StealthMiner: Specialized Time Series Machine Learning for Run-Time Stealthy Malware Detection based on Microarchitectural Features

Hossein Sayadi
California State University, Long Beach
hossein.sayadi@csulb.edu

Yifeng Gao
George Mason University
ygao12@gmu.edu

Hosein Mohammadi Makrani
University of California, Davis
hmakrani@ucdavis.edu

Tinoosh Mohsenin
University of Maryland, Baltimore
tinoosh@umbc.edu

Avesta Sasan
George Mason University
asasan@gmu.edu

Setareh Rafatirad
George Mason University
srafatir@gmu.edu

Jessica Lin
George Mason University
jessica@gmu.edu

Houman Homayoun
University of California, Davis
hhomayoun@ucdavis.edu

ABSTRACT

Hardware-Assisted Malware Detection (HMD) techniques deploy Machine Learning (ML) classifiers to detect patterns of malicious applications based on microarchitectural features captured by modern microprocessors' Hardware Performance Counters (HPCs). Existing HMD methods have limited their analysis on detecting malicious applications that are spawned as a separate thread during application execution, hence detecting embedded malware patterns at run-time still remains an important challenge. Embedded malware refers to harmful stealthy cyber attacks in which the malicious code is hidden within benign applications and remains undetected by traditional malware detection approaches. In HMD methods, when the HPC data is directly fed into a machine learning classifier, embedding malicious code inside the benign applications leads to contamination of HPC information, as the collected HPC features combine benign and malware microarchitectural events together. To address this challenge, in this paper we propose *StealthMiner*, a specialized time series machine learning approach to accurately detect embedded malware at run-time using branch instructions feature, the most prominent microarchitectural feature. The results indicate that *StealthMiner* can detect embedded malware at run-time with 94% detection performance on average with only one HPC feature, outperforming the detection performance of state-of-the-art HMD methods by 42%.

CCS CONCEPTS

• **Security and privacy → Security in hardware**;

KEYWORDS

Embedded Malware, Hardware-Assisted Malware Detection, Machine Learning, Stealthy Malware, Time Series Classification

ACM Reference Format:
Hossein Sayadi, Yifeng Gao, Hosein Mohammadi Makrani, Tinoosh Mohsenin, Avesta Sasan, Setareh Rafatirad, Jessica Lin, and Houman Homayoun. 2020. StealthMiner: Specialized Time Series Machine Learning for Run-Time Stealthy Malware Detection based on Microarchitectural Features . In *Proceedings of the Great Lakes Symposium on VLSI 2020 (GLSVLSI '20), September 7–9, 2020, Virtual Event, China.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3386263.3407585

GLSVLSI '20, September 7–9, 2020, Virtual Event, China

ACM ISBN 978-1-4503-7944-1/20/09…$15.00
https://doi.org/10.1145/3386263.3407585

1 INTRODUCTION

The ever-increasing complexity of modern computing systems has led to the proliferation of security vulnerabilities, making such systems suitable targets for sophisticated cyber attacks. The attackers make use of vulnerabilities to compromise systems and deploy malicious activities [21, 25]. The recent evolution of computing devices in mobile platforms and Internet-of-Things (IoT) domains further exacerbates the malware threats calling for effective security countermeasures against such attacks [11, 19]. Traditional software-based malware detection techniques have shown to be inefficient mostly imposing significant complexity and computational overheads on the system. In addition, such detection methods rely on the static signature analysis of the running programs and are not able to detect unknown attacks at run-time. The emergence of new variants of malware threats requires continuous updating the software-based solutions (e.g. off-the-shelf anti-virus) that further utilizes considerable memory and hardware relevant resources making such solutions less practical for emerging computing systems especially in embedded mobile and IoT devices [3, 13, 15, 18, 20].

In order to address the shortcomings of conventional malware detection techniques, Hardware-Assisted Malware Detection (HMD), by employing low-level features captured by Hardware Performance Counters (HPCs), has emerged as a promising solution [2, 24]. The HPCs are special-purpose registers built into modern microprocessors to capture the trace of hardware-related events [19, 26].

HMD techniques have shown that malware can be differentiated from normal programs using Machine Learning (ML) techniques applied on hardware performance counter features. In addition, HMDs reduce the latency of detection process by order of magnitude with small hardware overhead [19, 20].

