Indian Journal of Science and Technology, *Vol 10(21), DOI: 10.17485/ijst/2017/v10i21/100023, June 2017*

ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Malware Classification Framework for Dynamic Analysis using Information Theory

Ehsan Moshiri*, Azizol Bin Abdullah, Raja Azlina Binti Raja Mahmood and Zaiton Muda

Computer Science and Information Technology Department (CSIT), Universiti Putra, Malaysia (UPM);
ootkit.py@gmail.com, azizol@upm.edu.my, raja_azlina@upm.edu.my, zaitonm@upm.edu.my

Abstract

Objectives: 1. To propose a framework for Malware Classification System (MCS) to analyze malware behavior dynamically using a concept of information theory and a machine learning technique. 2. To extract behavioral patterns from execution reports of malware in terms of its features and generates a data repository. 3. To select the most promising features using information theory based concepts. **Methods/Statistical Analysis**: Today, malware is a major concern of computer security experts. Variety and in- creasing number of malware affects millions of systems in the form of viruses, worms, Trojans etc. Many techniques have been proposed to analyze the malware to its class accurately. Some of analysis techniques analyzed malware based upon its structure, code flow, etc. without executing it (called static analysis), whereas other techniques (termed as dynamic analysis) focused to monitor the behavior of malware by executing it and comparing it with known malware behavior. Dynamic analysis has proved to be effective in malware detection as behavior is more difficult to mask while executing than its underlying code (static analysis). In this study, we propose a framework for Malware Classification System (MCS) to analyze malware behavior dynamically using a concept of information theory and a machine learning technique. The proposed framework extracts behavioral patterns from execution reports of malware in terms of its features and generates a data repository. Further, it selects the most promising features using information theory based concepts. **Findings:** The proposed framework detects the family of unknown malware samples after training of a classifier from malware data repository. We validated the applicability of the proposed framework by comparing with the other dynamic malware analysis technique on a real malware dataset from Virus Total. **Application:** The proposed framework is a Malware Classification System (MCS) to analyze malware behavior dynamically using a concept of information theory and a machine learning technique.

Keywords: Information Theory, Malware Classification, Mutual Information, Neural Network

1. Introduction

Despite of numerous anti malware software, number of known and unknown malware incidents are increasing day by day. Now days, detection of malware is the focus of current research community in the field of computer security. However, malware analysis is labor oriented and time consuming task. Moreover, it does not scale well with the ever increasing prevalence of malware[1]. Various techniques have been proposed for detection and analysis of malware.

The important detection techniques include traditional signature based techniques, and dynamic behavior based tech- niques. Detection techniques further use analysis techniques to observe a malware and its intention[2]. Researchers proposed three types of approaches for analysis of malware that includes: static analysis, dynamic analysis and hybrid analysis. Static analysis works by analyzing malware based upon its structure, control flow, etc. without executing it[3]. Moreover, static analysis dependent on a pre-established signature database, it is unable to detect novel malware until the signature database is updated[4–6]. Whereas in dynamic analysis, the malware is observed for its behavior by executing it in a controlled environment. During dynamic analysis, reports are gener-

**Author for correspondence*

ated to conclude about the in- tent of malware based upon record behavior of malware like sequence of Application Programming Interface API. As malware authors are designing more complicated and sophisticated malware using obfuscation and encryption techniques, its static analysis is becoming very hard and unable to classify its behavior. The problem is solved by dynamic analysis by recording the behavior of malware by executing it in a controlled environment. So, dynamic analysis has a potential of providing more accurate results than static analysis and is wildly used for achieving more accurate malware detection. Keeping in view the better results of malware analysis, researchers have shifted their focus from traditional static based methods[3,7] to more dynamic and automatized methods for malware analysis[2,8]. Dynamic malware analyses are generally based on collecting malware behavior traces during execution of malware[9].

In this work, we propose a framework for dynamic analysis of malware based upon information theory concepts. The framework performs malware analysis in three phases namely: Feature Extraction, Feature Selection and Malware Classification. The working of the proposed framework is validated using a real time collection of dataset and by comparing it with representative techniques in the field of malware analysis.

Article overview: Section 2 presents the work related to malware analysis. A critical review of the state of art in the field is presented for better understanding and current trends of the field. The methodology adopted in the proposed work consisting of experimental setup, stages in the proposed works, malware dataset description is presented in Section 3. Section 4 highlights the proposed framework, and it's working. All the modules of the framework are described here. Section 5 presents the details of the results of experiments. The discussion of results is presented for proving its validity to the field. Finally, the paper concludes the framework and provides clues for future research in the field of malware analysis in Section 6.

Most of the traditional malware analysis and classification systems are based on static features. The static features are extracted from executable of malware without executing them. These features are generally extracted by using reverse-engineering methods. Many researchers proposed to detect malware by analyzing the API call sequence[6,10]. They believe that API call sequences are related to the behavior of the Portable Executable (PE) code in win- dows environment. However, the extracted information in terms of API sequences for PE files has been static. It may not properly represent the actual behavior of the PE code. So, these approaches can easily get failed by obfuscation techniques. Most of approaches proposed in early stages of malware analysis are signature based approaches. How- ever, these approaches have many weaknesses. The major weakness includes the continuous updating of signatures of malware that is a laborious, time consuming and a challenging task. Moreover, these approaches can be easily evaded by malware in polymorphic form[3].

To meet the limitations of these approaches, researchers shifted their focus to dynamic analysis of malware. Dynamic analysis of malware includes the execution of malware, monitors its behavior, and generates a profile. It detects the unknown malware by computing it's similarly of the known profile of malware[4]. Dynamic analysis of malware is either based upon control flow analysis or API call analysis[6]. Both techniques compare the behavior of malware by analyzing the similarity.

Recently, many researchers have been proposed to analyze API call sequences for their behavior. Some of them have used the API calls and their frequency[11], whereas other studies focused to mine API call sequence for each malware class[12]. Recent studies centered that low level system calls remain unchanged until function or intent of malware is changed[6].

