Performance Monitoring Counter based Intelligent Malware Detection and Design Alternatives

Journal:	*IEEE Access*
Manuscript ID	Access-2022-03407
Manuscript Type:	Regular Manuscript
Date Submitted by the Author:	06-Feb-2022
Complete List of Authors:	Pattee, Jordan; University of Massachusetts Lowell Department of Electrical and Computer Engineering Anik, Shafayat; University of Massachusetts Lowell Department of Electrical and Computer Engineering Lee, Byeong; University of Massachusetts Lowell Department of Electrical and Computer Engineering
Keywords: Please choose keywords carefully as they help us find the most suitable Editor to review:	Hardware acceleration, Machine learning, Malware, Pattern classification
Subject Category Please select at least two subject categories that best reflect the scope of your manuscript:	Computational and artificial intelligence, Computers and information processing
Additional Manuscript Keywords:	

AUTHOR RESPONSES TO IEEE ACCESS SUBMISSION QUESTIONS

Author chosen manuscript type:	Research Article
Author explanation /justification for choosing this manuscript type:	This article is a classic research article that has a hypothesis, investigation, model, experiment, and analysis. The topic is related to malware detection with machine learning techniques based on the profiles of processorâ€™s performance monitoring counters.
Author description of how this manuscript fits within the scope of IEEE Access:	This article covers an intelligent malware detection using the processorâ€™s behaviors based on the performance monitoring counters. Technical scopes include processor architecture, machine learning, classification, and hardware acceleration. The article fits the scope of IEEE Access.
Author description detailing the unique contribution of the manuscript related to existing literature:	This article emphasizes the importance of selecting architectural features for malware detection. The statistical differences between malware and benign were characterized based on the performance profiles. Most malware can easily be detected with basic characteristics, but some malware types are statistically very similar to benign, which need to be handled more in-depth. Hence, we focus on multiple steps to investigate critical issues of PMC-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 and hardware acceleration schemes.

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2021.Doi Number

Performance Monitoring Counter based Intelligent Malware Detection and Design Alternatives

Jordan Pattee, Shafayat Mowla Anik, and Byeong Kil Lee

Department of Electrical and Computer Engineering, University of Colorado, Colorado Springs, CO 80918 USA

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

This work was supported in part by UCCS and 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 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 [20-26] with multiple types of data including performance counter information. Garcia et. al. [21] discuss the feasibility of unsupervised learning to detect attacks. Conversely, Zhou et. al. [27] 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 [28], 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 [29] and Virus Sign [30]. 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 [34], rhythm box [35], 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 I. We make four events as a group since the processor we use for data collection has only 4 counters.

TABLE I
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

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 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.

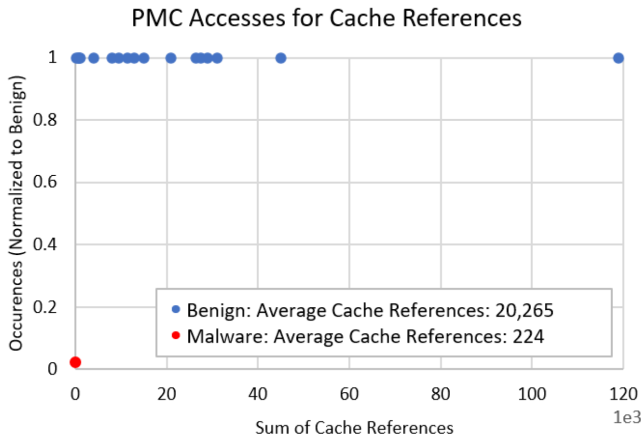


FIGURE 1. **PMC accesses for cache reference.**

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.

$$Degree\ of\ Distribution = \frac{Mean}{(Mean + StdDev)} \quad (1)$$

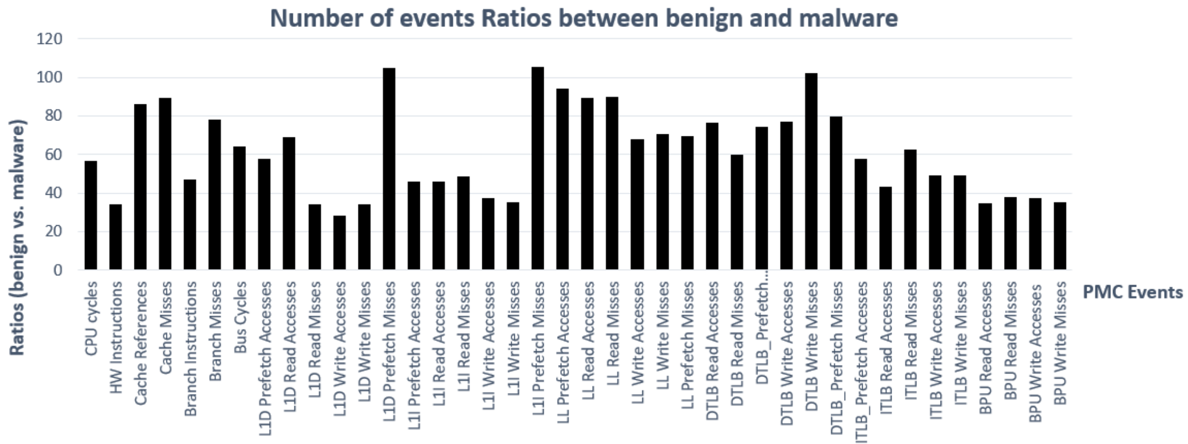


FIGURE 2. **The number of events ratios between benign and malware.**

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.

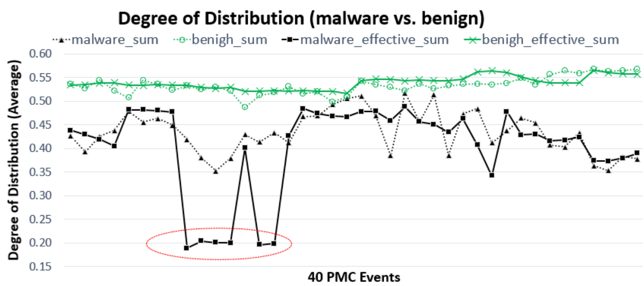


FIGURE 3. **Characterization of each PMC event to the Degree of distribution (average value) of 20 malware and 20 benign samples – sum and effective sum.**

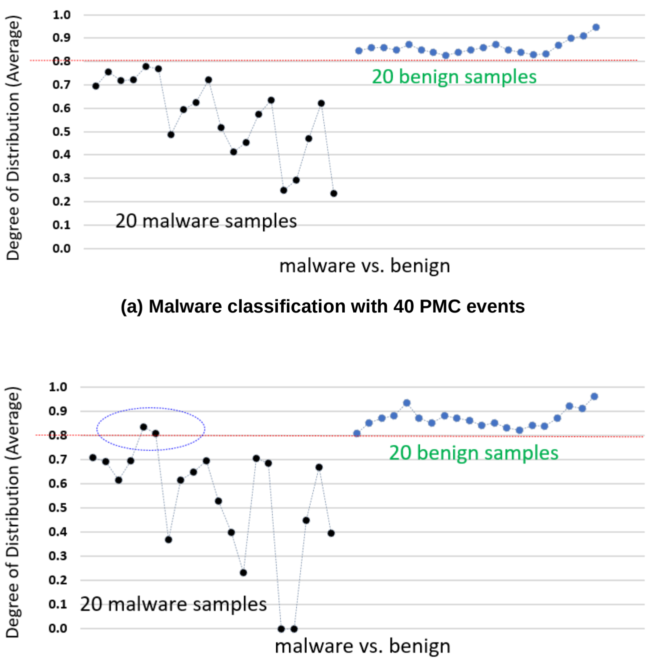
Figure 4 shows the comparison of malware classification with 6 events (tailored) and the case with 40 events (full). For each sample, we extract the average DoD value from 40 PMC events. The threshold lines (red-dotted line) for malware detection are based on the DoD values (average) for each sample, 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.