Malicious software attacks have continued to evolve in quantity and sophistication during the past decade. Due to ever-increasing complexity of malware attacks and financial motivations of attackers, malware trends are recently shifting towards *stealthy* attacks [16, 23]. Stealthy malware is a type of attack in which the malicious code is hidden inside the benign application for performing harmful purposes [5, 9, 29]. The main purpose of stealthy malware is to remain undetected for a longer period of time in the computing system. The longer the threat remains undiscovered in the system, the more opportunity it has to compromise computers and/or steal information before a suitable detection mechanism can be deployed to protect against it. Stolfo et al. discovered a new type of stealthy threat referred as *embedded malware* [23]. Under this threat, the attacker embeds the malicious file inside a benign program on the target host such that the benign and malicious applications are executed as a single thread on the system. It has been shown that traditional signature-based antivirus applications are unable to detect embedded malware even when the exact signature of malware is available in the detector database [9, 23]. In this work, we primarily focus on detecting stealthy malware where malicious code is hidden inside the benign program, both executed as a single thread on the target system making the detection more challenging.

The existing studies on hardware-based malware detection have primarily assumed that the malware is spawned as a separate thread while executing on the target host. However, in real-world scenarios malicious programs attempt to hide themselves within a benign application to bypass the detection mechanisms. In HMD methods the HPC data is directly fed to a detector, therefore, for embedded malicious code hidden inside the benign application, the HPC data becomes contaminated as the collected events include the combined benign and malware microarchitectural events. The recent work in [30] is the only HMD study on embedded malware in which they showed that one benign program infused with ransomware cannot be detected by traditional ML-based detection methods. However, they have not proposed any effective solution to tackle the challenge of detecting stealthy malware using HPC features.

In this work, we propose *StealthMiner*, a specialized lightweight time series machine learning approach to accurately detect the embedded malicious patterns hidden inside the benign programs using only one HPC feature (branch instruction). To the best of our knowledge, this is the first work that addresses the challenge of detecting stealthy/embedded malware at run-time using hardware performance counters features. The main objective of this work is to effectively detect the malicious application embedded inside the benign program using the least number of microarchitectural events (only one HPC feature) in which the traditional machine learning-based solutions are unable to detect them with even 8-16 features. To this aim, using an effective feature reduction technique, we first identify the most prominent low-level feature for embedded malware detection. Next, we propose a lightweight scalable time series-based Fully Convolutional Neural Network (FCN) model that automatically identifies potentially contaminated samples in HPC-based time series and utilizes them to distinguish the stealthy malware from benign applications at run-time using branch instructions as the most significant low-level feature.

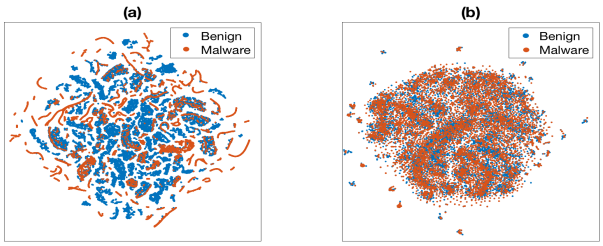

Figure 1: Visualizing the complete benign and malware dataset using t-SNE algorithm: a) malware spawned as a separate thread b) malware embedded inside benign applications

2 PROPOSED STEALTHY MALWARE DETECTION FRAMEWORK

In this section, we describe the proposed machine learning-based approach for effective hardware-based stealthy malware detection.

2.1 Challenge of Detecting Stealthy Malware

Figure 1 illustrates the challenge of detecting embedded malware. Figure 1-(a) visualizes the complete benign and malware HPC data (described in details in Section 2), when the malware is spawned as a separate thread, via t-distributed Stochastic Neighbor Embedding (t-SNE) algorithm a widely used algorithm for visualizing high dimensional data. As seen, the marginal area between malware and benign program is large when malware is spawned as a separate thread indicating that by using traditional ML models (prior works) the malware can be easily detected. However, the converted points of embedded malware data are mixed with each other in Figure 1-(b) depicting the impact of embedding malicious code inside benign applications. The figure highlights the challenge of stealthy malware detection indicating that due to the dense distribution of malware and benign applications features, traditional classification approaches are not able to achieve a high accuracy in detecting embedded malware. As a case study, by applying nearest neighbor classifier on both complete and embedded malware dataset, the classifier can achieve an accuracy of 90% in detecting the malware as a separate thread. However, the classifier can only achieve nearly 60% accuracy in embedded malware detection task when the malicious code is hidden inside the normal program.

