On the Classification of Microsoft-Windows ransomware using hardware profile

Sana Aurangzeb [Equal first author, 1], **Muhammad Aleem** [2], **Rao Naveed Bin Rais** [Corresp., Equal first author, 3], **Muhammad Arshad Islam** [1], **Muhammad Azhar Iqbal** [4]

[1] Department of Computer Science, National University of Modern Languages, Islamabad, Islamabad, ICT, Pakistan

[2] Department of Computer Science, National University of Computer and Emerging Sciences, Islamabad, Islamabad, ICT, Pakistan

[3] College of Engineering and Information Technology, Ajman University, Ajman, Ajman, United Arab Emirates

[4] School of Information Science and Technology (SIST), Southwest Jiaotong University, Chengdu, China

Corresponding Author: Rao Naveed Bin Rais
Email address: r.rais@ajman.ac.ae

Due to the expeditious inclination of online services usage, the incidents of ransomware proliferation being reported are on the rise. Ransomware is a more hazardous threat than other malware as the victim of ransomware cannot regain access to the hijacked device until some form of compensation is paid. In the literature, several dynamic analysis techniques have been employed for detection of malware including ransomware; however, to the best of our knowledge, hardware execution profile for ransomware analysis has not been used, as of today. In this study, we present that the hardware execution profile can be exploited for the identification of ransomware applications. We show that the true execution picture obtained via a hardware execution profile, is beneficial to identify the obfuscated ransomware too. We evaluate the features obtained from hardware performance counters to classify malicious applications into ransomware and non-ransomware categories using several machine learning algorithms such as Random Forest, Decision Tree, Gradient Boosting, and Extreme Gradient Boosting. The employed data set comprises 80 ransomware and 80 non-ransomware applications, which are collected using the VirusShare platform. The results revealed that extracted hardware features play a substantial part in the identification and detection of ransomware with an accuracy of 0.97.

On the Classification of Microsoft-Windows Ransomware Using Hardware Profile

Sana Aurangzeb [1], Muhammad Aleem[2], Rao Naveed Bin Rais[3], Muhammad Arshad Islam[2], Muhammad Azhar Iqbal[4]

[1] Department of Computer Science, National University of Modern Languages, Islamabad 44000, Pakistan
[2] Department of Computer Science, National University of Computer and Emerging Sciences, Islamabad 44000, Pakistan
[3] College of Engineering and Information Technology, Ajman University, Ajman, UAE
[4] School of Information Science and Technology (SIST), Southwest Jiaotong University, Chengdu 611756, P. R. China

Corresponding Author:
Rao Naveed Bin Rais[3]
College of Engineering and Information Technology, Ajman University, UAE
Email address: r.rais@ajman.ac.ae

Abstract

Due to the expeditious inclination of online services usage, the incidents of ransomware proliferation being reported are on the rise. Ransomware is a more hazardous threat than other malware as the victim of ransomware cannot regain access to the hijacked device until some form of compensation is paid. In the literature, several dynamic analysis techniques have been employed for detection of malware including ransomware; however, to the best of our knowledge, hardware execution profile for ransomware analysis has not been used, as of today. In this study, we present that the hardware execution profile can be exploited for the identification of ransomware applications. We show that the true execution picture obtained via a hardware execution profile, is beneficial to identify the obfuscated ransomware too. We evaluate the features obtained from hardware performance counters to classify malicious applications into ransomware and non-ransomware categories using several machine learning algorithms such as Random Forest, Decision Tree, Gradient Boosting, and Extreme Gradient Boosting. The employed data set comprises 80 ransomware and 80 non-ransomware applications, which are collected using the VirusShare platform. The results revealed that extracted hardware features play a substantial part in the identification and detection of ransomware with an accuracy of 0.97.

Keywords: malware; ransomware; performance counters; classification; machine learning

Introduction

Over the past half a decade, an exponential increase has been reported in ransomware attacks. Ransomware is the sub-class of malware that hijacks a device and blocks the victim to access the data until a compensation of some form is made. Typically, this compensation is in the form of money to concede access back to the victim. Ransomware contains abilities to harmfully affect various kinds of devices such as *personal computers, servers, smartphones, tablets,* etc. For instance, multiple new variants of ransomware including WannaCry ransomware, JAFF, Petya have been reported in 2017 (Hampton, et al., 2018). On May 12, 2017, within the span of a few hours, the WannaCry ransomware (Maurya, et al., 2018) infected more than 70,000 desktop devices in over 150 countries across the globe (Grant & Parkinson, 2018) as shown in Figure 1 (Krebsonsecurity, 2017).

The economic effects of ransomware can be quite devastating. For instance, CryptoWall_v3 ransomware (Cyber Threat Alliance, 2016; Sgandurra, et al., 2016) caused the loss of an estimated $325 million in the US from November 2015 to June 2016. Another ransomware attack, triggered by CryptoWall_v4 ransomware resulted in a loss of $7.1 million worldwide (Cyber Threat Alliance, 2016). Another recently reported ransomware attack, NotPetya costs $10 billion and WannaCry estimated to have cost $8 billion (Davies, et al. 2020). These attacks wreaked havoc in systems of various world organizations by halting and damaging their daily operations.

Typically, a ransomware displays a ransom note to the victim after encrypting her data. The majority of the ransomware also specify the instructions regarding compensation payment to regain access to the device. A ransomware employs different hijacking strategies such as behaving like an adware resulting in unwanted advertisements or by being hidden using rootkits to bypass Anti-Viruses (AV) (Demme, et al. 2013). A rootkit is a malware which alters the operating system (OS) and resides in the system for a prolonged period (Aurangzeb, et al., 2017).

Although, malware is deemed as a great threat over the years, yet ransomware is even more daunting threat compared to other malware due to its attacking and demanding nature (i.e., expecting a ransom in return). Before analyzing the ransomware, one of the mandatory steps is the accurate identification of a particular type of ransomware and differentiating it from other typical malware. Broadly, malware analysis techniques are categorized as: 1) *static* and 2) *dynamic* analysis (Chen, et al., 2017). Besides, various researchers have employed the combinations of the static and dynamic techniques in the form of hybrid analysis techniques. The procedure of scrutinizing a potential malware without executing the program is referred to as *static analysis*, whereas, the analysis performed via observing the execution behavior of a

malware is known as *dynamic analysis*. Most contemporary state-of-the-art *dynamic analysis* techniques detect and classify ransomware that hide themselves using various obfuscation techniques such as packed programs, compressed, or data transformation, indirect addressing, etc. (Behera & Bhaskari, 2015). Today, various anti-viruses tackle malware to dampen their caused and expected damages. However, the techniques employed by the anti-viruses are often limited to the prior knowledge (e.g., signatures, etc.) and lack a comprehensive *dynamic analysis* that could detect ransomware, employing the obfuscation techniques (Demme, et al., 2013).

On the other hand, Hardware Performance Counters (HPCs) have been typically used by the programmers to analyze and measure the performance of applications and to identify the execution bottlenecks of a program with the purpose of improving it on a target platform (Beneventi, et al. 2017). Initially, HPCs were employed for investigating the *static* and *dynamic analysis* of programs in order to detect any malicious amendments as mentioned in (Alam, et al.,2020) and (Malone, et al., 2011). In the study (Zhou, et al., 2018), the authors surveyed to identify whether HPCs are useful in differentiating the malware from benign applications. However, the study did not consider malware as ransomware; rather it considers revoking access to network activities. In this paper, we present a framework based on *dynamic analysis* that mainly focuses on the classification of ransomware from non-ransomware. Moreover, the classification of ransomware from *traditional malware* is essential because of their higher damaging impact in terms of informative data and financial loss. Compare to typical malware, it is more challenging to identify and kill ransomware even when it is discovered, and the damage can be potentially irreparable even after its deletion (Al-remy, et al., 2018) and (Zhang, et al., 2019). Hence, we require proactive and aggressive techniques to handle ransomware. Moreover, it is very challenging to recognize and isolate the malware from ransomware due to the similarity in nature. A ransomware is more menacing than malware, as it not only damages the system and results in loss of control from the system, but also demands a compensation in return. Therefore, there is a need to have proper distinction of ransomware from other malware (Aurangzeb et al, 2017; Kok et al., 2019 and Zhang, et al., 2019) in order to save billions of illegal transactions (Davies, et al 2020) in the name of ransom.

