monitoring. In our research, we come up with a new scheme to utilize both existing PMCs and newly added PMCs for malware detection. As shown in Figure 6 (a), two different operation modes can be designed: normal mode and performance tuning mode. In normal mode, two PMC modules will be used for malware detection which can provide more accuracy. In performance tuning mode, only half of the PMC will be used for detection while the other half will be used for profiling behaviors for performance tuning. Hardware cost estimation for additional PMC is also described in Figure 6 (b).


Performance tuning mode [profiling]
Performance tuning mode [detection]
Normal mode [detection mode]
existing PMC
duplicated PMC

(a) Additional PMC

Design features	Duplicated PMC *(vs. single PMC)*
Area	2X
Power	2X (normal mode) 1X~2X (perf tuning mode)
Latency	1X+Δ (more ALU ops)

(b) Hardware cost (for additional PMC)

FIGURE 6. Duplicated PMC to improve the detection accuracy in normal mode by using more profiling information. In performance tuning mode, only duplicated PMC will be used for malware detection.

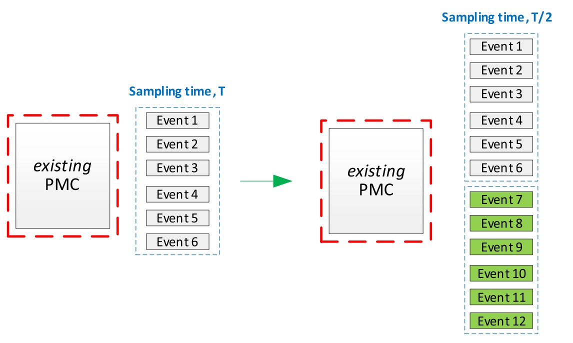
If we increase the PMC registers double, the area will be increased almost twice. Operation power in normal mode will be increased 2X, while power consumption in performance tuning mode will be 1X∼2X depending on the availability of the malware detection. The latency of malware detection will be slightly increased because of more computational latency to extract the statistical information from more profiling information. The latency will be 1X+Δ rather than 2X+Δ due to the parallel capturing of microarchitectural behaviors.

2) WITH EXISTING PMCs

Instead of adding more PMCs, a large number of profiling events can be captured with a shorter sampling period with the existing PMCs. As shown in Figure 7, assuming the existing PMC module has 6 monitoring counters, the PMC module can capture 6 events during the sampling time, T. With this scheme, the PMC module can capture 12 events during the same sampling time (T), where each event will be monitored only for T/2. Area and power consumption will not be changed with this scheme, but the latency can be leveraged by the number of distinctive features and monitoring time for each feature. Also, the accuracy of malware detection can be improved from more profile events.

B. HARDWARE ACCELERATION MODELS: PRE-PROCESSING MODULE vs. DEDICATED DETECTION MODULE

For the acceleration of detection, two different approaches can be considered depending on the design budget as described in Figure 8: (i) adding a pre-processing module to generate the statistical metadata which will be

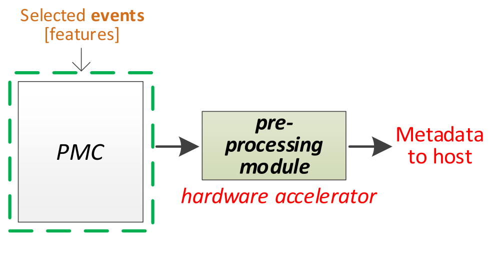


(a) Existing single PMC module with more profiling events

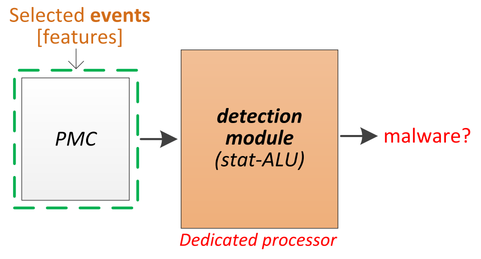
Design features	Single PMC
Area	1X
Power	1X
Latency	1X+Δ (more ALU ops)

(b) Hardware cost

FIGURE 7. Existing PMC module (6 event monitoring counter) with more profiling events and a shorter sampling period.



