ATLANTIS PRESS International Journal of Computational Intelligence Systems, Vol. 11 (2018) 1153-1169

Evaluation of Supervised Machine Learning Techniques for Dynamic Malware Detection

Hongwei Zhao [1] Mingzhao Li [2] , Taiqi Wu [2] , Fei Yang [3]

[1] College of Computer Science and Technology,
Jilin University, Changchun, China
State Key Laboratory of applied optics, Changchun
Key Laboratory of Symbolic Computation and Knowledge Engineering for Ministry of Education, Jilin University, Changchun
E-mail: zhao_hongwei6@sohu.com

[2] College of Computer Science and Technology,
Jilin University, China
E-mail: li_mingzhao@sina.cn

[3] College of Software,
Jilin University, China
E-mail: gulsha_ahuja@outlook.com

Received 21 May 2017

Accepted 28 May 2018

Abstract

Nowadays, security of the computer systems has become a major concern of security experts. In spite of many antivirus and malware detection systems, the number of malware incidents are increasing day by day. Many static and dynamic techniques have been proposed to detect the malware and classify them into malware families accurately. The dynamic malware detection has potential benefits over the static ones to detect malware effectively. Because, it is difficult to mask behavior of malware while executing than its underlying code in static malware detection. Recently, machine learning techniques have been the main focus of the security experts to detect malware and predict their families dynamically. But, to the best of our knowledge, there exists no comprehensive work that compares and evaluates a sufficient number of machine learning techniques for classifying malware and benign samples. In this work, we conducted a set of experiments to evaluate machine learning techniques for detecting malware and their classification into respective families dynamically. A set of real malware samples and benign programs have been received from VirusTotal, and executed in a controlled & isolated environment to record malware behavior for evaluation of machine learning techniques in terms of commonly used performance metrics. From the execution reports saved in the form of JSON reports, we extract a promising set of features representing behavior of a malware sample. The identified set of features is further employed to classify malware and benign samples. The Major motivation of this work is that different techniques have been designed to optimize different criteria. So, they behave differently, even in similar conditions. In addition to classification of malware and benign samples dynamically, we reveal guidelines for researchers to apply machine learning techniques for detecting malware dynamically, and directions for further research in the field.

Keywords: Dynamic Analysis, Malware detection, Machine Learning, Static Analysis

1. Introduction

The enormous growth of online services and resources has increased the number of Internet users through a variety of devices ranging from computers to embedded systems. This Internet connectivity has provided many services to the end users, like easy and quick communication. Nowadays, end users can enjoy online services anywhere-anytime through an Internet connected device like mobile phones, tablets, etc. This increasing number of Internet users also activated the malicious programmers to develop malicious applications/programs generally called malware. In the recent years, a huge amount of malware has been noticed as depicted in Figure 1.

Many antivirus, intrusion detection systems and other malware detection systems have been developed for the prevention of damage caused by these malicious programs. Still, there exist some issues that need immediate attention. Because of changing nature of malware and flaws in existing software. Various techniques from different disciplines have been proposed for effective malware detection. The techniques can be categorized broadly into two categories, namely the static, signature based techniques and dynamic, behavior based techniques. Static techniques analyze malware based upon its structure, control flow, etc. without executing it[15]. These techniques involve establishment of a signature database. The major limitation is that these techniques fail to detect a novel malware until its signature is updated[4,18,10]. Whereas, dynamic techniques analyze the malware samples during its execution. These techniques analyze behavior of malware samples from their execution reports. In recent years, malicious programmers are developing more complex and advanced malware using obfuscation and encryption techniques. Static techniques fail to detect malware accurately. Whereas, dynamic techniques have benefited over static techniques, because it is more difficult to mask the behavior of malware during its execution. Taking into consideration the benefits of dynamic techniques, the focus of current research has shifted to dynamic[23,15] and automated techniques for malware detection[6,1,7].

In the recent years, many researchers employed machine learning (ML) techniques to tackle constantly changing behavior of malware detection dynamically. The ML techniques take a labeled dataset as a training dataset and develop a model representing the behavior of malware and benign samples. The trained model is found to capable of classifying test samples. The ML techniques can learn from huge amount of labeled training data to enhance their predictive accuracy.

The ML techniques have been divided into different categories depending upon their working namely: 1) Rule based; 2) Tree based; 3) Function based; 4) Instance based; 5) Probability based; and 6) Ensemble learning based techniques [28]. Recently, many researchers employed ML techniques from different categories to dynamically detect malware.

In this work, we evaluate performance of representative machine learning techniques from different categories like decision tree based, probability based using a real malware dataset in terms of a variety of performance metrics. Evaluation of ML techniques on a number of metrics is important, because different ML techniques have been designed to optimize a different set of criteria. So, they behave differently in a similar environment. For instance, Mukkanmala et al. 2003[16] proved Support Vector Machine (SVM) designed to minimize the structural risk presents a better generalization ability than NN developed to minimize empirical risk for intrusion detection in terms of accuracy, test time and scalability. So, it may be possible that one technique may show optimal performance on one set of metrics. Whereas, suboptimal performance on another set of metrics.

In this work, we identify the best performing techniques for dynamic malware detection based upon a promising set of features extracted from execution reports of malware and benign samples. Major contributions of this work are:

- Extraction of dynamic behavior of real malware samples from VirusTotal by executing them in a virtually controlled environment of Cuckoo Sandbox.
- Selection of features representing dynamic behavior of malware to generate a real malware dataset.
- Evaluation of ML techniques category wise such

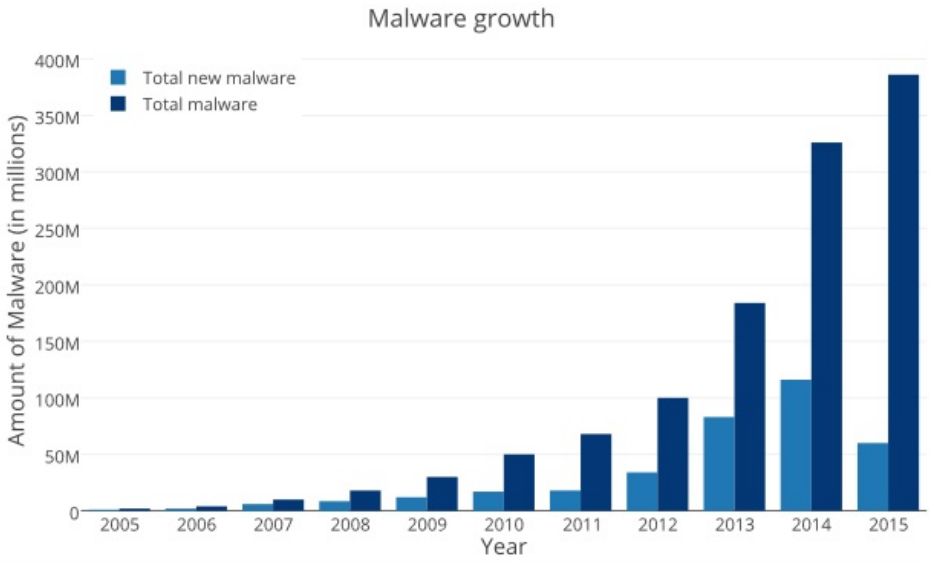

Fig. 1. Trend in malware growth

as decision tree based, probability based techniques, etc. to identify the promising techniques using a real malware dataset.