Several studies (Das et al., 2019; Demme, et al., 2013; Singh, et al., 2017; and Wang, et al., 2016) discuss potential implications of using Hardware Performance Counters (HPC) for application analysis, and the majority of them suggest that hardware execution profile can effectuate the detection of malware (Demme, et al., 2013; Singh, et al., 2017; and Wang, et al., 2016). Another study (Xu, et al., 2017) has utilized the hardware execution profiles to detect malware using machine learning algorithms, as malware changes data structures and control flow, leaving fingerprints on accesses to program memory. In this respect, they proposed a framework for detecting malware from benign applications that use machine learning to classify malicious behavior of malware based on access patterns of virtual memory. Hence, it is still an open research question whether to utilize HPC or not for detection of malware. However,

utilizing the hardware performance measurements and the profile of the low-level execution behavior has not been previously studied for the analysis and detection of ransomware applications. We argue that ransomware reveals itself by exhibiting peculiar HPCs (e.g., through clock cycles, cache misses and hits, branch instructions and misses, retired instructions, etc.). This paper contemplates HPCs to detect Microsoft Windows-based ransomware by analyzing the execution behavior of ransomware. We primarily focus to determine the potential use of HPCs in analyzing and proactively detecting ransomware. Moreover, the classification of ransomware from malware is imperative because the damages caused by ransomware drastically ensure the data and monetary loss. To address this concern, we propose a mechanism which utilizes the application execution profile for the classification and detection of ransomware from non-ransomware. For classification, the application's hardware related performance features are extracted from the data set of 160 malware (consisting of 80 ransomware and 80 non-ransomware). Afterward, these features are fed to some well-known machine learning classification models such as Decision Tree (Kohavi, 1996), Random Forest (Liaw, et al., 2002), Gradient Boosting (Friedman, 1999), and Extreme Gradient Boosting (Chen, et al., 2015). These four classifiers are generally used for classification tasks of various applications including *spam detection*, *face recognition,* and *financial predictions* (Jordan and Mitchell, 2015), etc. We employ these four classifiers as part of the proposed methodology to analyze their performance for ransomware detection. These models perform binary classification of malicious software into ransomware or non-ransomware classes. In summary, the main contributions of this paper are as follows:

- In-depth analysis of the current state-of-the-art to identify the merits and demerits of several existing approaches;
- A novel mechanism for the classification and detection of malicious applications into ransomware and non-ransomware; and
- Empirical investigation of the HPCs against state-of-the-art dynamic techniques using machine learning classifiers;

The outcomes revealed that the random forest classifier has outperformed decision tree, gradient boosting, and extreme gradient boosting by attaining accuracy of 0.97 for classification.

The rest of the paper is organized as follows. Section 2 describes the related work. Section 3 presents the proposed methodology, dataset, and feature extraction mechanism. In Section 4, the experimental setup details, results, and related discussions are presented and Section 5 concludes the paper.

Related Work

For *dynamic analysis*, it is necessary to collect key ransomware features at runtime. Most of the *dynamic analysis*-based research studies exploit the renowned malware databases[1] for the acquisition of malicious software and use quarantine environments (such as Cuckoo's sandbox (Kaur, Dhif and Singh 2017)) to execute the applications.

In (Zavarsky and Lindskog 2016), the authors presented an experimental analysis of Microsoft Windows and Android-based ransomware. This analysis demonstrates that ransomware detection could be performed by monitoring the abnormalities in the file system and registry activities. It was shown that a significant number of ransomware families exhibit very similar characteristics. Moreover, the authors concluded that changes in a particular set of registry keys are important aspects to be analyzed for ransomware detection. The authors discovered that Microsoft Windows 10 is reasonably effective against ransomware attacks. Moreover, this study also revealed that for the Android platform, the *Android Manifest* file and the permissions (required by an app) should also be considered for ransomware detection.

Several researchers utilized the hash information (i.e., comparing hash values) to detect the CryptoLocker ransomware (Song, Kim, and Lee 2016). The affected systems are recovered by the following ways: 1) process CryptoLocker, 2) comparing hash information with the encrypted data files 3) validating the key using the key-index information stored therein, and 4) proceeding to decode. Generally, this type of process consumes a lot of time for ransomware detection with a potential risk that another ransomware appears until a security company comes up with decryption keys of the old ransomware. Moreover, additional analysis is needed to detect new patterns of ransomware as the hackers persistently come up with new variants of ransomware. On the Android platform, another technique is proposed (Song, Kim, and Lee 2016) to prevent the ransomware intrusion. The technique requires intense monitoring of the executing processes and analysis of the particular file directories using the statistical techniques, such as *Next-generation Intrusion Detection Expert System* (NIDES) (Anderson et al, 1995) using *processor*, *memory usage*, and *I/O rates*, to uncover the applications exhibiting abnormal behavior (Song, Kim and Lee 2016).

Several other research studies have harnessed the machine learning-based approaches and dynamic or runtime features of executing applications to detect ransomware. Recently, HPC events and their features are being used widely in research to detect side-channel attacks and ransomware. Another research (Alam et al. 2020) uses HPC features to detect malware from benign applications. The study used machine learning techniques Recurrent Neural Networks (RNN) to examine HPC data. The authors proposed an anomaly detection technique to identify the malicious ransomware in few seconds with very few false positives. Maiorca et al., (Maiorca et al. 2017) proposed a supervised machine learning-based procedure, *R-PackDroid*, to detect

[1] www.virusshare.com

Android ransomware, which is a light-weight technique and does not require prior knowledge of ransomware's encryption mechanisms. However, the R-PackDroid technique uses fully encrypted code-files and is unable to analyze the applications that load the code at run-time. The R-PackDroid can be incorporated with the other *dynamic analysis* methods, such as the approach proposed by (Kimberly et al., 2015). Moreover, R-PackDroid based application analysis strongly depends on parsing capabilities of the *ApkTool* framework.

In the study (Narudin, et al. 2016), a machine learning-based malware analysis approach based on the anomaly detection mechanism is presented. The results indicated that Bayes network and Random Forest classifiers produce accurate results by attaining 99.97% *True-Positive Rate* (TPR) as compared to the multi-layer perceptron technique with only 93.03% TPR using the *MalGenome* data set. However, the accuracy of this scheme dropped to 85% for the latest malware experiments.

Desktop ransomware can easily bypass any counter-measures and thus results in the seizure of personal data. Authors (Al-rimy, Maarof, and Shaid 2017) presented an effective mechanism for early diagnosis and avoidance of the crypto-ransomware, which is based on machine learning techniques (*One-Class SVM* and n-gram technique (Zhang, et al., 2015)) and comprises three modules: 1) pre-processing, 2) features engineering and 3) detection module. The authors employed an adaptive anomaly detection mechanism that handles the dynamic characteristics of systems and frequently updates the normal profile built from the feature extraction (Al-rimy, et al., 2017) in order to improve the accuracy of detection.

The study (Kharraz, Roberstson, et al. 2015) presented the analysis of ransomware families (the year 2006—2014) and concluded that the suspicious activity of file system should be observed for ransomware detection. For instance, the changes in the types of *I/O Request Packets* (IRP) or the *Master File Table* (MFT) are usually formed to access the file system. The study concluded that a considerable number of ransomware families share related features as a core part of the attacks; however, there still lacks a reliable destructive function to successfully infect files of victims. In Table 1, we recapitulate several other prominent ransomware detections (Yang, et al. 2015; Andronio, Zanero and Maggi 2015; Kharraz, Arshad, et al. 2016) and prevention (Ahmadian, Shahriari and Ghaffarian 2015; Kim, Soh and Kim 2015; Lee, Moon and Park 2016; Brewer 2016) techniques.

Besides, the performance counters exhibit the true application execution behavior and are being employed by the researchers to analyze application performance (Mucci, et al., 1999) (Bahador, et al., 2014) (Demme, et al., 2013). However, none of the existing *dynamic analysis* techniques utilizes the important dynamic feature of HPCs to detect malicious applications. Malware can employ obfuscation techniques to deceive static analysis based anti-viruses. Furthermore, runtime behavior cannot be obfuscated and can be detected using dynamic analysis. We believe this fact should essentially be exploited and the hardware execution profile should be utilized to

execute applications for ransomware detection. Based on these aspects, we argue that HPCs are useful features that could be utilized for the detection and classification of ransomware. In this study, we employ various machine learning classifiers such as Decision Tree, Random Forest, Gradient Boosting, and Extreme Gradient Boosting along with the HPCs to address the following questions:

(1) How different are ransomware from malware at runtime considering machine learning techniques?
(2) Which of the hardware performance counters (HPC) play vital role in ransomware detection?

Motivation and Methodology

The dynamic analysis holds adequate potential to accurately detect the threat of ransomware because an executable program cannot hide its true characteristic. Therefore, most of the anti-virus vendors rely on automated dynamic analysis mechanisms to detect new variants of ransomware. Most of the antivirus applies the heuristics combined with the behavior analysis to deduce whether an executable is *benign* or *malware* (Sgandurra et al. 2016).

A wide range of CPU performance counters i.e., *clock cycles, cache hits, cache misses, branch instructions, branch misses, retired instructions*, etc. are used to observe the behavior of an executing application (Chiappetta, Savas and Yilmaz 2016). Usually, the symmetric encryption marks the cache-based events while the asymmetric encryptions does have an impact on the instruction and branching events as explained in (Alam, et al., 2020). The performance counters have been harnessed by many application developers to identify the computation and memory bottlenecks to improve the performance and reliability of the executing applications (Chiappetta et al. 2016). In this study, we utilize 11 performance counters for the classification of ransomware. For classification, we train the employed machine learning classifiers to analyze the dynamic behavior of ransomware and non-ransomware malicious programs. Moreover, the classification of Ransomware from *Traditional Malware* is essential due to the intensity of the damage caused in terms of informative data and financial loss. Unlike traditional malware, it is more troublesome to identify and kill ransomware even when it is discovered, and the damage is irreparable even after its removal (Al-remy, et al., 2018) and (Zhang, et al., 2019). Hence, it is very important to recognize and isolate the malware from ransomware due to the similarity in nature. Therefore, it is required to devise a formal classification mechanism to discriminate ransomware from other non-ransomware (Zhang, et al., 2019), (Aurangzeb et al, 2017) and (Kok et al., 2019) to avoid billions of transactions in the name of ransom.

