Deep Neural Network and Transfer Learning for Accurate Hardware-Based Zero-Day Malware Detection

Zhangying He
California State University, Long Beach
mandy.he@student.csulb.edu

Amin Rezaei
California State University, Long Beach
amin.rezaei@csulb.edu

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

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

ABSTRACT

In recent years, security researchers have shifted their attentions to the underlying processors' architecture and proposed Hardware-Based Malware Detection (HMD) countermeasures to address inefficiencies of software-based detection methods. HMD techniques apply standard Machine Learning (ML) algorithms to the processors' low-level events collected from Hardware Performance Counter (HPC) registers. However, despite obtaining promising results for detecting known malware, the challenge of accurate zero-day (unknown) malware detection has remained an unresolved problem in existing HPC-based countermeasures. Our comprehensive analysis shows that standard ML classifiers are not effective in recognizing zero-day malware traces using HPC events. In response, we propose *Deep-HMD*, a two-stage intelligent and flexible approach based on deep neural network and transfer learning, for accurate zero-day malware detection based on image-based hardware events. The experimental results indicate that our proposed solution outperforms existing ML-based methods by achieving a 97% detection rate (F-Measure and Area Under the Curve) for detecting zero-day malware signatures at run-time using the top 4 hardware events with a minimal false positive rate and no hardware redesign overhead.

CCS CONCEPTS

• **Security and privacy → Intrusion/anomaly detection and malware mitigation**; **Security in hardware**.

KEYWORDS

Deep Learning, Hardware-Based Malware Detection, Machine Learning, Transfer Learning, Zero-Day Attack

ACM Reference Format:
Zhangying He, Amin Rezaei, Houman Homayoun, and Hossein Sayadi. 2022. Deep Neural Network and Transfer Learning for Accurate Hardware-Based Zero-Day Malware Detection . In *Proceedings of the Great Lakes Symposium on VLSI 2022 (GLSVLSI '22), June 6–8, 2022, Irvine, CA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3526241.3530326

GLSVLSI '22, June 6–8, 2022, Irvine, CA, USA

ACM ISBN 978-1-4503-9322-5/22/06…$15.00
https://doi.org/10.1145/3526241.3530326

1 INTRODUCTION

Malicious software (also known as malware) is an emerging cyber-attack that is usually bundled with seemingly legitimate programs to lure incautious users [5, 21]. Recently, Hardware-Based Malware Detection (HMD) techniques have emerged as a promising substitute to overcome the inefficiencies and performance overheads of conventional software-based detection strategies. HMD techniques leverage low-level hardware events that are monitored via Hardware Performance Counters (HPCs) registers [4, 14, 17, 19, 20, 22, 23, 26]. HPCs are specialized registers built in modern microprocessors to collect the hardware events of running applications [6, 27]. HMD methods train standard Machine Learning (ML) algorithms on HPC events to develop accurate classifiers for detecting signatures of malicious software. Machine learning has recently gained a tremendous attention of researchers due to its potential to learn the hidden representation from the abundant data and to automate complex classification tasks [9, 16]. Previous HMD works have shown the effectiveness of standard ML techniques for detecting known malware patterns. However, as highlighted below, we identify some major challenges of existing HPC-based detection methods and propose a deep learning-based solution to realize an effective and accurate hardware-based zero-day malware detection.

Challenge 1: Determining Key Hardware Events. Identifying the most prominent low-level events is an essential step for effective hardware-based malware detection [19, 20]. There exist numerous events in modern microprocessors, each representing a different functionality; thus, monitoring all of them leads to data with high dimensionality. Moreover, analysis of such a raw dataset would result in a high computational complexity and delay [15] making it less feasible for efficient HMD solutions. Therefore, as different HPC events are employed for various purposes, it is important to effectively determine the most suitable hardware events to develop accurate ML-based countermeasures for malware detection.

Challenge 2: Detection of Zero-Day Malware. Zero-day attacks exploit potentially serious software security vulnerabilities that are undocumented (unknown) in the database of the detection mechanism [1]. Lack of signature history or clear remediation strategy has made detection of zero-day malware a long-standing challenge for anomaly detection in securing modern computer systems. In addition, existing ML-based detection methods have ignored the challenging problem of zero-day malware detection. Therefore, they are inherently unscalable and inflexible, as the inclusion of any new

malware types would require training of new models hardening the efficiency and applicability of the solution.