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 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.

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

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 [31-32]. 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* [33]. 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.



(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).

Binary Classification: A malware detection scheme is a binary classification: malware or not. There are many classifiers for binary classification [36]. For this experiment, we use 10 classifiers that include Bayes network, logistic classification, multilayer classification, OneR, decision trees, JRIP, Bagging, AdaBoostM1, KStar classification, and random forest [33]. 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 for training and 30% of the samples for testing data [37]. 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 [38].

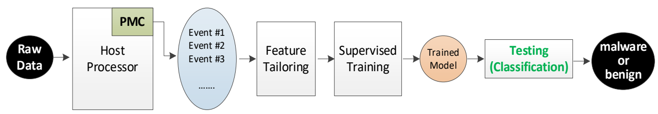


FIGURE 5. Overview of the complete learning framework from prepossessing to classification.

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 II 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 III 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) [39-41].

TABLE II
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 III, 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 III
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.

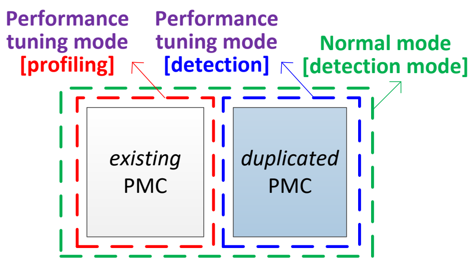
Based on our experiments and analysis, the detection rate is 10~20ms and classification accuracy is 90~97%. The detection rate depends on the sampling rate for capturing profile information, and the classification accuracy depends on the number of PMC registers. By adding more PMC registers, classification 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

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).



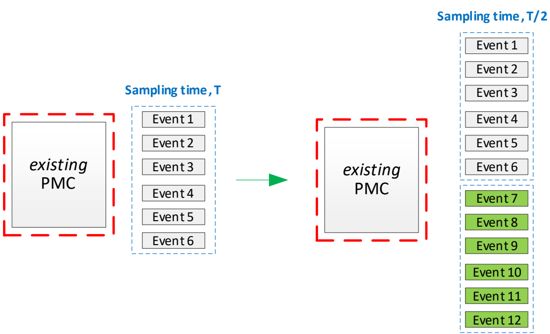
(a) Additional PMC

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.



(a) Existing single PMC module with more profiling events

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

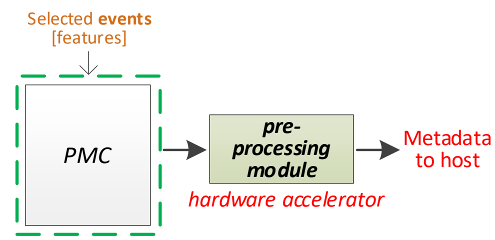
(b) Hardware cost

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

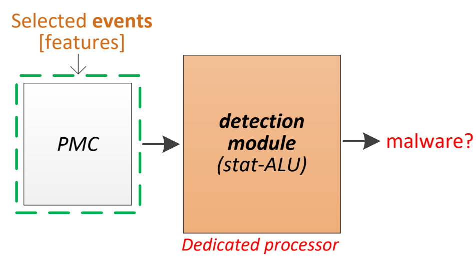
With Existing PMCs: Instead of adding more PMCs, a large number of profiling events can be captured with a shorter sampling period with the existing PMCs. As shown in Figure 7, assuming the existing PMC module has 6 monitoring counters, the PMC module can capture 6 events during the sampling time, T. 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 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.



(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.

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] Al-Asli et al., "Review of Signature-Based Techniques in Antivirus Products," IEEE ICCISci.2019.8716381, 2019.

[2] You et al., “Malware Obfuscation Techniques: A Brief Survey,” BWCCA, 2010.

[3] N. Patel, A. Sasan, H. Homayoun, "Analyzing Hardware Based Malware Detectors," DAC 2017.

[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. Virology Hacking Techn., vol. 13, no. 1, pp. 1–12, 2017.

[5] L. Nataraj, A Signal Processing Approach To 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, Australia, Dec. 2011.

[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: ACM, Nov. 2017.

[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.

[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), vol. 2, Jun. 2016.

[14] W. Huang, 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.

[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.

[16] M. Sebastián, R. Rivera, P. Kotzias, and J. Caballero, “Avclass: A tool for massive malware labeling,” in International Symposium on Research in Attacks, Intrusions, and Defenses. Cham, Switzerland: Springer, 2016.

[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] Bircher et al., "Complete system power estimation: A trickle-down approach based on performance events," ISPASS 2007.

[19] Demme et al., "On the feasibility of online malware detection with performance counters," ACM Computer Architecture News, 2013.

[20] Bahador et al., "Hpcmalhunter: Behavioral malware detection using hardware performance counters and singular value decomposition," ICCKE 2014.

[21] A. Garcia-Serrano, "Anomaly detection for malware identification using hardware performance counters," arXiv:1508.07482, 2015.

[22] Ozsoy et al., "Malware-aware processors: A framework for efficient online malware detection," The 21st International Symposium on High Performance Computer Architecture (HPCA) 2015.

[23] Hossein et al., “Ensemble Learning for Effective Run-Time Hardware-Based Malware Detection: A Comprehensive Analysis and Classification,” DAC 2018.

[24] Tobiyama et. al., “Malware Detection with Deep Neural Network Using Process Behavior,” International Computer Software and Applications Conference 2016.

[25] Dahl et al., “Large-scale malware classification using random projections and neural networks,” ICASSP 2013.

[26] B. Athiwaratkun and J. Stokes, “Malware Classification with LSTM and GRU language models and a character-level CNN,” ICASSP, 2017.

[27] B. Zhou, et al., "Hardware Performance Counters Can Detect Malware: Myth or Fact?," ASIACCS 2018.

[28] J. Pattee and B. K. Lee, "Design Alternatives for Performance Monitoring Counter based Malware Detection," International Performance Computing and Communications Conference (IPCCC), 2020.

[29] Virus Total Academic Malware Samples. http://www.virustotal.com/intelligence/ [upon request], Accessed on: Aug. 5, 2021.

[30] Virus Sign Malware Samples. http://samples.virussign.com/samples, Accessed on: Jun. 10, 2021.

[31] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," ICLR 2015.

[32] L. Bottou, et. al., "Global training of document processing systems using graph transformer networks," In Proc. of Computer Vision and Pattern Recognition, Puerto-Rico, 1997.

[33] Hall et al., "The weka data mining software: an update," ACM SIGKDD explorations newsletter, 2009.

[34] Firefox, https://www.mozilla.org/en-US/firefox/linux/, Accessed on: May. 20, 2021.

[35] Rhythm box, https://help.ubuntu.com/community/Rhythmbox, Accessed on: May. 20, 2021.

[36] V. Bahel, S. Pillai and M. Malhotra, "A Comparative Study on Various Binary Classification Algorithms and their Improved Variant for Optimal Performance," 2020 IEEE Region 10 Symposium (TENSYMP), 2020.

[37] D. M. W. Powers and A. Atyabi, "The Problem of Cross-Validation: Averaging and Bias, Repetition and Significance," 2012 Spring Congress on Engineering and Technology, 2012.

[38] B. Ghojogh and M. Crowley, "The Theory Behind Overfitting, Cross Validation, Regularization, Bagging, and Boosting: Tutorial," arXiv:1905.12787v1, 2019.

[39] 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 Proceedings of the 23rd Conference on Design, Automation and Test in Europe, March, 2020.