2.2 Experimental Setup and Configuration

This section provides the details of the experimental setup and data collection process. The benign and malware applications are executed on an Intel Xeon X5550 machine (4 HPC registers available) running Ubuntu 14.04 with Linux 4.4 Kernel and HPC features are captured using *Perf* tool available under Linux at sampling time of 10ms. *Perf* provides rich generalized abstractions over hardware specific capabilities. HPC-based profilers are currently built into almost every popular operating systems. *Linux perf* is a new implementation of performance counter support for Linux which is based on the Linux kernel subsystem *perf-event* and provides users

a set of commands to analyze performance and trace data. It exploits *perf-event-open* function call in the background which can measure multiple events simultaneously. In our experiments, we executed more than 3500 benign and malware applications for data collection. Benign applications include real-world applications comprising MiBench and SPEC2006 , Linux system programs, browsers, and text editors. Malware applications collected from virustotal and virusshare online repositories include Linux ELFs and scripts created to perform malicious activities and include 850 Backdoor, 640 Rootkit, and 1460 Trojan samples. The functionality of Backdoor applications is trying to provide remote access to the remote user (attacker) and facilitates information leakage; Rootkits provide the attackers with privilege access to modify the registers and authorized programs; and Trojans perform phishing of confidential information in the system.

In our experiments, the HPC information is collected by running applications in an isolated environment referred as Linux Containers (LXC) which unlike common virtual platforms such as VMWare or VirtualBox, provides access to actual hardware performance counters data instead of emulating HPCs. In order to effectively address the non-determinism and overcounting issues of HPC registers in hardware-based security analysis discussed in recent works [1, 30], we have extracted 56 low-level CPU events available under *Perf* tool using static performance monitoring approach where we can profile applications several times measuring different events each time. In other words, since Intel Xeon processor hosts only 4 HPC registers physically available [4], we can only measure 4 events at a time. As a result, multiple runs are required to fully capture all events. We divided 56 events into 14 batches of 4 events and executed each application 14 times at sampling time of 10ms to gather all microarchitectural events. Furthermore, to ensure that running malware inside the LXC does not contaminate the system's environment and also no contamination occurs in collected data due to the previous run of the program, the container is destroyed after each run.

2.3 Microarchitectural Features Analysis

Identifying the most important microarchitectural features is an crucial step for building efficient ML classifiers [17]. For selection of features, we first use Correlation Attribute Evaluation to rank all captured features by calculating Pearson correlation between each attribute and class. Next, we apply Principle Component Analysis (PCA) to find the best HPCs suited for training the ML-based malware detectors. PCA is a class of dimensionally reduction techniques that captures most of the data variation by rotating the original data to a new variable in a new dimension [22, 28]. We employ PCA to reduce the features and apply a hierarchical clustering technique to group similar features and identified the top 4 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 which include branch instructions, cache references, branch misses, and node-stores.

2.4 Stealthy Malware Threat Models

For modeling the embedded malware threats, we have considered persistent malicious attacks which occur once in the benign application with notable amount of duration attempting to infect the system. Persistent malicious codes are primarily a subset of Advanced Persistent Threat (APT) which is comprised of stealthy and continuous computer hacking processes, mostly crafted to perform a specific malfunction activities. For the purpose of thorough analysis, we deployed various malware types for embedding the malicious code inside the benign application including Backdoor, Rootkit, Trojan, and Hybrid (Blended) attacks. For per-class embedded malware analysis, malware traces taken from one category of malware, are randomly embedded inside the benign applications and the proposed detection approach attempts to detect the malicious pattern. Furthermore, the Hybrid threat combines the behavior of all classes of malware.

In order to create embedded malware time series and model the real-world applications scenario, with capturing interval of 10ms for HPC features monitoring, we consider 5 sec. infected running application (benign application infected by embedded malware). For this study, 10,000 test experiments were conducted in which malware appeared at a random time during run of a benign program. In our experiments, three different sets of data including training, validation, and testing sets are created for comprehensive evaluation of the *StealthMiner* approach. Each dataset contains 10,000 complete benign HPC time series and 10,000 embedded malware HPC time series. Since the attacker can deploy unseen malware program to attack system, we create these three datasets with three groups of recorded malware HPC time series consisting of 33.3% for training, 33.3% for validation, and the remaining of whole recorded data for testing evaluation.