Challenge 3: High False Positive Rate. Conventional ML-based malware detection methods also suffer from a high false positive rate issue in which the benign application is classified as malware. When detecting unknown malware the ML models are often confused benign as malware. Our experiments across different branches of standard ML algorithms demonstrate that standard ML models used in existing HMDs falsely detect benign applications as malware on zero-day test with a significantly high false positive rate on an average across different algorithms. Thus, this challenge creates a disruption and low accuracy to the security countermeasure against emerging cyber-attacks that needs to be addressed urgently.

Our comprehensive examination across different types of malware and machine learning algorithms used for HPC-based malware detection indicates that standard machine learning classifiers (widely used in prior works) fail in recognizing the signature of zero-day (unknown) malware with a high detection performance and low false positive rate. In particular, the results show a clear performance degradation for standard ML classifiers used for HPC-based zero-day malware detection. In response to the challenges and deficiencies of existing ML-based malware detection solutions, we first identify the most prominent hardware events for accurate HPC-based malware detection using an effective feature selection technique based on Mutual Information (MI) method.

Next, we propose *Deep-HMD*, a two-stage Deep Neural Network (DNN)-based approach for accurate and effective hardware-based zero-day malware detection. *Deep-HMD* first transforms HPC-based malware and benign data to images, and then leverages a lightweight deep learning approach to obtain a high malware detection performance (for both known and unknown tests) despite using a small number of hardware events captured at run-time by existing HPCs. To the best of our knowledge, *Deep-HMD* is the first DNN-based framework for accurate hardware-based zero-day malware detection that enables a lightweight and efficient transfer learning strategy on HPC-based data of new malware types (in an image format), therefore, it is extensible and generalizable.

2 PROPOSED METHODOLOGY

This section presents the proposed two-stage deep learning-based approach for accurate hardware-based zero-day malware detection.

2.1 Feature Engineering

In our experiments, the benign and malware programs are profiled on an Intel Xeon X5550 machine. To effectively address the non-determinism and overcounting issues of HPC registers in hardware-based security analysis discussed in recent works [3, 28], we have extracted low-level CPU events available under *Perf* tool using a static performance monitoring approach where we can profile applications several times measuring different events each time. HPC events are monitored with a sampling time of 10ms within Linux Containers (LXC) as an isolated profiling environment. More than 5,000 benign and malware applications are executed for data acquisition. Benign applications include real-world applications comprising MiBench [7] and SPEC2006 [10], Linux system programs, browsers, and text editors. Malware applications, collected

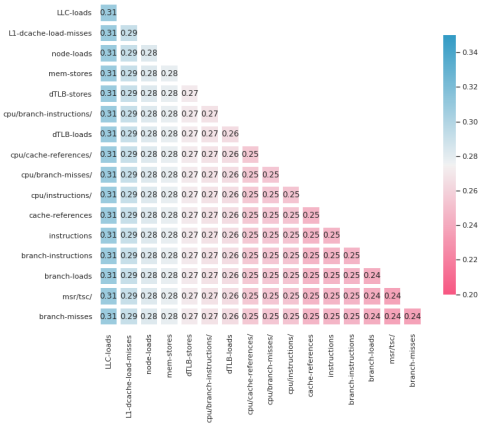

Figure 1: Top 16 HPC features' accumulated mutual information gain to Y. The more blue highlight, the more contribution of the feature to the label Y.

from and categorized by VirusShare and VirusTotal online repositories, comprise nine types of malware including worm, virus, botnet, ransomware, spyware, adware, trojan, rootkit, and backdoor.

As highlighted before, feature selection (e.g., analyzing the importance of the hardware events) is an important step in developing accurate ML models for hardware-based malware detection [21, 22]. To address the *Challenge 1* of existing HMD methods, we employ Mutual Information (MI) method in information theory to analyze the HPC events and select the most prominent features. We use Scikit Learn library's $mutual\_info\_classif$ algorithm [18] to estimate MI from k-nearest neighbor statistics [13]. Regarding features X and label Y, the MI measure $I(X, Y)$ is obtained by estimating the marginal entropies $H(X)$, $H(Y)$, and the joint entropy $H(X, Y)$ as follows:

$$I(X, Y) = H(X) + H(Y) - H(X, Y) \quad (1)$$

For each data point i, the MI method computes I^i based on its neighboring data points. It first finds the k-closest neighbors falling inside of the distance to point i. Using $\psi(\cdot)$ as the digamma function, N is the total samples, Nx_i is the data sample falling within the distance d with k neighbors, and m_i is the total number of neighbors in the dataset. The estimated MI is defined as below:

$$I^i = \psi(N) - \psi(Nx_i) + \psi(k) - \psi(m_i) \quad (2)$$

Figure 1 illustrates the heatmap of the top 16 features from our MI implementation. We select the top four hardware events that show significant accumulated information gains to train a model, considering that most modern microprocessors' counters can only monitor a limited number of events at once during applications execution time [19, 20]. The selected four hardware events include node-loads, LLC-loads, L1-dcache-load-misses, and mem-stores.