DATASET COLLECTION

For the experimentation, we have investigated randomly selected 160 Windows-based malware from *VirusShare*. Afterward, each malware is labeled as a non-ransomware or ransomware based

on the information provided by renowned anti-viruses such as VirusShare. These labeled tags are then further validated with the tags available from *VirusShare* for the sake of confirmation. In this study, benign binary files are not considered because the main aim of the study is to classify between the ransomware and other malicious applications. Therefore, we consider the malicious applications category *Trojan* (as a non-ransomware sample) due to their similarity in activities with the ransomware (Gazet 2010). The employed classifiers are trained using the behavioral features for ransomware and non-ransomware with explicit labeling (i.e., Ransomware/Non-Ransomware). Furthermore, a disjoint data set is used for training and testing purposes.

FEATURE EXTRACTION

All malware in the data set are executed in a quarantine environment and their data related to hardware performance counters are collected using *perf* (an instrumentation and performance analysis tool (Weaver, 2013) (Alam et al., 2020)). To ensure the reliability and accuracy of the results, mean values of three rounds of experiments are reported.

For binary classification, we employ hardware performance counters as features, i.e., 1) *task clock,* 2) *context switching,* 3) *CPU utilized,* 4) *CPU migrations,* 5) *page faults,* 6) *CPU cycles,* 7) *cache-misses,* 8) *instructions retired,* 9) *branches taken,* 10) *branch-misses,* and 11) *execution time*, (illustrated in Table 2) to train the machine learning classifier. Feature selection plays a significant role in achieving precise training of the employed machine learning models; thereby attaining accurate results with efficient performance and low overhead (Li, et al., 2017). Correlation matrix among the employed features is generated to analyze the pattern that leads to selection of features. Two features are considered negatively correlated if a change of one feature inversely impacts the value of the other feature. The features correlation analysis is presented in Figure 1. If two numerical features are highly correlated, then one of them can be ignored. Therefore, we employed a sub-set of those features which are not co-related to reduce the computation overhead during the training process of the machine learning models. For E.g., Figure 1 shows that the *Cache Misses* related hardware feature has a low positive correlation with all the other features showing that the increase in the *Cache Misses* does not necessarily cause an increment in other hardware features. On the other hand, the Task Clock feature has a strong relationship with the *Context Switches, Cycles, Instructions, Branches,* and *Branches Misses*, which indicates that with the increase in *Task Clock*, the other highly correlated features also increase. The features having higher rank are deemed as potential features for classification than low ranked features as shown in Table 3.

In the training phase, hardware features are extracted by executing known malware and non-malware application in containing environment system units as shown in Figures 2a and 2b. Depending upon the labels assigned by the *VirusShare*, each executed malware is labeled as ransomware or non-ransomware. The vectors consisting of hardware performance features with the application category and classification label (ransomware or non-ransomware), are provided

to the machine learning classifiers. 80% of the employed data set is used for training and 20% is used for testing. The goal of the supervised machine learning is to find a function that is trained using the employed features such that the error is minimum for the new or unseen data. In the training phase, the classification model is trained using the hardware performance features as shown in Table 2. The testing or validation methodology is performed after the training of the classifiers.

CLASSIFICATION MODEL

The machine learning classification algorithms namely *Decision Tree, Random Forest, Gradient Boosting,* and *Extreme Gradient Boosting* are used for classification purpose that including *phishing detection, facial recognition*, and *financial predictions* (Jordan and Mitchell, 2015), etc. We employ these four classifiers as part of the proposed methodology to analyze their performance for ransomware detection.

The *decision tree* is a tree-based classifier, which contains a root, internal nodes, and leaf nodes. The class label is assigned to each leaf node and the decisions are rendered by the internal nodes (Tan, et al., 2006). *Random Forest* (RF) classifier is based on a combination of multiple decision tree predictors such that each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest (Tian, et al. 2009). The *Extreme Gradient Boosting* and *Gradient Boosting* follow the same basic principle however, there are a few differences in their modeling details. Specifically, extreme gradient boosting utilizes a more regularized model formalization to control *over-fitting problem* that may occur due to linear fitting over noisy data to provide better performance (Jbabdi, et al., 2012).

Results and Discussion

For experimentation, we utilize a system with Intel core i7 processor, 8 GBs of memory and Ubuntu 12.10 OEM as operating system. For classification, a machine learning tool *Scikit-learn* (Pedregosa, et al., 2011), is employed. To evaluate the results, standard evaluation measures i.e., *precision, recall,* and *F-Measure* are calculated to determine the accuracy of each classifier. Equations 1—4 provide the mathematical description of accuracy, precision, recall, and f-measure, respectively. The terms used in Equations 1—4 are explained as follows: *True Positive* (TP) rate shows the number of predicted positives that are correct, while the *False Positive* (FP) rate refers to the number of predicted positives that are incorrect. Similarly, *True Negative* (TN) rate shows the number of predicted negatives that are correct while the *False Negative* (FN) rate refers to the number of predicted negatives that are incorrect. The recall is the sensitivity for the most relevant result. F-measure is the value that estimates the entire system performance by calculating the harmonic mean of precision and recall. The maximum value of 1.000 for accuracy precision and recall indicates the best result (Narudin, et al., 2016).

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

Precision denotes the proportion of Predicted Positive cases that are correctly Real Positives.

$$\text{Precision} = \frac{TP}{TP + FP} \quad (2)$$

The recall is the proportion of Real Positive cases that are Predicted Positive

$$\text{Recall} = \frac{TP}{TP + FN} \quad (3)$$

$$\text{F-Measure} = 2x\frac{(\text{Precision} \times \text{Recall})}{(\text{Precision} + \text{Recall})} \quad (4)$$

Receiver Operating Characteristic (ROC) curves (Metz, 1978) are extensively being applied in significant researches to measure the accuracy of the machine learning models that are being trained to achieve actual performance (Bradley, 1997). Furthermore, ROC curves are applied in numerous systematic approaches that merge multiple clues, test results, etc., and are plotted and evaluated to characterize a qualitative feature of the particular. ROC is a plot wherein Y-axis is reserved for *True Positive Rate* (TPR) and X-axis is reserved for *False Positive Rate* (FPR). For all possible classification such as the output class, the TPR rate depends on the set-up where the real classification is considered to be as positive and the number of times the classifier has predicted the result to be as positive. The FPR can be defined as how the classifier incorrectly labelled positive to those that are actually classified to be as negative. Together the TPR and FPR values lies in-between 0—1 in a way that 0 label as poor prediction however 1 labelled to be as highly-accurate prediction. The area under ROC (AUC) is now applied for weighing classifiers to get their performance updates (Narudin, et al., 2016).

The results based on the decision tree classifier can be clearly seen in Figure 3. The ROC curve for both classes (i.e., ransomware as class “1” and non-ransomware as class “0”) is the same having value 0.94 which signifies the excellent prediction. However, the precision-recall curve area of class 0 i.e., for *Non-Ransomware* it is 0.89 or 89% whereas for class 1 i.e., ransomware the AUC value is 0.93. The F-measure score of the Decision Tree is 0.94 as shown in Table 4.

The results obtained using the Random Forest classifier for two classes (i.e., *ransomware* and *non-ransomware*) are shown in Figure 4. The higher accuracy results are evident from the similar ROC curve value i.e., 0.99 for both the ransomware and non-ransomware classes. The Random Forest-based classification model outperformed decision tree-based classification by attaining the accuracy of 0.94, as shown in Table 8). However, the value of F-measure for both the classes is 0.97 (as shown in Table 8).

The gradient boosting classification-based results are shown in Figure 5. The results revealed that the ROC curve values for both the classes (i.e., *ransomware* and *non-ransomware*) are the same (i.e., 1.0) and the precision-recall curve of both classes is 1.0. The F-measure score of the gradient boosting classifier is 0.93 for ransomware and 0.94 for non-ransomware (as shown in Table 6).

The extreme gradient boosting classification model-based results are shown in Figure 6 and Table 7. The ROC curve and Precision-Recall Curve of both classes (i.e., ransomware and non-ransomware) are the same (i.e., 1.0). The extreme gradient boosting based model's F-measure score is 0.97, which is similar to the gradient boosting and random forest-based classification as shown in Table 8. The model has attained an improvement of 3% than the decision tree-based classification. The model shows similar results of 0.97 as observed for random forest and gradient boosting.

This study has demonstrated the possibility of exploiting HPCs as the potential features for ransomware detection. After analyzing the sets of ransomware and non-ransomware, the features obtained from HPCs have been analyzed to classify malicious applications into ransomware and non-ransomware categories using several machine learning algorithms such as *Decision Tree, Random Forest, Gradient Boosting,* and *Extreme Gradient Boosting.* The results of detailed experiments as stated earlier in the section have revealed that extracted hardware features play a significant role in the detection and identification of ransomware. Among all the employed machine learning classifiers, the random forest-based model has outperformed by yielding an accuracy of 0.97 followed by a decision tree with an accuracy of 0.94. Moreover, the features cache misses, task clock, and branches obtained through HPCs could be deemed as potential parameters in classifying ransomware from non-ransomware.

