2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering

Customized Machine Learning-Based Hardware-Assisted Malware Detection in Embedded Devices

Hossein Sayadi, Hosein Mohammadi Makrani, Onkar Randive, Sai Manoj P D, Setareh Rafatirad, Houman Homayoun
George Mason University, Fairfax, VA 22030
{hsayadi, hmohamm8, orandive, spudukot, srafatir, hhomayou}@gmu.edu

***Abstract*— The emerging embedded systems, which account for a wide range of applications are often highly resource-constrained challenging the conventional software-based methods traditionally deployed for detecting and containing malware in general purpose computing systems. In addition to the complexity and cost (computing and storage), the software-based malware detection methods mostly rely on the static signature analysis of the running programs, requiring continuous software update in the field to remain accurate in capturing emerging malware, which is not affordable for embedded systems with limited computing and communication bandwidth. Hardware-assisted Malware Detection (HMD) though found to be more efficient, limited computing power and resources in embedded systems as well as the small number of available Hardware Performance Counter (HPC) registers that can be simultaneously captured, make accurate runtime malware detection in embedded devices a challenging problem. In response, this work proposes a lightweight customized HMD approach which takes advantage of HPC features to effectively detect and further classify various malware classes at runtime. To realize a runtime solution that relies on limited available HPCs and to enhance the accuracy of malware detection, we use customized HMD for individual class of malware that utilizes various Machine Learning (ML) classifiers to detect malware using the four most important HPC features.**

Keywords—malware detection, embedded systems, hardware performance counter, machine learning.

I. INTRODUCTION

Recent advances in digital electronics and wireless communications have enabled a widespread proliferation of embedded systems ranging from micro-sensors, cell phones and PDAs to smart homes and military applications. Security is an important issue in embedded systems due to the adoption of embedded systems in many mission and safety-critical systems [13,14,21]. Malware is a piece of code designed by attackers to perform various malicious activities in computer systems [5,15]. Many of the existing software attacks on conventional computing platforms such as servers and desktops can be launched on the embedded systems due to their similarities in using general purpose processors computing systems and their connectivity to internet. In addition, the advancement of computing devices in the form of mobile devices and Internet-of-Things (IoT) further exacerbates the malware attacks calling for developing effective malware detection solutions.

The constant interaction between physical and cyber worlds has made security one of the important concerns in the design of embedded systems [10,13,19]. Ensuring security in embedded systems translates into several design challenges, imposed by the unique features of these systems. The existing malware detection approaches such as signature-based detection and semantics-based solution (off-the-shelf Norton and McAfee antivirus tools) are ineffective for resource-constrained embedded systems due to the limited available computing resources [4,5,11]. Hardware-assisted Malware Detection (HMD) methods have emerged recently as an effective solution to improve the security of computing systems by using Machine Learning (ML) models to detect pattern of malicious applications. The ML classifiers are trained using low-level features such as processor Hardware Performance Counters (HPCs) data captured at runtime to appropriately represent the application behavior. HPCs are a set of special-purpose registers in the processing units to capture the trace of hardware events for a running application [11,22]. While HPCs have been widely employed to predict the performance and energy efficiency of computing systems [7,16,17,18,20], here we leverage HPCs to improve the security. Hardware-assisted malware detection has two advantages: 1) robustness, as manipulating the hardware events is more complex compared to code obfuscation; and 2) the latency of detecting malware is smaller by order of magnitude with low hardware costs [4,5].