[40] J. Davis and M. Goadrich, "The relationship between precision-recall and ROC curves," In Proceedings of the 23rd International Conference on Machine Learning, 2006.

[41] J. Keilwagen, I. Grosse, and J. Grau, "Area under precision-recall curves for weighted and unweighted data," PLoS ONE, 9(3):e92209, 03 2014.

JORDAN PATTEE received her bachelor's degree from University of Colorado Colorado Springs in 2021, and she worked for Symetrix Corporation as an application and design engineer. She is a graduate student in Kyushu University in Japan. Her research interest includes malware detection, deep learning, resistive memory, and modeling. She worked at LiCA (Laboratory for Intelligent Computer Architecture) as an undergraduate researcher, and she got an excellent student award on her graduation.

SHAFAYAT MOWLA ANIK is a Ph.D. student at University of Colorado Colorado Springs. He received his bachelor's and master's degree from University of Dhaka, Bangladesh in 2015 and 2017 respectively. His research interest includes deep learning, computer architecture, malware detection, cybersecurity, low power and performance modeling. He is working at LiCA (Laboratory for Intelligent Computer Architecture) as a graduate research assistant.

BYEONG KIL LEE (M'87) (M'05-SM'11) received the Ph.D. degree in computer engineering from the University of Texas at Austin, Austin, in 2005. He is currently an Assistant Professor in the Department of Electrical and Computer Engineering at the University of Colorado at Colorado Springs. 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. Prior to joining the faculty at UCCS, he worked for Samsung as a Vics President for 5 years and he was an assistant professor at the University of Texas at San Antonio (UTSA) for 4 years. Also, he was a senior design engineer and senior research staff at TI (Texas Instruments) and ADD (Agency for Defense Development) for 5 and 10 years respectively. His research is published in several international conferences and journals, and he is also a reviewer for conferences and journals. He serves on the technical program committee and organizing committee of some conferences and workshop such as ISCA, ISPASS, IISWC, ICCD, HPCA, ASAP, PACT, ICPP and UCAS.

JORDAN PATTEE received her bachelor's degree from University of Colorado Colorado Springs in 2021, and she worked for Symetrix Corporation as an application and design engineer. She is a graduate student in Kyushu University in Japan. Her research interest includes malware detection, deep learning, resistive memory, and modeling. She worked at LiCA (Laboratory for Intelligent Computer Architecture) as an undergraduate researcher, and she got an excellent student award on her graduation.

SHAFAYAT MOWLA ANIK is a Ph.D. student at University of Colorado Colorado Springs. He received his bachelor's and master's degree from University of Dhaka, Bangladesh in 2015 and 2017 respectively. His research interest includes deep learning, computer architecture, malware detection, cybersecurity, low power and performance modeling. He is working at LiCA (Laboratory for Intelligent Computer Architecture) as a graduate research assistant.

BYEONG KIL LEE (M'87) (M'05-SM'11) received the Ph.D. degree in computer engineering from the University of Texas at Austin, Austin, in 2005. He is currently an Assistant Professor in the Department of Electrical and Computer Engineering at the University of Colorado at Colorado Springs. 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. Prior to joining the faculty at UCCS, he worked for Samsung as a Vics President for 5 years and he was an assistant professor at the University of Texas at San Antonio (UTSA) for 4 years. Also, he was a senior design engineer and senior research staff at TI (Texas Instruments) and ADD (Agency for Defense Development) for 5 and 10 years respectively. His research is published in several international conferences and journals, and he is also a reviewer for conferences and journals. He serves on the technical program committee and organizing committee of some conferences and workshop such as ISCA, ISPASS, IISWC, ICCD, HPCA, ASAP, PACT, ICPP and UCAS.

Original Manuscript ID: Access-2021-30014

Original Article Title: "Performance Monitoring Counter based Intelligent Malware Detection and Design Alternatives"

To: IEEE Access Editor

Re: Response to reviewers

Dear Editor,

Thank you for allowing a resubmission of our manuscript, with an opportunity to address the reviewers' comments.

We are uploading (a) our point-by-point response to the comments (below) (response to reviewers), (b) an updated manuscript with yellow highlighting indicating changes, and (c) a clean updated manuscript without highlights (PDF main document).

Best regards,

Byeong Kil Lee et al.

Reviewer#1 Recommendation: Accept (minor edits)

Comments: We see all of the previously articulated concerns addressed and taken care of. We hope that this has contributed the enhancing the quality of the resulting paper.

Reviewer#1, Concern: N/A

Reviewer#2 Recommendation: Reject (updates required before resubmission)

Comments: Paper which entitled "Performance Monitoring Counter based Intelligent Malware Detection and Design Alternatives" proposed hardware based malware detection with several good points, however, there are some issues which needed to be fixed before publication.

Reviewer#2, Concern # 1: As I understood 20 benign applications and 20 malware applications are analyzed, it must be used more malware and benign samples

Author response: Thanks for the detailed comments on applications and data samples. In our experiments, each malware sample includes a combined 10 profiles of the same category of malware. Each profile is captured for 30 minutes of processor behaviors. So, 200 benign and malware profiles used for our experiments, respectively. All benign and malware samples are used for training and classification using 3-fold and 5-fold cross-validation. We should have clearly mentioned it before the submission. We corrected the relevant sentences.

Author action: We updated the manuscript (p.2) by reviewer's comments. The followings are the corresponding actions for the Concern #1 and are included in section II-A (p.2).

- 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 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.

Reviewer#2, Concern # 2: In 1 paragraph, you should represent the data distribution, for example how many virus, worm, rootkit, ransomware, downloader, etc. your dataset has or in how many percentage.

Author response: Thanks for the detailed comments on data distribution of malware sample. Among many types of malware, we use Trojans, spyware, adware, worms and keyloggers. Some types of malwares including rootkits and ransomware were excluded in our experiment due to the lack of sources. The distribution of malware types is Trojans (40%), spyware (20%), adware (15%), worms (15%) and keyloggers (10%). We included this information in the paper.

Author action: We updated the manuscript (p.2) by reviewer's comments. The followings are the corresponding actions for the Concern #2 and are included in section II-A (p.2).

- The distribution of malware types used in our experiments is Trojans (40%), spyware (20%), adware (15%), worms (15%) and keyloggers (10%). Some types of malwares including rootkits and ransomware were excluded in our experiment due to the lack of sources.

Reviewer#2, Concern # 3: In 1 paragraph, you should explain how your method can handle previously unseen malware. Is it efficiently to detect new malware variants?

Author response: Thanks for the detailed comments on detection efficiency for new malware variants. The proposed malware detection method is based on hardware components' activities, therefore malware types including previously unseen malwares will not affect the detection accuracy. In addition to the 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 5% error rate. We properly updated the paper with this information.

Author action: We updated the manuscript by reviewer's comments. The followings are the corresponding actions for the Concern #3 and are included in section IV-B (p.6).

- The proposed malware detection method is based on hardware components' activities, therefore malware types including previously unseen malwares will not affect the detection accuracy. In addition to the 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 5% error rate.

Reviewer#2, Concern # 4: You can also add detection rate as well as classification accuracy

Author response: Thanks for the detailed comments on performance analysis. Based on our experiments and analysis, detection rate is 10~20ms and classification accuracy is 90~97%. 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.

Author action: We updated the manuscript by reviewer's comments. The followings are the corresponding actions for the Concern #4 and are included in section IV-C (p.6).

- Based on our experiments and analysis, 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.

Reviewer#2, Concern # 5: When more references are used in the same sentences, usage of parenthesis should be the same in everywhere. For example, in text I found [1][2] and [6-17], you should correct them for consistency.

Author response: Thanks for the detailed comments on the format of multiple references. We update all the formats with one format such as [1-2], [20-26], etc.