(a) Pre-processing module



(b) Dedicated detection module

FIGURE 8. Additional hardware modules for improving detection latency. Pre-processing module as a hardware accelerator vs. dedicated detection module for ALU and decision operation. Hardware cost varies on budget and goal.

sent to the host processor for machine learning operation; (ii) adding a dedicated detection hardware module to dynamically calculate statistical data and learning-based decision module. Two hardware approaches for malware detection can be applied to either existing embedded processors or new application-specific processors. Additional hardware cost varies on design goal and budget. Based on our design estimation, the complexity of the hardware acceleration module will be less than 1% of the entire chip for both approaches.

Operations: All processes will be monitored through the proposed detection mechanism with the information from PMCs. The period of data capturing per event process can be calibrated depending on the demand and resource availability. Selected events (features) can be dynamically or statically updated according to the learning results to improve detection accuracy. Based on our performance estimation, both combinational schemes provide accuracy improvement (5∼10%) and detection speedup (up to 10%) with the additional hardware cost.

VI. CONCLUSION

Malware detection with hardware solutions is becoming more important as malware becomes more advanced. Many existing hardware solutions use behavioral data from PMCs. However, due to the limited number of PMCs, the selection of architectural features is a critical issue to provide high-quality data for malware detection. To address the issue, we come up with a metric called Degree of Distribution (DoD) as one of the differentiation criteria. Our experimental results show that the DoD can differentiate malware from benign samples and can also provide highly accurate malware detection through the machine learning framework. The accuracy comes from both a statistical feature with a smaller number of events and machine learning schemes to boost the detection accuracy with limited PMC registers. Based on our analysis, hardware acceleration modules, as well as additional PMC registers are required for more accurate malware detection in real-time.

It will be highly possible for malicious software designers to be aware of the proposed detection algorithm when it is widely used. As one of the solutions, the periodic update of the tailored features could prevent form any tricks by reflecting the latest malware behaviors.

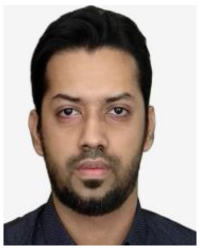
In future works, a more detailed architectural design for a dedicated accelerator to provide more efficiencies in chip area, power, and processing time will be investigated. Also, malware workloads need to be architecturally categorized, so that specific architectural features can be reflected in the hardware design of the detection module.

REFERENCES

[1] M. Al-Asli and T. Ghaleb, "Review of signature-based techniques in antivirus products," in *Proc. ICCI*, Apr. 2019, pp. 1–6.
[2] I. You and K. Yim, "Malware obfuscation techniques: A brief survey," in *Proc. BWCCA*, 2010, pp. 297–300.
[3] N. Patel, A. Sasan, and H. Homayoun, "Analyzing hardware based malware detectors," in *Proc. DAC*, 2017, pp. 1–6.
[4] A. Damodaran, F. Di Troia, C. A. Visaggio, T. H. Austin, and M. Stamp, "A comparison of static, dynamic, and hybrid analysis for malware detection," *J. Comput. Virol. Hacking Techn.*, vol. 13, no. 1, pp. 1–12, 2017.
[5] L. Nataraj, *A Signal Processing Approach Malware Analysis*. Santa Barbara, CA, USA: Univ. California, 2015.
[6] M. Alazab, S. Venkatraman, P. Watters, and M. Alazab, "Zero-day malware detection based on supervised learning algorithms of API call signatures," in *Proc. 9th Australas. Data Mining Conf.*, vol. 121, Ballarat, NSW, Australia, Dec. 2011, pp. 1–20.
[7] S. Huda, J. Abawajy, M. Alazab, M. Abdollalihian, R. Islam, and J. Yearwood, "Hybrids of support vector machine wrapper and filter based framework for malware detection," *Future Gener. Comput. Syst.*, vol. 55, pp. 376–390, Feb. 2016.
[8] E. Raff, J. Sylvester, and C. Nicholas, "Learning the PE header, malware detection with minimal domain knowledge," in *Proc. 10th ACM Workshop Artif. Intell. Secur.*, New York, NY, USA, Nov. 2017, pp. 121–132.