2.2 Machine Learning Classifiers

2.2.1 Standard ML Classifiers. We examine the suitability of various standard machine learning classifiers for known and unknown malware detection. These ML models include RandomForest (RF), DecisionTree (DT), Gaussian Naive Bayes (GNB), Logistic Regression (LR), ExtraTreeClassifier (ExtraTree), RidgeClassifier (Ridge), KNN, SVM, and BaggedDT. These ML models cover a diverse range of algorithms and the final predictor can be a binary classification model which is aligned with the malware detection task.

2.2.2 Deep Neural Network (DNN). Traditional ML classifiers are primarily used to train on structured data such as tabular data stored in CSV files or relational databases. On the other hand, Deep Neural Networks (DNNs) are most commonly used on unstructured data such as images and natural language processing. Computer vision-based DNN models can recognize hierarchical relationships in analyzing simple to complex features, and characterize the visual system as a hierarchical and feedforward system. While the neurons in the early layers of a DNN have small receptive fields and are sensitive to local features, they can capture more generalized patterns in deeper layers. Recent results of very deep neural networks from the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) have shown that the neural network can achieve a top 5 error rate of 3.57%. Inspired by computer vision applications, in this work we leverage DNN to develop an accurate and intelligent malware detection framework based on image-based HPC data.

2.2.3 Transfer Learning. Recent studies have reported that a well-trained DNN could transfer its knowledge of generalized features and feature extraction ability from one domain to another. The work in [12] highlight that transfer learning can also share the architecture-related parameters in a new field while maintaining a high-performance rate. Notably, they present that a CNN architecture can transfer its knowledge trained on the ImageNet domain to a new problem domain with high accuracy and stable results. Motivated by such advances, in this work we develop a transfer learning strategy combined with a DNN model in zero-day hardware-based malware detection. ImageNet is significantly different from malware and benign datasets, where ImageNet contains more generic images in everyday lives. In contrast, malware and benign datasets have numerical features from the processor's HPC events. Our work is the first in the field that explores the functionality and effectiveness of leveraging DNN network transferred from ImageNet to the tabular-like zero-day malware detection domain based on only a few hardware tracing events monitored at run-time.

2.3 Overview of *Deep-HMD* Framework

In this work, we propose *Deep-HMD* framework to address the *Challenges 1 & 2* of existing HMD methods and to overcome the limitations of standard ML classifiers in detecting zero-day malware. To this aim, we first explore the performance of standard ML classifiers trained with the most prominent HPC events. We train the ML models with default parameter settings as our base learners. We then examine the models across various metrics on the known test and zero-day test datasets. The ML classifiers are implemented using scikit-learn [18] and are used to analyze how well they can perform on known and zero-day test datasets. Next, given the weak performance of ML models (as we will show in Section 3), we present the *Deep-HMD* framework as the target hardware-based zero-day malware detector.

We investigate state-of-art deep learning model architecture that has trained over ImageNet Large Scale Visual Recognition Challenge (ILSVRC). ILSVRC uses the smaller portion of the ImageNet which consists of 1000 categories with a total of 1.3 million training images, 50,000 validation images, and 100,000 testing images. There are several advantages in using transfer learning. Firstly, the pre-trained model has already learned to recognize patterns from millions of images. Secondly, training from scratch requires a larger dataset and high training time, while using a pre-trained model and transfer learning can maintain high accuracy when fine tuned in a new field. The work in [25] studied that among the many popular DNN networks, ResNet is an appropriate choice in network-based deep transfer learning. ResNet is a convolutional neural network that implements residual blocks of "skip connections" to alleviate the issue of vanishing gradient by setting up an alternate shortcut for the gradient to pass through. In addition, they enable the model to learn an identity function. This ensures that the higher layers of the model do not perform any worse than the lower layers. ResNet is also a simple architecture such that residual blocks do not add any major complexity to the network so that all the common optimization methods can be used in training residual networks. Therefore, we implemented our proposed transfer learning scheme based on ResNet18 which has 18 layers in total.