Conclusions

The origination of new variants of ransomware and expeditious increase in its families has adhered to the software developers to efficiently detect and deal with such applications. In the literature, numerous studies have been performed to address different applications of ransomware. However, these schemes contain some deficiencies that allow cybercriminals to bypass security measures. The addition of hardware support and hardware performance analysis could be deemed as potential measures to deal against ransomware to new grounds. The hardware-based analysis and diagnosing the potential threat at the early stages could beneficiate the process of ransomware detection before its malicious activity. In this paper, the analysis of HPCs has been presented for Windows ransomware classification. The results have revealed that the HPCs hold the considerable potential to expose hidden indicators of the executing applications such as malicious codes and ransomware. Performance counters, i.e., *cache misses, task clock*, and *branches* have played a pivotal role in classifying ransomware in a way that if there are a high number of cache misses or a high number of branch mispredictions (where

control flow becomes detectably anomalous) are good indicators that help in indicating a potential attack (Foreman, 2018). The proposed technique holds adequate potential to provide sufficient detection accuracy by attaining the F-measure score of 0.97. This study demonstrated the possibility of exploiting HPCs as the potential feature for the detection of ransomware. However, this topic needs further investigation. In the future, we intend to scrutinize other dynamic features with the combination of call graphs to detect and classify ransomware. Moreover, the application of machine learning algorithms has shown very promising results in ransomware detection. In the future, we will expand this study to perform in-depth static analysis as well as dynamic analysis with the combination of HPCs in the detection of that ransomware that usually hides by implementing various obfuscation techniques (like packed or compressed programs, or indirect addressing (Behera & Bhaskari, 2015)). One major challenge and limitation of this research is in ransomware detection of false positives and false negatives. Consider the case of Qwerty ransomware, which uses a benign GPG executable to perform encryption. Perhaps the proposed solution would correctly detect the GPG binary when used in this way, but we suspect it would also detect it in a benign case. Since in this work we did not evaluate benign executables, it is not clear how the system performs with software that performs encryption and/or compression tasks which is the limitation of this research that will be investigated in our future work.

References

Ahmadian, M. M., Shahriari, H. R. & Ghaffarian, S. M., 2015. *Connection-monitor & connection-breaker: A novel approach for prevention and detection of high survivable ransomwares.* s.l., IEEE, pp. 79-84.

Alam, M., Sinha, S., Bhattacharya, S., Dutta, S., Mukhopadhyay, D. and Chattopadhyay, A., 2020. RAPPER: Ransomware prevention via performance counters. arXiv preprint arXiv:2004.01712.

Al-rimy, B. A. S., Maarof, M. A. & Shaid, S. Z. M., 2017. *A 0-Day Aware Crypto-Ransomware Early Behavioral Detection Framework.* s.l., Springer,Cham, pp. 758-766.

Al-rimy, B.A.S., Maarof, M.A. and Shaid, S.Z.M., 2018. Ransomware threat success factors, taxonomy, and countermeasures: A survey and research directions. *Computers & Security*, 74, pp.144-166.

Ammons, G., Ball, T. & Larus, J. R., 1997. Exploiting hardware performance counters with flow and context sensitive profiling. *ACM Sigplan Notices,* 32(5), pp. 85-96.

Anderson, D., Thane, F. & Alfonso, V., 1995. Next-generation intrusion detection expert system (NIDES): A summary.

Andronio, N., Zanero, S. & Maggi, F., 2015. *Heldroid: Dissecting and detecting mobile ransomware.* s.l., Springer International Publishing, pp. 382-404.

Aurangzeb, S., Aleem, M., Iqbal, M. A. & Islam, M. A., 2017. Ransomware: A Survey and Trends. *Journal of Information Assurance and Security,* Volume 12, pp. 048-058.

Bahador, M. B., Abadi, M. & Tajoddin, A., 2014. *Hpcmalhunter: Behavioral malware detection using hardware performance counters and singular value decomposition." ,.* s.l., IEEE, pp. 703-708.

Batcheller, A. et al., 2017. Building on the Success of Building Security. *IEEE Security & Privacy,* 15(4), pp. 85-87.

Behera, C. K. & Bhaskari, D. L., 2015. Different obfuscation techniques for code protection.. *Procedia Computer Science,* Volume 70, pp. 757-763.

Beneventi, F., Bartolini, A., Cavazzoni, C. & Benini, . L., 2017. *Continuous learning of HPC infrastructure models using big data analytics and in-memory processing tools.* s.l., IEEE, pp. 1038-1043.

Bradley, A., 1997. The use of the area under the ROC curve in the evaluation of machine learning algorithms.. *Pattern Recognit,* 30(7), pp. 1145-1159.

Brewer, R., 2016. Ransomware attacks: detection, prevention and cure. *Network Security,* 2016(9), pp. 5-9.

Chen, Q. & Robert, A. B., 2017. Automated Behavioral Analysis of Malware A Case Study of WannaCry Ransomware.. Volume arXiv preprint arXiv:1709.08753.

Chen, T., Tong, H. & Michael, B., 2015. Xgboost: extreme gradient boosting. *R package,* 0.4(2), pp. 1-4.

Chen, W. et al., 2016. *More Semantics More Robust: Improving Android Malware Classifiers.* s.l., ACM, pp. 147-158.

Chen, Z., Kang, H., Yin, S. & Kim, S., 2017. *Automatic Ransomware Detection and Analysis Based on Dynamic API Calls Flow Graph..* s.l., s.n., pp. 196-201.

Chiappetta, M., Savas, E. & Yilmaz, C., 2016. Real time detection of cache-based side-channel attacks using hardware performance counters. *Applied Soft Computing,* Volume 49, pp. 1162-1174.

Choi, K. S., Scott, T. M. & LeClair, D. P., 2016. Ransomware Against Police: Diagnosis of Risk Factors via Application of Cyber-Routine Activities Theory. *International Journal of Forensic Science & Pathology,* IV(7), pp. 253-258.

Das, S., Werner, J., Antonakakis, M., Polychronakis, M. and Monrose, F., 2019, May. SoK: The challenges, pitfalls, and perils of using hardware performance counters for security. In *2019 IEEE Symposium on Security and Privacy (SP)* (pp. 20-38). IEEE.

Demme, J. et al., 2013. On the feasibility of online malware detection with performance counters. *ACM SIGARCH Computer Architecture News,* June, 41(3), pp. 559-570.

Davies, S.R., Macfarlane, R. and Buchanan, W.J., 2020. Evaluation of live forensic techniques in ransomware attack mitigation. Forensic Science International: Digital Investigation, 33, p.300979.

Egele, M., Scholte, T., Kirda, E. & Kruegel, C., 2012. A survey on automated dynamic malware-analysis techniques and tools. *ACM computing surveys (CSUR),* 44(2), p. 6.

Foreman, J.C., 2018. A Survey of Cyber Security Countermeasures Using Hardware Performance Counters. *arXiv preprint arXiv:1807.10868*.

510 Friedman, J. H., 1999. Reitz Lecture. 29(2001), pp. 1189-1232.
511 Gazet, A., 2010. Comparative analysis of various ransomware virii. *Journal in computer*
512 *virology,* 6(1), pp. 77-90.
513 Grant, L. & Parkinson, S., 2018. Identifying File Interaction Patterns in Ransomware Behaviour.
514 In: *Guide to Vulnerability Analysis for Computer Networks and Systems.* s.l.:Springer, pp.
515 317-335.
516 Hampton, N., Baig, Z. & Zeadall, S., 2018. Ransomware Behavioural Analysis on Windows
517 Platform. *Journal of Information Security and Applications,* Volume 40, pp. 44-51.
518 Jordan, M. I. & Mitchell, T. M., 2015. Machine learning: Trends, perspectives, and prospects.
519 *Science,* 349(6245), pp. 255-260.
520 Jbabdi, S., Sotiropoulos, S.N., Savio, A.M., Graña, M. and Behrens, T.E., 2012. Model-based
521 analysis of multishell diffusion MR data for tractography: how to get over fitting
522 problems. Magnetic resonance in medicine, 68(6), pp.1846-1855.
523 Kaur, G., Dhir, R. & Singh, M., 2017. Anatomy of ransomware malware: detection, analysis and
524 reporting. *International Journal of Security and Networks,* 12(3), pp. 188-197.
525 Kharraz, A., Arshad, S., Mulliner, C. & Robertson, W. K., 2016. UNVEIL: A Large-Scale,
526 Automated Approach to Detecting Ransomware. *USENIX Security Symposium,* pp. 757-
527 772.
528 Kharraz, A. et al., 2015. *Cutting the gordian knot: A look under the hood of ransomware*
529 *attacks.*. s.l., Springer, Cham, 2015., pp. 3-24.
530 Kimberly, T., Salahuddin J, K., Aristide, F. & Lorenzo, C., 2015. CopperDroid: Automatic
531 Reconstruction of Android Malware Behaviors. *NDSS.*
532 Kim, D., Soh, W. & Kim, S., 2015. Design of quantification model for prevent of cryptolocker.
533 *Indian Journal of Science and Technology,* 8(19).
534 Kohavi, R., 1996. Scaling up the accuracy of Naive-Bayes classifiers: a decision-tree hybrid.. *In*
535 *KDD,* 96(Citeseer), p. In KDD.
536 Kok, S., Abdullah, A., Jhanjhi, N. and Supramaniam, M., 2019. Ransomware, threat and
537 detection techniques: A review. *Int. J. Computer Science and Network Security*, 19(2),
538 p.136.
539 Krebsonsecurity. 2017. *Microsoft Issues WanaCrypt Patch for Windows 8, XP*. Available at
540 https://krebsonsecurity.com/2017/05/microsoft-issues-wanacrypt-patch-for-windows-8-
541 xp/
542 Labs, M., 2017. *Threat Predictions Ransomware Infographic",* s.l.: McAfee Labs Threat
543 Predictions report 2017.
544 Lee, J. K., Moon, S. Y. & Park, J. H., 2016. CloudRPS: a cloud analysis based enhanced
545 ransomware prevention system. *The Journal of Supercomputing,* 73(7), pp. 3065-3084.
546 Liaw, A. & Wiener, M., 2002. Classification and regression by randomForest. *R news,* 2(3), pp.
547 18-22.
548 Li, J. et al., 2017. Feature selection: A data perspective. *ACM Computing Surveys (CSUR),* 50(6),
549 p. 94.

