2022 11th International Conference on Renewable Energy Research and Application (ICRERA) | 978-1-6654-7140-4/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICRERA55966.2022.9922745

Machine Learning based Malware Detection for Secure Smart Grids

Kevin Immanuel Gubbi
University of California, Davis
Davis, CA, USA
kgubbi@ucdavis.edu

Han Wang
Temple University, Philadelphia
Philadelphia, PA, USA
han.wang0004@temple.edu

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

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

***Abstract*—The development of the current electrical networks into smart grids, which promises higher dependability, lower production costs, and more environmentally friendly energy generation, signals the beginning of a new era for the electricity business. Numerous projects throughout the world, sponsored by both business and academics, are a reflection of the growing concern around both the immense advantages and the significant threats brought about by this progress. The security of the smart grid and the smart home network and the detection of malicious malware, which we present as a crucial component of the smart grid, is the main topic of this manuscript. We seek to show that Hardware Performance Counters (HPCs) can be used to efficiently detect malware based attacks on the smart grid. Machine Learning methods are used in our a state-of-the-art malware detection method. The manuscript provides a brief overview and in-depth analysis of HPCs for malware detection using ensemble learning. Finally, detection results are discussed in detail with relevant accuracy graphs and metrics.**

***Index Terms*—Hardware Security, Malware Detection, Smart grids, machine learning**

I. Introduction and Background

A smart grid is a digitally-based power network that uses two-way digital communication to deliver electricity to users. To increase productivity, cut costs and consumption of energy, and increase the transparency and dependability of the energy supply chain, this system enables monitoring, analysis, control, and communication within the supply chain. By utilising smart net metres, the smart grid was created with the intention of eliminating the drawbacks of traditional electricity networks.

With the introduction of the Smart Grid, the energy sector will enter a new age of dependability, availability, and efficiency that will improve both our economic situation and the environment. To make sure that the advantages we anticipate from the Smart Grid become a reality throughout the transition phase, it will be crucial to conduct out testing, technical advancements, consumer education, establishment of standards and laws, and information exchange amongst projects. Although, smart grids and smart homes provide great opportunities and convenience to the citizens of a nation, without proper security measures, they are vulnerable to all sorts of attack vectors and provide an opportunity for malicious entities to access the grid or home network [1], [2].

According to [3], the smart grid system must contain three primary security goals: **1) availability of uninterrupted power supply in accordance with user needs; 2) integrity of communicated information;** and **3) confidentiality of user data**.

Existing research in security for smart grids have proposed anomaly detection and resiliency-oriented countermeasures to protect smart grid against cyber-security attacks [4], [5]. [6] and [7] show innovative ways to integrate security into smart-grids. With the advancements in machine learning (ML), it is important to develop ML based or assisted smart grid security solutions to safeguard the grid from malicious attacks. Recent work in malware detection using ML methods, [8]–[10], prove that machine learning is efficient and great at sniffing out malware or other malicious firmware/software running on hardware using Hardware Performance Counters (HPCs). These HPCs are embedded inside the hardware used to develop these smart grids. Some commercial processors have HPCs by default and can be used for the purpose of performance measurement or, in our case, real-time detection of malware on a smart grid system. Given the importance of the smart grid to a nation's security, it is vital to integrate security measures and malware detection methods into smart grids and smart home networks [11].

In order to develop learning approaches that provide more accurate predictions and higher generalization performance, ensemble learning and joint decision procedures are frequently used. In this research, we implement and evaluate the performance of two ensemble learning techniques for successful malware detection even with fewer HPCs. We employ bagging and boosting techniques in this operation. One of the most popular ensemble learning techniques for improving the performance of ML algorithms is boosting. An ensemble learning

methodology called Bagging, or Bootstrap Aggregation, is utilized for classification and regression issues [12].