2.5 Overview of *StealthMiner*

StealthMiner malware detection framework is based on a lightweight Fully Convolutional Neural Network (FCN)-based time series classification. Primarily, it attempts to automatically identify potentially contaminated intervals in HPC-based time series at runtime and utilize them to distinguish the embedded malware from benign applications. The overview of *StealthMiner* and its comparison with prior works is described in Figure 2. The network is a simplified version of neural network models inspired from previous general convolutional neural network-based time series classification models [6, 27]. As shown in Figure 2-(a), our proposed solution is based on the least number of HPCs and targets detecting stealthy malware that have been ignored in prior studies. Furthermore, as seen in Figure 2-(b), the proposed FCN-based malware detector is created by stacking two 1-D convolution layers with 16 and 2 kernels, respectively. The size of kernel in these two convolution layers is 2 and 3, respectively. These convolution layers aim at selecting the subsequence of HPC time series for identifying the malware. Next, a global average pooling layer is applied to convert the output of the convolution layer into low dimension features. These features are then fed into a fully connected neural network to distinguish the embedded malware from benign applications.

Concretely, given a time series of HPC features of $x = x_1, x_2, \ldots, x_N$, where N is the length of the time series in the first 1-D convolution layer, an output of k_{th} kernel is computed by:

$$t_{i,k}^{(1)} = \sum_{j \in 1,2} w_{k,j,1} x_{i+j-1} + b_1 \quad (1)$$

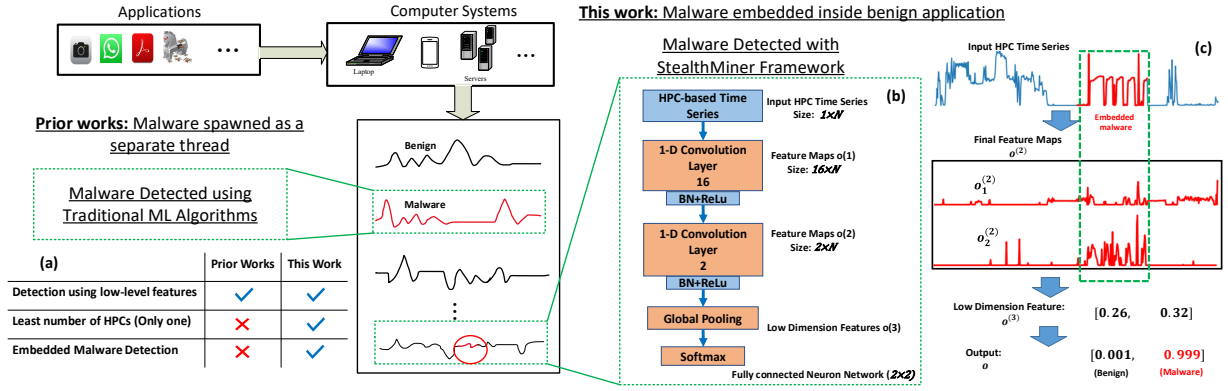

Figure 2: Overview of *StealthMiner*, the proposed specialized time series machine learning approach for embedded malware detection

where 2-d vector $[w_{k,1,1}, w_{k,2,1}] \in \mathbf{w}$ is the weight of k_{th} kernel and $\mathbf{w} = \{w_{k,j,1} | k = 1 \ldots, 16, j = 1, 2\}$ is a 16×2 matrix that describes all weights of first layer. Given $t_k^{(1)} = [t_{1,k}^{(1)}, \ldots, t_{N,k}^{(1)}]$, a batch normalization function, $t_k^{(2)} = BN(t_k^{(1)})$, and a ReLu activation function, $o_k^{(1)} = ReLu(t_k^{(2)})$, are then applied. $BN(.)$ is a function which normalizes mean and variance of the $t_k^{(1)}$ to 0 and 1, respectively, and ReLu activation function sets any negative value in $t_k^{(2)}$ to 0. Also, $o_k^{(1)}$ is a N dimension feature map generated from the k_{th} kernel. We denote $o^{(1)} = [o_1^{(1)}, \ldots, o_{16}^{(1)}]$ as the output of the convolution layer. Intuitively, convolution layer converts original time series of length N into 16 different N dimensional feature maps capturing different potential local features that can be used to classify the input data [27].