Prior studies on HMD have shown the effectiveness of ML classifiers in classifying anomalies in low-level feature spaces captured by HPCs to differentiate the malware from normal programs [1,2,4,5,6,11,12,22], but in high performance computing domain under the assumption of availability of large number of HPCs. However, malware detection problem in embedded systems domain poses primarily two challenges. First, the systems are mainly limited in terms of computing power and resources as such the use of heavyweight ML classifiers like Neural Network and Bayesian models is not efficient. Second, the malware detection accuracy is proportional to the number of HPC events used by the ML classifier, while, due to physical limitations in microprocessors the number of HPCs that can be read simultaneously in miniature embedded systems is limited to 2 to 6 [4,5,11]. For instance, widely used processors such as ARM Cortex-A5 and Cortex-A8 can afford to have only 2 and 4 HPCs available, respectively, that can be read concurrently. As such, runtime detection of malware in resource-constrained embedded systems needs capturing the most prominent microarchitectural events limited to the number of HPCs available and achieve a high malware detection accuracy, which has been ignored in existing work. Running an application multiple times aid in capturing more microarchitectural events to achieve higher accuracy but is not a practical solution for runtime malware detection, since the hardware can only count a small subset of events concurrently. Furthermore, the prior work on HMD are either limited to one or very few classes of malware and does not evaluate different ML classifiers and its suitability to resource-constrained embedded devices.

In this work, we propose a customized runtime malware detection framework using limited number (only 4) of HPC

2324-9013/18/31.00 ©2018 IEEE
DOI 10.1109/TrustCom/BigDataSE.2018.00251

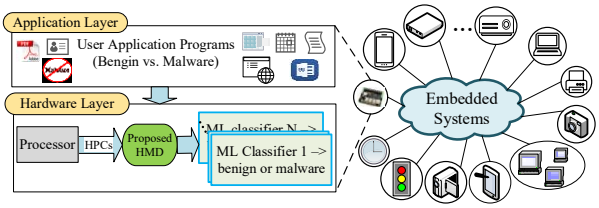

Fig. 1. Application of proposed HMD in embedded systems

features, addressing the challenge faces resource-constrained embedded systems. Fig. 1 illustrates the applicability of the proposed HMD in various embedded devices. The proposed HMD comprises of predicting the malware class using a lightweight multiclass classifier succeeded by a customized per class malware detector. The customized per class malware detector is a detector trained with the HPC event data of the corresponding malware class which is the target of detection. In addition, a comprehensive analysis of malware detection using various ML classifiers and their suitability to resource-constrained embedded systems based on their hardware footprint and latencies are presented. To evaluate the effectiveness of proposed HMD, we thoroughly examine the malware detectors in terms of detection accuracy, and hardware implementation overheads to determine the most suited ML classifiers per malware class for effective malware detection with limited number of HPCs.

II. Proposed Hardware-Assisted Malware Detction

In this section, we present a customized HMD for distinguishing the malware programs from benign applications and determining the class of malware at runtime.

A. Experimental Setup and Data Collection

Fig. 2-(a) illustrates the overview of data collection process in proposed customized HMD. Since each malware class has a different behavior, it allows a customized detector that is trained specifically for that malware class to more effectively perform the classification. We implemented customized detectors for three classes of malware including Rootkit, Trojan, and Backdoor. The applications (both malware and benign) are executed on an Intel Atom C2758 processor running Ubuntu 14.04 with Linux 4.4 Kernel and various HPCs data are collected with the aid of *Perf* tool. We executed more than 3000 benign and malware applications. Benign applications include MiBench benchmark suite [8], Linux system programs, browsers, text editors, and word processor. For malware applications, Linux malware is collected from virustotal.com [3]. After collecting microarchitectural events using *Perf*, we use the WEKA tool [9] to evaluate the accuracy and performance of different ML classifiers. HPC information is collected by running all groups of regular and malware programs within Linux Containers (LXC) which is an isolated environment. Training ML classifiers involves profiling the incoming application with *Perf* tool under Linux and extracting low-level feature values for each training program, reducing the extracted features to the most vital HPCs, and developing a learning model from the training data. It is important to note that the input variables in our classifiers are the HPCs extracted every 10ms interval from running applications, and the output variable is the class of an application. In order to validate each of the utilized ML classifiers, a standard 70%-30% dataset split for training and testing is followed.