In this paper, We investigate the efficacy of ensemble learning models in **1) lowering the quantity of performance counters needed to implement efficient ML classifiers for run-time malware detection**, and **2) enhancing the performance of weak but affordable classifiers in malware detection with a small quantity of HPCs.** Since we suggest using HPCs in smart grid hardware, all implementations and results must be scrutinized from the standpoint of smart grid security.

The remainder of this manuscript is organized as follows. The suggested hardware-based malware detection methodology is presented in Section II and the specifics of the experimental setup are presented in Section III. Finally, in the conclusion, we will discuss challenges and potential avenues for research in this direction.

II. Framework for Malware Detection in Smart Grids

We outline the cutting-edge run-time hardware-based malware detection method in this part.

A. Experimental Setup - Data Collection

The specifics of the data and experimental setup gathering process are provided in this section. On an Intel Xeon-X5550 computer running Ubuntu 14.04 with the Linux 4.4 Kernel, all apps are executed, and we gather different HPCs statistics. This CPU has four performance counter registers and is based on Intel's Nehalem architecture. We employ the Linux-compatible Perf programme to retrieve the HPC data. Rich generic abstractions over hardware-specific features are provided by Perf. It makes use of the background function call perf_event_open, which may measure several events at once. For HPC data collecting, we have run over 100 programmes, both good and bad. The MiBench benchmark suite [13], Linux system programmes, browsers, text editors, and word processors are examples of benign software. In terms of malware programmes, virustotal.com is used to gather malware [14]. Python, Perl, Python, and Bash scripts are examples of apps that can be made malicious to carry out destructive actions. Malware applications also include Linux ELFs. We utilise the WEKA tool [15] to assess the efficacy and effectiveness of several machine learning classifiers after collecting microarchitectural events using Perf.

Figure 1 shows an overview of the suggested hardware-based malware detection method and how to build machine learning classifiers to anticipate harmful application behavior. In general, it consists of a number of processes, including feature extraction, feature reduction, and the creation of ML classifiers (both general and ensemble) for malware detection. The relevant sections will go into further information about these stages. By running all apps in the isolated environment known as Linux Containers (LXC) [16], HPC data is gathered. Operating system level virtualization is accomplished via LXC, which utilizes the host operating system's kernel. Due to the fact that LXC provides access to real performance counters data rather than only mimicking HPCs, it is preferred in this work over other popular virtual platforms like VMWare or VirtualBox. 44 CPU events were collected using the Perf tool. Four events can only be measured at once on Intel Xeon, since there are only 4 counter registers accessible [14]. Multiple runs are therefore necessary to completely record all occurrences. To collect all microarchitectural events, we split 44 events into 11 batches of 4 events and ran each programme 11 times at a sampling rate of 10 ms. Malware installation inside the container might pollute the environment, which can interfere with data collecting in the future. The container is destroyed after each run to make sure that the preceding run did not contaminate the data that was obtained.

B. Feature Selection

As was previously noted, describing programmes at the micro-architectural level is necessary for malware detection using machine learning models. The dataset produced by this procedure has a very high dimension. Running ML algorithms on massive HPCs would be difficult and time-consuming. Additionally, the classifier [9] would have lesser accuracy due to the inclusion of extraneous characteristics. Therefore, a subset of HPCs is chosen that contains the most crucial characteristics for classification rather than taking into consideration all of the collected features. Irrelevant data is then detected and deleted using a feature reduction technique. Each learning algorithm receives the features, and the learning algorithm looks for a relationship between the feature values and the behaviour of the application to identify if the programme is malicious or benign. Finding the proper characteristics to describe the incoming data is crucial for creating an accurate detector, as was previously described. From a base of 44 performance counters, we began. Following feature extraction, the feature reduction method minimises the quantity of low-level features, as seen in Figure 2.We start by monitoring the most important microarchitectural metrics in WEKA on our training set to capture application characteristics. The technique of feature scoring is then used to assign scores to the features based on their significance and relevance to the target variable. The sixteen hardware performance counters that are most closely connected are identified and ranked in order of significance for malware detection using the feature reduction approach. Table 3 has a list of these HPCs. They serve as input parameters for our prediction model. The characteristics that were chosen include HPCs that reflect pipeline front-end, pipeline back-end, cache subsystem, and main memory behaviors that influence the performance of typical applications.