Author action: We updated the manuscript by reviewer's comments. The followings are the corresponding actions for the Concern #5 and are included in section I (p.1 and p.2), section II (p.2) and section III-A (p.5).

- 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].

- Recently, machine learning techniques have been used for classifying malware [20-26] with multiple types of data including performance counter information.
- 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].
- As a pre-processing strategy of machine learning, feature selection is very important, and will determine the quality of results and processing time [31-32].

Reviewer#2, Concern # 6: The language of the paper can be improved further.

Author response: Thanks for the detailed comments on the language improvement. We use the grammar checking tools for the correction, and we also get a proofreading from other engineers.

Author action: We updated the manuscript with all the corrections. The followings are the corresponding actions for the Concern #6.

- Abstract (p.1)
  - can be easily compromised by intelligent malwares.
    → can be easily compromised by intelligent malware.
  - based on statistical learning blocks with abnormal features of system calls, network traffics or processor behaviors.
    → based on statistical learning blocks with abnormal features of system calls, network traffics, or processor behaviors.
- Abstract (p.1)
  - but some malwares are statistically very similar to
    → but some malware types are statistically very similar to
  - because the priority has always been given to performance,
    → because priority has always been given to performance,
- I (p.1)
  - automated analysis and detection of malware remain an open issue.
    → automated analysis and detection of malware remain open issues.
- I (p.2)
  - The main purpose of PMCs is to profile and tune the system performance in the architectural level
    → The main purpose of PMCs is to profile and tune the system performance at the architectural level
- II-A (p.2)
  - To avoid any contamination or infection from malwares under the experiment
    → To avoid any contamination or infection from malware under the experiment
- II-B (p.3)
  - as unique criteria that separated malware from benign applications.
    → as unique criteria that separate malware profiles from benign profiles.
  - The sum of events can be used to detect malwares,
    → The sum of events can be used to detect malware,

- numerous malwares have zero counts for most of the sampling time.
  → numerous malware samples have zero counts for most of the sampling time.

- The ratios between benign and malware of the effective sum are ranged from 1x to 66x.
  → The ratios between benign and malware of the effective sum are ranging from 1x to 66x.
- The ratio of 1x indicates that some malwares have very similar behaviors to
  → The ratio of 1x indicates that some malware types have very similar behaviors to

- For a group of malwares,
  → For a group of malware,

- II-B (p.4)
  - Figure 3 shows the characterization results of each PMC event with respect to the DoD
    → Figure 3 shows the characterization results of each PMC event to the DoD

- III-A (p.5)
  - Malware detection scheme is a binary classification: malware or not.
    → A malware detection scheme is a binary classification: malware or not.

  - Cross validation which is the N-fold split method
    → Cross-validation which is the N-fold split method

- IV-C (p.6)
  - Generally, malwares are active only for a very short period, and some malwares hibernate until a specific event occurs.
    → Generally, malware is active only for a very short period, and some malware hibernates until a specific event occurs.

  - in modern microprocessors to monitor large number of profiling events.
    → in modern microprocessors to monitor a large number of profiling events.

- V-A (p.7)
  - Instead of adding more PMCs, large number of profiling events can be captured
    → Instead of adding more PMCs, a large number of profiling events can be captured

  - With this scheme, the PMC module can capture 12 events during same sampling time
    → With this scheme, the PMC module can capture 12 events during the same sampling time

- V-B (p.8)
  - Based our performance estimation,
    → Based on our performance estimation,

- VI (p.8)
  - can differentiate malwares from benign applications
    → can differentiate malware from benign applications

  - hardware acceleration module, as well as additional PMC registers
    → hardware acceleration modules, as well as additional PMC registers

  - In future works, more detailed architectural design for a dedicated accelerator
    → In future works, a more detailed architectural design for a dedicated accelerator

Note: *References suggested by reviewers should only be added if it is relevant to the article and makes it more complete. Excessive cases of recommending non-relevant articles should be reported to ieeeaccesseic@ieee.org*

Design Alternatives for Performance Monitoring Counter based Malware Detection

Jordan Pattee and Byeong Kil Lee
Department of Electrical and Computer engineering
University of Colorado, Colorado Springs (UCCS)
Colorado Springs, CO 80918 USA
{jpattee, blee}@uccs.edu

Abstract—**Hardware-based malware detection is becoming increasingly important as software-based solutions can be easily compromised by attackers. Many of the existing hardware solutions are based on statistical learning blocks with processor behavioral information, which can be captured from the PMC (performance monitoring counters). The performance of the learning techniques relies primarily on the quality of data. However, due to the limited number of PMCs in a processor, only a few behavioral events can be monitored simultaneously. In this paper, 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 complexity and accuracy; and (iv) design alternatives for PMC-based malware detection. Our experimental results show that the proposed detection scheme can provide highly accurate malware detection. As architectural implications, hardware acceleration as well as additional PMC registers are discussed for more accurate malware detection in real-time.**

Keywords—malware detection, performance counter, machine learning, hardware acceleration, workload characterization

I. Introduction

Software based malware detection can remove harmful programs with a static signature-based mechanism. However, the detectors can be easily compromised with the obfuscation techniques and 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 the priority has always been given to performance, power, and area (PPA). Consequently, inherent security risks exist that are associated with architectural modules. These vulnerabilities are difficult to remove due to the conflict of performance and security. As a hardware solution of security, ARM TrustZone can be operated without burdening the host processor. However, it still shares physical resources, which leads to the risk of side-channel leakage [1]. To improve accuracy, researchers recently focus on machine learning, but there are still various issues that exist for applying machine learning techniques to security.

In this paper, we focus on multiple steps to resolve critical issues of PMC-based malware detection, including statistical characterization, feature selection for learning, tradeoff analysis and design alternatives for hardware solutions. Based on our analysis, we observe that a programmable hardware acceleration module that includes more PMC registers is required to get more accurate malware detection in real-time.

Related work: The main purpose of PMCs is to profile and tune the system performance in architectural level [2][3][4]. Recently, PMCs are widely used in various domains including system power estimation, malware detection [1][3], etc. Also, machine learning has been used for classifying malware [4][5][6] with multiple types of data. Conversely, Zhou et. al. [7] claims incapability and difficulty of malware detection with the PMC in terms of detection accuracy. Our research focuses on improving the detection accuracy as well as latency by adding additional hardware modules.

II. Malware Characterizations

The *perf* of the Ubuntu 18.04 on the Intel Xeon processors is used to collect architectural information from 20 benign applications and 20 malware applications, and each workload is characterized from the architectural aspects. We collected malware samples from multiple sources including Virus Total [8] and Virus Sign [9]. Among *perf* attributes, we captured 40 hardware events which are based on two types of events, HARDWARE and HW_CACHE.

Based on our characterizations, we observe that malwares show a very small sum of each hardware event compared to benign applications. Only considering the sum as a detection criterion can skew the results because malware datasets are irregularly distributed. Therefore, we refer to the feature as the *effective sum* - the sum of events divided by the sum of non-zero events. With more in-depth analysis and observation, we come up with a metric called *Degree of Distribution (DoD)* as one of the differentiation criteria for malware detection, *mean / (mean + standard_deviation)*. If the standard deviation is 0, the DoD value will be 1. For a group of malwares, DoD values will be relatively small due to the intermittent events. Distinct features can be observed between benign and malware applications with the *effective sum* datasets.