Lee and Mody represented malware samples with sequences of system calls and proposed to use string edit distance to classify them[13]. Whereas defined the behavior of malware in terms of not-transient state changes that malware causes on the system and apply Normalized Compression Distance (NCD) as a similarity mea- sure for classifying malware samples[14]. Rieck et al. used the information contained in the analysis reports created by CWSandBox to generate behavioral profiles and train Support Vector Machines (SVM) to build classifiers for malware families[15]. Most of the work on malware classification discriminates between malicious and benign exe- cutables. In contrast to these works, we aim to discriminate between different malware families and classify samples into their respected families.

Another focus of researchers is clustering of malware samples into groups with similar behavior[15,16]. They proposed to cluster the malware into groups and detect new families of malware by comparing its similarity with existing clusters. However, malware representation is a challenging task in clusters based techniques.

Some of the researchers have focused on network behavior of the malware[17]. They proposed to analyze network traces in the form of pcap files to extract the network flow information. Further, they proposed to represent network flow information graphically and extracted network behavior based features. Unknown samples of malware are classified based upon trained classifier using labeled dataset of network features.

Recently, it is proposed an approach to detect malware based on API call sequence analysis[6]. They utilized DNA sequence alignment algorithm along with Longest Common Sequence (LCS) concept to find the similarities among the patterns. They compared the results with that of representative techniques in the field. They reported better results than others techniques. However, they excluded benign LCSs from their database to reduce the false positives. It may be the fact behind their reporting of better results.

However, most of the researchers utilized one subset of features of malware to represent its behavior pattern and ignored other ones. For example, some studies have used only API based sequences but not considered network based features and vice versa. It may be possible that a feature subset used for predicting malware family contain some redundant and/or irrelevant features leading to the extra computation overhead and reduced accuracy. It may also be possible that some feature may be irrelevant but become relevant and provide important information for predicting class of malware in the presence of some other feature.

In this work, we propose a framework for dynamic analysis of malware. The proposed framework is focused to extract dynamic behavior of malware during its execution in term of different features including duration, network features, API frequencies along with their sequences and count of various files read, written or created etc. For each program under consideration, a feature vector is generated for further analysis of its behavior and intent. Most promising features out of the representing feature vector are selected to represent behavior patterns of malware based upon information theory concepts. A labeled data repository is generated for training a machine learning technique as a classifier. Further, the trained classifier is used to predict the class of test sample as a malware family or a benign. Specifically, we consider the use of mutual information to select promising features and Multi-Layer Perceptron based Neural Network for classification of malware in this work. To determine the effectiveness of the proposed approach, we compare our detection results to the results obtained by using static analysis and dynamic analysis. We show that significantly stronger results can be obtained using the dynamic approach.

2. Methodology

This section is devoted to present the overall methodology followed in the present research work.

2.1 Experimental Setup

Dynamic analysis of malware has attracted lots of attention recently. Multiple systems have been proposed, such as CWSandbox[18], and Anubis[19]. Those systems can execute malware binaries within an instrumented environment and monitor their behaviors for analysis and development of defense mechanisms.

For further analysis, Malheur was developed to cluster and classify malware by processing the malware behaviors[20], CWSandbox[18] was employed for monitoring malware behaviors and represented the results in MIST format, by means of n-grams algorithm and several related approaches. Malheur can classify the malware to a predefined set of classes and find novel classes by clustering. Unfortunately, CWSandbox and MIST[21] are not open source, so we use Cuckoo sandbox[22] as a replacement.

In this work, we set up a virtual environment to run malicious programs. We observer dynamic the behavior of the malware sample by executing it in controlled and virtual environment created by Cuckoo sandbox[22]. Cuckoo sandbox is an open source automated malware analysis system. It can analyze PE, PDF, MS Office, PHP scripts, etc. In our experiments, the output from Cuckoo environment is stored in a file format of JSON report.

2.2 Extraction of raw features and their statistics from JSON files

To extract the raw features and their statistics from JSON reports, a Python language based automated system has been developed with the main steps as:

1. Read sections of JSON file
2. Extract features including
    - Basic features
    - Network feature statistics based upon protocol headers

- CPU and memory usage statistics
- Statistics of APIs based upon their categories
- Statistics of file system activities in terms of number of files written, delated, read, commands executed,
- services started, services created
- Resolved APIs
- Number of sub processes generated

3. Labeling of malware samples

The Python based system extracts the raw features from JSON files into a CSV file format for further analysis of malware.

2.3 Labeling of Malware Samples

It is most critical task in malware analysis using machine learning techniques in supervised mode. It is found that different antivirus vendors label malware samples differently. In most of cases, the labels are inconsistent with each other. So, the labeling of malware may be less accurate when employed to dynamic analysis of malware samples. Since, we are using the dataset for our experiments same as that used in our anchor paper proposed in [6]. They used Kaspersky detection for labeling the malware samples[6]. Therefore, in this work we decided to use the labeling of malware categories as reported by Kaspersky anti-virus during their execution in Cuckoo environment. Because, used it in the anchor paper[6].

2.4 Malware Dataset

After labeling of malware samples, we are able to generate a malware dataset from JSON reports from Cuckoo sandbox. We have chosen 23,146 malware samples randomly from the malware dataset of the VirusTotal[23]. In the dataset, we found a large number of classes of malware. For testing purpose of the proposed frame work, we categorized malware samples of different families into categories as reported in[6], the statistics of the collected malware with families are depicted in Table 1.

In this set of experiments, we used randomly 70% of malware samples as training dataset, whereas rest of 30% is used as test dataset. Malware dataset contains symbolic as well as continuous features. The dataset is pre-processed before it is used for training and testing the classifiers. The pre-processing steps involve mapping of symbolic value features to numeric value and scaling of feature values to a uniform scale. We have mapped malware class labels in dataset to numeric numbers in range of 0 to 16 as depicted in Table 1.