2.3.1 Threat Model. Recent ML-assisted malware detection methods using HPC events have mainly considered two major validation methods including cross validation and percentage split to examine the effectiveness of their models. The cross validation method splits the dataset into $K(1, ..., n)$ folds and selects one of them as a target testing dataset while the rest of the folds are used for the training dataset. And in the percentage split method, the dataset is divided into two sections based on the percentage setting allocated to training and the other to the testing set. However, the major issue with these validation techniques is that the testing data is split from the large dataset and is part of the same data type used in the training dataset. Hence, such validation methods could not imitate the zero-day or unknown testing scenarios occurring in real-world applications in which the trained machine learning classifiers should have never seen the testing dataset.

To model the zero-day malware threat type in our experiments, among all nine malware types, we held out all of the four types of malware from rootkit, backdoor, virus, and ransomware as the target zero-day test data. These four types of malware are not presented in the training and known test datasets, thus, the zero-day malware set is totally unknown from the training dataset. For benign, we held out 30% of all benign data aside as a zero-day test benign dataset. We kept both malware and benign aside to imitate the zero-day testing in real-world scenarios where the malware is undocumented in the training database of the detection mechanism. The rest of the five types of malware including trojan, spyware, botnet, worm, and adware as well as the rest of benign samples are considered for training and known test purposes, and we randomly split them into 70% for training and 30% for known testing. The difference between known-test and zero-day-test in our experiments is that the known-test data contains the same malware types with the training dataset but with different unseen data and the zero-day-test data contains different malware types from the training dataset that are considered as new unknown attacks. After data are split, we relabel all types of malware as malware and leave benign as benign. Notably, our *Deep-HMD* framework uses the same datasets during training, known-test, and zero-day test same as all classifiers. Also, classical ML models use the tabular format data, and *Deep-HMD* employs the image data converted from corresponding tabular data.

2.3.2 Architecture of Deep-HMD. *Deep-HMD* is a two-stage intelligent and salable framework that achieves an accurate and robust

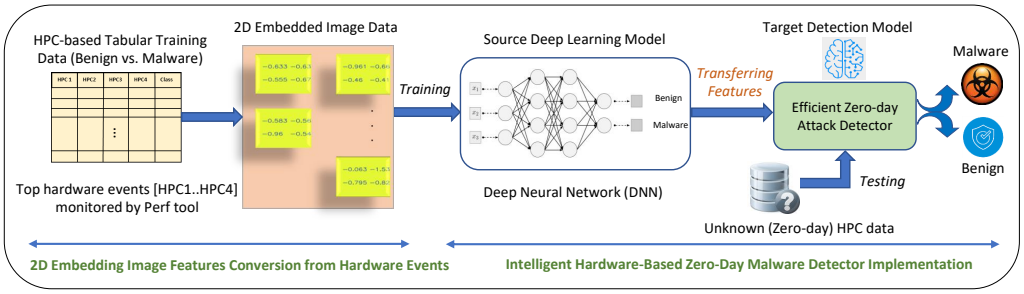

Figure 2: Overview of *Deep-HMD*, the proposed zero-day malware detection framework using 4 selected hardware events.

zero-day malware detection performance. The general overview of the *Deep-HMD* is depicted in Figure 2. As seen, during the first level the *Deep-HMD* converts tabular malware and benign hardware events data (that are monitored from the underlying processor) to image formatted data using an effective 2D embedding image features conversion method. A deep neural network can train on both tabular and image data. However, DNN-based architectures on images have achieved far-reaching performance with high top 1/top 5 accuracy and low top 5 error rates. Next, *Deep-HMD* leverages a high performance ResNet architecture to recognize the zero-day malicious images at run-time. The work in [24] presented a method that projects features in tabular format into two-dimensional images before feeding images to fine-tune CNN models. However, there exist no similar experiments on the application of such methods in detecting the signatures of unknown malware and in particular with the emphasis on developing effective security countermeasures at the processors' hardware level. Our *Deep-HMD* method explores this space by employing a novel deep neural network and transfer learning training strategy on image-based hardware events data to achieve state-of-the-art performance for zero-day malware detection using a limited number of HPC events.