C. Training & Testing the Malware Detectors

The specifics of developing and putting to the test ML classifiers for malware detection are covered in this section. For training, the incoming application is profiled using the Linux Perf tool, and low-level feature values are collected for each training program. The retrieved features are then condensed to the most important performance counters, and a

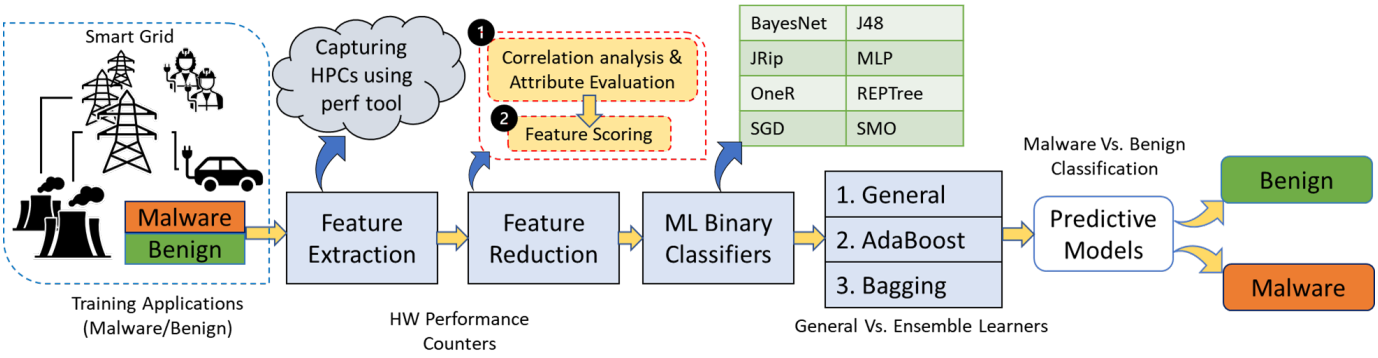

Fig. 1. Hardware-based malware detection approach for smart grid systems

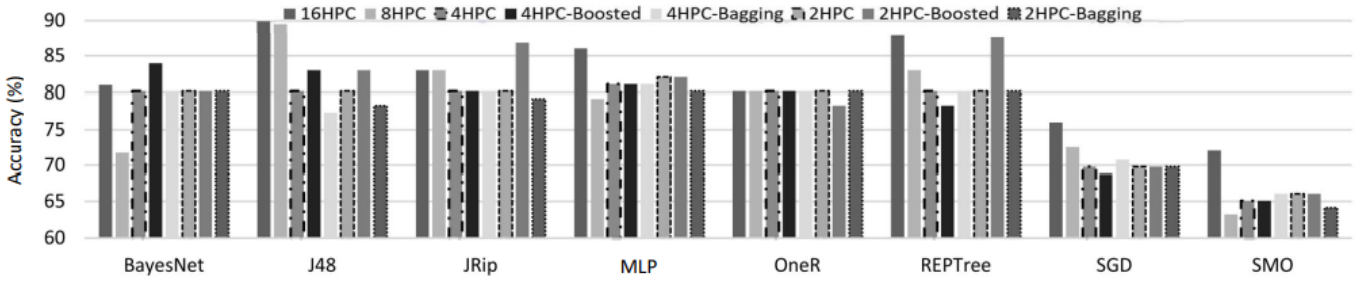

Fig. 2. Results of accuracy tests for several ML classifiers with different numbers of HPCs