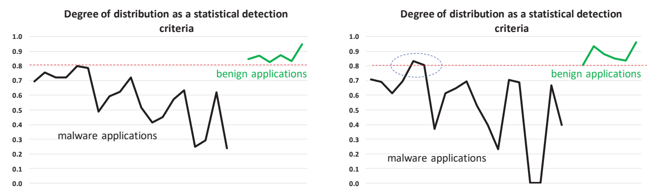
III. Proposed Malware Detection

The DoD values from malware and benign applications are used as threshold lines for malware detection as shown in Figure 1, where the case with 40 events has slightly better threshold line than the case with 6 events. The DoD value with an appropriate threshold will have exceptions, so static threshold number will not be a good idea. In our research, we combine the statistical information with a supervised learning approach for binary classification to solve those issues.

The proposed statistical features were applied to machine learning framework, and the results were compared to the ones with the existing attribute evaluation based features extracted

2020 IEEE 39th International Performance Computing and Communications Conference (IPCCC) | 978-1-7281-9829-3/20/$31.00 ©2020 IEEE | DOI: 10.1109/IPCCC50635.2020.9391559

from a tool called *Weka* [10]. The top 6 features, which are based on evaluation ranking, are selected for further training/testing for classification. Data sets are split with 3 different methods – standard, 3-fold and 5-fold cross-validation. The N-fold split method provides more training/testing cases and can reduce overfitting and underfitting. Based on the extracted features, we see that malware characteristics are closely related to data read and write behaviors to import malicious data.



(a) DoD with 40 events **(b)** DoD with 6 events

Figure 1: The DoD as a detection criterion. The case with 6 events

IV. EVALUATION

Table 1 shows the malware detection results from supervised learning using the proposed feature tailoring methods. Five accuracy metrics were used for comparison. Normally, the performance is considered as acceptable if the false positive is less than 5%. Both methods show less than 0.5%, but the proposed DoD-standard based detector shows the best result.

TABLE 1. ACCURACY COMPARISON

	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) curve
AUC (PRC): Area Under Curve - PRC (associated Precision/ReCall) curve

Tradeoff Analysis: Detection accuracy is very important, but effective (dynamic) accuracy will be worsened if proper and enough datasets cannot be provided. Therefore, an additional hardware module to extract statistical information with additional PMC registers is required to simultaneously collect more profiles and to promptly extract meaningful statistical information.

If we increase the PMC registers to double, 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. Latency of malware detection will be slightly increased because 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. We also can assume two different modes: *performance tuning mode* and *normal mode*. In normal mode, two PMCs will be used for malware detection which can provide more accuracy. In performance tuning mode, only half of PMCs will be used for detection while the other half will be used for performance tuning. Instead of adding additional PMCs, large number of profiling events can be captured with shorter sampling period with existing PMCs. 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.

Design Alternatives for Hardware Accelerations: Along with additional PMCs, two different acceleration hardware can be considered depending on the design budget: (i) adding only a pre-processing module to generate meta data which will be sent to host processor for machine learning operation; (ii) adding a dedicated detection and decision module. Additional hardware cost varies on design goal and budget. The period of data capturing and feature (event) selection should be dynamically updated to improve detection accuracy.

Two hardware approaches for malware detection can be applied to either existing processors or new processors. Additional hardware cost varies on design goal and budget. Based on the design estimation, the complexity of hardware acceleration is 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 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. Both combinational schemes provide accuracy improvement (5~10%) and detection speedup (up to 10%) with the additional hardware cost (less than 1% of the chip complexity).

V. CONCLUSION

Due to the limited number of PMCs, the selection of architectural features is one of the critical issues 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 malware detection criteria. Our experimental results show that the DoD can differentiate malware applications from benign applications and can also provide highly accurate malware detection through the machine learning framework. For more accurate malware detection in real-time, hardware acceleration module as well as additional PMC registers are required. In future works, a more detailed architectural design for dedicated accelerator to provide more efficiencies in chip area, power and processing time will be investigated.

REFERENCES

[1] N. Patel, A. Sasan, H. Homayoun, "Analyzing Hardware Based Malware Detectors," DAC 2017.
[2] Bircher et al., "Complete system power estimation: A trickle-down approach based on performance events," ISPASS 2007.
[3] Demme et al., "On the feasibility of online malware detection with performance counters," ACM Computer Architecture News, 2013.
[4] Bahador et al., "Hpcmalhunter: Behavioral malware detection using hardware performance counters and singular value decomposition," ICCKE 2014.
[5] A. Garcia-Serrano, "Anomaly detection for malware identificat-ion using hardware performance counters," arXiv:1508.07482, 2015.
[6] Ozsoy et al., "Malware-aware processors: A framework for efficient online malware detection," HPCA 2015.
[7] B. Zhou, et al., "Hardware Performance Counters Can Detect Malware: Myth or Fact?," ASIACCS 2018.
[8] Virus Total. http://www.virustotal.com/intelligence/.
[9] Virus Sign Incorporation. http://www.virussign.com/.
[10] Hall et al., "The weka data mining software: an update," ACM SIGKDD explorations newsletter, 2009.

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2021.Doi Number

Performance Monitoring Counter based Intelligent Malware Detection and Design Alternatives

Jordan Pattee, Shafayat Mowla Anik, and Byeong Kil Lee

Electrical and Computer Engineering Department, University of Colorado, Colorado Springs, CO 80918 USA

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

This work was supported in part by UCCS and 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 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 [20-26] with multiple types of data including performance counter information. Garcia et. al. [21] discuss the feasibility of unsupervised learning to detect attacks. Conversely, Zhou et. al. [27] 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 [28], 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 (accuracy and latency) 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 [29] and Virus Sign [30]. 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 [34], rhythm box [35], 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~~ – for 30 minutes per application~~. The 40 events are based on two types of events which are HARDWARE and HW_CACHE as shown in Table I. We make four events as a group since the processor we use for data collection has only 4 counters.

TABLE I
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

Some malware applications 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 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 applications can be distinguishable characteristics based on our observation. The executable malware has single counter magnitudes up to 100x smaller than benign applications. 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.

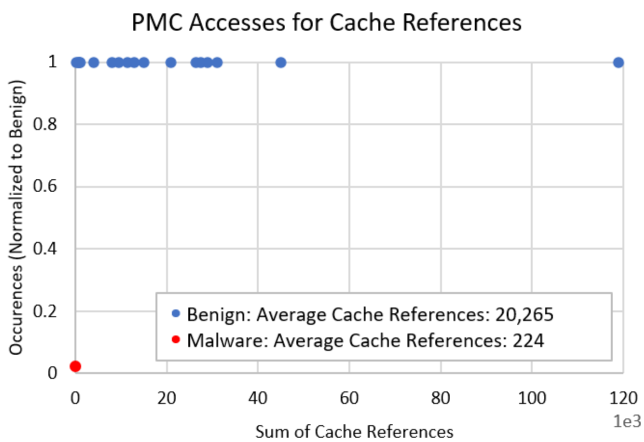


FIGURE 1. **PMC accesses for cache reference.**

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.

$$Degree\ of\ Distribution = \frac{Mean}{(Mean + StdDev)} \qquad (1)$$

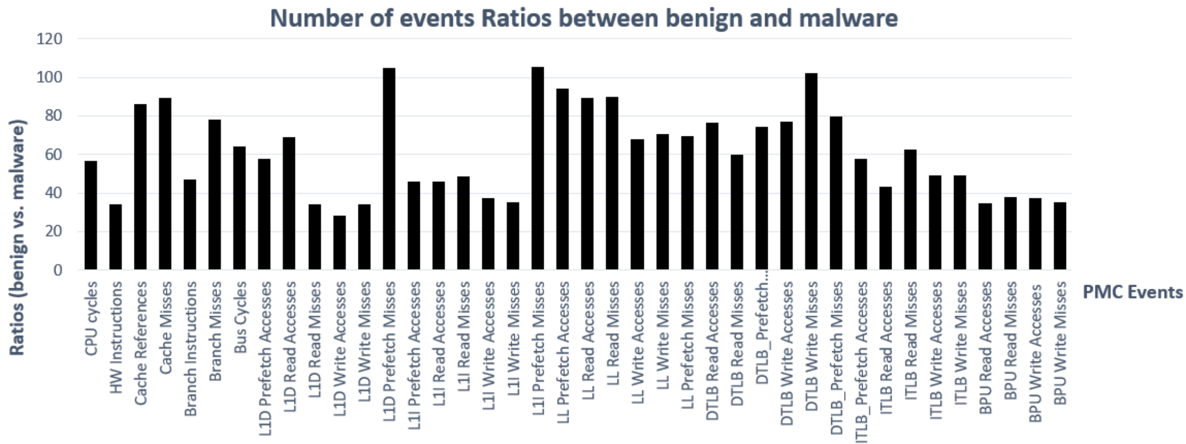


FIGURE 2. **The number of events ratios between benign and malware.**

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.