B. Feature Reduction

In order to perform runtime malware detection using HPC information, it is essential to represent the applications in the form of low-level microarchitectural features. Such a representation can lead to a high-dimensional data processing which involves large computational overheads and complexity. Thus, it is non-trivial to determine the correct features in building an accurate predictor. In this work, we apply *Principle Component Analysis (PCA)* technique on the extracted HPCs to select the best HPCs for detecting malwares with one single run. PCA analysis allows us to monitor and determine the most vital and distinct microarchitectural parameters to extract application characteristics [7]. Next, we apply clustering and attribute evaluation technique to identify the most relevant features.

We apply PCA to find the best HPCs suited for training the ML-based malware detectors. We started with 16 HPCs for each malware class, we reduced the features to 8 and 4 most significant HPCs to capture the behavior of specific class of malware. The feature reduction results indicate that the identified prominent 4 HPCs are the same across various classes of malware. These HPCs include branch instructions, cache references, branch misses, and node-stores which provide a unique opportunity to identify the class of malware ahead of time for unknown malware. These 4 HPC features are then supplied to each ML-based malware detector as input parameters for each malware class. These features reflect pipeline front-end, pipeline back-end, cache subsystem, and main memory behaviors and are influential in the performance of standard applications.

C. Malware Detection Approach

The overview of the customized malware detection process is depicted in Fig. 2-(b). As shown, first an effective multiclass classification model is developed to estimate the class of the running application. Next, depending on the predicted application type, customized ML classifiers are chosen for the predicted malware class with limited number of HPCs.

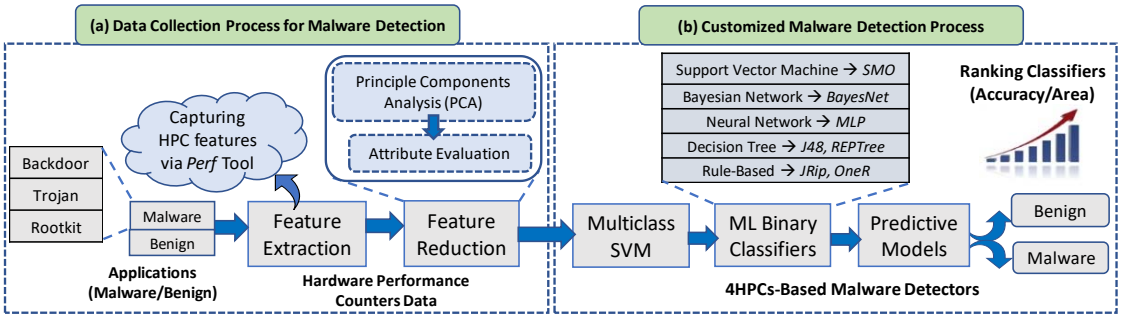

Fig. 2. Proposed hardware-assisted malware detection framework

Application Class Prediction: Customized malware detector is a ML classifier which is trained specifically with the samples that belong to one particular malware class. As each of the customized detectors is trained to classify a different class of malware, they are each answering a different classification question. Initially, the system is unaware of existence of malware in the application, as such the use of customized detectors cannot be effective. As such, for each class of malware experimented, one customized malware detector is employed. To address this concern, we first propose to predict the behavior of the application (benign or a particular malware class) using a multiclass Support Vector Machine (SVM). The SVM classifier is a supervised learning that analyzes data used for classification and regression analysis. The multiclass SVM primarily converts the basic binary classification to a multiclass problem, i.e. with more than two possible discrete outcomes.

In this work, the output of multiclass SVM is corresponding to the set of feasible classes of applications including 4 individual classes, one for "benign" program and 3 malware classes namely "Rootkit", "Trojan", and "Backdoor". The SVM model is trained using extensive set of HPCs data captured by running various benign and malware programs. The inputs to the SVM consists of the 4 features described previously. Given the inputs during runtime, the proposed multiclass SVM can then calculate a probability of each candidate application class being executed. The output with the highest probability is then chosen as the current application type during runtime. The evaluation results of the proposed SVM model show that lowering the number of HPCs to the top 4 features retains an accuracy of 70% and an average Area Under the Curve (AUC) of 0.81 indicating the effectiveness of SVM model to predict the right type of running application by using only 4 HPC features.