Hardware Performance Counters		
1 - branch_instructions	2 - branch_loads	3 - iTLB_load_misses
4 - dTLB_load_misses	5 - dTLB_store_misses	6 - L1_dcache_stores
7 - cache_misses	8 - node_loads	9 - dTLB_stores
10 - iTLB_loads	11 - L1_icache_load_misses	12 - branch_load_misses
13 - branch_misses	14 - LLC_store_misses	15 - node_stores
16 - L1_dcache_load_misses		

Fig. 3. Hardware performance counters features (HPCs) used in order of importance

learning model is created using the training data. It is vital to note that the HPCs retrieved from the running apps at intervals of 10ms serve as the input variables for our classifiers, and the application's type is the output variable (malware vs. benign). We build general and ensemble models (AdaBoost and Bagging) for each ML classifier to identify malware. A typical dataset split of 70% to 30% is used for training and testing in order to validate each of the used ML classifiers. In order to prevent biased splitting, training applications consist of 70% benign and 30% malware, while testing applications consist of 70% benign and 30% malware.

III. Experimental Results

In this section, we present the evaluation results for different machine learning classifiers. We thoroughly compare these learning techniques in terms of the prediction accuracy, robustness, performance, and the hardware implementation costs.

A. Detection Accuracy

We calculate the proportion of samples that are successfully categorised in order to assess the malware classifiers' detection accuracy. A thorough accuracy comparison of several ML classifiers (general and ensemble) employed for malware detection is shown in Figure 2. We constructed two ensemble learning algorithms, eight generic ML classifiers, and measured their accuracy in identifying malicious and benign apps. The claimed malware detection accuracy is based on 16, 8, 4, and 2 hardware performance counters. The majority of ML classifiers perform well before feature reduction (16 HPCs), typically offering accuracy rates of above 80%. Several classifiers' accuracy is considerably impacted by feature reduction. But even after feature reduction, OneR classifiers function effectively. Because OneR classifier only uses one performance counter (branch_instructions) to forecast malware activity, it is not impacted by feature reduction and displays accuracy results that are almost constant. Figure 2 shows that for some classifiers, such as BayesNet, JRip, OneR, REPTree, and SMO, a greater or comparable accuracy level to 8/16 HPC models is attained by limiting the number of hardware performance counters to 2 or 4. This intriguing discovery supports the efficacy of employing ensemble learning to increase classifier accuracy. As an illustration, REPTree uses 16 HPCs to produce accuracy that is about 88%. However, we find that using the AdaBoost ensemble approach and decreasing the number of essential performance counters to two yields results that are virtually as accurate (88%) as when using 16 HPCs.

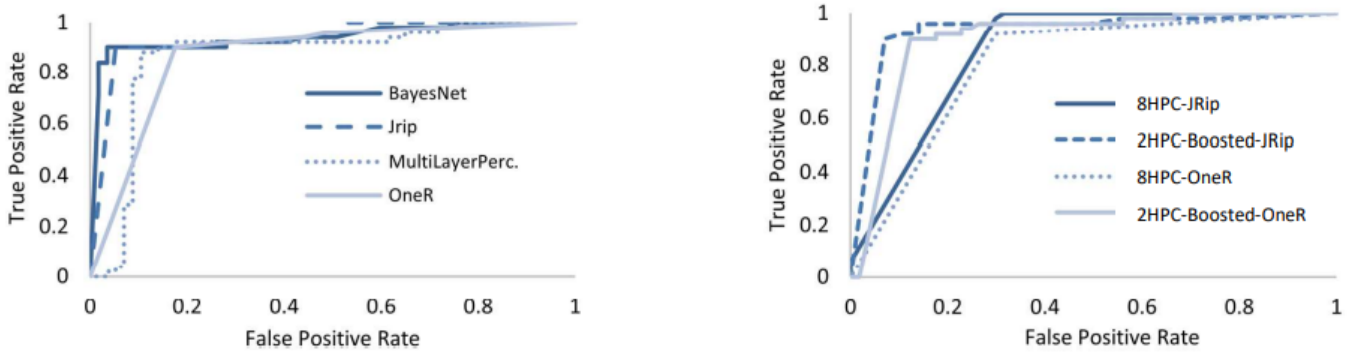

Fig. 4. The ROC graphs for various ensemble methods with varying HPC count - 4HPC(Bagging), 8HPC vs. 2HPC(Boosted).