- Empirical comparative analysis of the promising ML techniques to identify the best performing technique for effective malware detection, so as to use it as a candidate technique for developing dynamic malware detection systems.

So, we evaluate the performance of supervised ML techniques and compared them for malware detection using a real data set from VirusTotal. To the best of our knowledge, this experimental work is the most comprehensive evaluation of ML techniques in terms of number of techniques, variety of metrics and promising set of malware features.

Article overview: Section 2 presents the work related to dynamic malware detection and their comparative evaluation. The methodology adopted in the proposed work consisting data generation phase, data extraction phase, classification phase and performance analysis phase of the proposed work is presented in Section 3. Section 4 presents detailed results of experiments and their discussion. Finally, the paper concludes the proposed work and provides directions for future research in the field in Section 5.

2. Related work

Conventional malware techniques have been proposed for analysis of malware static features derived from its structure without executing them. Here, they follow a reverse engineering approach to detect the malware. Many researchers used API call sequences to analyze behavior of portable executable (PE) code[10,8]. Most of these techniques fail due to use of obfuscation techniques for development of malware. Some of conventional techniques are signature based techniques, which requires a continuous updating of signatures of malware. Updating the database is a time consuming and a challenging task. These techniques can be easily evaded by malware in polymorphic form[15].

The remedy to limitations of static analysis techniques is offered by dynamic analysis techniques for malware detection. Dynamic techniques involve the steps of executing a malware sample, monitoring its behavior, creating and analyzing a profile representing its dynamic behavior[4]. For an unknown sample, its profile is compared to a known profile for finding its similarity and predict family of malware sample. Dynamic techniques involve generation of a profile of malware samples either by analyzing control flows or API calls [10]. Both of these profile generation approaches involve finding the similarity of

test samples with an existing profile.

In the recent years, ML techniques have been the focus of the researchers to classify malware and benign samples. The significant researches in this field are as described in following text. Samra et al. 2007[21] proposed the notion of using effective signals to improve android security, and introduced risk signals combining information about the permissions requested by a mobile application, function category of application as well as what permissions other apps request. They reported detection accuracy of 68.50% for normal applications and 93.38% of malicious programs.

In their static analysis, Huang et al. 2013 [9] evaluated four ML technique, namely AdaBoost, NB, DT48, and SVM for detecting anomalies for malware detection. The reported results indicate that the Naive Bayes technique can detect malware up to 81% of detection accuracy. Whereas, Vyas et al. 2017[26] performed a static analysis of detecting PE on the basis of a small set of features and four supervised machine learning techniques. The authors proved that random forest technique performed better in terms of detection accuracy of 98.6% with a false positive rate of 1.8%. Similarly, Shabtai et al. 2012[22] evaluated six ML techniques, namely DTJ48, NB, BN, k-Means, histogram, and logistic regression to identify the best technique, detecting the malware based upon anomalies. They used feature selection techniques like Chi-square, Fisher score and information gain to select the most promising features. The reported results indicate the achievement of 99.9% detection accuracy, using information gain method for feature selection and decision tree (i.e. J48) technique as a classifier based on a self written malware dataset.

Dini et al. 2012[5] suggested a multi-level anomaly detector system based upon the k-nearest neighbors (KNN) technique. They reported to achieve 93% accuracy for ten malware. However, the proposed system fails to detect malware that contains no system call with root permission. For instance, SMS malware that is invisible in the kernel. On similar lines, SVM ML technique based system has been proposed by Zhao et al. 2012[30] for detecting unknown malware in mobile devices like Gemini, DroidDream and Plankton. They attempted to detect privacy information leakage and hidden payment services.

Rieck et al. 2008[19] used analysis reports created by CWSandBox to create behavioral profiles and train SVM technique to build classifiers for detection of malware families. Narudin et al. 2016[17] evaluated ML techniques for mobile malware detection based upon public as well as private malware dataset. But, they considered a limited number of ML techniques in their work.

Recently, Ye et al. 2017[29] proposed a heterogeneous deep learning framework composed of an AutoEncoder stacked up with multi layer restricted Boltzmann machines (RBMs) and a layer of associative memory to detect newly unknown malware using Windows API calls extracted from PE profiles. The proposed framework consists of two phases: pre-training and fine-tuning. The pre-training phase involves utilization of both labeled and unlabeled file samples to pre-train multiple layers for feature learning. Further, it is followed by construction of an accurate classification model which can differentiate malware from benign files by supervised parameter fine-tuning. The authors performed a comprehensive experimental study on a real and large file collection from Comodo Cloud Security Center to compare various malware detection approaches. The experimental results demonstrated that the proposed method can further improve the overall performance in malware detection compared with traditional shallow learning methods. However, the improved performance is achieved at the cost of increased complexity in the system. Moreover, the authors focused on only API based features for training the proposed method.

The aforementioned text demonstrates the use of applying ML techniques for effective malware detection. But, the prior evaluation related work is limited to a certain amount of malware samples, or a few numbers of ML techniques or focus on a specific set of features like API calls based features. Different researchers evaluated different ML techniques in their studies based upon different malware datasets. So, the reported results cannot be critically analyzed to identify promising technique for

effective malware detection. This work evaluates ML techniques from different categories belonging to decision trees, probability categories using a common platform of same and similar preprocessed real malware dataset.

The major motivation behind the proposed work is to evaluate a sufficient number of ML techniques to identify the techniques that achieve better results in terms of detection rate, and false positive rate using a real malware dataset.

3. Methodology

This section presents the overall methodology followed in this work as depicted in Figure 2. It includes dynamic malware detection using ML technique consisting of data generation phase, data extraction phase, classification phase, and performance metric computation phase. The data generation phase executes the benign and malware PE in a controlled environment of Cuckoo sandbox and produces its execution report in the form of a JSON file. The data extraction phase extracts features from JSON files that represents the dynamic behavior of samples and labels each sample as benign or malware. It generates a real malware dataset that is further used as training and test dataset by the classification phase. The performance metric computation phase computes malware detection results in terms of variety of metrics.

The detailed description of these phases is described in following subsections.

3.1. Data generation phase

Many systems have been proposed for observing the dynamic behavior of PEs, such as CWSandbox[27], Anubis[2], Cuckoo sandbox[20] etc. These environments allow the execution of malware and benign binaries within an isolated environment, analyze and record their behavior. Unfortunately, CWSandbox and MIST[24] are not open source. Whereas, Cuckoo sandbox[20] is an open source sandbox environment. So, it is selected for the proposed work. Here, we set up a virtual environment of the Cuckoo Sandbox to execute malware and benign binaries received from VirusTotal[25]. As an output of the Cuckoo Sandbox setup, we collected the JSON reports for the dynamic behavior of executing binaries. For the proposed work, we used a Linux based PC with Core i3-2330M 2.20 GHz CPU and 2 GB RAM and employed WEKA implemented ML techniques.