Table 1. Categories of malware and number of malware samples in dataset

Malware family	**Class label**	**No of samples**
Backdoor	0	501
Benign	1	1336
DangerousObject	2	531
Email-Worm	3	4536
Adware	4	672
Net-Worm	5	2592
Packed	6	2100
Trojan	7	873
Trojan-Downloader	8	2011
Trojan-Dropper	9	552
Trojan-FakeAV	10	1465
Trojan-GameThief	11	550
Trojan-PSW	12	588
Trojan-Ransom	13	895
Trojan-Spy	14	1120
Virus	15	1678
Worm	16	1080
	Total	23080

2.5 API Categories

The malware samples are executed in a controlled environment of Cuckoo sandbox and it behaviour is recorded in form of JSON reports. The behaviour of a malware sample is recorded in terms of API calls. It is observed that there are about 2727 different API names in our dataset[6]. These API names are categorized into different categories depending upon its intended use. The API categories are depicted in Table 2. The major API categories contains the API calls related to Exceptions, File system operations, Internet explorer, Network, OLE, Processes, Registry, Synchronization, User interface, and Miscellaneous as categories reported by Kaspersky antivirus program in JSON reports of malware behaviour.

2.6 Feature Selection

For dynamic analysis of malware, we extracted a large number of raw features from JSON reports produced in Cuckoo environment representing the dynamic behaviour of malware. In theory, higher dimensions of the data improve the classification accuracy of the algorithm. But, practically, it is not true. All the raw features of the data are not important to understand it. However, the higher dimensions of the data suffer from difficulty called curse of dimensionality[24–27]. In addition, high dimensional data requires more computational overhead and leads to delay in detection of malware, which is not desirable. In order to tackle this difficulty of analyzing the high dimensional data, we identified a filter based feature selection technique proposed in[28]. The identified feature selection tech- nique keeps the original features as such and select subset of the features that predicts the target class variable with maximum classification accuracy using expression 1.

$$PEF = RelevanceTerm - \beta * RedundancyTerm + \gamma * ClassCondtionalTerm$$

$$= MI(X_n, Y) - \beta * \sum_{k=1}^{n} MI(X_n, X_k) + \gamma * \sum_{k=1}^{n} MI(X_n, X_k|Y) \quad (1)$$

Where MI ( ) gives the mutual information between two variables. The parameters β, γ are the weights assigned to redundancy term and class conditional interaction information term respectively. The parameter β regulates the relative significance between candidate feature and already selected set of the features with respect to the target class variable. Only the MI with the target class variable is considered for each feature selection ignoring redundancy and interaction information by putting $\beta = 0, \gamma = 0$. A large value of β ensures high penalty for redundancy, and net MI of candidate feature is discounted by a quantity equivalent to its redundancy with already-selected features. $\gamma = 0$ assumes the non-interaction of features. A large value of γ results the addition of interaction information of candidate feature with already selected features to its net MI.

For effective feature selection, an automated system is developed in Python language that implements the identified feature selection technique. The system takes a raw dataset as an input and selects most promising features using mutual information. It gives an output as a list indexes selected corresponding to reduced dataset. The reduced dataset is generated using only selected indexes from the raw dataset in a CSV format. Selected dataset is further used to train the classifier model for prediction of unknown malware samples.

2.7 Malware Classification

A large number of machine learning techniques in supervised mode have been proposed that are used for classifying malware dataset into a set of malware classes. For instance, Artificial Neural Networks are designed to mimic the human brain. They have the capability to learn any nonlinear relationship between input and desired output even in presence of noisy input training data. So, keeping advantages of neural networks in mind, we utilized MLP neural network as a classifier to learn the behavior of malware samples[29]. The MLP classifier is trained using raw dataset as well as reduced dataset containing only selected features. The trained model of MLP classifier is further used to predict the class of unknown malware samples.

2.8 Implementation

To perform experiments and evaluate the performance of the proposed framework, we implemented it as an automated system in Python language. In these experiments, we used Python on a Linux PC with Core i3-2330M 2.20.

GHz CPU and 2 GB RAM. Identified feature selection technique and MLP ANN based classifier is implemented in Python.

3. Proposed Framework

In this section, we describe the proposed framework for malware classification based on dynamic behavior represented in terms of feature vectors. Figure 1 shows a schematic overview of the proposed framework.

Functioning of the major modules of the proposed framework is summarized:

3.1 Cuckoo Sandbox Environment

This module is designed to observe the behavior of malware by executing it in a controlled environment. We used Cuckoo sandbox to execute the malware and record its behavior in the form JSON reports. These reports contain the detailed recording of duration, system calls, its arguments, network information etc. The output of the module is set of reports in JSON format for malware and benign.

3.2 Feature Extraction Module

The module extracts the raw features of program for its dynamic behavior from reports in JSON format pro-

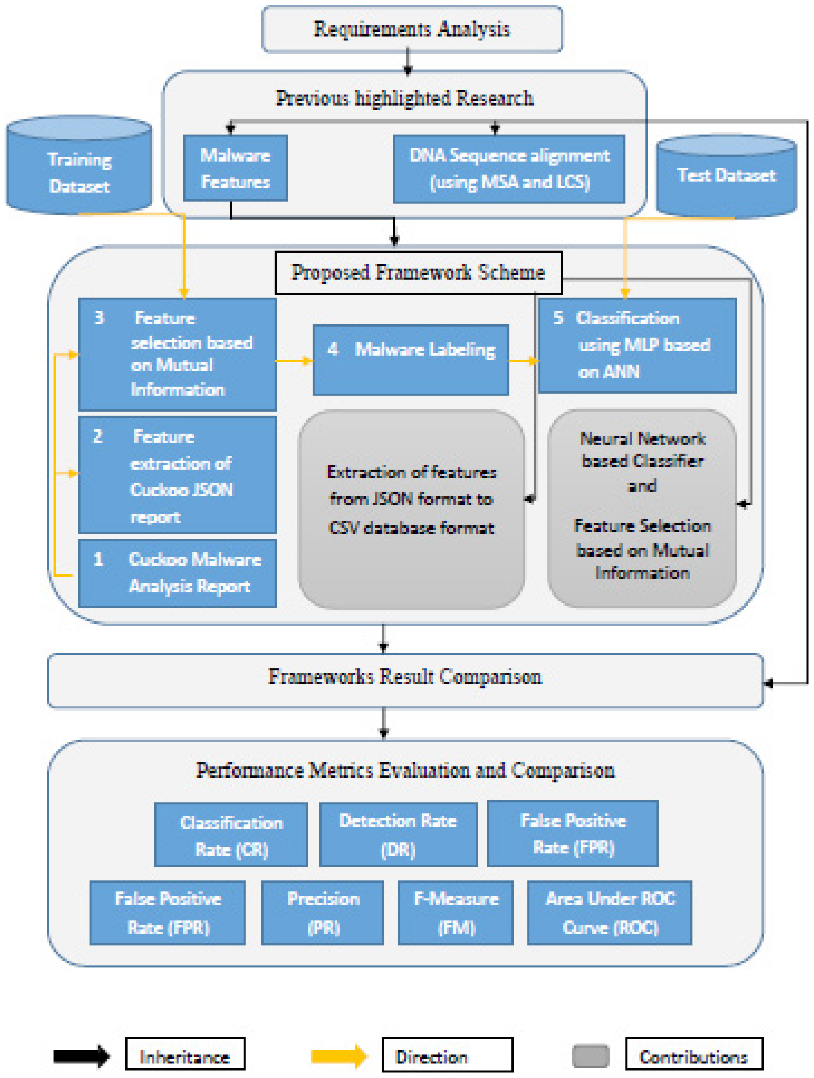
Requirements Analysis
Previous highlighted Research
Training Dataset
Malware Features
DNA Sequence alignment (using MSA and LCS)
Test Dataset
Proposed Framework Scheme
3 Feature selection based on Mutual Information
4 Malware Labeling
5 Classification using MLP based on ANN
2 Feature extraction of Cuckoo JSON report
Extraction of features from JSON format to CSV database format
Neural Network based Classifier and Feature Selection based on Mutual Information
1 Cuckoo Malware Analysis Report
Frameworks Result Comparison
Performance Metrics Evaluation and Comparison
Classification Rate (CR)
Detection Rate (DR)
False Positive Rate (FPR)
False Positive Rate (FPR)
Precision (PR)
F-Measure (FM)
Area Under ROC Curve (ROC)
Inheritance
Direction
Contributions