[9] M. Rhode, P. Burnap, and K. Jones, "Early-stage malware prediction using recurrent neural networks," *Comput. Secur.*, vol. 77, pp. 578–594, Aug. 2018.
[10] H. S. Anderson, A. Kharkar, B. Filar, and P. Roth, *Evading Machine Learning Malware Detection*. New York, NY, USA: Black Hat, 2017.
[11] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
[12] J. Saxe and K. Berlin, "Deep neural network based malware detection using two dimensional binary program features," in *Proc. 10th Int. Conf. Malicious Unwanted Softw. (MALWARE)*, Oct. 2015, pp. 11–20.
[13] S. Tobiyama, Y. Yamaguchi, H. Shimada, T. Ikuse, and T. Yagi, "Malware detection with deep neural network using process behavior," in *Proc. IEEE 40th Annu. Comput. Softw. Appl. Conf. (COMPSAC)*, Jun. 2016, pp. 577–582.
[14] W. Huang and J. W. Stokes, "MTNet: A multi-task neural network for dynamic malware classification," in *Proc. Int. Conf. Detection Intrusions Malware, Vulnerability Assessment*. Cham, Switzerland: Springer, Jul. 2016, pp. 399–418.
[15] T. Shibahara, T. Yagi, M. Akiyama, D. Chiba, and T. Yada, "Efficient dynamic malware analysis based on network behavior using deep learning," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2016, pp. 1–7.
[16] M. Sebastin, R. Rivera, P. Kotzias, and J. Caballero, "AVclass: A tool for massive malware labeling," in *Proc. Int. Symp. Res. Attacks, Intrusions, Defenses*. Cham, Switzerland: Springer, 2016, pp. 230–253.
[17] S. Ni, Q. Qian, and R. Zhang, "Malware identification using visualization images and deep learning," *Comput. Secur.*, vol. 77, pp. 871–885, Aug. 2018.
[18] W. Bircher and L. John, "Complete system power estimation: A trickle-down approach based on performance events," in *Proc. ISPASS*, 2007, pp. 158–168.
[19] J. Demme, "On the feasibility of online malware detection with performance counters," *ACM Comput. Archit. News*, vol. 41, no. 3, pp. 559–570, 2013.
[20] M. Bahador, "HPCMalHunter: Behavioral malware detection using hardware performance counters and singular value decomposition," in *Proc. ICCKE*, 2014, pp. 703–708.
[21] A. Garcia-Serrano, "Anomaly detection for malware identification using hardware performance counters," 2015, *arXiv:1508.07482*.
[22] M. Ozsoy, C. Donovick, I. Gorelik, N. Abu-Ghazaleh, and D. Ponomarev, "Malware-aware processors: A framework for efficient online malware detection," in *Proc. IEEE 21st Int. Symp. High Perform. Comput. Archit. (HPCA)*, Feb. 2015, pp. 651–661.
[23] H. Hossein, "Ensemble learning for effective run-time hardware-based malware detection: A comprehensive analysis and classification," in *Proc. DAC*, Jun. 2018, pp. 1–6.
[24] G. Dahl, "Large-scale malware classification using random projections and neural networks," in *Proc. ICASSP*, 2013, pp. 3422–3426.
[25] B. Athiwaratkun and J. Stokes, "Malware classification with LSTM and GRU language models and a character-level CNN," in *Proc. ICASSP*, 2017, pp. 2482–2486.
[26] B. Zhou, "Hardware performance counters can detect malware: Myth or fact?" in *Proc. ASIACCS*, 2018, pp. 457–468.
[27] J. Pattee and B. K. Lee, "Design alternatives for performance monitoring counter based malware detection," in *Proc. IEEE 39th Int. Perform. Comput. Commun. Conf. (IPCCC)*, Nov. 2020, pp. 1–2.
[28] *Virus Total Academic Malware Samples*. Accessed: Aug. 5, 2021. [Online]. Available: https://www.virustotal.com/intelligence/
[29] *Virus Sign Malware Samples*. Accessed: Jun. 10, 2021. [Online]. Available: https://samples.virussign.com/samples
[30] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. ICLR*, 2015, pp. 1–14.
[31] L. Bottou, "Global training of document processing systems using graph transformer networks," in *Proc. Comput. Vis. Pattern Recognit.*, 1997, pp. 489–494.
[32] M. Hall, "The weka data mining software: An update," in *ACM SIGKDD Explor. Newslett.*, vol. 11, no. 1, pp. 10–18, 2009.
[33] *Firefox*. Accessed: May 20, 2021. [Online]. Available: https://www.mozilla.org/en-U.S./firefox/linux/
[34] *Rhythm Box*. Accessed: May 20, 2021. [Online]. Available: https://help.ubuntu.com/community/Rhythmbox
[35] V. Bahel, S. Pillai, and M. Malhotra, "A comparative study on various binary classification algorithms and their improved variant for optimal performance," in *Proc. IEEE Region Symp. (TENSYMP)*, Jun. 2020, pp. 495–498.
[36] D. M. W. Powers and A. Atyabi, "The problem of cross-validation: Averaging and bias, repetition and significance," in *Proc. Spring Congr. Eng. Technol.*, May 2012, pp. 1–5.
[37] B. Ghojogh and M. Crowley, "The theory behind overfitting, cross validation, regularization, bagging, and boosting: Tutorial," 2019, *arXiv:1905.12787*.
[38] M. Rottmann, K. Maag, R. Chan, F. Hüger, P. Schlicht, and H. Gottschalk, "Detection of false positive and false negative samples in semantic segmentation," in *Proc. 23rd Conf. Design, Automat. Test Eur.*, Mar. 2020, pp. 1–16.
[39] J. Davis and M. Goadrich, "The relationship between precision-recall and ROC curves," in *Proc. 23rd Int. Conf. Mach. Learn.*, 2006, pp. 233–240.
[40] J. Keilwagen, I. Grosse, and J. Grau, "Area under precision-recall curves for weighted and unweighted data," *PLoS ONE*, vol. 9, no. 3, Mar. 2014, Art. no. e92209.