3.2. Data extraction phase

The Cuckoo sandbox is configured to execute PEs and produce the output in the form of JSON report. These reports contain behavior of programs executed in a controlled environment of Cuckoo sandbox. For extracting features representing dynamic behavior of executing programs from JSON reports, a Python language based automated system has been developed. The main steps for data extraction phase are as below:

1. Read sections of JSON file
2. Extract features including
   - Basic features such as duration of execution of PEs
   - Network features based upon protocol headers
   - CPU and memory usage features
   - Statistics of APIs based upon their categories
   - Statistics of file system activities in terms of number of files written, deleted, read, commands executed, services started, services created
   - Count of APIs in each category like network APIs, Registry APIs etc.
   - Number of sub processes generated
3. Labeling of malware samples

The most critical task in malware analysis using supervised ML techniques is the labeling of samples as malicious and benign. In Cuckoo sandbox, JSON reports contain malware detection results of a number of ant viruses. However, detection results of different ant viruses are not consistent with each other. Most of the researchers used the labeling detected by Kaspersky antivirus [10]. So, in the proposed work, we used the labeling of malware samples as

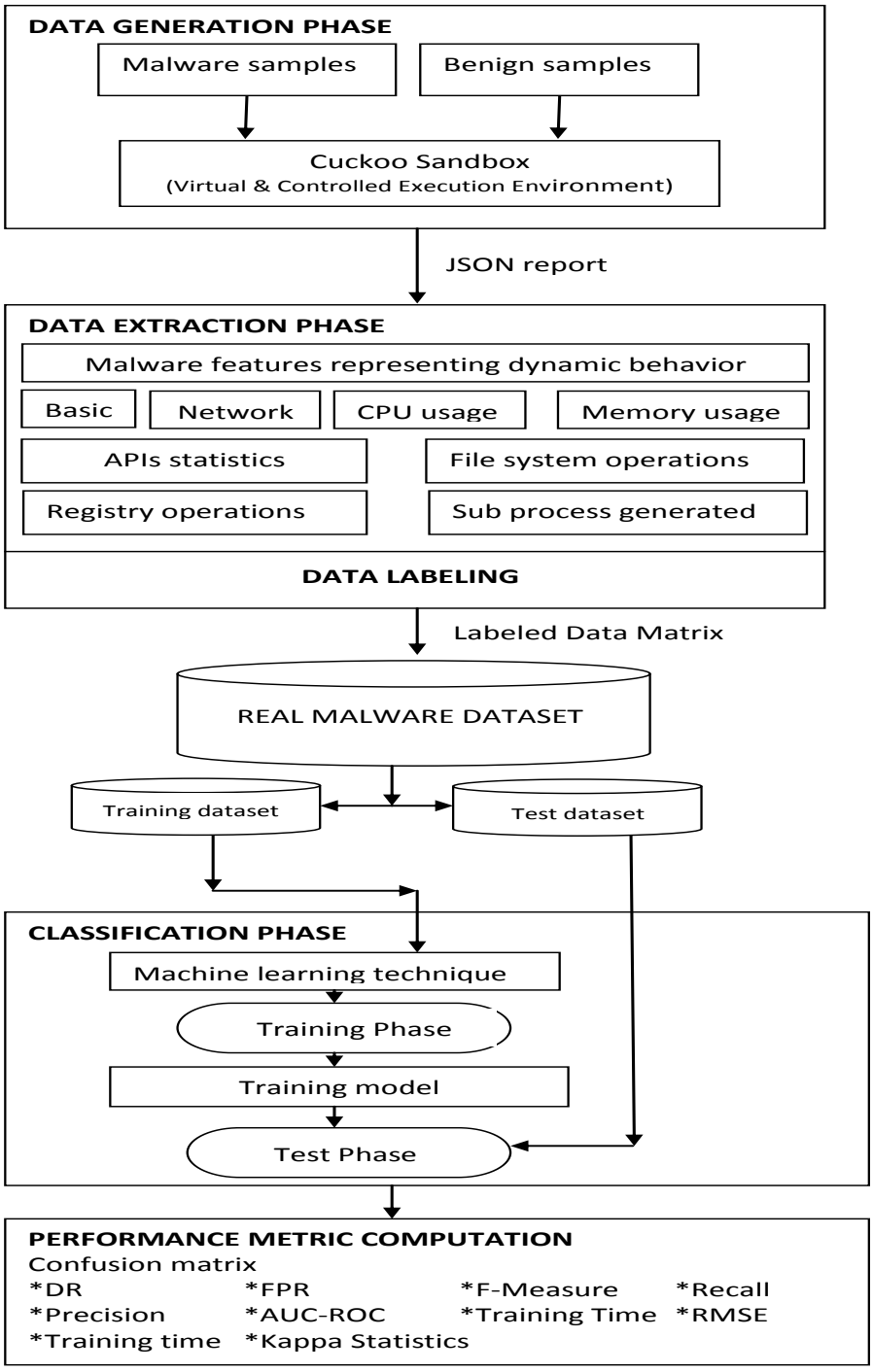

Fig. 2. Schematic flow of the proposed work.

reported by Kaspersky antivirus during their execution in Cuckoo environment.

For dynamic analysis of malware, we extracted a large number of raw features from JSON reports produced in Cuckoo environment representing the dynamic behavior of malware. All the raw features of the data are not important to understand it. However, the higher dimensions of the data suffer from a problem called curse of dimensionality[3]. In addition, high dimensional data require more computational overhead and leads to delay in detection of malware, which is not desirable. In order to tackle this difficulty of analyzing high dimensional data, we identified a filter based feature selection technique proposed by Kumar & Kumar 2012[13] keeping the benefits of using mutual information and entropy concepts to compute the relevance of features in the feature selection process into consideration. As mutual information is an information metric used to measure the relevance of features taking into account higher order statistical structures existing in the data. After applying the identified mutual information based feature selection ITFS technique[13], we selected promising features, among raw features extracted from JSON files as depicted in Table 1. Here, feature titled API name and its frequency give the name of API call and the number of times it is called during execution of a sample.

Further, this phase transforms the features from JSON files into a CSV format for analysis of malware in the classification phase by ML technique and applies the identified feature selection technique to select promising features. Each record in the CSV file of selected features is a sequence of 41 features labeled with malware family as detected by Kaspersky.

In this work, we selected 41 features and 01 features as a family name of malware. The description of selected features is as depicted in Table 2. Finally, records are shuffled using a pre-processor offered by WEKA to ensure data randomization while splitting of malware dataset into training and test dataset for experimental validity. The randomization of the dataset is done using an unsupervised instance filter implemented in WEKA.