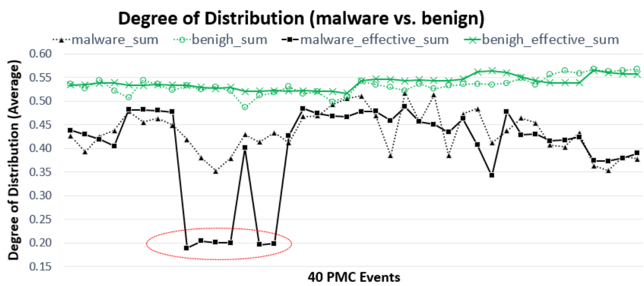


FIGURE 3. **Characterization of each PMC event to the Degree of distribution (average value) of 20 malware and 20 benign samples – sum and effective sum.**

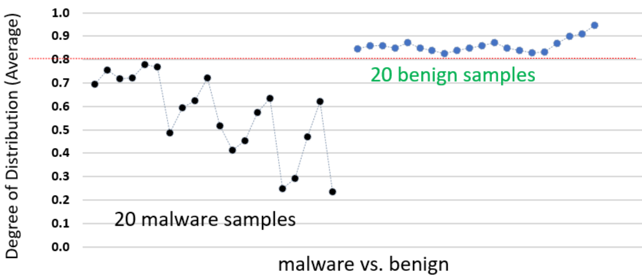
Figure 4 shows the comparison of malware classification with 6 events (tailored) and the case with 40 events (full). For each sample, we extract the average DoD value from 40 PMC events. The threshold lines (red-dotted line) for malware detection are based on the DoD values (average) for each sample, 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.

III. MALWARE DETECTION BASED ON STATISTICAL CHARACTERIZATION

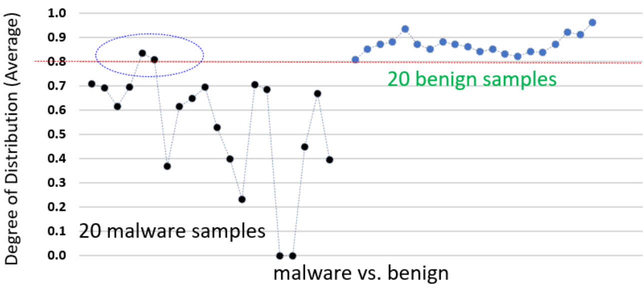
Generally, hardware-based malware detection has some advantages: it can provide a capability for dynamic mechanisms without relying on static signatures, and hardware-based detection also delivers faster processing time. However, one of the disadvantages is the cost of architectural resources (e.g.,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

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

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 [31-32]. 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* [33]. 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.



(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).

Binary Classification: A malware detection scheme is a binary classification: malware or not. There are many classifiers for binary classification [36]. For this experiment, we use 10 classifiers that include Bayes net, logistic classification, multilayer classification, Bagging, decision trees, JRIP, OneR, AdaBoostM1, KStar classification, and random forest [33]. 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 for training and 30% of the samples for testing data [37]. 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 [38].

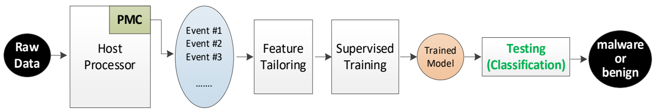


FIGURE 5. Overview of the complete learning framework from prepossessing to classification.

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 II 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 III 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) [39-41].

TABLE II
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

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

As shown in Table III, 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 III
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.

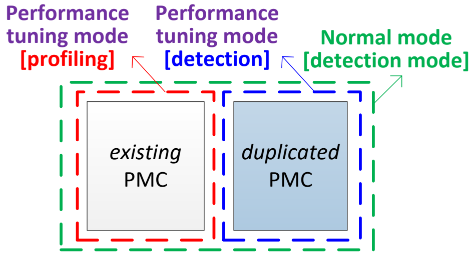
Based on our experiments and analysis, the detection rate is 10~20ms and classification accuracy is 90~97%. The detection rate depends on the sampling rate for capturing profile information, and the classification accuracy depends on the number of PMC registers. By adding more PMC registers, classification 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

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).



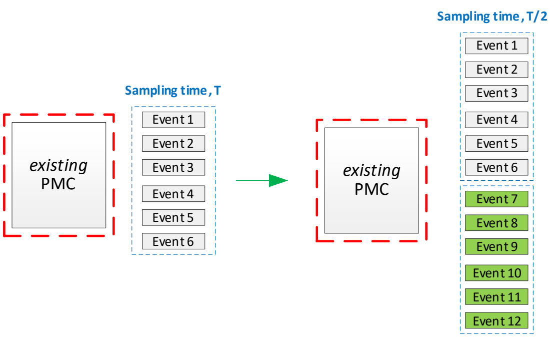
(a) Additional PMC

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.



(a) Existing single PMC module with more profiling events

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

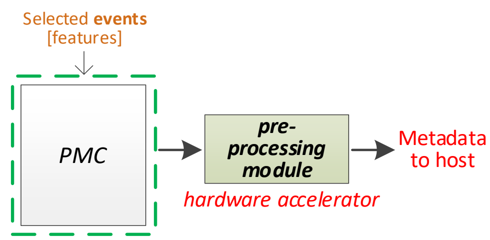
(b) Hardware cost

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

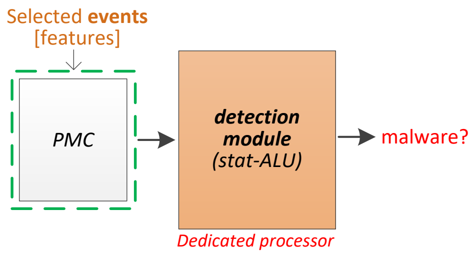
With Existing PMCs: Instead of adding more PMCs, a large number of profiling events can be captured with a shorter sampling period with the existing PMCs. As shown in Figure 7, assuming the existing PMC module has 6 monitoring counters, the PMC module can capture 6 events during the sampling time, T. 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 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.



(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.

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] Al-Asli et al., "Review of Signature-Based Techniques in Antivirus Products," IEEE ICCISci.2019.8716381, 2019.

[2] You et al., “Malware Obfuscation Techniques: A Brief Survey,” BWCCA, 2010.

[3] N. Patel, A. Sasan, H. Homayoun, "Analyzing Hardware Based Malware Detectors," DAC 2017.

[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. Virology Hacking Techn., vol. 13, no. 1, pp. 1–12, 2017.

[5] L. Nataraj, A Signal Processing Approach To 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, Australia, Dec. 2011.

[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: ACM, Nov. 2017.

[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.

[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), vol. 2, Jun. 2016.

[14] W. Huang, 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.

[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.

[16] M. Sebastián, R. Rivera, P. Kotzias, and J. Caballero, “Avclass: A tool for massive malware labeling,” in International Symposium on Research in Attacks, Intrusions, and Defenses. Cham, Switzerland: Springer, 2016.