550 Maiorca, D. et al., 2017. *R-PackDroid: API package-based characterization and detection of*
551 *mobile ransomware.* s.l., ACM, pp. 1718-1723.
552 Malone, C., Zahran, M. and Karri, R., 2011, October. Are hardware performance counters a cost
553 effective way for integrity checking of programs. In *Proceedings of the sixth ACM*
554 *workshop on Scalable trusted computing* (pp. 71-76).
555 Martinelli, F., Mercaldo, F. & Saracino, A., 2017. *Bridemaid: An hybrid tool for accurate*
556 *detection of android malware. ..* s.l., In Proceedings of the 2017 ACM on Asia
557 Conference on Computer and Communications Security ACM, pp. 899-901.
558 Maurya, A., Kumar, N., Agrawal, A. & Khan, R., 2018. Ransomware: Evolution, Target and
559 Safety Measures. *International Journal of Computer Sciences and Engineering,* 6(1).
560 Metz, C. E., 1978. Basic principles of ROC analysis. *Seminars in nuclear medicine,* 8(4).
561 Micro, T., 2016. *Ransomware,* s.l.: Trend Micro Incorporated Labs report, 2016.
562 Milletary, J., 2012. Citadel trojan malware analysis. *Luettavissa: http://botnetlegalnotice.*
563 *com/citadel/files/Patel_Decl_Ex20. pdf. Luettu*.
564 Mucci, P. J., Browne, S., Deane, C. & Ho, G., 1999. *PAPI: A portable interface to hardware*
565 *performance counters.* s.l., Proceedings of the department of defense HPCMP users
566 group conference.
567 Narudin, F. A., Feizollah, A., Anuar, N. B. & Gani, A., 2016. Evaluation of machine learning
568 classifiers for mobile malware detection. *Soft Computing,* 20(1), pp. 343-357.
569 Pedregosa, F. et al., 2011. Pedregosa, Fabian, et al. Scikit-learn: Machine learning in Python..
570 *Journal of machine learning research ,* pp. 2825-2830.
571 Perf , L., 2016. Linux profiling with performance counters.
572 Powers & David, M., 2011. Evaluation: from precision, recall and F-measure to ROC,
573 informedness, markedness and correlation..
574 Sgandurra, D., Muñoz-González, . L., Mohsen, . R. & Lupu, E. C., 2016. Automated Dynamic
575 Analysis of Ransomware: Benefits, Limitations and use for Detection. *arXiv preprint*
576 *arXiv:1609.03020.*
577 Singh, B. et al., 2017. On the detection of kernel-level rootkits using hardware performance
578 counters.. *In Proceedings of the 17th Asia Conference on Computer and Communications*
579 *Security (AsiaCCS). ACM,* pp. 483-493.
580 Song, S., Kim, B. & Lee, S., 2016. The effective ransomware prevention technique using process
581 monitoring on android platform.
582 Tan, P.-N., Steinbach, . M. & Kumar, V., 2006. Classification: basic concepts, decision trees,
583 and model evaluation.. *Introduction to data mining,* Volume 1, pp. 145-205.
584 Term, R. C., 2000. R language definition. *Vienna, Austria: R foundation for statistical*
585 *computing.*
586 Tian, R., Batten, L., Islam, R. & Versteeg, S., 2009. *An Automated Classification System Based*
587 *on the Strings of Trojan and Virus.* s.l., IEEE, pp. 23-30.

588 Wang, X. et al., 2016. Hardware performance counter-based malware identiication and detection
589 with adaptive compressive sensing. *Transactions on Architecture and Code Optimization*
590 *(TACO).*
591 Weaver, V. M., 2013. Linux perf_event features and overhead.. *The 2nd International Workshop*
592 *on Performance Analysis of Workload Optimized Systems, FastPath.,* Volume 13.
593 Xu, Z., Ray, S., Subramanyan, P. & Malik, S., 2017. Malware detection using machine learning
594 based analysis of virtual memory access patterns.. *In Proceedings of the Conference on*
595 *Design, Automation & Test in Europe European Design ,* pp. 169-174.
596 Yang, . T., Yang, Y., Qian, K. & Tao, L., 2015. *Automated detection and analysis for android*
597 *ransomware.* s.l., IEEE, pp. 1338-1343.
598 Zavarsky, P. & Lindskog, D., 2016. Experimental Analysis of Ransomware on Windows and
599 Android Platforms: Evolution and Characterization. *Procedia Computer Science,*
600 94(2016), pp. 465-472.
601 Zhang, M., Xu, B. & Wang, D., 2015. *An anomaly detection model for network intrusions using*
602 *one-class SVM and scaling strategy.* Cham, Springer, pp. 267-278.
603 Zhang, H., Xiao, X., Mercaldo, F., Ni, S., Martinelli, F. and Sangaiah, A.K., 2019. Classification
604 of ransomware families with machine learning based on N-gram of opcodes. *Future*
605 *Generation Computer Systems,* 90, pp.211-221.
606 Zhao, M., Zhang, T., Ge, F. & Yuan, Z., 2012. RobotDroid: A Lightweight Malware Detection
607 Framework On Smartphones. *JNW,* 7(4), pp. 715-722.
608 Zhou, B. et al., 2018. Hardware Performance Counters Can Detect Malware: Myth or Fact?.. *In*
609 *Proceedings of the 2018 on Asia Conference on Computer and Communications Security.*
610 *ACM.,* pp. 457-468.

Table 1(on next page)

Summary of literature review along with their key points, drawbacks and implementation design approach

1

2 Table 1. Summary of literature review along with their key points, drawbacks and

3 implementation design approach.

Reference	Methodology	Strengths	Limitations
Narudin et al., (2016)	• Machine learning-based study • Filter TCP packets, extract network traffic features • Evaluate Bayes, Random Forest, KNN, J48, & MLP	• Accurate detection based on ML classifiers. • BN and RF produces 99.97% TPR • Bayes, MLP with ROC 0.995 and RF with 0.991	• Applicable for Android platform only
Zavarsky and Lindskog (2016)	• the life cycle of Windows-based Ransomware study. • Implement basic static and basic dynamic • MD5 method, Cuckoo Sandbox used. • For android Analyze AndroidManifest.xml, administrative privilege • For Windows analyze Filesystems, registry activities, and network operations	• Explained the detailed analysis, working, and functionality of Ransomware • Performed analysis on both the Windows and Android-based RW • PEiD tool is used for windows ransomware detection	• Performed only basic static and dynamic analysis. • No machine learning-based approach to detect zero-day ransomware • Lack of experimental analysis
Song, et al., (2016)	• Proposed techniques on three modules: Configuration, Monitors, and Processes sing • the hash information method is used for detection of CryptoLocker type ransomware	• The proposed technique monitors the processes and specific file directories • monitor file events using statistical methods on Processor usage, Memory usage, and I/O rates	• Not applicable for Windows-based ransomware • No classifier is used • Does not install applications and execute for prevention and detection • Results are not analyzed quantitatively
Al-rimy et al., (2017)	• Machine learning n-gram, EFCM, • Information Gain, • Sliding window • Static + dynamic conf • SVM for behavioral detection	• Proposed framework inclines to share the pre-encryption data space as the main defense step against crypto-ransomware attacks	• No classification • No experimental work • No results evaluation details
Kharraz et al., (2015)	• Analyzed 15 ransomware families • Proposed various mitigation approaches to decoy resources to detect malicious file access.	• Provide evolution-based study of RW attacks from a long-term study 2006-2014 • Detailed analysis of Bitcoin for monetization	• Assumed that every file system access to delete or encrypt decoy resources • However, they didn't implement any concrete solution to detect or defend against these attacks