Table 1. Raw features extracted from JSON reports

Category	Feature	Data type
Info	Duration	Integer
Network	UDP_requests	Integer
	IRC_requests	Integer
	http_requests	Integer
	smtp_requests	Integer
	tcp_requests	Integer
	hosts_contacted	Integer
	DNS_requests	Integer
	domains_contacted	Integer
	ICMP_requests	Integer
Usage	CPU_usage	Integer
	mem_usage	Integer
Dropped	Dropped	Integer
API categories	Noti_API	Integer
	Certi_API	Integer
	Crypto_API	Integer
	exception_API	Integer
	file_API	Integer
	iexplore_API	Integer
	misc_API	Integer
	netapi_API	Integer
	network_API	Integer
	ole_API	Integer
	process_API	Integer
	registry_API	Integer
	resource_API	Integer
	services_API	Integer
	Syn_API	Integer
	system_API	Integer
	ui_API	Integer
	other_API	Integer
API summaries	files_accessed	Integer
	files_written	Integer
	files_deleted	Integer
	Mutexes	Integer
	executed_cmds	Integer
	started_services	Integer
	files_read	Integer
	resolved_APIs	Integer
	created_services	Integer
Processes	processes_generated	Integer
API name and its frequency	API (321)	Integer
Malware Family	Family	Categorical

Table 2. Selected Features

Category	Feature	Data type
Info	Duration	Integer
Network	UDP_requests	Integer
	IRC_requests	Integer
	http_requests	Integer
	smtp_requests	Integer
	tcp_requests	Integer
	hosts_contacted	Integer
	DNS_requests	Integer
	domains_contacted	Integer
	ICMP_requests	Integer
Usage	CPU_usage	Integer
	mem_usage	Integer
Dropped	Dropped	Integer
Processes	processes_generated	Integer
API categories	Noti_API	Integer
	Certi_API	Integer
	Crypto_API	Integer
	exception_API	Integer
	file_API	Integer
	iexplore_API	Integer
	misc_API	Integer
	netapi_API	Integer
	network_API	Integer
	ole_API	Integer
	process_API	Integer
	registry_API	Integer
	resource_API	Integer
	services_API	Integer
	Syn_API	Integer
	system_API	Integer
	ui_API	Integer
	other_API	Integer
API summaries	files_accessed	Integer
	files_written	Integer
	files_deleted	Integer
	Mutexes	Integer
	executed_cmds	Integer
	started_services	Integer
	files_read	Integer
	resolved_APIs	Integer
	created_services	Integer
Malware Family	Family	Categorical

3.3. Classification phase

A large number of supervised ML techniques have been designed for classifying malware dataset into a set of malware categories. For instance, Artificial Neural Networks are designed to mimic the human brain. They have the capability to learn any non-linear relationship between input and desired output even in the presence of noisy training data. Interested readers may explore review of ML techniques mentioned in the studies[11,14,28]. The ML techniques from different categories implemented in ML tool WEKA are used to produce trained models for malware dataset having bifurcation of 70% as training and 30% test dataset. In the present work, we have been using default parameters of different ML techniques implemented in WEKA. However, fine tuning of the parameters may lead in further improvement of classification results of ML techniques. The trained model of ML technique is further used to predict malware family of unknown samples. The output of this phase is a report consisting of confusion matrix and other details. The generated report is used by the security experts for further compute other performance metrics and derive policy decisions.

3.4. Performance metric computation phase

The performance metric computation phase calculates the identified performance metrics from the confusion matrix after testing phase. The confusion matrix provides the values of False Positives (FP), True Negatives (TN), False Negatives (FN), and True Positives (TP). It derives the weighted average of identified performance metrics like TPR (also known as Recall), FPR, RMSE, Detection Accuracy, Precision, F-measure, AUC-ROC from the values of TN, FN, TP, and TN for comparative evaluation of the ML techniques. We used a weighted average of different metrics for deriving the metrics like AUC-ROC by following one vs rest approach.

4. Experiments and results

This section presents details of malware dataset and comparative performance evaluation of various ML

techniques in terms of identified metrics.

4.1. Evaluation metrics

Here, we are focused to compute confusion matrix and derive different performance metrics namely: Accuracy, True Positive Rate (TPR), False Positive Rate (FPR), Precision, F-measure, AUC-ROC in addition to training time, root mean square error, and kappa statistics to evaluate the performance of the ML techniques [12]. The metrics can be derived from values of FP, FN, TN, and TP as depicted in Table 3.

Table 3. Performance metrics

Performance metric	Expression
Accuracy	(TP+TN)/(TP+TN+FP+FN)
False Positive rate(FPR)	FP/(FP+TN)
True Positive rate(TPR)	TP/(TP+FN)
Precision	TP/(TP+FP)
F-measure	2*TP/(2*TP+FP+FN)

Here, TPR (also known as Recall) gives the weighted average value of predicted malware classified correctly, whereas FPR indicates the weighted average value of benign data incorrectly predicted as malware. Precision gives the rate of relevant results rather than irrelevant results. Whereas, Recall provides the sensitivity for the most relevant results. F-Measure is the weighted average value that estimates the entire system performance by representing precision and recall into a single number. Similarly, AUC-ROC is also a weighted average of each class by following one vs rest approach. These results are provided by WEKA. For details of computation of results, interested readers may refer to WEKA manual [28].

4.2. Evaluation malware dataset

In this work, we used the malware and benign samples from VirusTotal[25]. The VirusTotal is a web site that offers the analysis of suspicious files and URLs to detect types of malware including viruses, worms, and trojans. VirusTotal aggregates many antivirus products and online scan engines to check for viruses that the user's own antivirus may have missed, or to verify against any false positives. Files up to 256 MB can be uploaded or sent via email to the website. Antivirus software vendors can receive copies of files that were flagged by other scans, but passed by their own engine, to improve their software and, by extension, VirusTotal's own capability. Users can also scan suspect URLs and search through the VirusTotal dataset. VirusTotal uses Cuckoo sandbox for dynamic analysis of malware. We extracted 13,236 malware and benign samples randomly from the malware dataset of the VirusTotal[25]. The malware and benign are executed for extracting features as described in Section 3.1 and 3.2. In resulting dataset, a large number of malware families were found. For evaluation purpose of ML techniques, we categorized malware samples into different families, as per their basic functions. The similar categorization is also reported by Ki et al. 2015[10]. For uniform and comprehensive analysis of the proposed work, malware dataset is divided randomly into training and test dataset. The training dataset contains 70% of samples and test dataset contains 30% samples. The category wise number of samples in the training data set and test data set are as described in Table 4.

4.3. Results

We employed the supervised ML techniques from different categories to select the most promising ML technique using a real malware dataset with the help of ML tool WEKA[28]. The details of performance metrics and malware dataset are as described in Section 4.1 and Section 4.2 respectively. It is observed that different techniques took different training time for obtaining a trained model as depicted in Figure 3. We computed the confusion matrix for the test results based upon malware test dataset. From confusion matrix, rests of identified metrics and weighted averages are derived as summarized in Table 5. WEKA [28] provides values of computed as well as derived metrics. From the reporting results, it can be clearly noticed that the instance based technique called IBk have shown better performance than the other techniques in terms of identified performance metrics. Each class of malware data set is analyzed based upon best identified ML technique IBk as shown in Table 6.