[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] Bircher et al., "Complete system power estimation: A trickle-down approach based on performance events," ISPASS 2007.

[19] Demme et al., "On the feasibility of online malware detection with performance counters," ACM Computer Architecture News, 2013.

[20] Bahador et al., "Hpcmalhunter: Behavioral malware detection using hardware performance counters and singular value decomposition," ICCKE 2014.

[21] A. Garcia-Serrano, "Anomaly detection for malware identification using hardware performance counters," arXiv:1508.07482, 2015.

[22] Ozsoy et al., "Malware-aware processors: A framework for efficient online malware detection," The 21st International Symposium on High Performance Computer Architecture (HPCA) 2015.

[23] Hossein et al., “Ensemble Learning for Effective Run-Time Hardware-Based Malware Detection: A Comprehensive Analysis and Classification,” DAC 2018.

[24] Tobiyama et. al., “Malware Detection with Deep Neural Network Using Process Behavior,” International Computer Software and Applications Conference 2016.

[25] Dahl et al., “Large-scale malware classification using random projections and neural networks,” ICASSP 2013.

[26] B. Athiwaratkun and J. Stokes, “Malware Classification with LSTM and GRU language models and a character-level CNN,” ICASSP, 2017.

[27] B. Zhou, et al., "Hardware Performance Counters Can Detect Malware: Myth or Fact?," ASIACCS 2018.

[28] J. Pattee and B. K. Lee, "Design Alternatives for Performance Monitoring Counter based Malware Detection," International Performance Computing and Communications Conference (IPCCC), 2020.

[29] Virus Total Academic Malware Samples. http://www.virustotal.com/intelligence/ [upon request], Accessed on: Aug. 5, 2021.

[30] Virus Sign Malware Samples. http://samples.virussign.com/samples, Accessed on: Jun. 10, 2021.

[31] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," ICLR 2015.

[32] L. Bottou, et. al., "Global training of document processing systems using graph transformer networks," In Proc. of Computer Vision and Pattern Recognition, Puerto-Rico, 1997.

[33] Hall et al., "The weka data mining software: an update," ACM SIGKDD explorations newsletter, 2009.

[34] Firefox, https://www.mozilla.org/en-US/firefox/linux/, Accessed on: May. 20, 2021.

[35] Rhythm box, https://help.ubuntu.com/community/Rhythmbox, Accessed on: May. 20, 2021.

[36] V. Bahel, S. Pillai and M. Malhotra, "A Comparative Study on Various Binary Classification Algorithms and their Improved Variant for Optimal Performance," 2020 IEEE Region 10 Symposium (TENSYMP), 2020.

[37] D. M. W. Powers and A. Atyabi, "The Problem of Cross-Validation: Averaging and Bias, Repetition and Significance," 2012 Spring Congress on Engineering and Technology, 2012.

[38] B. Ghojogh and M. Crowley, "The Theory Behind Overfitting, Cross Validation, Regularization, Bagging, and Boosting: Tutorial," arXiv:1905.12787v1, 2019.

[39] 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 Proceedings of the 23rd Conference on Design, Automation and Test in Europe, March, 2020.

[40] J. Davis and M. Goadrich, "The relationship between precision-recall and ROC curves," In Proceedings of the 23rd International Conference on Machine Learning, 2006.

[41] J. Keilwagen, I. Grosse, and J. Grau, "Area under precision-recall curves for weighted and unweighted data," PLoS ONE, 9(3):e92209, 03 2014.

JORDAN PATTEE received her bachelor's degree from University of Colorado Colorado Springs in 2021, and she worked for Symetrix Corporation as an application and design engineer. She is a graduate student in Kyushu University in Japan. Her research interest includes malware detection, deep learning, resistive memory, and modeling. She worked at LiCA (Laboratory for Intelligent Computer Architecture) as an undergraduate researcher, and she got an excellent student award on her graduation.

SHAFAYAT MOWLA ANIK is a Ph.D. student at University of Colorado Colorado Springs. He received his bachelor's and master's degree from University of Dhaka, Bangladesh in 2015 and 2017 respectively. His research interest includes deep learning, computer architecture, malware detection, cybersecurity, low power and performance modeling. He is working at LiCA (Laboratory for Intelligent Computer Architecture) as a graduate research assistant.

BYEONG KIL LEE (M'87) (M'05-SM'11) received the Ph.D. degree in computer engineering from the University of Texas at Austin, Austin, in 2005. He is currently an Assistant Professor in the Department of Electrical and Computer Engineering at the University of Colorado at Colorado Springs. 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. Prior to joining the faculty at UCCS, he worked for Samsung as a Vics President for 5 years and he was an assistant professor at the University of Texas at San Antonio (UTSA) for 4 years. Also, he was a senior design engineer and senior research staff at TI (Texas Instruments) and ADD (Agency for Defense Development) for 5 and 10 years respectively. His research is published in several international conferences and journals, and he is also a reviewer for conferences and journals. He serves on the technical program committee and organizing committee of some conferences and workshop such as ISCA, ISPASS, IISWC, ICCD, HPCA, ASAP, PACT, ICPP and UCAS.

Extended contents from IPCCC poster

[28] J. Pattee and B. K. Lee, "Design Alternatives for Performance Monitoring Counter based Malware Detection," 2020 IEEE 39th International Performance Computing and Communications Conference (IPCCC), 2020

- We published our early version of research to International Performance Computing and Communications Conference (IPCCC), 2020. It is a 2-page of poster paper. We expanded the poster paper with more characterization, additional modeling, more experiments and analysis. We believe it is <35% similarity to our previous work.

- Details on expanded part include:

  - Our previous work is referenced and mentioned in the new article:

    Based on our previous research [28], 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.

  - Detailed characterization of benign and malware applications in terms of performance monitoring counter's accesses (e.g., cache reference, the number of events ratios, etc.) and represented as graphs.

    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.

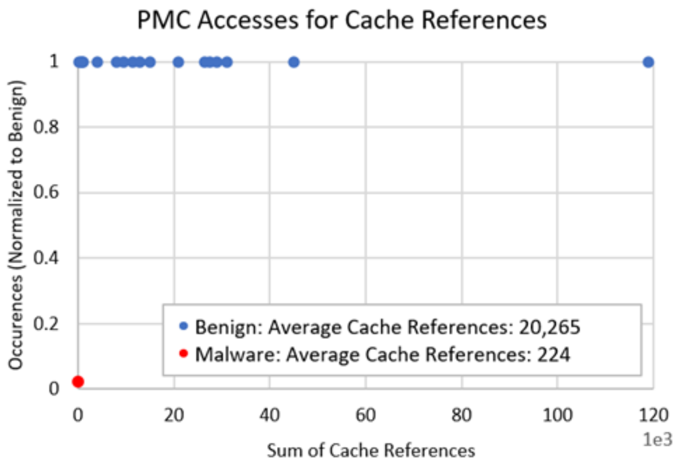


FIGURE 1. PMC accesses for cache reference.

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.