Algorithm 1 Converting Tabular Data to 2D Images

Input: HPC features in tabular format X = {x1,x2,x3,x4}
Output: image data equivalent to X
repeat
  forall *for each row of X={x1, x2, x3, x4}* **do**
    normalize HPCs to range [-1,+1]
    set font size as 50, resolution as 256 x 256 x3
    set 2 columns 2 rows per image
    apply OpenCV's cv2.putText() to draw HPC number on image
    save converted image to folder
  end
until *all rows of fitted tabular data are converted to images*

Stage 1 in *Deep-HMD*: In its first stage, our proposed HMD framework employs the encoding method described in Algorithm 1 to convert each row of 4 HPC tabular data to one image data. We firstly normalized all rows of data using the standard scaler in the Scikit Learn, which removes the mean and scales the data to unit variance. Next, we use OpenCV [2] library to evenly project the four numeric data to a 256 x 256 x 3 resolution image with equal spacing and no-overlapping.

Stage 2 in *Deep-HMD*: The second step includes using the generated image data as inputs to train an accurate and effective DNN-based model for zero-day malware detection. We implement *Deep-HMD* with a customized transfer learning training strategy based on the ResNet18 architecture [8]. Algorithm 2 describes the training process in *Deep-HMD*. In this stage, the generated two-dimensional images are resized to 224 x 224 x 3 and fed into the *Deep-HMD* network initialized with the ImageNet pre-trained weights. As described further in the next subsection, we remove the last output layer first and reshape the last layer to two output nodes. We then fine-tune the whole network over malware and benign datasets, and perform known and zero-day test.

Algorithm 2 Training Process in *Deep-HMD*

Input: HPC features in tabular format X, and target label Y
Output: *Deep-HMD* DNN Model for Binary Classification

Feature Selection:
  forall *tabular HPC-based dataset* **do**
    calculate MI by applying scikitlearn mutual_info_classif on all HPC features
    select top 4 features and fit to whole dataset
    train/known test/zero-day test split on fitted dataset
end

Image Conversion: - see Algorithm 1

while *training* **do**
  load batch data, resize images to 224 x 224 x3 and initialize ImageNet parameters
  while *validation loss >training loss* **do**
    for every 7 to 20 epochs:
      apply cyclical learning rate (clr) to find optimized learning rate (lr)
      apply found lr to the next training steps (7-20 epochs)
  end
  save model
end

2.4 Training and Testing in *Deep-HMD*

Figure 3 demonstrates the overall training and testing process of *Deep-HMD* in four steps. Steps 1 and 2 are dedicated to the training process, and steps 3 and 4 are for the testing phase.

In **Step** ❶, we first examine how the baseline model of the *Deep-HMD* performs on our known and zero-day test datasets. To this aim, we load the ImageNet parameter of *Deep-HMD* in fast.ai, run two tests, and calculate various metrics. Fast.ai [11] is a deep learning library with PyTorch as its underlined backbone that can quickly train high-performance deep learning models and supports application domains in computer vision, natural language processing, and tabular models. The baseline of *Deep-HMD* tests how much knowledge it learns from ImageNet on generic features such as lines and strokes. We found it can detect 92% of benign applications but only 6% of actual malware. This experiment indicates that the pre-trained model trained on a sizeable ImageNet has obtained the valuable knowledge of finding a generic spatial relationship in images. With fine-tuning model parameters on malware dataset, it continues to learn on domain knowledge, particularly the pattern of malware from benign. In particular, we examine the possibility of transferring the knowledge of pattern recognition to a new domain with a limited number of training samples and fast training time.

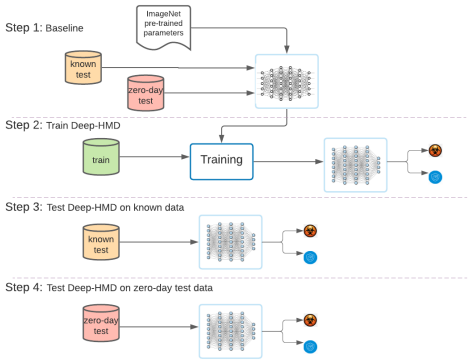

Figure 3: Overview of training and testing steps in *Deep-HMD*.

During **Step** ❷, we load the pre-trained baseline model with the ImageNet parameter, remove the last layer of the network, replace it with a Softmax function that outputs binary classification, and train the architecture. Firstly, we train it for five steps with a batch size of 64 to find an optimized learning rate for the next training cycle. In this step, generic features learned from ImageNet are transferred to the *Deep-HMD* network. In the customized training, we apply cyclical learning rates, oversampling, and weight decay techniques. We monitor training loss and validation loss decreasing until the validation loss is close to the training loss. We save checkpoints periodically and use the best model for the testing phase.