Chen and Robert, (2017)	• Dynamic behavioral analysis of wanna cry • Present a method to extract features of malware from hosts logs • TF-IDF approach gives better results for analyzing wanna cry	• Research helps in further manual analysis of logs from ambient system logs in forensic efforts. • Automatically generate behavior analysis of malware samples from sandbox log data	• Presentation and experimented results are outside the scope of the paper • Study not help in analyzing automatic pattern generation
Kharraz et al., (2016)	• dynamic approach • Monitors file system I/O activity • Detect screen locking mechanism, • used Tesseract-OCR	• new ransomware family were detected that was not detected previously • The long-term study analyzed 148223 malware samples and correctly detect and verified 13637 ransomware samples • 96.3% TP rate and 0 FPs	• Accuracy is not that good. For example, the system correctly detects 7,572 ransomware whereas only one unknown was detected
Sgandurra et al., (2016)	• Dynamically monitor file system activity on windows platform • Classify between goodware and ransomware using ML • Mutual Information and Regularized Logistic Regression classifier used. • Proposed machine learning approach EldeRan	• e□ective and entirely automated tool to analyze new software and enhance the detection capabilities of AV software • registry key and API calls are the two classes with the most relevant features. • EldeRan achieves ROC curve of 0.995, detection rate 96.3%	• Despite good results, EldeRan still not be used as a replacement for AV • the current settings have no other applications running in the VM, except the ones coming with a fresh installation of Windows, • initial dataset was larger • Unable to analyze RW that shows silent behavior, or wait for the user to do something
Kim and Kim (2015)	• present a quantification model based on social engineering technique to avoid and identify any cryptographic operations in the local drive	• explains the file-based intrusion detection system and IP traceback algorithm	• Lack of experimental results • Suggests guidelines online
Demme et al., (2013)	• Dynamic approach • Android Malware detection with performance counters • Applied ML algorithms (KNN, Decision tree)	• Major support is that runtime behavior can capture using HW performance counters are essential to detect malware • 90% accuracy with 3% FP	• Able to detect some variants whereas some were not detected • Malware label data might not accurate
Alam et al. 2020	• Dynamic Analysis • Implement Artificial Neural Network and Fast Fourier Transformation • Disk encryption detection module process used	• Two-step detection framework named as RAPPER • an accurate, fast, and reliable solution to detect ransomware. • Used minimal tracepoints • Provide a comprehensive solution • to tackle standard benchmark, • disk encryption and regular high • computational processes • HPCs were used to analyze files using perf tool	• Observe 5 events of HPCs only i.e., instruction, cache-references, cache-misses, branches, and branch-misses • Analyze and present all the case studies by giving a comparison with WannaCry only • Lack of detailed experimental results and

			accuracies.

4

5

6

Table 2(on next page)

Features Set used in this work for performance evaluation (HPCs)

Table 2. Features Set used in this work for performance evaluation (HPCs).

S.no	Hardware Features	Description
1	Task-clock	The task-clock explans the amount of time spent on the task (Kuznetsova et al. 2017)
2	CPU utilization	CPU-clock is based on the total time spent on the CPU.
3	Context Switching	explains the number of times the software switched off the CPU from one process/thread to another (Kuznetsova et al. 2017)
4	CPU Migration	CPU migration refer to equality in a workload distribution across all cores. (Kuznetsova et al. 2017)
5	Page Faults	Page-faults occur when a program's virtual content has to be copied to the physical memory (Kuznetsova et al. 2017)
6	Instructions per cycle	The average number of instructions executed for each clock cycle
7	Branch	A branch is an instruction in a computer program that can cause a computer to begin executing a different instruction sequence and thus deviate from its default behavior of executing instructions in order
8	Branch Misses	Branch misprediction occurs when a processor mispredicts the next instruction to process in branch prediction, which is aimed at speeding up execution.
9	Cycles	Perf-CPU-cycles is a count of CPU cycles that traces to a hardware counter (Flater & Flater, 2014)
10	Cache Misses	Cache misses is a state of not getting data that is being processed by a component or application that is not found in the cache.
11	Total Time elapsed	It's the total execution time in seconds

Table 3(on next page)

Features Rank List

Table 3. Features rank list.

Rank	Score	Feature
1	0.20145	cache misses
2	0.181887	taskClock
3	0.153562	Branches
4	0.10867	secondsTimeElapsed
5	0.086973	Instructions
6	0.085666	branchMisses
7	0.044272	contextSwitches
8	0.042727	pageFaults
9	0.040087	CPU migration
10	0.028564	Cycles
11	0.026142	CPUsUtilized

Table 4(on next page)

Decision Tree precision, recall and F-measure score for malware classes (0,1)

1

2 Table 4. Decision Tree precision, recall and F-measure score for malware classes (0,1).

Malware Class	Precision	Recall	F- Measure
Ransomware (class label 1)	1.0	0.88	0.93
Non- Ransomware (class label 0)	0.89	1.0	0.94

3

4

Table 5(on next page)

Random Forest Precision Recall and F-Measure Score against classes 0 and 1

1

2 Table 5 Random Forest Precision Recall and F-Measure Score against classes 0 and 1.

Malware Class	Precision	Recall	F- Measure
Ransomware (class label 1)	1.0	0.94	0.97
Non- Ransomware (class label 0)	0.94	1.0	0.97

3

Table 6(on next page)

Gradient Boosting precision, recall and F-measure score for malware classes

1 Table 6 Gradient Boosting precision, recall and F-measure score for malware classes (0,1)

Malware Class	Precision	Recall	F- Measure
Ransomware (class label 1)	1.0	0.88	0.93
Non- Ransomware (class label 0)	0.89	1.0	0.94

2

3

Table 7 (on next page)

Extreme Gradient Boosting precision, recall and F-measure score for malware

Table 7 Extreme Gradient Boosting precision, recall and F-measure score for malware classes (0,1).

Malware Class	Precision	Recall	F- Measure
Ransomware (class label 1)	1.0	0.94	0.97
Non- Ransomware (class label 0)	0.94	1.0	0.97

Table 8(on next page)

Four classifiers result and their comparison F-measure score

1 Table 8 Four classifiers result and their comparison F-measure score

Classifier	F- Measure
Decision Tree	0.94
Random Forest	0.97
Gradient Boosting	0.94
Extreme Gradient Boosting	0.97

2

3

4

Figure 1

A map tracking the global spread of Wanna.Cry ransomware (malwaretech.com)