JORDAN PATTEE received the bachelor's degree from the University of Colorado Colorado Springs, in 2021. She worked at Symetrix Corporation as an Application and Design Engineer. She worked at the Laboratory for Intelligent Computer Architecture (LiCA) as an Undergraduate Researcher and she got an Excellent Student Award on her graduation. She is currently a Graduate Student with Kyushu University, Japan. Her research interests include malware detection, deep learning, resistive memory, and modeling.



SHAFAYAT MOWLA ANIK received the bachelor's and master's degrees from the University of Dhaka, Bangladesh, in 2015 and 2017, respectively. He is currently pursuing the Ph.D. degree with the University of Colorado Colorado Springs. He is working at the Laboratory for Intelligent Computer Architecture (LiCA) as a Graduate Research Assistant. His research interests include deep learning, computer architecture, malware detection, cybersecurity, low power, and performance modeling.

BYEONG KIL LEE (Senior Member, IEEE) received the Ph.D. degree in computer engineering from The University of Texas at Austin, Austin, in 2005. He is currently an Assistant Professor with the Department of Electrical and Computer Engineering, University of Colorado at Colorado Springs. Prior to joining the Faculty at UCCS, he worked at Samsung as the Vice President for five years. For four years, he was an Assistant Professor at The University of Texas at San Antonio (UTSA). For five years, he was also a Senior Design Engineer at Texas Instruments (TI). For ten years, he was also a Senior Research Staff at the Agency for Defense Development (ADD). His research is published in several international conferences and journals. His research interests include computer architecture, application-specific embedded systems (mobile processors), deep-learning-based intelligent computing, workload characterization of emerging applications, parallel computing and parallel architecture design, performance modeling, low power design, and early-stage power estimation. He serves on the technical program committee and organizing committee for some conferences and workshop, such as ISCA, ISPASS, IISWC, ICCD, HPCA, ASAP, PACT, ICPP, and UCAS. He is a reviewer for conferences and journals.