Figure 1. Schematic overview of the proposed framework.

vided as output of Cuckoo environment. The extracted features are transformed into a tabular form and saved to a CSV file containing all possible dynamic features of a malware. The dynamically extracted features includes features related to Dynamic Imports, File Operations, Mutex Operations, Network Operations, Processes Created, injected or terminated, Registry Operations, Windows API Calls and their frequency. The working of the feature extraction module is summarized in Figure 2.

After extracting the features from JSON reports, the values are summarized and further are converted to a feature vector to represent the malware dynamic behavior. The labeling of the feature vector is done on the basis of detection results by Kaspersky Antivirus into categories of malware family. The process is repeated for all JSON reports to extract all feature vectors of malware samples.

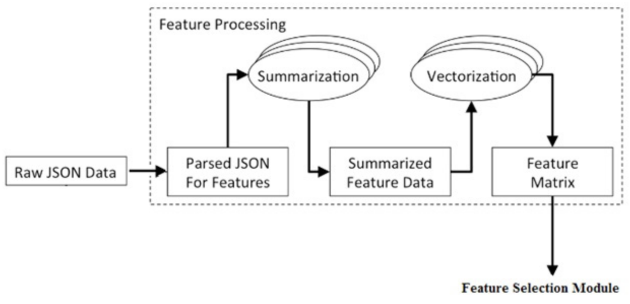

Figure 2. Schematic overview of the feature extraction module.

In this proposed work, we extracted 361 features and 01 feature as family of malware. The description of extracted features is as depicted in Table 2.

The last feature named API name and its frequency contains the name of 321 different APIs and their occurrences in JSON report.

Table 2. Features extracted from JSON report

Category	Feature	Data type
Info	Duration	Numeric
Network	UDP requests	Numeric
	IRC requests	Numeric
	http requests	Numeric
	smtp requests	Numeric
	tcp requests	Numeric
	hosts contacted	Numeric
	DNS requests	Numeric
	domains contacted	Numeric
	ICMP requests	Numeric
Usage	CPU usage	Numeric
	mem usage	Numeric
Dropped	Dropped	Numeric
API categories	Noti API	Numeric
	Certi API	Numeric
	Crypto API	Numeric
	exception API	Numeric
	file API	Numeric
	iexplore API	Numeric
	misc API	Numeric
	netapi API	Numeric
	network API	Numeric
	ole API	Numeric
	process API	Numeric
	registry API	Numeric
	resource API	Numeric
	services API	Numeric
	Syn API	Numeric
	system API	Numeric
	ui API	Numeric
	other API	Numeric
API summaries	files accessed	Numeric
	files written	Numeric
	files deleted	Numeric
	Mutexes	Numeric
	executed cmds	Numeric
	started services	Numeric
	files read	Numeric
	resolved APIs	Numeric
	created services	Numeric
Processes	processes generated	Numeric
API name and its frequency	API (321)	Numeric
Malware Family	Family	Categorical

3.3 Feature Selection Module

The module is responsible for selecting most promising features from the raw features dataset provided by the extraction module of the proposed framework. The malware dataset contains some irrelevant and redundant features. Processing of these irrelevant and redundant features leads to many problems including, 1) Undesirable delay in classification task which in turn loses the real time capability of MCS; 2) Increase computation overhead in terms of memory and time; and 3) Deteriorate the classification and prediction accuracy. To solve this problem, we employed an information theoretic approach to feature selection suggested by[28]. This feature selection approach is a filter approach and independent of any classification technique. Thus, relevant features of malware dataset are selected using the approach. The reduced dataset is not dependent upon any classification technique. Here, we utilized Mutual Information to compute the relevance of features to predict the class labels. The reduced malware dataset contains 50 features of the original number of instances as depicted in Table 3. The most promising features are selected by considering the relevance, redundancy and class interaction information of features in predicting the class of malware as per expression 1. The output of this module is a set of indices of selected features in the feature vector of the dataset.