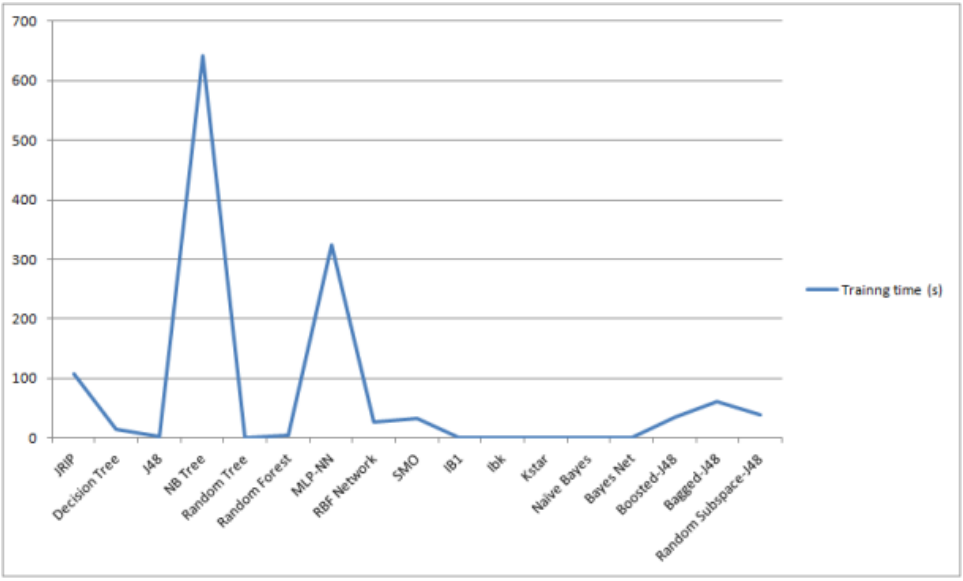

Fig. 3. Training time for ML techniques.

Table 4. Categories and number of samples in dataset

Sr No	Class name	Number of samples
Training dataset		
1	Benign	3433
2	Trojan	4447
3	Virus	265
4	Worm	326
5	Packed	430
6	Backdoor	233
7	Hoax	41
8	DangerousObject	28
9	Adware	62
	Total	9265
Test dataset		
1	Benign	1451
2	Trojan	1889
3	Virus	115
4	Worm	140
5	Packed	206
6	Backdoor	109
7	Hoax	20
8	DangerousObject	13
9	Adware	28
	Total	3971

The result comparison of ML techniques is graphically represented in terms of performance metrics as shown in Figs. 4 to 10.

4.4. Discussion

This section explains the reporting results of the proposed work. Better results to detect a particular malware type indicate its probable use in malware detection systems to pre-filter newly registered malware instances. It should be noticed that future work will be devoted to optimize the technique such that a pre-filtering system can be developed to identify novel malware samples and sort out legacy malware that have minor changes. Figure 3 indicates that different techniques take different training time to build the prediction model. Because, different techniques are designed to optimize the criteria involving different levels of computational overhead.

It can be concluded from the Table 6 that in most of the malware classes, the IBk technique has reported 100% performance in terms of TPR, precision, F-Measure and Area Under Curve ROC with FPR equal to 0%. However, in case of a Packed class of malware, it reported the results close to 100% with 0%. Low detection results for some of malware classes may be due to an imbalance in sam-

Table 5. Performance evaluation of ML techniques

Category	ML technique	RMSE	Kappa	Accuracy	TPR	FPR	Precision	F-Measure	AUC-ROC
Rule based	JRIP	0.148	0.818	0.888	0.888	0.086	0.891	0.884	0.926
	Decision Tree	0.197	0.778	0.861	0.862	0.087	0.869	0.859	0.955
Tree based	J48	0.076	0.95	0.968	0.969	0.016	0.968	0.968	0.995
	NB Tree	0.09	0.946	0.966	0.966	0.023	0.966	0.966	0.944
	Random Tree	0.004	0.999	0.999	0.992	0.014	0.932	0.991	0.99
Function based	MLP-NN	0.191	0.615	0.769	0.769	0.162	0.755	0.752	0.902
	RBF Network	0.229	0.432	0.664	0.664	0.253	0.669	0.639	0.806
	SMO	0.288	0.414	0.662	0.663	0.258	0.617	0.618	0.73
Instance based	IB1	0.151	0.835	0.897	0.897	0.055	0.896	0.897	0.921
	IBk	0.005	0.999	0.999	1	0	1	1	1
	Kstar	0.032	0.992	0.994	0.995	0.003	0.995	0.995	1
Probability based	Naive Bayes	0. 372	0.218	0.318	0.319	0.04	0.75	0.375	0.679
	Bayes Net	0.247	0.562	0.68	0.68	0.043	0.85	0.738	0.937
Ensemble learning	Boosted-J48	0.005	0.999	0.997	1	0	1	1	1
	Bagged-J48	0.068	0.968	0.98	0.98	0.015	0.98	0.98	0.998
	Random Subspace	0.065	0.968	0.979	0.98	0.015	0.98	0.98	0.997
	Random Forest	0.0346	0.997	0.998	0.998	0.001	0.998	0.998	1

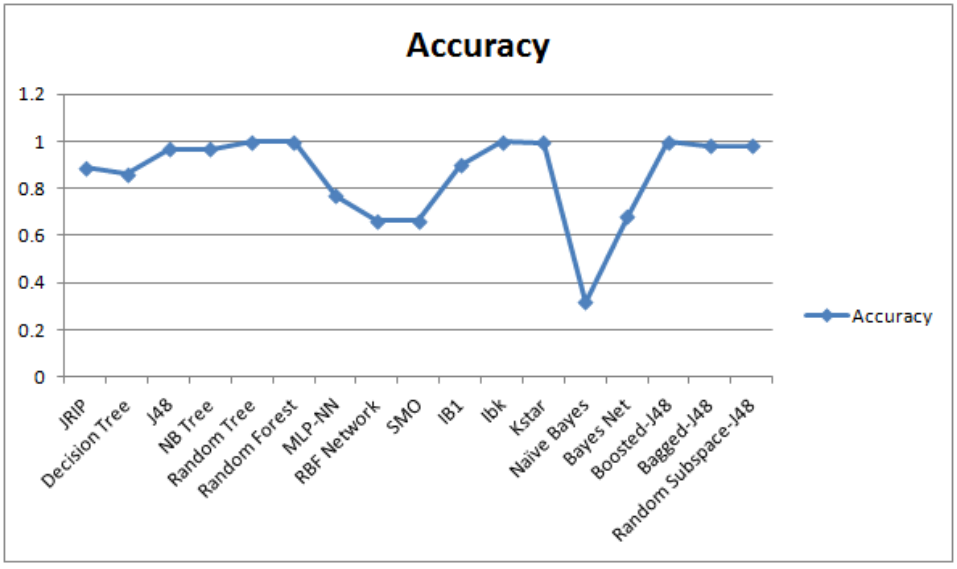

Fig. 4. Performance comparison of ML techniques in terms of accuracy.