The $o^{(1)}$ is then fed into next convolution layer with total number of kernels equal to 2. This layer summarizes $o^{(1)}$ into two different feature maps which can be computed via:

$$t_{i,k'}^{(3)} = \sum_{k=1}^{16} \sum_{j=1}^{3} w_{k',k,j,2} o_{i+j-1,k}^{(1)} + b_2 \tag{2}$$

where the weight of all kernels is a 3-d tensor $w_{k',k,j,2}$ of size $2 \times 16 \times 3$. For each $t_i^{(3)}$, $BN(.)$ and $ReLu(.)$ functions are further applied and four feature maps (denoted as $o^{(2)} = [o_1^{(2)}, o_2^{(2)}]$) are generated. Intuitively, stacking two convolution layers can increase the accuracy of the framework and the ability of model to detect complicated features which are not possible to be captured by single convolution layer [27]. Note that any positive value inside the $o_1^{(2)}, o_2^{(2)}$ indicates the potential HPC intervals used to determine whether the input HPC time series contains an embedded malware. Next, we conduct a global average pooling step to convert feature map $o^{(2)}$ into low dimension features. In particular, given a feature map of $o_k^{(2)} \in o^{(2)}$, we deploy the average value of all elements inside $o_k^{(2)}$ as the low dimension feature. As a result, this step converts $o^{(2)}$ into a 2-d vector (denoted as $o^{(3)}$).

Finally, $o^{(3)}$ is fed into a fully connected neural network with softmax activation function formulated below where a standard neural network layer is designed for our target classification task in detecting embedded malware:

$$o = Softmax(W^T o^{(3)} + b_3) \tag{3}$$

where $Softmax(x) = \frac{e^{x_i}}{\sum_{k=1}^{2} e^{x_k}}$. Eq. (3) first converts $o^{(3)}$ into a new 2-d real value vector via linear transformation $W^T o^{(3)} + b_3$, where W is a 2×2 matrix and b_3 is a 2×1 vector. Next, all elements in the vector is mapped to [0,1] via $Softmax$ function. The final output is a 2-d vector $o = [o_1, o_2]$ which describes the possibility that the time series is benign or infected by malware.

Suppose that we denote all the weights and the output of network is Θ and $\Theta(x) = [\Theta_1(x), \Theta_2(x)]$, respectively. Given a training dataset $\mathcal{D}$ and the network weights Θ, we update Θ by minimizing the binary cross-entropy loss as follows:

$$L = \sum_{(\mathbf{x_i}, y_i) \in \mathcal{D}} -y_i \log(\Theta_1(\mathbf{x}_i)) - (1 - y_i) \log(\Theta_2(\mathbf{x}_i))) \tag{4}$$

where $\mathbf{x}_i$ and y_i is the HPC time series and the associated ground true label of the i_{th} record in $\mathcal{D}$. And $y_i \in \{0, 1\}$ indicates whether the time series is benign or contains malware. Equation 4 can be minimized via standard back propagation algorithm, a widely used model for training various types of neural networks [6, 27]. It primarily updates weights in neural network by propagating the loss function value from the output to the input layer and iteratively minimizes the loss function for each layer via gradient descent method. In this work, for each layer, the weights are optimized via Adam optimizer [8], a stochastic gradient descent method used to efficiently update weights of neural network.

In order to demonstrate the functionality of the *StealthMiner* approach in identifying the malware embedded inside the benign program, a detection case study is presented in Figure 2-c. As shown, an HPC-based time series is the input to the classifier which contains an embedded rootkit malware (the embedded malware is highlighted in red). To identify the hidden malicious pattern, *StealthMiner* generates two feature maps $o_1^{(2)}, o_2^{(2)}$ via the proposed fully convolution neural network. The $o_1^{(2)}$ and $o_2^{(2)}$ are then categorized as a 2-d feature vector $o^{(3)}$ by calculating the simple average of all the value in the feature map. In the given example, $o^{(3)}$ is equal to [0.26, 0.32]. This 2-d feature is then fed into a fully connected neural network layer and the detector analyzes the input HPC time

series and attempts to find that whether the input trace contains an embedded malware or not. In this case it successfully identifies the embedded malware with significantly high probability (0.999).

We implemented the proposed embedded malware detection framework via Pytorch deep learning library. For evaluating *StealthMiner* framework using accuracy and F-measure (described in Section 3.1), the proposed detector determines whether the input time series contains embedded malware by computing the $argmax(o)$. And for measuring the the Area Under the Curve (AUC), we directly use the $output$ computed via equation (3). Different from existing neural network time series classification models proposed in prior works, *StealthMiner* framework has small total number of kernels and layers which dramatically reduces the number of parameters and the cost of detecting malware in new HPC time series. For instance, in the latest neural network introduced by [6], to classify a time series the proposed solution needs more than 150,000 parameters. Hence, applying such heavyweight classification models to our embedded malware detection problem would significantly increase the overhead and complexity of our design, which certainly makes the solution impractical. In contrast, *StealthMiner* framework only contains 200 parameters. Having small number of parameters enhances the efficiency of the proposed ML-based malware detection solution highlighting the effectiveness and applicability of *StealthMiner* to efficiently identify the embedded malware.