2.4.1 Cyclical Learning Rates. The learning rate strategy could have a significant impact on the effectiveness of model training. We use the fast.ai wrapper library [11] with the plain PyTorch backbone and applied the cyclical learning rate technique to find the most optimal learning rate for every several training steps. We set each training cycle with seven to twenty epochs depending on how soon we observe it stops learning by monitoring the validation loss and validation accuracy as depicted in Figure 4. We apply fast.ai's learning rate finder, set the start learning rate of 0.0001 to an end learning rate for 100 epochs, and stop in the case of divergence. Then, we apply the best learning rate range in the next training cycle. The validation accuracy starts to grow and stabilize from epoch 4, as shown in Figure 4. As a result, the best performing model is selected for the testing phase.

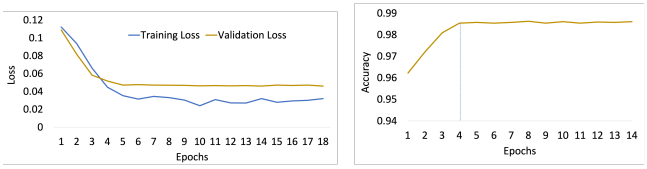

Figure 4: Applied cyclical learning rate (CLR) during training. The left figure shows the training loss and validation loss, and the right figure depicts the validation accuracy during training.

2.4.2 Over Sampling for Imbalanced Datasets. To overcome the challenges associated with the imbalanced dataset samples and remove the potential bias towards the majority class (benign in our case), we apply the oversampling technique during the training process. Oversampling technique involves duplicating examples from the minority class and adding them to the whole training dataset to create a more balanced training dataset. During each epoch, samples from the minority class are selected randomly with replacement. Once the training epoch completes, they are released back to the original dataset to be chosen again in the following training epochs. Doing so over the minority class dataset creates a more balanced training dataset for all classes and helps with model convergence in our *Deep-HMD* network.

In **Steps** ❸ and ❹, we convert the same sets of HPC data stored in tabular format to images and store the images into sub-folders of malware or benign. In fast.ai, we load the pre-trained *Deep-HMD* model, resize the input image to 224 x 224 x 3, and then run in the batch size of 64 images to process the prediction. Lastly, we accumulate the predictions, calculate the zero-day test metrics, and report the zero-day test results in the experimental results section.

3 EXPERIMENTAL RESULTS

In this section, we evaluate the effectiveness of the proposed malware detection approach. As described in Section 2, various ML models are implemented and tested using both known and unknown zero-day datasets to explore the feasibility of the standard learning classifiers in detecting malware based on hardware events. Thus, to further highlight the challenge of unknown malware detection, we have evaluated the standard ML classifiers that are widely used in state-of-the-art HMD methods considering both known and unknown conditions.

The F-measure (F1-score) results for known and zero-day malware detection (with 4 HPC events) are shown in Figure 5. As observed, the performance of standard ML models on zero-day attack detection substantially drops by more than 40% in GNB, Logistic Regression, Ridge, and SVM classifiers. When examined by the unknown (zero-day) test data, the trained machine learning classifiers have never seen the testing malware types. Even for the most robust ML model, Random Forest, the F-measure for zero-day malware drops by 14% as compared with the scenario of detecting known malware. The results confirm the limitation of standard ML algorithms in recognizing the signatures of unknown malware using HPC events and further highlights the importance of proposing an effective mechanism to enhance the detection rate of the hardware-based zero-day malware detection process.

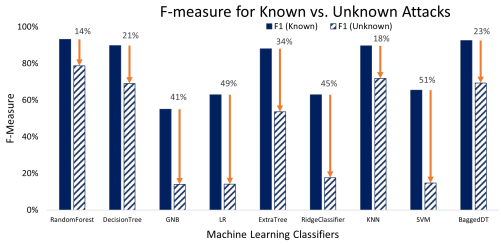

Figure 5: Evaluation of standard ML classifiers for known and unknown (zero-day) malware detection.