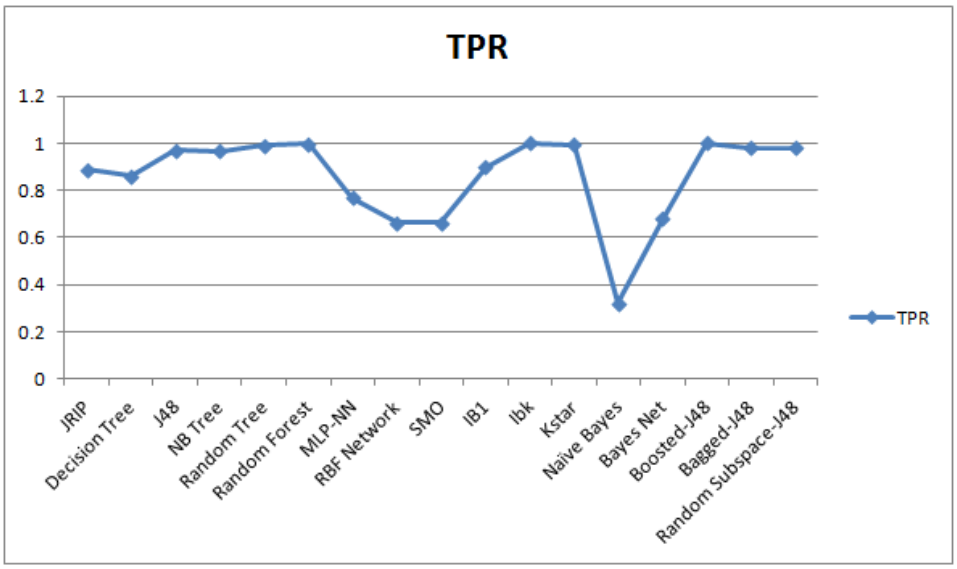

Fig. 5. Performance comparison of ML techniques in terms of TPR.

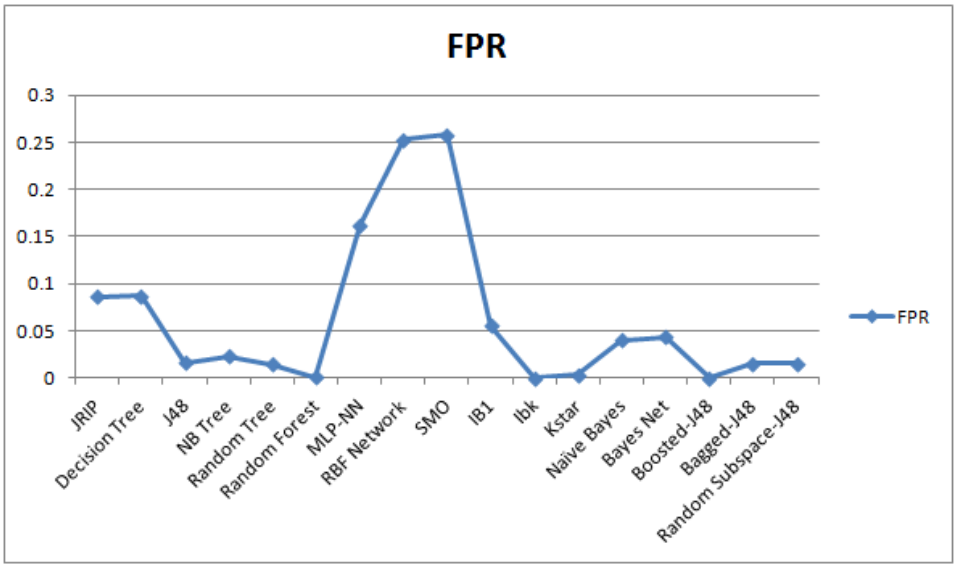

Fig. 6. Performance comparison of ML techniques in terms of FPR.

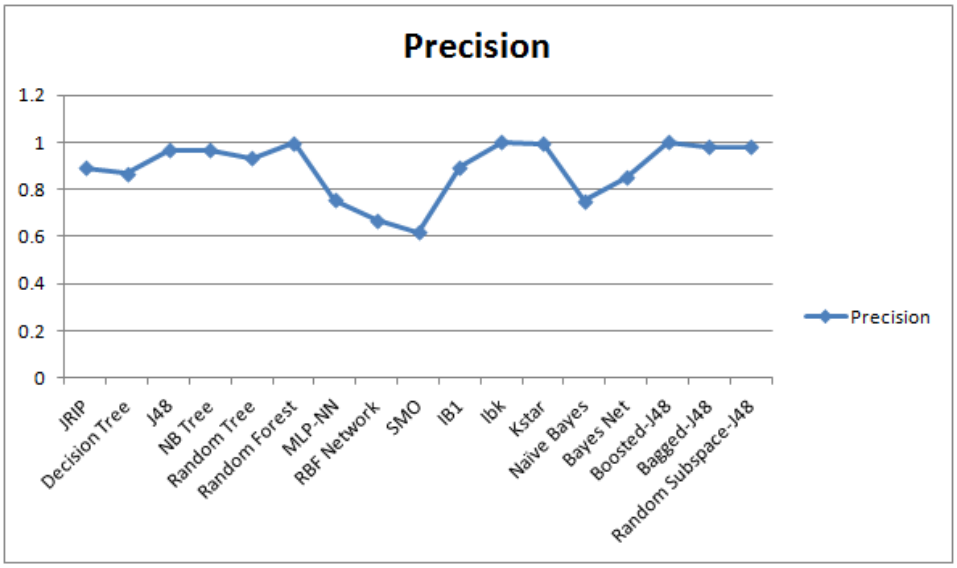

Fig. 7. Performance comparison of ML techniques in terms of Precision.

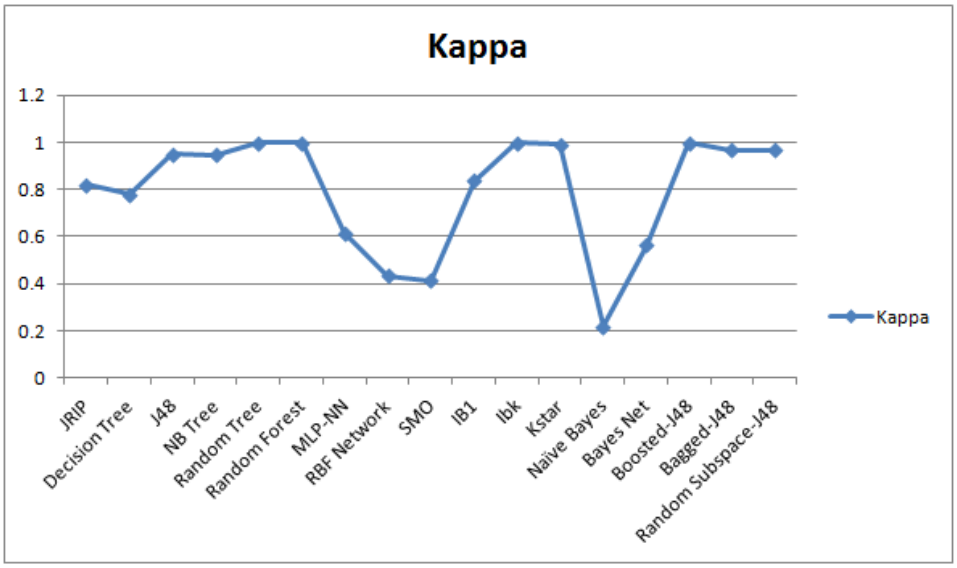

Fig. 8. Performance comparison of ML techniques in terms of kappa statistics.