3 EXPERIMENTAL RESULTS

3.1 Performance Evaluation Criteria

The *StealthMiner* approach is evaluated using precision, recall, F-score, and detection accuracy (the overall rate of correctly classified samples). In binary classification techniques the precision (p) is the proportion of the sum of true positives versus the sum of positive instances. For instance, it is the probability for a positive sample to be classified correctly. The recall (r) is the proportion of instances that are predicted positive and are also actually positive of all the instances that are positive. The F-score or F-measure is the weighted harmonic mean of the precision and recall reaches its best value at 1 and worst at 0 and is formulated as $\frac{2 \times (p \times r)}{p+r}$.

Since the F-measure and accuracy are not the only metrics to determine the performance of the ML-based malware detectors, we also evaluate *StealthMiner* using Receiver Operating Characteristics (ROC) graphs. The ROC curve represents the fraction of true positives versus false positives for a binary classifier as the threshold changes. We further deploy the Area under the Curve (AUC) measure for ROC curves which corresponds to the probability of correctly identifying malware and benign programs. and is more related to the robustness of the classifier. Robustness is referred to how well the classifier distinguishes between binary malware and benign classes, for all possible threshold values. The AUC of the best possible classifier is equal to 1, meaning that we can find a discrimination threshold under which the classifier obtains 0% false positives and 100% true positives.

3.2 Evaluation of *StealthMiner*

For the purpose of comprehensive evaluation, we compare our proposed approach with both the recent general time series classification approaches and recent traditional machine learning-based HMD techniques. We studied two general time series classification methods including a k-Nearest Neighbour (KNN) classifier, a classical time series classification method, and Bag-of-Pattern-Features (BOPF) [10] classifier, which is a recently proposed scalable time series classification approach. Given the input time series, KNN classifier will assign same class label to the input time series based on the most similar observed time series in training set in which the similarity is measured by Euclidean distance.

Table 1: Evaluation results of *StealthMiner* for validation set

Type	Precision	Recall	F-score	Accuracy
Hybrid	0.85	0.89	0.88	0.87
Rookit	0.93	0.88	0.91	0.91
Trojan	0.91	0.87	0.89	0.89
Backdoor	0.88	0.94	0.91	0.91
Average	0.89	0.9	0.9	0.89

As described before, Bag-of-Pattern-Features based time series classification approach is one of the recent fast time series classification algorithm that has a significantly low time and computational complexity compared with other existing time series classification approaches while maintaining a very high accuracy. As a result, for a comprehensive comparison of *StealthMiner* with state-of-the-arts, we implemented different ML-based HMD techniques and time series classification presented in recent prior works including JRIP [14, 19], J48 [14, 19], Logistic Regression [7, 12, 14], KNN [2], and BOPF [10] that are representing the rule-based, decision tree, regression-based, and time series machine learning classifiers and have demonstrated high accuracy for detecting malware (spawned as separate thread) in recent works.

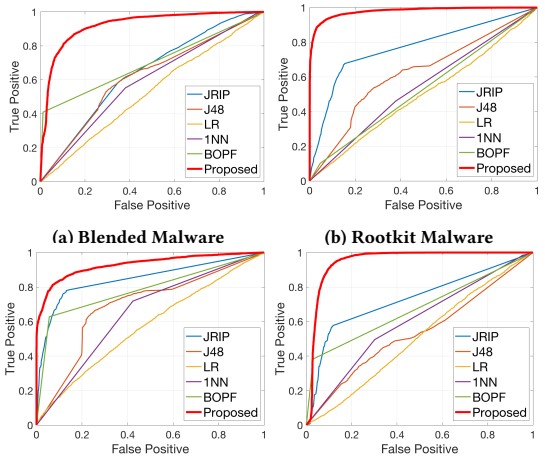

Figure 3: ROC graphs of *StealthMiner* vs. various state-of-the-arts for detecting different classes of embedded malware

Table 1 presents the evaluation results of malware detection for different classes of embedded malware for validation set. The results show that our proposed lightweight neural network-based solution can achieve average accuracy, precision, recall and F-score of nearly 0.9 across all types of experimented embedded malware only by using the most prominent HPC feature (branch instructions). This makes the run-time detection of stealthy malware feasible which is primarily eliminating the need to execute applications multiple times to capture various low-level features suitable for HMD.