3.4 Classification Module

The working of this module involves two phases namely, training phase and testing phase. 1) Training Phase: A machine learning based technique specifically MLP-

Table 3. Most promising features selected from malware dataset

Sr No	Feature name	Sr No	Feature name
1	Duration	26	CreateThread
2	UDP req	27	CreateToolhelp32Snapshot
3	http req	28	Process32FirstW
4	tcp req	29	NtAllocateVirtualMemory
5	hosts contacted	30	NtCreateSection
6	ICMP req	31	NtOpenProcess
7	CPU usage	32	NtOpenSection
8	Dropped	33	NtProtectVirtualMemory
9	Noti API	34	NtResumeThread
10	filesystem API	35	NtSuspendThread
11	misc API	36	NtTerminateThread
12	network API	37	NtUnmapViewOfSection
13	process API	38	RegCloseKey
14	registry API	39	GetLocalTime
15	system API	40	GetSystemTime
16	other API	41	GetSystemTimeAsFileTime
17	resolved APIs	42	NtDelayExecution
18	FindFirstFileExW	43	NtQuerySystemTime
19	NtQueryInformationFile	44	IsDebuggerPresent
20	NtReadFile	45	LdrGetDllHandle
21	NtSetInformationFile	46	LdrGetProcedureAddress
22	GetCursorPos	47	NtClose
23	GetSystemMetrics	48	SetWindowsHookExA
24	recv	49	FindWindowA
25	socket	50	other

ANN is being trained using selected features only for learning the behavior of malware. The output of this phase is trained model of the classifier. 2) Testing Phase: Here, the trained model is given input of Test dataset in terms of selected features only to predict the class label of malware. A report is generated as an output that can be used by security analysts for further policy decisions.

3.5 Performance Analysis Module

After the testing phase, performance analysis module computes the defined performance metrics. The performance metrics divided into three classes: threshold, ranking and probability metrics[30]. Threshold metrics include Classification Rate (CR), F-Measure (FM) and Cost Per Example (CPE). It is not important how close a prediction is to a threshold, only if it is above or below threshold. The value of threshold metrics lies in [0, 1]. Ranking metrics include False Positive Rate (FPR), Detection Rate (DR), Precision (PR) and area under ROC curve (ROC). The value of ranking metrics lies in [0, 1]. These metrics depend on the ordering of the cases, not the actual predicted values. As long as ordering is preserved, it makes no difference. These metrics measure how well the negative instances are ordered before positive instances and can be viewed as a summary of model performance across all possible thresh- olds. Probability metrics include Root Mean Square Error (RMSE). Value of RMSE lies between 0 and 1. The metric is minimized when the predicted value for each negative class coincides with the true conditional probability of that class being normal class. These metrics are computed from confusion matrix. The matrix gives the values of True Positives (TP), True Negatives (TN), False Positives (FP) and False Negatives (FN).

In this work, we focus to compute confusion matrix and derive different performance metrics namely: Accuracy, TPR, FPR, FNR, Precision, Recall, Specificity, Sensitivity, and F-measure to evaluate the performance of the proposed approach.

4. Results and Discussion

This section describes the evaluation dataset, and experimental setup. The section presents a comparison of the proposed method and other representative dynamic malware analysis method in terms of defined performance metrics. In order to evaluate the performance of proposed framework, we conducted the experiments to evolve MLP-ANN with parameters as described in Table 4 based upon selected the dataset for malware analysis.

Table 4. Configuration setting of MLP

Input nodes	Number of features of the malware dataset
Hidden layer	1
Number of hidden nodes	300
Output nodes	Number of malware families in the dataset

For our experiment, we set up a virtual environment of the Cuckoo Sandbox to run malicious programs. A detailed report is fetched containing full set of features in a JSON format. We executed malicious programs of different families as well benign program and collected their JSON reports.

We performed experiments by randomly selecting instances from malware dataset as described in above cited section. We computed the defined performance metrics from confusion matrix for different malware classes in terms of True Positives (TP), True Negatives (TN), False Positives (FP) and False Negatives (FN) as depicted in Table 5.

Table 5. Performance metrics

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

4.1 Results for Malware Dataset having Full Feature Set

The confusion matrix for test results of MLP based upon malware dataset having full feature set is computed. Subsequently, identified performance metrics have been computed from confusion matrix and depicted in Table 6.

4.2 Results for Malware Dataset having a Selected Feature Set

We employed the identified feature selection technique to select the promising features from malware dataset. The most promising selected features are depicted in Table 3.

We run our experiments with dataset having selected feature set and number of instances are same as described in Table 1. The confusion matrix for test results of MLP based upon malware dataset having selected feature set is computed. Subsequently, identified performance metrics have been computed from confusion matrix and are depicted in Table 7.

It can be observed from values mentioned in Table 6 and Table 7 that our proposed framework leads to improve the detection of malware to their respective classes.

In order to prove the practicality of the proposed approach, we compare the results of the proposed paper with representative paper in the field proposed in [6]. It can be observed from Table 8 that the proposed approach proved the results comparable to that of provided by [6] and [31]. To summarize for the proposed framework, the FPR is 0, Recall is 0.999, and Precision is 1. This implies that the proposed framework for malware classification system is highly reliable for a real malware dataset. Moreover, the proposed framework provides a small number of more

Table 6. Performance metrics for malware dataset having full feature set

Malware category	Accuracy	TPR	FPR	FNR	Precison	Recall	Specificity	Sensitivity	F-measure
Backdoor	0.99	0.64	0.00	0.36	0.80	0.64	1.00	0.64	0.71
benign	0.98	0.75	0.00	0.25	0.91	0.75	1.00	0.75	0.83
DangerousObject	0.99	1.00	0.01	0.00	0.77	1.00	0.99	1.00	0.87
Email-Worm	1.00	0.99	0.00	0.01	1.00	1.00	1.00	1.00	1.00
Adware	1.00	0.95	0.00	0.05	1.00	1.00	1.00	1.00	1.00
Net-Worm	1.00	0.99	0.00	0.01	1.00	1.00	1.00	1.00	1.00
Packed	1.00	1.00	0.00	0.00	1.00	1.00	1.00	1.00	1.00
Trojan	0.97	0.45	0.01	0.55	0.78	0.45	1.00	0.45	0.57
Trojan-Downloader	0.99	0.99	0.01	0.01	0.93	1.00	0.99	1.00	0.96
Trojan-Dropper	0.99	0.78	0.00	0.22	0.99	0.78	1.00	0.78	0.87
Trojan-FakeAV	1.00	0.97	0.00	0.03	1.00	1.00	1.00	1.00	1.00
Trojan-GameThief	1.00	0.96	0.00	0.04	0.99	1.00	1.00	1.00	1.00
Trojan-PSW	0.99	0.85	0.00	0.15	0.87	0.85	1.00	0.85	0.86
Trojan-Ransom	0.98	0.97	0.02	0.03	0.70	1.00	0.98	1.00	0.82
Trojan-Spy	1.00	1.00	0.00	0.00	0.96	1.00	1.00	1.00	0.98
Virus	0.97	0.71	0.00	0.29	0.94	0.71	1.00	0.71	0.81
Worm	0.98	1.00	0.02	0.00	0.68	1.00	0.98	1.00	0.81