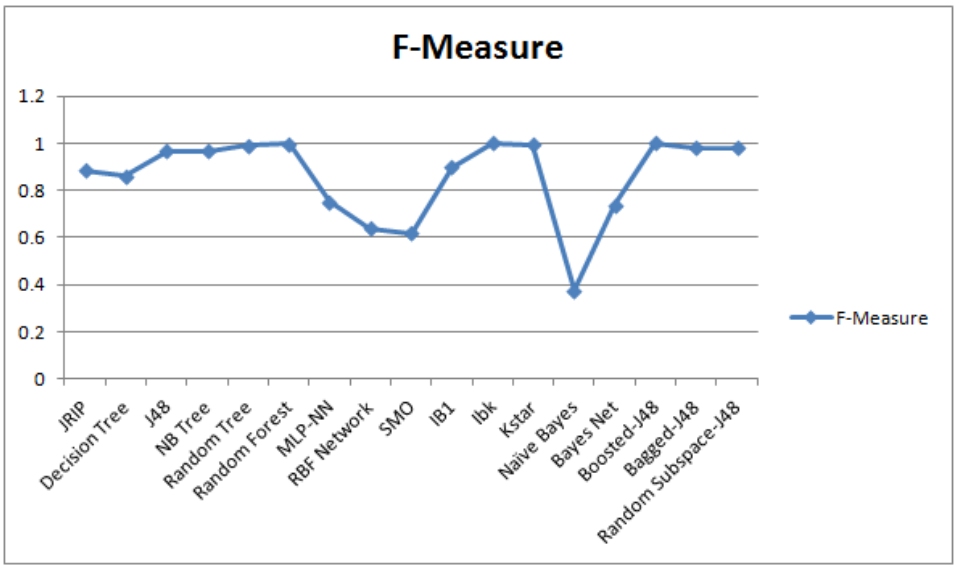

Fig. 9. Performance comparison of ML techniques in terms of F-Measure.

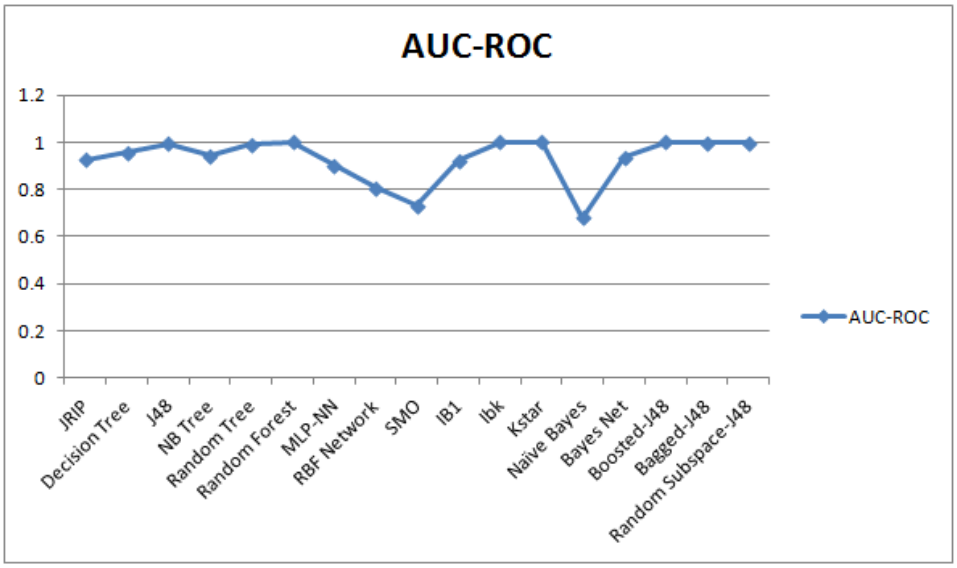

Fig. 10. Performance comparison of ML techniques in terms of AUC-ROC.

Table 6. Performance of IBk ML technique for each class

Class	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area
Benign	1	0	0.999	1	1	1
Trojan	1	0	1	1	1	1
Virus	1	0	1	1	1	1
Worms	1	0	1	1	1	1
Packed	0.995	0	1	0.995	0.998	1
Backdoor	1	0	1	1	1	1
Hoax	1	0	1	1	1	1
DangerousObject	1	0	1	1	1	1
Adware	1	0	1	1	1	1
Weighted Avg.	1	0	1	1	1	1

pling of malware dataset or lack of sufficient amount of training data. The imbalance in samples of malware dataset generally leads to biasing of the IBk technique.

It can also be concluded from the reporting results that ensemble based techniques have shown better performance for malware detection. It is in line with theoretical as well, practically proved facts about ensemble techniques that they generally outperform the individual techniques. Here, the IBk technique has also shown comparable performance with the ensemble of J48 technique using boosting. It can be considered as a candidate ML technique for building an effective malware detection system.

Tree structure based J48 ML techniques reported the second highest results in terms of TPR of 96.9% and FPR of 0.016%. The reported ROC area of 99.5% indicates its better performance for malware detection. The reporting results are satisfactory for malware detection, so can be considered as a candidate ML technique for building an effective malware detection system.

Furthermore, these techniques can also be considered to formulate base classifiers for designing ensembles for effective malware detection in real systems.

5. Conclusions and future work

In this work, an evaluation of supervised ML techniques is done empirically for detecting malware using a real malware dataset in terms of a variety of evaluation metrics. The major motivation behind using a variety of evaluation metrics is that different techniques are developed to optimize different set of criteria. To evaluate ML techniques comprehensively, a promising set of features has been extracted from malware and benign executable samples using a Cuckoo Sandbox and a Python based automated system to form a real malware dataset.

In this work, we identify the best techniques for effective malware detection based upon a real malware dataset in terms of identified performance metrics. The proposed work contributed for extracting the dynamic behavior of real malware samples from VirusTotal by executing them in a virtual and controlled environment, transforming the dynamic behavior into a matrix having a promising feature set to design a malware dataset, and identifying the promising techniques for each malware family. The reporting results indicate that Instance based IBk and Tree Based J48 ML techniques has shown the results of TPR close to 100% and FPR close to 0%. So, these techniques can be considered as a candidate technique for building an effective malware detection system for novel and known malware samples dynamically. Moreover, the reported performance of these techniques is also comparable to that of their ensemble techniques using a boosting algorithm. This comparable performance validates the set of novel features extracted from the execution reports for effective malware detection. The proposed

features resulted a fast and effective malware detection by processing a small number of features.