Table 2: AUC values of testing set results of *StealthMiner* vs. traditional ML-based detectors in prior works

Attack Type	***StealthMiner***	JRIP	J48	LR	KNN	BOPF
Hybrid	**0.92**	0.64	0.62	0.53	0.6	0.7
Rookit	**0.98**	0.77	0.62	0.5	0.54	0.53
Trojan	**0.93**	0.85	0.69	0.57	0.65	0.79
Backdoor	**0.91**	0.73	0.54	0.51	0.6	0.68
Average	**0.94**	0.75	0.62	0.52	0.58	0.67

Figure 3 illustrates the ROC graphs of the proposed approach compare to state-of-the-art HMD and time series classification techniques. The correspondent AUC values for each embedded malware category are further presented in Table 2. A higher AUC value means that the ROC graph is closer to the optimal threshold and the classifier is performing better in terms of identifying the stealthy malware and classification of malware and benign applications. The ROC results clearly indicate the effectiveness of the proposed approach in this work as compared with prior ML-based malware detection and time series classification. As can be seen, our proposed approach, *StealthMiner*, achieves an average AUC value of 0.94 across all experimented categories of embedded malware. Furthermore, *StealthMiner* significantly outperforms the traditional ML algorithms used in recent HMD works, JRip, J48, and LR, by up to 0.48, and further outperforms tested time series classification approaches by up to 0.45 for embedded Rootkit detection.

4 CONCLUSION

Detection of applications' malicious patterns using microarchitectural features has emerged recently as a promising alternative solution to address the inefficiency of software-based malware detection mechanisms. Prior works on Hardware-Assisted Malware Detection (HMD) using Machine Learning (ML) have primarily considered that the malicious software is spawned as a separate thread while infecting the target computer system. This essentially means that the HPCs data captured at run-time inserted to the classifier belongs only to the malware program. In real-world applications however, the malware can be embedded inside a benign application, rather than spawning as a separate thread producing a more harmful attack. Therefore, the HPCs information collected at run-time could belong to both malware and benign applications. Our analysis showed that this HPCs data pollution could result in performance degradation of traditional ML-based malware detectors. In response to this challenge, in this work we proposed *StealthMiner*, a lightweight specialized time series-based Fully Convolutional Neural Network approach to effectively detect the stealthy malware inside the the benign applications at run-time. Our experimental results demonstrated that the proposed detector, using only the most prominent HPC feature, branch instructions, can detect the embedded malware with 94% detection performance on average at run-time outperforming the detection performance of state-of-the-art hardware-based malware detection methods and general time series classification methods by up to 42% and 36%, respectively.

5 ACKNOWLEDGMENT

This research was supported in part by DARPA SSITH program under the award number 97774952.