Customized Machine Learning Classifiers: As observed before, using SVM classifier does not provide a high malware detection accuracy when using limited number of HPCs. To address this challenge, we add the second stage of detection using various type of machine learning techniques. We propose using customized malware detector which is a ML classifier individually trained using the characteristics of specific class of malware. These ML classifiers are shown in Fig. 2. The rationale for selecting these machine learning models are: First, they are from different branches of ML methods covering a diverse range of learning algorithms which are inclusive to model both linear and non-linear problems. Second, the prediction model produced by them can be a binary classification which is compatible with our malware detection.

III. Evaluation Results

In this section, the proposed HMD is evaluated in terms of malware detection accuracy, hardware overheads as well as the accuracy per unit of area to analyze hardware suitability.

A. Malware Detection Accuracy

The accuracy is defined as the percentage of correctly classified samples by the malware detector. Fig. 3 depicts the accuracy results of proposed runtime HMD for different ML classifiers with three different classes of malware experimented. In addition, the accuracy when employing the first level of detection (multiclass SVM) is also illustrated where the class of application is predicted (referred as MSVM). As seen, using only one-level classifier i.e., only multiclass SVM as a first level provides an accuracy of 70% when using 4 HPCs which is low

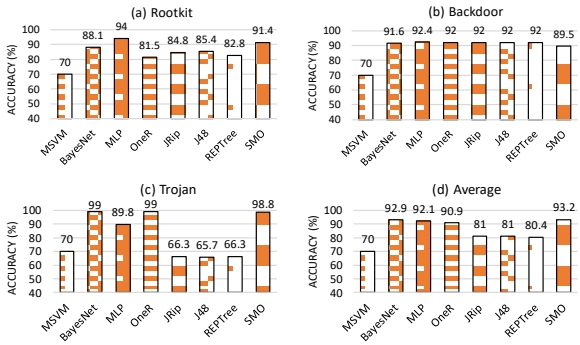

Fig. 3. Accuracy of various malware detectors: comparison of multiclass SVM (MSVM) and proposed customized HMD for (a) Rootkit, (b) Backdoor, (c) Trojan, (d) average results

for security-critical embedded devices. The results show that the proposed approach enhances the malware detection accuracy in majority of the cases.

In case of Rootkit, the accuracy is improved by more than 20% when using MLP and SMO classifiers. Similar observations are made for Trojan and Backdoor. MLP performs best for Rootkit and Backdoor detection, whereas OneR and BayesNet outperforms MLP in terms of accuracy in Trojan. In addition, a reduction in the rule-based and decision tree-based techniques such as JRip, J48, and REPTree is observed for Trojan. The OneR and BayesNet customized classifiers are both delivering the highest accuracy of 99%. Fig. 3-(d) also shows the average accuracy results for various ML classifiers. On an average, the accuracy of malware detection is increased by up to 23.2% (in SMO classifier) across all the three tested classes of malware. As seen, the HMD employing customized SMO and BayesNet classifiers achieves highest malware detection accuracy of 93.2% and 92.9%, respectively.

B. Hardware Overhead Analysis

As the embedded systems are resource-constrained, the hardware footprint and the involved latency play a crucial role in adopting ML classifier for runtime malware detection. While complex algorithms such as Neural Networks can deliver high accuracy, they also add significant area overhead and are slower in terms of hardware implementation making them unsuitable for resource-limited embedded devices. Hence, we evaluate the hardware overheads posed by different ML classifiers. For hardware implementation, we implemented and synthesized ML classifiers using Vivado HLS for Xilinx Virtex-7 FPGA. In order to evaluate the hardware implementation cost, in Table I, we report the results for different ML classifiers that use 4 HPCs for runtime malware detection. Latency unit is represented in terms of the number of clock cycles (cycles @10 ns) required to classify input vector and the area is the total number of utilized LUTs, FFs, and DSP units inside FPGA.