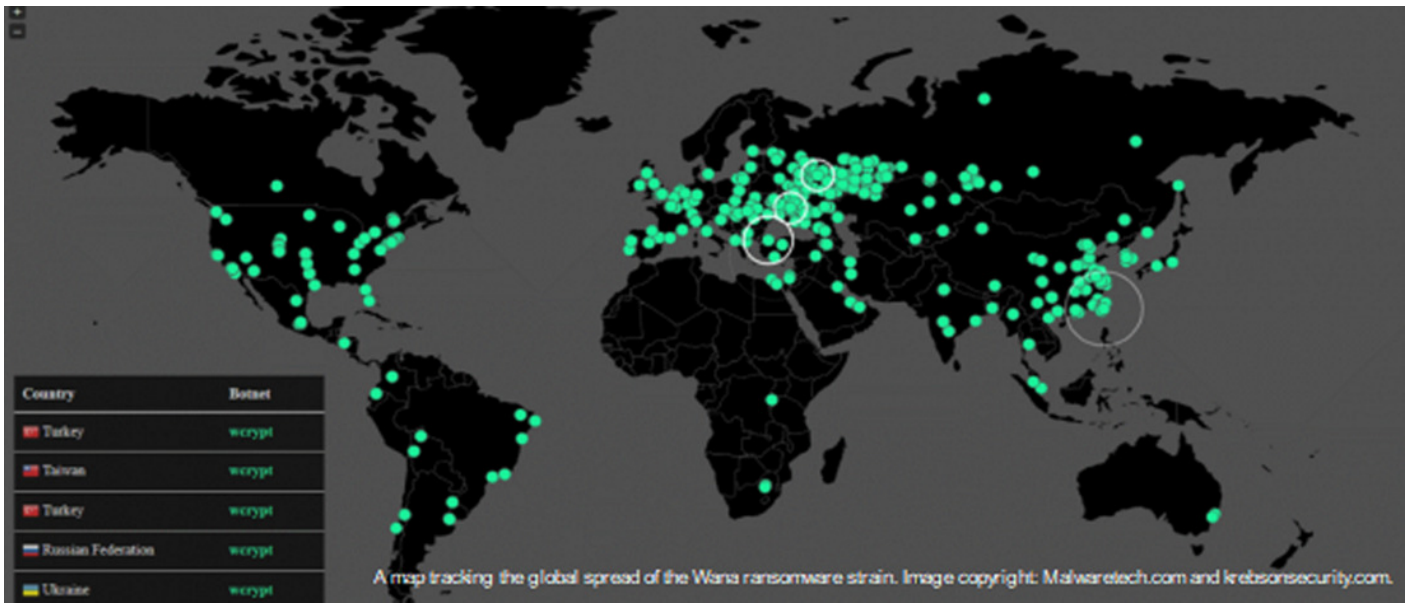

Figure 2

Feature Set correlation analysis

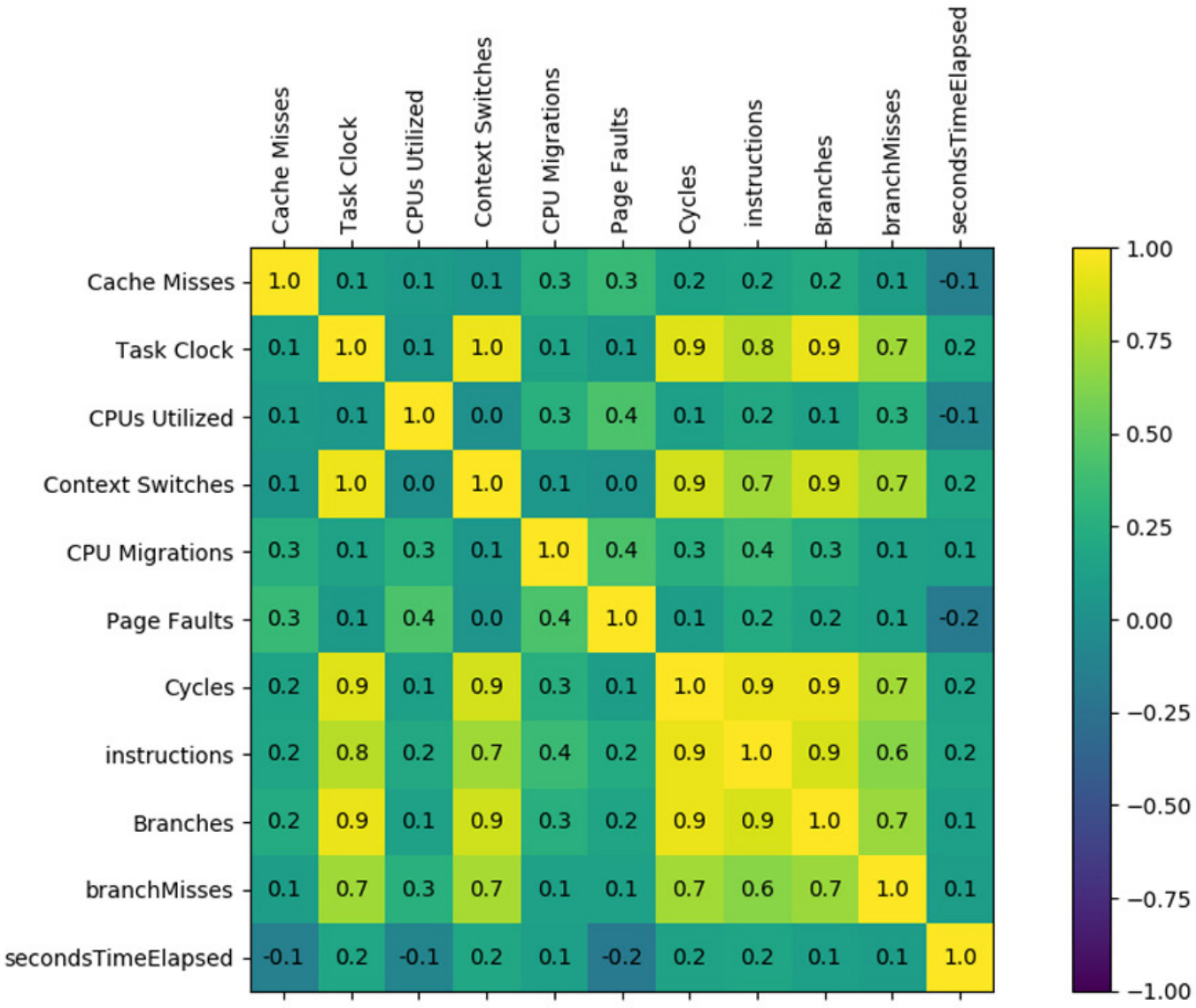

Figure 2 Feature Set correlation analysis

Figure 3

Workflow of training and testing phases

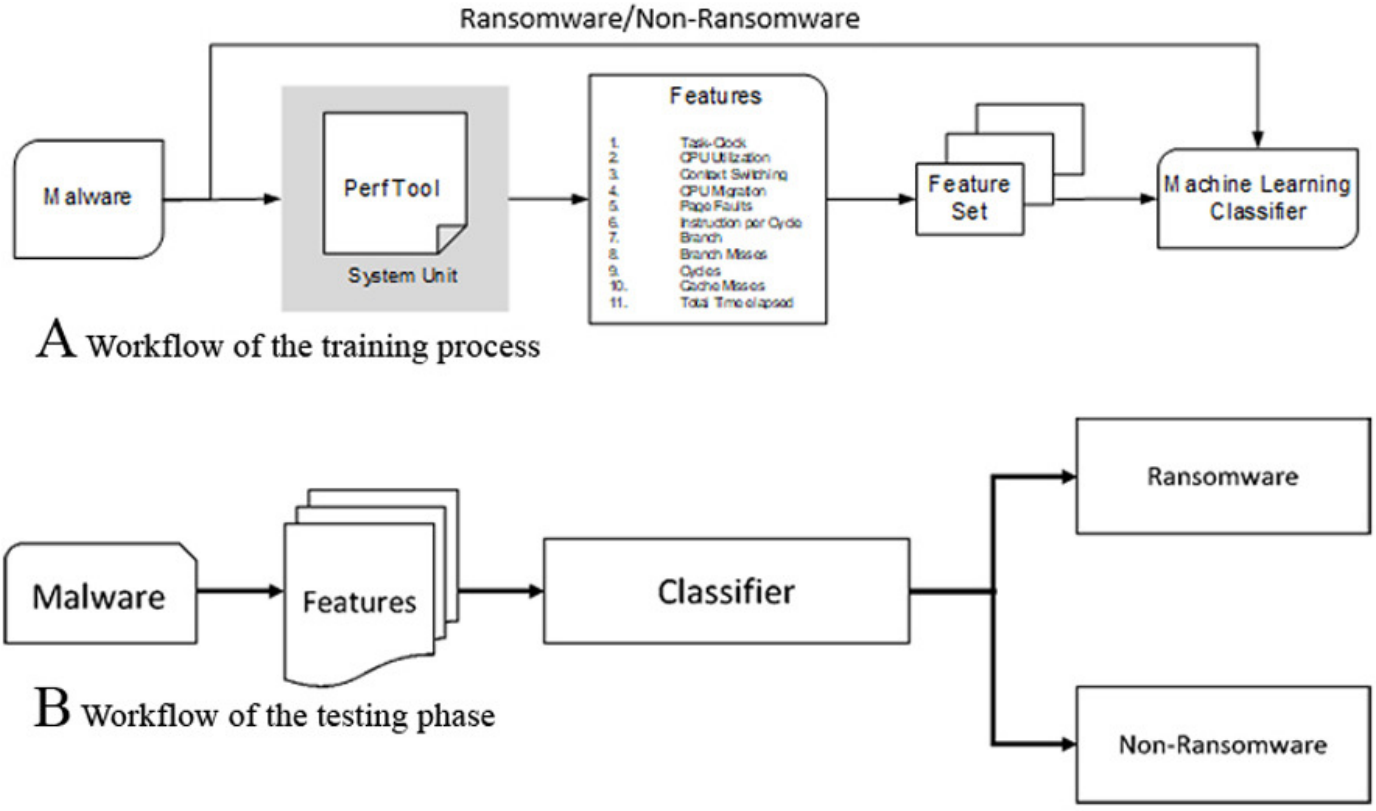

Figure 3 Workflow of the training and testing phases

Figure 4

Decision Tree performance metrics

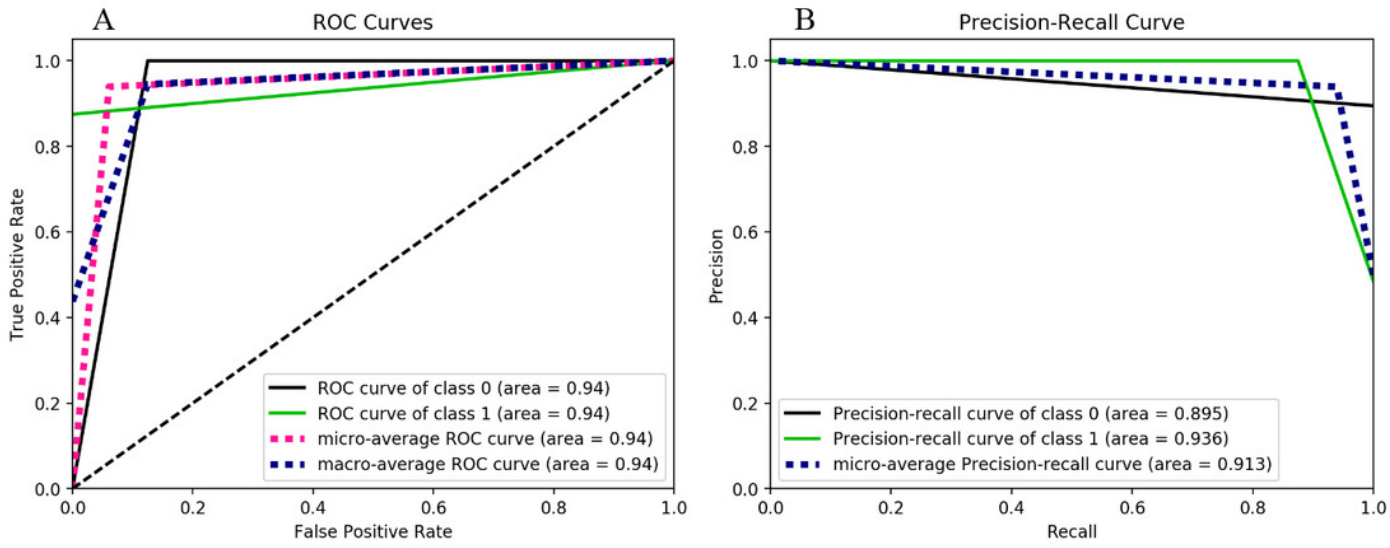

Figure 4 Decision Tree Performance Metrics