Classifier	16HPC	8HPC	4HPC	4HPC-Boosted	4HPC-Bagging	2HPC	2HPC-Boosted	2HPC-Bagging
BayesNet	0.92	0.92	0.92	0.92	0.94	0.92	0.87	0.93
J48	0.88	0.88	0.81	0.94	0.85	0.81	0.92	0.82
Jrip	0.86	0.86	0.81	0.88	0.93	0.81	0.93	0.88
MLP	0.9	0.9	0.89	0.92	0.86	0.9	0.93	0.87
OneR	0.81	0.81	0.81	0.9	0.87	0.81	0.9	0.87
REPTree	0.85	0.85	0.81	0.85	0.88	0.81	0.92	0.91
SGD	0.74	0.74	0.72	0.89	0.74	0.71	0.71	0.71
SMO	0.65	0.65	0.65	0.88	0.85	0.68	0.89	0.83

Fig. 5. Area Under the Curve (AUC) values for different general and ensemble based malware detectors

B. Classification Robustness

Receiver Operating Characteristics (ROC) graphs are employed to assess the precision and dependability of machine learning classifiers in identifying malware. Plotting the ratio of true positives to false positives for a binary classifier as the threshold increases results in the ROC curve. Thus, the ideal classifier would produce a point at position (0,1) in the upper left corner of the ROC space, signifying 100% true positives and 0% false positives.

When evaluating each ML classifier, we look at its robustness using the Area Under the Curve (AUC) ROC curve metric. Area under the curve analysis offers useful information for choosing potentially ideal ML classifiers appropriate for malware detection and for eliminating inferior detectors.

The ROC curves for two distinct ensemble learning models with varying numbers of performance counters are shown in Figure 4. Due to a lack of space, we must give the ROC graphs for a few chosen ML classifiers here. We also demonstrate how ensemble learning approaches affect AUC robustness. Figure 3 displays the ROC graphs for four machine learning classifiers that were enhanced by bagging ensemble learning and were created using four performance counters. The BayesNet and JRip classifiers, which give the best robustness with only 4 performance counters, have maximum AUC values of 0.937 and 0.932, respectively, as shown in this figure in Table 5. Figure 3 shows the efficiency of the AdaBoost algorithm on two distinct detectors when the number of HPCs is decreased from 8 to 2. As seen, each classifier's boosting model greatly raises the AUC of the ROC curve, which increases the effectiveness of the ML classifier in terms of classification robustness.

To effectively detect malicious malware in Smart Grid systems, in order to evaluate and compare the performance of malware detectors, we utilize the product of accuracy and area under the ROC graph (ACC*AUC) as a performance measure. This metric simultaneously accounts for both measurements and combines the effects of accuracy and resilience in the categorization of malware. Since performance is a more thorough statistic that takes into consideration the effects of the detection accuracy and AUC values, we used it as the final comparison metric for different ML classifiers. The key point is that extracting 16 or 8 hardware performance counters, which undoubtedly impose significant implementation cost overhead to the smart grid system in terms of resource utilisation and power consumption, is more efficient than gathering fewer HPCs (four or two) in the smart grid system and improving the performance of the ML classifier with one of the ensemble learning approaches.

IV. Conclusion

This manuscript discusses the security of smart grids and employs an ensemble learning-based machine learning malware detection strategy to search for possible harmful software injected into smart grid systems. Hardware-based detectors utilize run-time HPC data and machine learning classifiers. A review of current efforts on ML-based malware detectors demonstrates that no one generic classifier consistently outperforms other classifiers across a range of malware kinds in terms of performance (accuracy and resilience), area overhead, and detection time. Results of the detection process and analysis are given and described. Utilizing ML-based malware detection techniques is a crucial first step in safeguarding the networks of smart grids, which are an essential component of national security.