Table 1 reports the performance results of *Deep-HMD* versus different ML-based detectors for zero-day malware detection using 4 HPC events. We choose the ML-based detectors that are widely been adopted in existing HMD techniques. We trained the most robust machine learning classifiers over the tabular data, and applied the *Deep-HMD* network on the two-dimensional embedded image data generated from the same tabular dataset that has been fed to the standard ML classifiers. As the results indicate, our proposed *Deep-HMD* achieves 97% in F-measure, 97% in Area Under the

Table 1: Performance results of *Deep-HMD* and various ML-based detectors for zero-day malware detection

Model	Accuracy	F1	AUC	TPR	FPR	TNR	Precision	Recall
Deep-HMD	0.98	0.97	0.97	0.99	0.01	0.97	0.96	0.99
RandomForest	0.87	0.79	0.87	0.68	0.32	0.98	0.94	0.68
DecisionTree	0.81	0.69	0.79	0.58	0.42	0.94	0.86	0.58
GNB	0.62	0.14	0.40	0.09	0.91	0.92	0.38	0.09
Logistic Regression	0.62	0.14	0.40	0.09	0.91	0.92	0.39	0.09
ExtraTreeClassifier	0.74	0.54	0.70	0.41	0.59	0.93	0.78	0.41
RidgeClassifier	0.63	0.18	0.46	0.11	0.89	0.93	0.48	0.11
KNN	0.83	0.72	0.81	0.61	0.39	0.95	0.87	0.61
SVM	0.64	0.15	0.49	0.09	0.91	0.96	0.56	0.09
BaggedDT	0.82	0.69	0.82	0.56	0.44	0.97	0.92	0.56

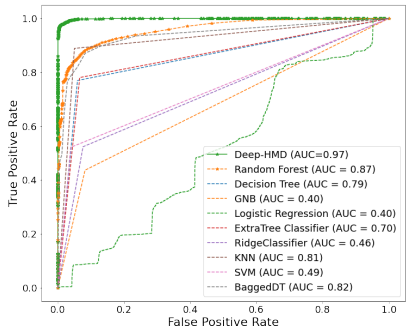

Figure 6: ROC curves of *Deep-HMD* as compared with standard ML models for zero-day malware detection using 4 hardware events.

Curve (AUC), and 98% accuracy for unknown malware detection. In addition, it obtains 96% in precision and 99% in recall. Whereas, the best performing standard ML classifier, Random Forest, can only achieve 79% in F-measure, 87% in AUC and accuracy, and 68% in recall when used for detecting unknown malware.

Overall, the results show that our proposed zero-day hardware-based malware detection method, *Deep-HMD*, is the most accurate model among all tested classifiers. While achieving an F-measure of 97% on the unknown zero-day test, *Deep-HMD* also offers 99% true positive rate and only 1% false positive rate, which is significantly outperforming the best standard ML (RF classifier) results with 68% true positive rate and 32% in false positive rate.

Furthermore, Figure 6 illustrates the ROC curves of zero-day malware detectors with *Deep-HMD* represented by the solid green line against all other classifiers. As observed, *Deep-HMD* achieves a higher detection rate than other classifiers on unknown zero-day malware detection, thanks to its ability to lower the false positive rate significantly. The dotted orange line for Random Forest as the best classical ML algorithm show an AUC of 0.87. Our proposed intelligent method improves the ROC curve for the zero-day test from 0.87 in the Random Forest classifier to 0.97, with a 11% enhancement, highlighting the effectiveness of *Deep-HMD* in improving the robustness of the zero-day malware detection process.

4 ACKNOWLEDGMENT

This research was supported by the 2021-2022 COE RSCA, and ORSP Multidisciplinary Research Awards from CSULB.

5 CONCLUSION

In this work, we examined the applicability of various standard machine learning classifiers for hardware-based zero-day malware detection and demonstrate that such methods are not capable of detecting the unknown malware patterns with a high detection performance and low false positive rate. This is because the zero-day malware HPC data does not match any seen attack applications' signatures in the existing database, which makes it a more challenging problem to address. In response, we proposed *Deep-HMD*, a two-stage DNN-based approach equipped with a flexible transfer training strategy, to accurately detect zero-day malware using a small number of hardware events. The experimental results demonstrated the superior performance of our novel intelligent solution as compared to state-of-the-art ML-based detection methods. The *Deep-HMD* method is capable of recognizing unknown malware signatures with 97% in both F-measure and AUC metrics using a limited number of HPC events and with only 1% false positive rate.