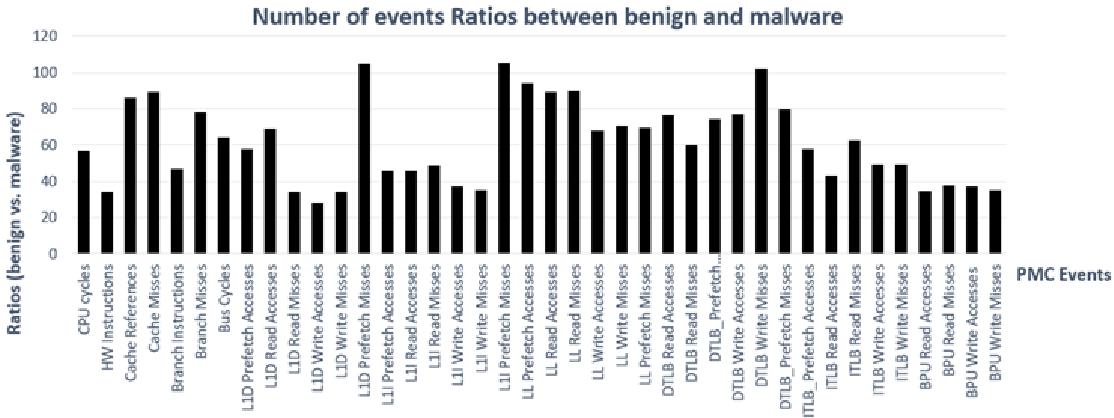


FIGURE 2. The number of events ratios between benign and malware.

- More detailed characterization of each PMV event with respect to degree of distribution for malware and benign applications and represented as graphs.

  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 with respect to the DoD.

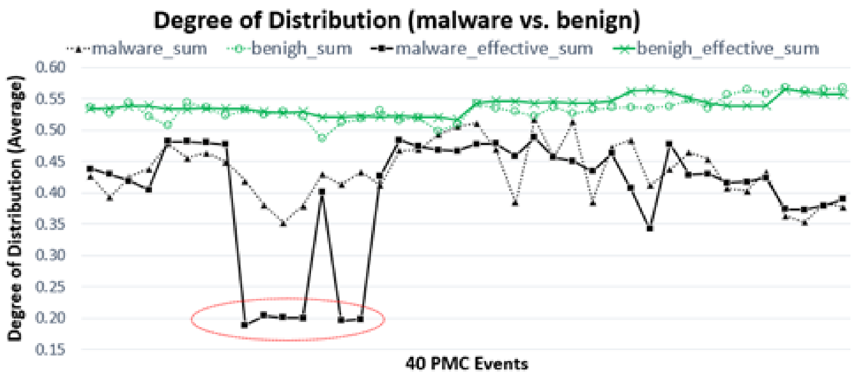
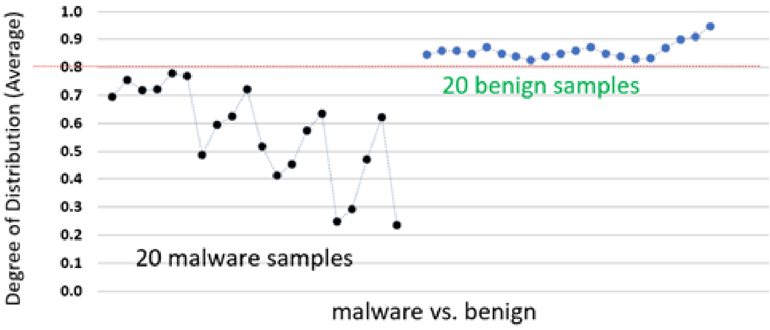


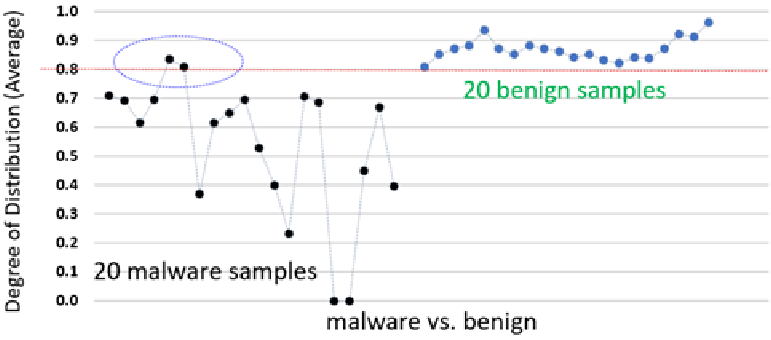
FIGURE 3. Characterization of each PMC event to the Degree of distribution (average value) of 20 malware and 20 benign samples – sum and effective sum.

- Extended investigation on malware classification. Experiments with more malware and benign samples with 40 PMC events and 6 PMC events.

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.



(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).

- Extended experiments with more binary classifiers to extract 6 attributes from the tool WeKa.

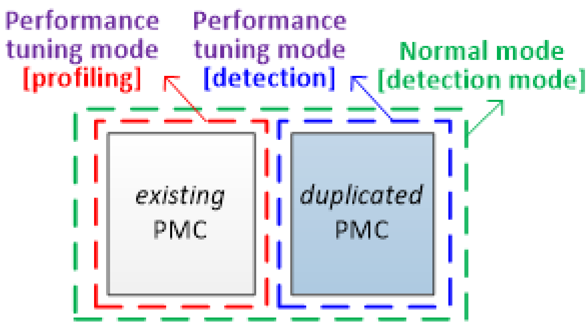
Binary Classification: A malware detection scheme is a binary classification: malware or not. There are many classifiers for binary classification [36]. For this experiment, we use 10 classifiers that include Bayes network, logistic classification, multilayer classification, OneR, decision trees, JRIP, Bagging, AdaBoostM1, KStar classification, and random forest [33].

TABLE II
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

- We added an implication section for hardware design for accuracy improvement. We proposed two methods: with additional PMCs and with existing PMCs. Conceptual architecture and operations are described, and hardware costs are compared.

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).



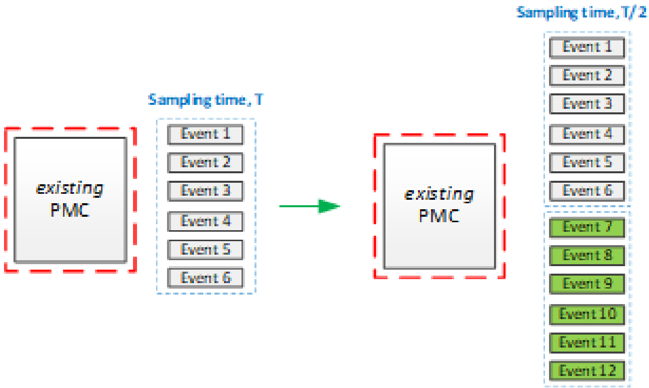
(a) Additional PMC

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.

Instead of adding more PMCs, 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 same sampling time (T), where each event will be monitored only for T/2.



(a) Existing single PMC module with more profiling events

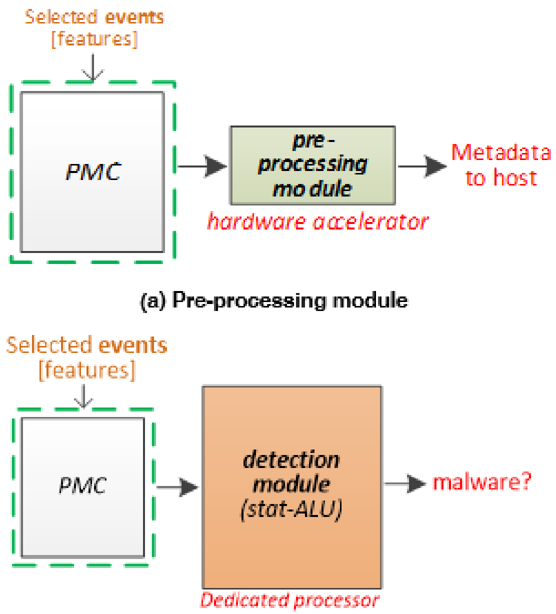
Design features	Single PMC
Area	1X
Power	1X
Latency	1X+Δ (more ALU ops)

(b) Hardware cost

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

- We added two detection modules for improving detection latency: a preprocessing module and with a dedicated detection module. We added accuracy improvement and detection speedup based on our performance estimation.

  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 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.



(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.