As expected, the area and the latency for heavyweight ML classifiers such as MLP and BayesNet are much larger compared

TABLE I: HARDWARE IMPLEMENTATION RESULTS

Classifier	Latency (@10ns)	Hardware Area
BayesNet	6	7645
J48	3	584
JRip	2	156
MLP	102	25667
OneR	1	292
REPTree	3	377
SMO	22	2466

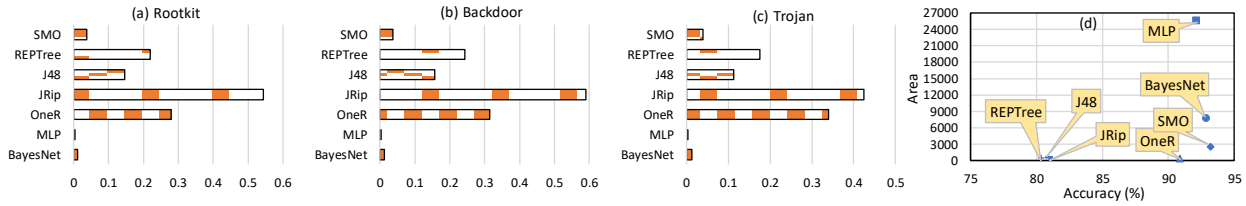

Fig. 4. Accuracy/Area comparison of malware detectors in proposed HMD for (a) Rootkit, (b) Backdoor, (c) Trojan, (d) Accuracy vs. Area trade-offs

to other classifiers. The rule-based and tree-based classifiers have significant smaller hardware footprint compared to more complex classifiers (up to ~80x smaller area, and 5-30x faster execution time). Table I also presents the delay imposed by individual ML classifiers. Malware detection process with the proposed two-level HMD is performed in less than *ms*, whereas the malware has an execution time of few *ms* [4,5]. As such, the proposed HMD can perform efficient runtime malware detection with high accuracy and low overhead.

C. Efficiency of ML Classifiers in Proposed HMD

Lastly, to accordingly account for malware detection accuracy and area overhead impact, in Fig. 4 (a-c) we compare accuracy over a unit of hardware area (Accuracy/Area) for various ML classifiers across different class of malware. We use detection accuracy over area to identify malware detectors that require small area and yet can detect the maliciousness of program with high accuracy and performance. A classifier with a higher ratio is considered better than with lower ratio. Among all classifiers, the rule-based and tree-based classifiers are found to be more efficient compared to the highly accurate but complex BayesNet and MLP. We evaluate the combined impact of ML classifier accuracy and area overhead to see the accuracy per unit of required area. Interestingly, the pattern is quite same across different malware classes. Among all ML classifiers, the JRip is seen to have a very high Accuracy/Area, whereas the MLP has the least due to its large hardware overhead. Furthermore, it is seen that rule-based and tree-based classifiers (JRip, OneR, and J48) outperform other classifiers in terms of accuracy per unit area, indicating the best suitability for resource-constrained embedded systems.

As shown in Fig. 4-(d), a clear trade-off is seen between area and accuracy achievable for runtime hardware-assisted malware detection. The ML classifiers such as MLP has high accuracy, but also higher complexity and hardware footprint. The techniques such as BayesNet, SMO, and OneR show relatively smaller area footprint with high malware detection accuracy. For highly resource-constrained embedded systems, techniques such as J48 and JRip provide smallest hardware overhead, while achieving an accuracy of close to 81% on average. Clearly, the results show trade-offs between detection accuracy, latency, and area overhead. Therefore, it is important to compare ML classifiers for effective malware detection in embedded systems by taking all these parameters into consideration.