Figure 5

Random Forest performance metrics

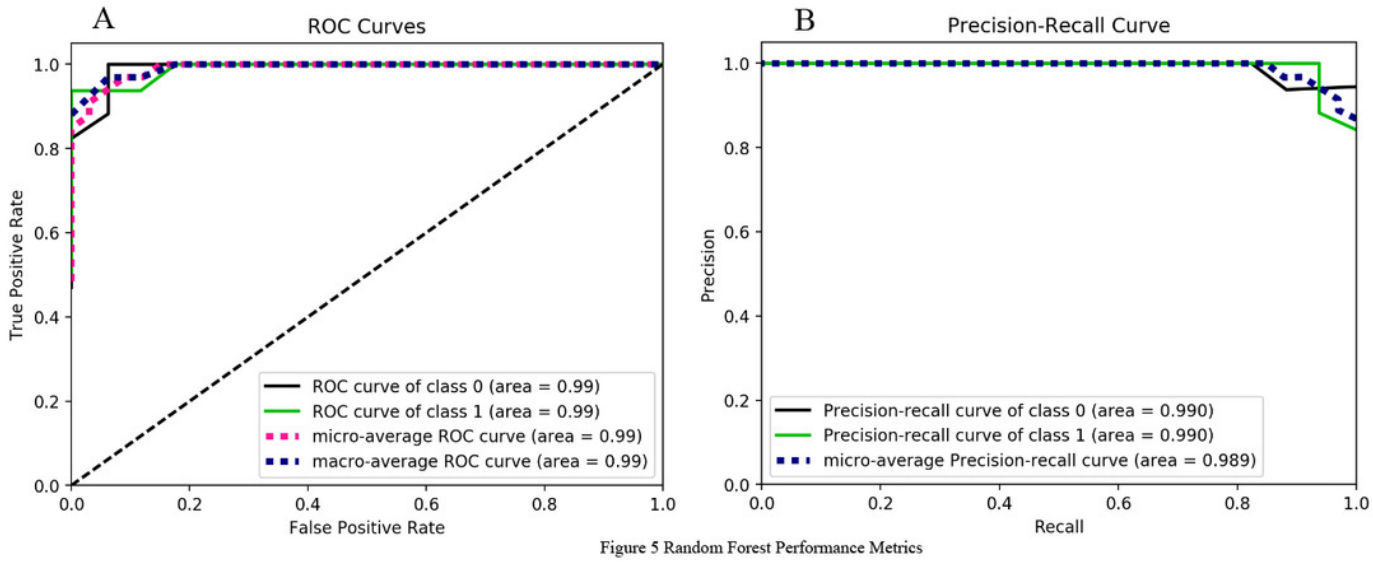

Figure 5 Random Forest Performance Metrics

Figure 6

Gradient Boosting performance metrics

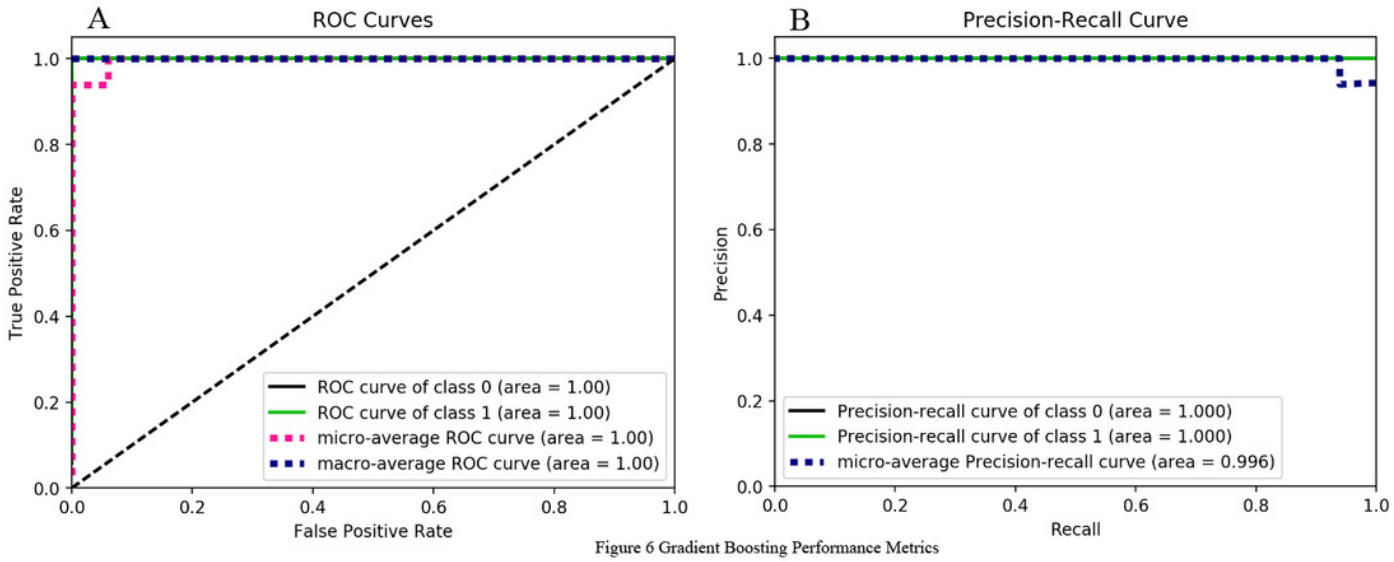

Figure 6 Gradient Boosting Performance Metrics

Figure 7

Extreme Gradient Boosting performance metrics

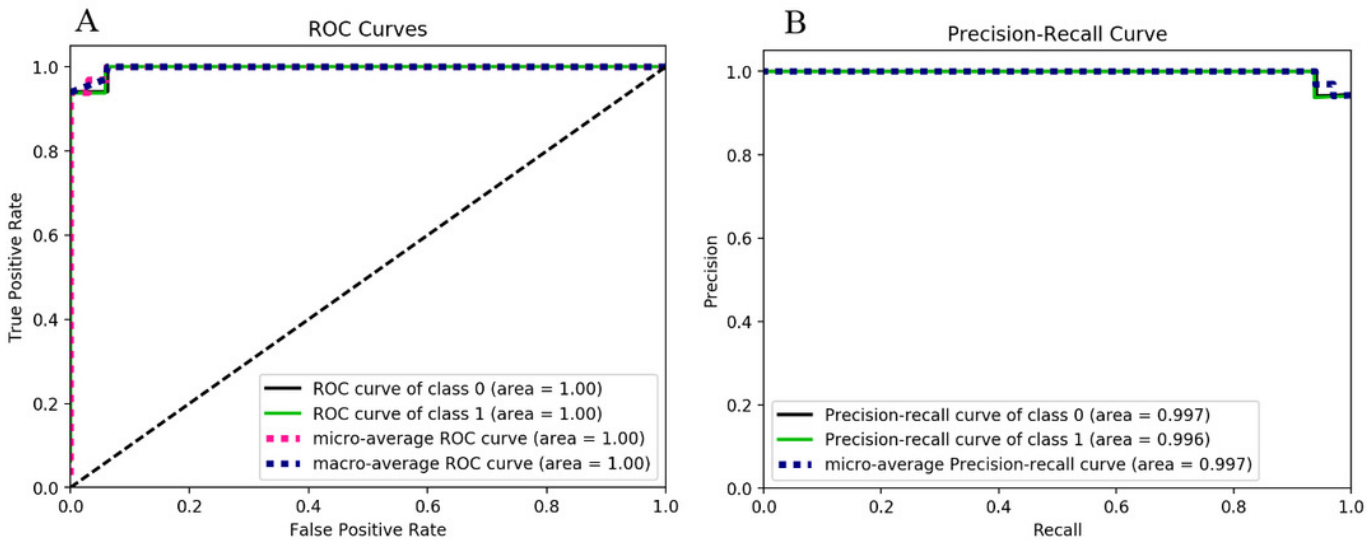

Figure 7 Extreme Gradient Boosting Performance Metrics