Table 7. Performance metrics for malware dataset having a selected feature set

Malware category	Accuracy	TPR	FPR	FNR	Precison	Recall	Specificity	Sensitivity	F-measure
Backdoor	0.99	0.64	0.00	0.36	0.80	0.64	1.00	0.64	0.71
Benign	0.98	0.75	0.00	0.25	0.91	0.75	1.00	0.75	0.83
DangerousObject	0.99	1.00	0.01	0.00	0.77	1.00	0.99	1.00	0.87
Email-Worm	1.00	0.99	0.00	0.01	1.00	1.00	1.00	1.00	1.00
Adware	1.00	0.95	0.00	0.05	1.00	1.00	1.00	1.00	1.00
Net-Worm	1.00	0.99	0.00	0.01	1.00	1.00	1.00	1.00	1.00
Packed	1.00	1.00	0.00	0.00	1.00	1.00	1.00	1.00	1.00
Trojan	0.97	0.45	0.01	0.55	0.78	0.45	1.00	0.45	0.57
Trojan-Downloader	0.99	0.99	0.01	0.01	0.93	1.00	0.99	1.00	0.96
Trojan-Dropper	0.99	0.78	0.00	0.22	0.99	0.78	1.00	0.78	0.87
Trojan-FakeAV	1.00	0.97	0.00	0.03	1.00	1.00	1.00	1.00	1.00
Trojan-GameThief	1.00	0.96	0.00	0.04	0.99	1.00	1.00	1.00	1.00
Trojan-PSW	0.99	0.85	0.00	0.15	0.87	0.85	1.00	0.85	0.86
Trojan-Ransom	0.98	0.97	0.02	0.03	0.70	1.00	0.98	1.00	0.82
Trojan-Spy	1.00	1.00	0.00	0.00	0.96	1.00	1.00	1.00	0.98
Virus	0.97	0.71	0.00	0.29	0.94	0.71	1.00	0.71	0.81
Worm	0.98	1.00	0.02	0.00	0.68	1.00	0.98	1.00	0.81

Table 8. Comparative summary of the malware classification systems

Malware Classification System	FPR	FNR	Recall	Precision	F-score
APIMDS [6]	0	0.0011	0.998	1	0.999
MSPMD [31]	0.613	0.038	0.962	0.959	0.960
(using ANN classifier)					
The proposed framework	0.596	0.016	0.984	0.929	0.956
(Full feature set)					
The proposed framework	0	0.000459	0.999	1	0.999
(Selected feature set)					

abstract features to predict the malware family which leads to low computational overhead. Low computational overhead leads to fast detection of malware and hence minimize the damage of resources. In order to prove the practicality of the proposed framework, we compare the results of the proposed framework with representative techniques in the field. It can be observed from Table 8 that the proposed framework provided the results comparable to that of provided by[6] and best results proved by [31]. To summarize for the proposed framework, the FPR is 0, Recall is 0.999, and Precision is 1. This implies that the proposed framework for malware classification system is highly reliable for a real malware dataset. Moreover, the proposed framework provides a small number of more abstract features to predict the malware family which leads to low computational overhead. Low computational overhead leads to fast detection of malware and hence minimize the damage of resources.

The comparison of results of the proposed approach with the representative proves its practical capability in the real world.

Therefore, we draw a conclusion that the proposed framework can lead to better results in malware analysis.

4.3 Discussion

This section elaborates on the reporting results. If the results for a particular malware type are good enough to be used in a future system that can pre-filter newly registered malware samples. It should be noticed that future work will be devoted to optimize the classifier such that a pre-filtering system can be developed to identify novel malware samples and sort out legacy malware that have minor changes.

It can be concluded from the Table 7 that in most of malware classes, the proposed framework is capable to report TPR very close to 1 and FPR close to 0. However, in some cases, it reported low detection results. Low detection results for some of malware classes may be due to imbalance in sampling of malware dataset. The imbalance in samples of malware dataset generally leads to biasing of the classifier MLP towards a majority class and poor classification of minority class samples.

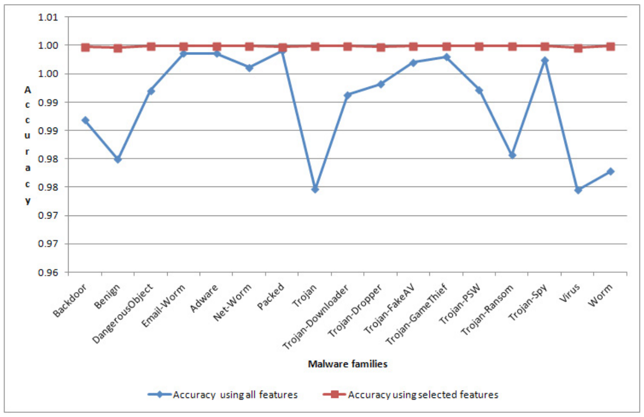

Figure 3. Comparative results of the proposed framework in terms of accuracy.

The problem of having a small number of Trojan samples classified as Adware can be caused by the fact that some of the Adware samples actually are Trojans, which have been used to install the Adware while running the experiment. With this small amount of FPs the classifier still performs satisfactory for Trojan. Similarly, for Trojan Ransom the proposed framework reported satisfactory results of accuracy of 98% with FPR 2%.

For Trojan-PSW, the proposed framework performs well regardless of the low amount of samples, which could be because of its distinct behavior. Therefore, it can be used as pre-filtering criteria, but in order to ensure a good performance, more samples should be used to train the model. Similar case can be considered for other low amount of samples. We compared the performance of the proposed framework over two set of malware data containing all features and selected set of features in terms of accuracy as depicted in Figure 3. The results indicate that the proposed framework has improved the results of malware classes to a significant level by using a selected set of features over use of all features.