IV. Conclusion

In this work, we proposed an effective machine learning-based hardware-assisted malware detection framework for embedded devices which makes use of a limited number (only 4) of low-level features of microprocessor i.e., HPC events to facilitate the runtime malware detection. Compared to traditional single-stage HMD methods, the proposed customized HMD enhances the accuracy up to 29%. Furthermore, the experimental results indicated that while heavyweight classifiers such as MLP, BayesNet, and SMO have higher average malware detection accuracy, the lightweight ML classifiers such as JRip and OneR show very high accuracy per unit of area across all tested classes of malware with MLP being the least. Based on the achieved accuracy, area, and latency, we provided valuable insights that aid in choosing accurate and hardware-suitable ML classifiers for malware detection in embedded system. This comprehensive analysis helps the designers to understand and navigate the trade-offs between several design parameters offered by each learning algorithm.

References

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

[2] J. Demme et al., "On the feasibility of online malware detection with performance counters", In ACM SIGARCH Computer Architecture News, Vol. 41, 2013.

[3] Virustotal intelligence service. http://www.virustotal.com/intelligence/. Accessed: December 2017.

[4] H. Sayadi et al., "Ensemble learning for effective run-time hardware-based malware detection: A comprehensive analysis and classification", In Design Automation Conference (DAC'18), CA, June 2018.

[5] N. Patel et al., "Analyzing hardware based malware detectors", In DAC, June 2017.

[6] A. Tang et al., "Unsupervised anomaly-based malware detection using hardware features", In RAID'14, Springer, 2014.

[7] H. Sayadi et al., "Machine learning-based approaches for energy-efficiency prediction and scheduling in composite cores architectures", In International Conference on Computer Design (ICCD'17), November 2017.

[8] Guthaus et al., "Mibench: A free, commercially representative embedded benchmark suite", In IISWC, 2001.

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

[10] D. Serpanos et al., "Security challenges in embedded systems," In TECS, 2009.

[11] H. Sayadi et al., "Comprehensive assessment of run-time hardware-supported malware detection using general and ensemble learning", In ACM Computing Frontiers (CF'18), Ischia, Italy, May 2018.

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

[13] H. Sayadi., et al., "A data recomputation approach for reliability improvement of scratchpad memory in embedded systems", In Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT'14), Amsterdam, NL, October 2014.

[14] H.M. Makrani, et al., "Evaluation of sofware-based fault-tolerant techniques on embedded OS's components", In DEPEND'14, pp. 51-57, 2014.

[15] G. Jacob et al., "Behavioral detection of malware: from a survey towards an established taxonomy", Journal of Computer Virology, 2008

[16] H. Sayadi et al., "Power conversion efficiency-aware mapping of multithreaded applications on heterogeneous architectures: A comprehensive parameter tuning" In Asia and South Pacific Design Automation Conference (ASP-DAC'18), Jan 2018.

[17] H.M. Makrani et al., "MeNa: A Memory navigator for modern hardware in a scale-out environment", In IISWC'17, October 2017.

[18] H. Sayadi et al., "Scheduling multithreaded applications onto heterogeneous composite cores architecture," In IGSC'17, October 2017.

[19] H.M. Kamali et al., "A fault tolerant parallelism approach for implementing high-throughput pipelined advanced encryption standard", In JVSC, Vol. 25, 2016.

[20] M. Malik, et al., "Co-Locating and concurrent fine-tuning MapReduce applications on microservers for energy efficiency", In IISWC'17, October 2017.

[21] S. Manoj et al., "A Scalable network-on-chip microprocessor with 2.5D integrated memory and accelerator", in IEEE TCAS-I, vol.64, no.6, pp.1432-1443, June 2017.

[22] Wang et al., "Confirm: Detecting firmware modifications in embedded systems using hardware performance counters", In ICCAD'15, 2015.