To the best of our knowledge, this experimental work is a comprehensive evaluation of ML techniques in terms of number of techniques and number of metrics for detecting malware. Empirical comparative analysis of the promising ML techniques is done to identify the overall best performing techniques for effective malware detection, so as to use them as candidate techniques for detecting novel and known malware samples in the future dynamic malware detection systems. Our future research work will focus on more experiments using real malware datasets and design ensemble techniques based on identified promising techniques as base classifiers for effective malware detection.

References

1. SF Ahmad, SZ Ahmad, SR Xu, and B Li. Next generation malware analysis techniques and tools. In *Electronics, Information Technology and Intellectualization: Proceedings of the International Conference EITI 2014, Shenzhen, China, 16-17 August 2014*, page 17. CRC Press, 2015.
2. Ulrich Bayer, Andreas Moser, Christopher Kruegel, and Engin Kirda. Dynamic analysis of malicious code. *Journal in Computer Virology*, 2(1):67–77, 2006.
3. R. Bellman. Adaptive control processes: a guided tour princeton university press. *Princeton, New Jersey, USA*, 1961.
4. Silvio Cesare and Yang Xiang. *Software similarity and classification*. Springer Science & Business Media, 2012.
5. Gianluca Dini, Fabio Martinelli, Andrea Saracino, and Daniele Sgandurra. Madam: a multi-level anomaly detector for android malware. In *International Conference on Mathematical Methods, Models, and Architectures for Computer Network Security*, pages 240–253. Springer, 2012.
6. Manuel Egele, Theodoor Scholte, Engin Kirda, and Christopher Kruegel. A survey on automated dynamic malware-analysis techniques and tools. *ACM Computing Surveys (CSUR)*, 44(2):6, 2012.
7. Christian Gorecki, Felix C Freiling, Marc Kührer, and Thorsten Holz. Trumanbox: Improving dynamic malware analysis by emulating the internet. In *Stabilization, Safety, and Security of Distributed Systems*, pages 208–222. Springer, 2011.
8. Kent Griffin, Scott Schneider, Xin Hu, and Tzi-Cker Chiueh. Automatic generation of string signatures for malware detection. In *Recent advances in intrusion detection*, pages 101–120. Springer, 2009.
9. Chun-Ying Huang, Yi-Ting Tsai, and Chung-Han Hsu. Performance evaluation on permission-based detection for android malware. In *Advances in Intelligent Systems and Applications-Volume 2*, pages 111–120. Springer, 2013.
10. Youngjoon Ki, Eunjin Kim, and Huy Kang Kim. A novel approach to detect malware based on api call sequence analysis. *International Journal of Distributed Sensor Networks*, 2015:4, 2015.
11. Sotiris B Kotsiantis, Ioannis D Zaharakis, and Panayiotis E Pintelas. Machine learning: a review of classification and combining techniques. *Artificial Intelligence Review*, 26(3):159–190, 2006.
12. G. Kumar and K. Kumar. Ai based supervised classifiers: an analysis for intrusion detection. In *Proc. of International Conference on Advances in Computing and Artificial Intelligence*, pages 170–174. ACM, 2011.
13. G. Kumar and K. Kumar. An information theoretic approach for feature selection. *Security and Communication Networks*, 5(2):178–185, 2012.
14. G. Kumar, K. Kumar, and M. Sachdeva. The use of artificial intelligence based techniques for intrusion detection: a review. *Artificial Intelligence Review*, 34(4):369–387, 2010.
15. Andreas Moser, Christopher Kruegel, and Engin Kirda. Limits of static analysis for malware detection. In *Computer security applications conference, 2007. ACSAC 2007. Twenty-third annual*, pages 421–430. IEEE, 2007.
16. S. Mukkamala and A.H. Sung. A comparative study of techniques for intrusion detection. In *Proc. of 15th IEEE International Conference on Tools with Artificial Intelligence, 2003*, pages 570–577. IEEE, 2003.
17. Fairuz Amalina Narudin, Ali Feizollah, Nor Badrul Anuar, and Abdullah Gani. Evaluation of machine learning classifiers for mobile malware detection. *Soft Computing*, 20(1):343–357, 2016.
18. Philip O'Kane, Sakir Sezer, and Keiran McLaughlin. Obfuscation: The hidden malware. *Security & Privacy, IEEE*, 9(5):41–47, 2011.
19. Konrad Rieck, Thorsten Holz, Carsten Willems, Patrick Düssel, and Pavel Laskov. Learning and classification of malware behavior. In *Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 108–125. Springer, 2008.
20. Cuckoo Sandbox. Automated malware analysis, 2013.
21. Bhaskar Pratim Sarma, Ninghui Li, Chris Gates, Rahul Potharaju, Cristina Nita-Rotaru, and Ian Molloy. Android permissions: a perspective combining risks and benefits. In *Proceedings of the 17th ACM symposium on Access Control Models and Technologies*, pages 13–22. ACM, 2012.

22. Asaf Shabtai, Uri Kanonov, Yuval Elovici, Chanan Glezer, and Yael Weiss. andromaly: a behavioral malware detection framework for android devices. *Journal of Intelligent Information Systems*, 38(1):161–190, 2012.
23. Monirul Sharif, Vinod Yegneswaran, Hassen Saidi, Phillip Porras, and Wenke Lee. Eureka: A framework for enabling static malware analysis. In *Computer security-ESORICS 2008*, pages 481–500. Springer, 2008.
24. Philipp Trinius, Carsten Willems, Thorsten Holz, and Konrad Rieck. A malware instruction set for behavior-based analysis. 2009.
25. VirusTotal. Virustotal-free online virus, malware and url scanner., 2016.
26. Rushabh Vyas, Xiao Luo, Nichole McFarland, and Connie Justice. Investigation of malicious portable executable file detection on the network using supervised learning techniques. In *Integrated Network and Service Management (IM), 2017 IFIP/IEEE Symposium on*, pages 941–946. IEEE, 2017.
27. Carsten Willems, Thorsten Holz, and Felix Freiling. Cwsandbox: Towards automated dynamic binary analysis. *IEEE Security and Privacy*, 5(2):32–39, 2007.
28. I.H. Witten, E. Frank, and M.A. Hall. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2011.
29. Yanfang Ye, Lingwei Chen, Shifu Hou, William Hardy, and Xin Li. Deepam: a heterogeneous deep learning framework for intelligent malware detection. *Knowledge and Information Systems*, pages 1–21, 2017.
30. Min Zhao, Tao Zhang, Fangbin Ge, and Zhijian Yuan. Robotdroid: a lightweight malware detection framework on smartphones. *Journal of Networks*, 7(4):715–722, 2012.