References

[1] H. Khurana, M. Hadley, N. Lu, and D. A. Frincke, "Smart-grid security issues," *IEEE Security & Privacy*, vol. 8, no. 1, pp. 81–85, 2010.
[2] T. Flick and J. Morehouse, *Securing the smart grid: next generation power grid security*. Elsevier, 2010.
[3] F. Aloul, A. Al-Ali, R. Al-Dalky, M. Al-Mardini, and W. El-Hajj, "Smart grid security: Threats, vulnerabilities and solutions," *International Journal of Smart Grid and Clean Energy*, vol. 1, no. 1, pp. 1–6, 2012.

[4] H. Shahinzadeh, A. Mahmoudi, J. Moradi, H. Nafisi, E. Kabalci, and M. Benbouzid, "Anomaly detection and resilience-oriented countermeasures against cyberattacks in smart grids," in *2021 7th International Conference on Signal Processing and Intelligent Systems (ICSPIS)*. IEEE, 2021, pp. 1–7.

[5] H. Shahinzadeh, A. Mahmoudi, G. B. Gharehpetian, S. Muyeen, M. Benbouzid, and E. Kabalci, "An agile black-out detection and response paradigm in smart grids incorporating iot-oriented initiatives and fog-computing platform," in *2022 International Conference on Protection and Automation of Power Systems (IPAPS)*, vol. 16. IEEE, 2022, pp. 1–8.

[6] J. Moradi, H. Shahinzadeh, H. Nafisi, M. Marzband, and G. B. Gharehpetian, "Attributes of big data analytics for data-driven decision making in cyber-physical power systems," in *2020 14th international conference on protection and automation of power systems (IPAPS)*. IEEE, 2019, pp. 83–92.

[7] J. Moradi, H. Shahinzadeh, H. Nafisi, G. B. Gharehpetian, and M. Shaneh, "Blockchain, a sustainable solution for cybersecurity using cryptocurrency for financial transactions in smart grids," in *2019 24th Electrical Power Distribution Conference (EPDC)*. IEEE, 2019, pp. 47–53.

[8] H. Sayadi, N. Patel, S. M. PD, A. Sasan, S. Rafatirad, and H. Homayoun, "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)*. IEEE, 2018, pp. 1–6.

[9] H. Sayadi, A. Houmansadr, S. Rafatirad, and H. Homayoun, "Comprehensive assessment of run-time hardware-supported malware detection using general and ensemble learning," in *Proceedings of the 15th ACM International Conference on Computing Frontiers*, 2018, pp. 212–215.

[10] S. M. P. Dinakarrao, H. Sayadi, H. M. Makrani, C. Nowzari, S. Rafatirad, and H. Homayoun, "Lightweight node-level malware detection and network-level malware confinement in iot networks," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2019, pp. 776–781.

[11] N. Komninos, E. Philippou, and A. Pitsillides, "Survey in smart grid and smart home security: Issues, challenges and countermeasures," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 4, pp. 1933–1954, 2014.

[12] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

[13] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown, "Mibench: A free, commercially representative embedded benchmark suite," in *Proceedings of the fourth annual IEEE international workshop on workload characterization. WWC-4 (Cat. No. 01EX538)*. IEEE, 2001, pp. 3–14.

[14] "MS Windows NT kernel description," https://www.virustotal.com/intelligence/, accessed: 2022-08-06.

[15] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," *ACM SIGKDD explorations newsletter*, vol. 11, no. 1, pp. 10–18, 2009.

[16] "MS Windows NT kernel description," https://developer.ibm.com/tutorials/l-lxc-containers/, accessed: 2022-08-06.