We analyzed the performance of the proposed framework over two set of malware data containing all features and selected set of features in terms of TPR as depicted in Figure 4. The results indicate that the proposed framework has improved the results of malware classes to a significant level by using a selected set of features over use of all features.

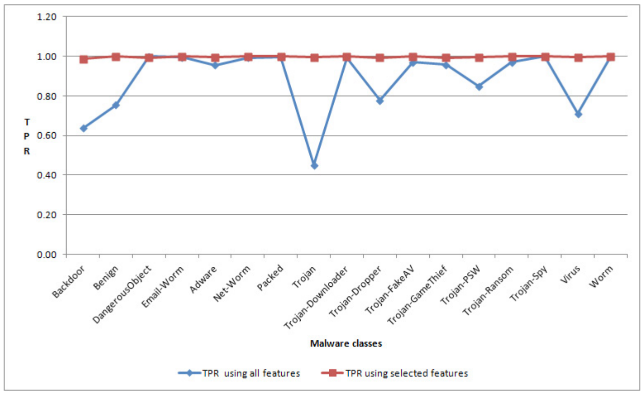

Figure 4. Comparative results of the proposed framework in terms of TPR.

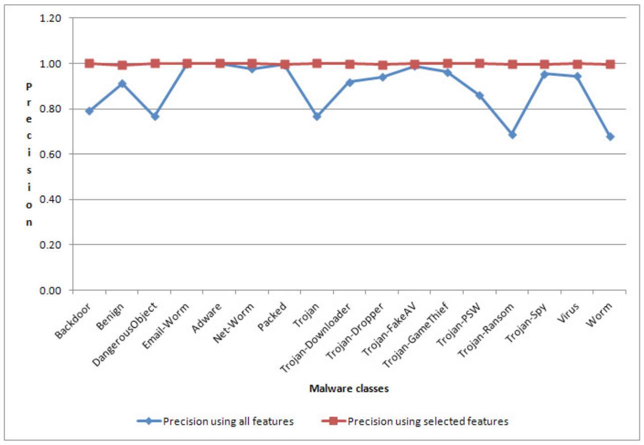

Figure 5. Comparative results of the proposed framework in terms of precision.

Figures 5–7 depicts the result comparison of the proposed framework in terms of precision, F- measure and FPR respectively. The results indicate that the proposed framework has improved the results of malware classes to a significant level by using a selected set of features over use of all features in terms of precision, F-measure and FPR.

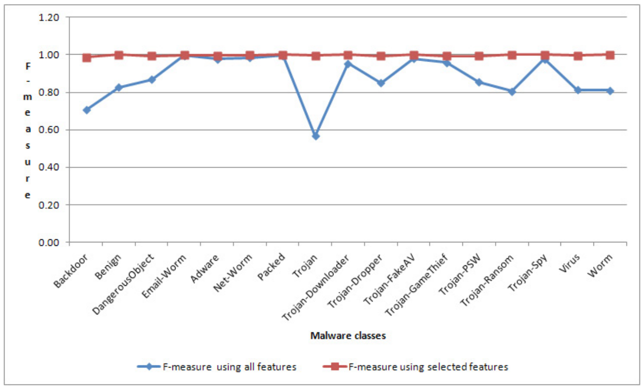

Figure 6. Comparative results of the proposed framework in terms of F-measure.

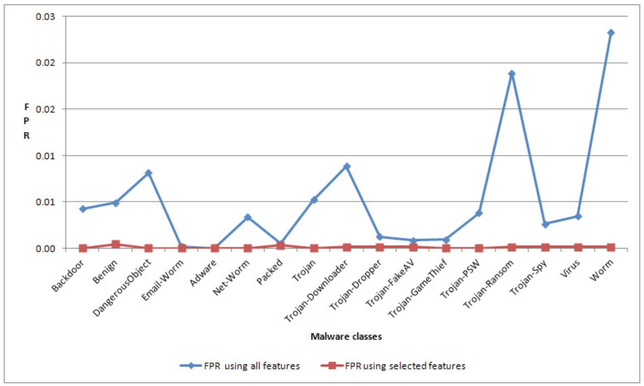

Figure 7. Comparative results of the proposed framework in terms of FPR.

It can be concluded above cited paragraphs that the proposed framework has shown significant improvements of results in terms of accuracy, TPR, FPR, precision and F-measure using selected set of features over all features of malware dataset. Use of a selected set of features in the proposed work is not only able to improve the malware classification results but also requires less computation cost in terms of CPU usage and memory usage.

Figure 8 depicted the comparison of the results for the proposed technique with representative techniques in the field.

It can be noticed from Figure 8 that the proposed framework reported improved results than representative techniques and proposed framework with all features in terms of identified metrics.

In a nutshell, most of the malware classes have been categorized by the proposed framework satisfactory using a subset of features quickly. This proves its validity for real time detection of malware.

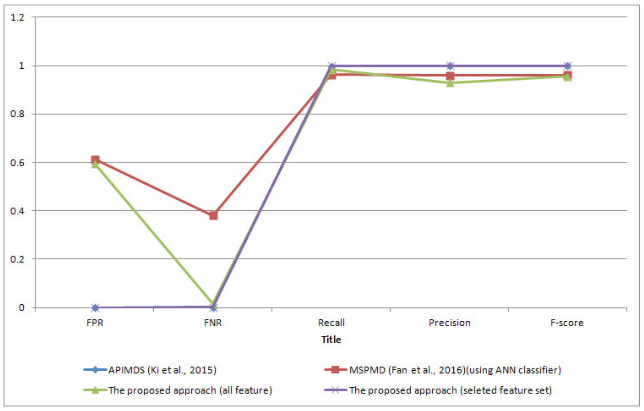

Figure 8. Comparative summary of the malware classification systems.