REFERENCES

[1] L. Bilge and T. Dumitraş. 2012. Before We Knew It: An Empirical Study of Zero-Day Attacks in the Real World. In *CCS'12 (CCS '12)*. ACM, 833–844.
[2] G. Bradski. 2000. The OpenCV Library. *Dr. Dobb's Journal of Software Tools* (2000).
[3] S. Das et al. 2019. SoK: The Challenges, Pitfalls, and Perils of Using Hardware Performance Counters for Security. In *IEEE SP*. 20–38.
[4] J. Demme et al. 2013. On the Feasibility of Online Malware Detection with Performance Counters. In *ISCA'13*. ACM, 559–570.
[5] A. AE Elhadi et al. 2012. Malware detection based on hybrid signature behaviour application programming interface call graph. *American Journal of Applied Sciences* 9, 3 (2012), 283.
[6] Y. Gao et al. 2021. Adaptive-HMD: Accurate and Cost-Efficient Machine Learning-Driven Malware Detection using Microarchitectural Events. In *IOLTS'21*. IEEE, 1–7.
[7] M. R. Guthaus et al. 2001. MiBench: A free, commercially representative embedded benchmark suite. In *IISWC'01*. 3–14.
[8] K. He et al. 2016. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 770–778.
[9] Z. He et al. 2021. When machine learning meets hardware cybersecurity: Delving into accurate zero-day malware detection. In *ISQED'21*. IEEE, 85–90.
[10] J. L. Henning. 2006. SPEC CPU2006 Benchmark Descriptions. *SIGARCH Comput. Archit. News* 34, 4 (Sept. 2006), 1–17.
[11] J. Howard et al. 2021. fastai. https://github.com/fastai/fastai.
[12] Simon Kornblith et al. 2019. Do Better ImageNet Models Transfer Better? arXiv:1805.08974 [cs.CV]
[13] A. Kraskov et al. 2004. Estimating mutual information. *Phys. Rev. E* 69 (Jun 2004), 066138. Issue 6. https://doi.org/10.1103/PhysRevE.69.066138
[14] Prashanth Krishnamurthy et al. 2019. Anomaly detection in real-time multi-threaded processes using hardware performance counters. *IEEE Transactions on Information Forensics and Security* 15 (2019), 666–680.
[15] H. Liu et al. 2012. *Feature selection for knowledge discovery and data mining*. Vol. 454. Springer Science & Business Media.
[16] H. M. Makrani et al. 2021. Adaptive Performance Modeling of Data-Intensive Workloads for Resource Provisioning in Virtualized Environment. *ACM ToMPECS* 5, 4, Article 18 (mar 2021), 24 pages.
[17] M. Ozsoy et al. 2015. Malware-aware processors: A framework for efficient online malware detection. In *HPCA'15*. 651–661.
[18] F. Pedregosa et al. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.
[19] H. Sayadi et al. 2018. Ensemble learning for effective run-time hardware-based malware detection: A comprehensive analysis and classification. In *DAC'18*. 1–6.
[20] H. Sayadi et al. 2019. 2SMaRT: A Two-Stage Machine Learning-Based Approach for Run-Time Specialized Hardware-Assisted Malware Detection. In *DATE'19*. 728–733.
[21] H. Sayadi et al. 2020. Recent Advancements in Microarchitectural Security: Review of Machine Learning Countermeasures. In *MWSCAS'20*. 949–952.
[22] H. Sayadi et al. 2020. StealthMiner: Specialized Time Series Machine Learning for Run-Time Stealthy Malware Detection Based on Microarchitectural Features. In *GLSVLSI'20*. 175–180.
[23] B. Singh et al. 2017. On the Detection of Kernel-Level Rootkits Using Hardware Performance Counters. In *ASIACCS'17*. 483–493.
[24] Baohua Sun et al. 2019. SuperTML: Two-Dimensional Word Embedding for the Precognition on Structured Tabular Data. arXiv:1903.06246 [cs.CV]
[25] Chuanqi Tan et al. 2018. A Survey on Deep Transfer Learning. In *ICANN 2018*, Věra Kůrková et al. (Eds.). Springer, Cham, 270–279.
[26] A. Tang et al. 2014. Unsupervised Anomaly-Based Malware Detection Using Hardware Features. In *RAID'14*. Springer, 109–129.
[27] H. Wang et al. 2020. Mitigating cache-based side-channel attacks through randomization: A comprehensive system and architecture level analysis. In *DATE'20*. IEEE, 1414–1419.
[28] B. Zhou et al. 2018. Hardware Performance Counters Can Detect Malware: Myth or Fact?. In *ASIACCS'18*. 457–468.