REFERENCES

[1] S. Das and et al. 2019. SoK: The challenges, pitfalls, and perils of using hardware performance counters for security. In *IEEE Symposium on Security and Privacy.*
[2] J. Demme and et al. 2013. On the Feasibility of Online Malware Detection with Performance Counters. In *International Symposium on Computer Architecture (ISCA'13).* ACM, 559–570.
[3] A. S. Gazafroudi and et al. 2017. Energy flexibility assessment of a multi agent-based smart home energy system. In *IEEE 17th International Conference on Ubiquitous Wireless Broadband (ICUWB).* 1–7.
[4] Intel. 2016. Intel 64 and ia-32 architectures software developer manual, volume 3b: System programming guide.
[5] X. Jiang and et al. 2007. Stealthy Malware Detection Through Vmm-based "Out-of-the-box" Semantic View Reconstruction. In *ACM Conference on Computer and Communications Security (CCS'07).* 128–138.
[6] Fazle Karim and et al. 2018. LSTM fully convolutional networks for time series classification. *IEEE Access* 6 (2018), 1662–1669.
[7] Kh. N. Khasawneh and et al. 2015. Ensemble Learning for Low-Level Hardware-Supported Malware Detection. In *International Workshop on Recent Advances in Intrusion Detection (RAID'15).* 3–25.
[8] D. P. Kingma and J. Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
[9] Wei-Jen Li and et al. 2007. A Study of Malcode-Bearing Documents. In *Detection of Intrusions and Malware Vulnerability Assessment (DIMVA'07).* Springer, Berlin, Heidelberg, 231–250.
[10] X. Li and et al. 2017. Linear Time Complexity Time Series Classification with Bag-of-Pattern-Features. In *ICDM'17.* 277–286.
[11] A. Mosenia and N. K. Jha. 2017. A Comprehensive Study of Security of Internet-of-Things. *IEEE Transactions on Emerging Topics in Computing* 5, 4 (Oct 2017), 586–602.
[12] M. Ozsoy and et al. 2015. Malware-aware processors: A framework for efficient online malware detection. In *HPCA'15.* 651–661.
[13] S. M. P. Dinakarrao and et al. 2019. Lightweight Node-level Malware Detection and Network-level Malware Confinement in IoT Networks. In *Design, Automation Test in Europe Conference Exhibition (DATE).* 776–781.
[14] N. Patel and et al. 2017. Analyzing Hardware Based Malware Detectors. In *DAC'17.* ACM, 25:1–25:6.
[15] S. Rezaei and et al. 2018. Scalable Multi-Queue Data Transfer Scheme for FPGA-based Multi-Accelerators. In *IEEE 36th International Conference on Computer Design (ICCD'18).* IEEE, 374–380.
[16] E. M. Rudd and et al. 2017. A Survey of Stealth Malware Attacks, Mitigation Measures, and Steps Toward Autonomous Open World Solutions. *IEEE Communications Surveys Tutorials* 19, 2 (2017), 1145–1172.
[17] H. Sayadi and et al. 2017. Machine learning-based approaches for energy-efficiency prediction and scheduling in composite cores architectures. In *35th International Conference on Computer Design (ICCD'17).* 129–136.
[18] H. Sayadi and et al. 2018. Comprehensive Assessment of Run-Time Hardware-Supported Malware Detection Using General and Ensemble Learning. In *ACM International Conference on Computing Frontiers (CF'18).*
[19] H. Sayadi and et al. 2018. Ensemble learning for effective run-time hardware-based malware detection: A comprehensive analysis and classification. In *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC'18).* 1–6.
[20] H. Sayadi and et al. 2019. 2SMaRT: A Two-Stage Machine Learning-Based Approach for Run-Time Specialized Hardware-Assisted Malware Detection. In *Design, Automation Test in Europe Conference Exhibition (DATE'19).* 728–733.
[21] Nader Sehatbakhsh and et al. 2019. REMOTE: Robust external malware detection framework by using electromagnetic signals. *IEEE Trans. Comput.* 69, 3 (2019), 312–326.
[22] Aghaei E. Serpen G. 2018. Host-based misuse intrusion detection using PCA feature extraction and kNN classification algorithms. In *Intelligent Data Analysis.*
[23] S. J. Stolfo and et al. 2007. Towards Stealthy Malware Detection. In *Malware Detection.* Springer US, Boston, MA, 231–249.
[24] A. Tang and et al. 2014. Unsupervised Anomaly-Based Malware Detection Using Hardware Features. In *International Workshop on Recent Advances in Intrusion Detection (RAID'14).* Springer, 109–129.
[25] Han Wang and et al. 2020. Mitigating Cache-Based Side-Channel Attacks through Randomization: A Comprehensive System and Architecture Level Analysis. In *Design, Automation Test in Europe Conference Exhibition (DATE'20).*
[26] Han Wang and et al. 2020. SCARF: Detecting Side-Channel Attacks at Real-time using Low-level Hardware Features. In *International Symposium on On-Line Testing and Robust System Design (IOLTS'20).* IEEE.
[27] Z. Wang and et al. 2017. Time series classification from scratch with deep neural networks: A strong baseline. In *IJCNN'17.* IEEE, 1578–1585.
[28] S. Wold and et al. 1987. Principal component analysis. *Chemometrics and Intelligent Laboratory Systems* 2, 1 (1987), 37 – 52. Proceedings of the Multivariate Statistical Workshop for Geologists and Geochemists.
[29] H. Zhang and et al. 2014. Detection of stealthy malware activities with traffic causality and scalable triggering relation discovery. In *ASIACCS'14.*
[30] B. Zhou and et al. 2018. Hardware Performance Counters Can Detect Malware: Myth or Fact?. In *ASIACCS'18.* 457–468.