5. Concluding Remarks and Future Directions

In the study, we proposed a framework for effective analysis and classification of malware. It extracts the dynamic raw features from JSON reports and selects the most promising features by using mutual information based upon information theory. The features are selected by taking relevance, redundancy and class conditional interaction information into consideration. The feature selection process helps to reduce the amount of data required for effective malware analysis without compromising the accuracy. A MLP-NN based classifier is trained based upon selected feature set of malware training data and further used to predict the family of unknown malware. Here, feature selection is based upon computation of mutual information of all feature with target class and process is repeated to compute relevance, redundancy and class conditional interaction information. So, it is computationally expensive. This is a major limitation of the proposed framework, which can be addressed using parallel computing. Moreover, quality of MLP-NN classifier can be further improved by training using quality malware dataset. Availability of quality training dataset is a challenging task. We validated proposed approach using a small subset of malware dataset only. The applicability of the proposed approach can be tested for different malware real datasets. So, our future research will be to carry out more experiments by choosing different malware datasets to improve the classification results. Another direction for research work is to explore different techniques to reduce the training time of MLP-NNs.

6. References

1. Krister KM. Automated analyses of malicious code. Springer. 2006; 2:67–77.
2. Egele M, Scholte T, Kirda E, Kruegel A. A survey on automated dynamic malware-analysis techniques and tools. ACM Computing Surveys (CSUR) USA. 2012; 44(2):6.
3. Moser C, Kruegel E, Kirda K. Limits of static analysis for malware detection. Computer Security Applications Conference, ACSAC Twenty-Third Annual, IEEE, Vienna; 2007. p. 421–30. Crossref
4. Cesare S, Xiang Y. Software similarity and classification. Springer Science & Business Media; 2012. Crossref
5. Kane PO, Sezer S, Laughlin K. Obfuscation: The hidden malware. Security & Privacy, IEEE. 2011; 9(5):41–7. Crossref
6. Ki Y, Kim K, Kim HK. A novel approach to detect malware based on API call sequence analysis. International Journal of Distributed Sensor Networks; 2015. p. 4.
7. Sharif M, Yegneswaran V, Saidi H, Porras P, Lee W. Eureka: A framework for enabling static malware analysis. Computer Security-ESORICS Springer. 2008; 5283:481–500. Crossref
8. Ahmad S, Ahmad S, Xu S, Li B. Next generation malware analysis techniques and tools. Electronics, Information Technology and Intellectualization: Proceedings of the International Conference EITI Shenzhen, China,CRC Press; 2015. p. 17. Crossref
9. Gorecki C, Freiling FC, Kührer M, Holz T. Trumanbox: Improving dynamic malware analysis by emulating the internet. Stabilization, Safety, and Security of Distributed Systems, Springer; 2011. p. 208–222. Crossref
10. Griffin K, Schneider S, Hu X, Chiueh TC. Automatic generation of string signatures for malware detection. Recent advances in intrusion detection, Springer; 2009. p. 101–120. Crossref
11. Tian R, Islam R, Batten L, Versteeg S. Differentiating malware from cleanware using behavioural analysis. 5th International Conference on Malicious and Unwanted Software (MALWARE), IEEE, Australia; 2010. p. 23–30. Crossref
12. Shankarapani MK, Ramamoorthy S, Movva RS, Mukkamala S. Malware detection using assembly and API call sequences. Journal in Computer Virology. 2011; 7(2):107–19. Crossref
13. Lee T, Mody JJ. Behavioral classification. EICAR Conference, USA. 2006; 45(4):1–17.
14. Bailey M, Oberheide J, Andersen J, Mao ZM, Jahanian F, Nazario J. Automated classification and analysis of internet malware. Recent Advances in Intrusion Detection, Springer; 2007. p. 178–97. Crossref
15. Rieck K, Holz T, Willems C, DüsselP, Laskov L. Learning and classification of malware behavior. Detection of intrusions and malware, and vulnerability assessment, Springer; 2008. p. 108–25. Crossref

16. Bayer U, Comparetti PM, Hlauschek C, Kruegel C, Kirda K. Scalable, behavior-based malware clustering. NDSS, Citeseer, 9; 2009. p. 8–11.
17. Nari S, Ghorbani AA. Automated malware classification based on network behavior. 2013 International Conference on Computing, Networking and Communications (ICNC), IEEE, Canada; 2013. p. 642–7. Crossref
18. Willems C, Holz T, Freiling F. Cwsandbox: Towards automated dynamic binary analysis. IEEE Security and Privacy. 2007; 5(2):32–9. Crossref
19. Bayer U, Moser A, Kruegel C, Kirda K. Dynamic analysis of malicious code. Journal in Computer Virology. 2006; 2(1):67–77. Crossref
20. Rieck K. Malheur-automatic analysis of malware behavior. 2015.
21. Trinius P, Willems C, Holz T, Rieck K. A malware instruction set for behavior-based analysis; 2009. p. 1–11.
22. Sandbox C. Automated malware analysis. Germany; 2013. p. 1–11.
23. VirusTotal, Virustotal-free online virus, malware and url scanner [Internet]. [cited 2016 Jun 16]. Available from: www.virustotal.com.
24. Bellman R. Adaptive control processes: a guided tour. Princeton university press, Princeton, New Jersey, USA; 2007.
25. Sharma UM. Hybrid feature based face verification and recognition system using principal component analysis and artificial neural network. Indian Journal of Science and Technology. 2015; 8(S1):115–20. Crossref
26. Das K, Ray J, Mishra D. Gene selection using information theory and statistical approach. Indian Journal of Science and Technology. 2015; 8(8):695. Crossref
27. Radhika S, Arumugam S. Improved non mutual information based multi-path time delay estimation. Indian Journal of Science and Technology. 2014; 7(8):1101–06.
28. Kumar G, Kumar K. An information theoretic approach for feature selection. Security and Communication Networks. 2012; 5(2):178–85. Crossref
29. Shekar MS, Krishna PM, Venkatesan M. Artificial neural network based prediction of pressure drop in heat exchangers. Indian Journal of Science and Technology. 2015; 8(S9):87–92. Crossref
30. Kumar G, Kumar K. Ai based supervised classifiers: An analysis for intrusion detection. Proceedings of International Conference on Advances in Computing and Artificial Intelligence, ACM, USA; 2011. p. 170–4. Crossref
31. Fan Y, Ye Y, Chen L. Malicious sequential pattern mining for automatic malware detection. Expert Systems with Applications. 2016; 52:16–25.