A Novel Malware Analysis for Malware Detection and Classification using Machine Learning Algorithms

Kamalakanta Sethi
Indian Institute of Technology Bhubaneswar
Orissa, India
ks23@iitbbs.ac.in

Shankar Kumar Chaudhary
Indian Institute of Technology Bhubaneswar
Orissa, India
sc16@iitbbs.ac.in

Bata Krishan Tripathy
Indian Institute of Technology Bhubaneswar
Orissa, India
bt10@iitbbs.ac.in

Padmalochan Bera
Indian Institute of Technology Bhubaneswar
Orissa, India
plb@iitbbs.ac.in

ABSTRACT

Nowadays, Malware has become a serious threat to the digitization of the world due to the emergence of various new and complex malware every day. Due to this, the traditional signature-based methods for detection of malware effectively becomes an obsolete method. The efficiency of the machine learning model in context to the detection of malware files has been proved by different researches and studies. In this paper, a framework has been developed to detect and classify different files (*e.g exe, pdf, php, etc.*) as benign and malicious using two level classifier namely, Macro (for detection of malware) and Micro (for classification of malware files as a *Trojan, Spyware, Adware, etc.*). Cuckoo Sandbox is used for generating static and dynamic analysis report by executing files in the virtual environment. In addition, a novel model is developed for extracting features based on static, behavioral and network analysis using analysis report generated by the Cuckoo Sandbox. Weka Framework is used to develop machine learning models by using training datasets.

The experimental results using proposed framework shows high detection rate with an accuracy of 100% using J48 Decision tree model, 99% using SMO (Sequential Minimal Optimization) and 97% using Random Forest tree. It also shows effective classification rate with accuracy 100% using J48 Decision tree, 91% using SMO and 66% using Random Forest tree. These results are used for detecting and classifying unknown files as benign or malicious.

CCS CONCEPTS

• **Security and privacy → Intrusion/anomaly detection and malware mitigation**; **Malware and its mitigation**;

SIN '17, October 13–15, 2017, Jaipur, IN, India

ACM ISBN 978-1-4503-5303-8/17/10. . . $15.00
https://doi.org/10.1145/3136825.3136883

KEYWORDS

Malware Detection, Malware Classification, Static and Dynamic Analysis, Cuckoo Sandbox, SMO

ACM Reference Format:
Kamalakanta Sethi, Shankar Kumar Chaudhary, Bata Krishan Tripathy, and Padmalochan Bera. 2017. A Novel Malware Analysis for Malware Detection and Classification using Machine Learning Algorithms. In *Proceedings of SIN '17.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3136825.3136883

1 INTRODUCTION

We are living in a Digital Era, where the use of technology and internet becomes the vital part of our daily life. As a result, the trend of Internet usage is also growing exponentially. Different types of computer applications is downloaded by the public, professional or private users via Internet at massive scale. Criminals and intruders are using the Internet for black marketing where they develop malware and other exploits for breaching system security. This provides a strong incentive for the hackers to modify and increase the complexity of the malicious code in order to improve the obfuscation to decrease the chances of being detected by anti-virus programs. This leads to multiple forks or new implementations of the same type of malicious software that can propagate out of control. As a result, the vulnerability of using the Internet is increased due to increasing threats of Malware[5], which get shipped within the software and files via Internet. Based on AV-Test [5], approximately 390,000 new malware samples are registered every day, which gives rise to the problem of processing the huge amount of unstructured data obtained from malware analysis. A report on the malware evolution from Symantec [12] says "New sophistication and innovation marked seismic shifts in the focus of attacks. Zero-day vulnerabilities and sophisticated malware were used less as nation states devolved from espionage to straight sabotage."

Malware is a malicious software which is used with the intention of breaching a computer systems security policy with respect to confidentiality, integrity and availability of data. Malware is of different types like virus, Trojan horse, spyware, adware,trapdoor, etc. according to their way of imposing threats to the system. It can add, change, or remove any program from the system to intentionally harm the

system's required functions and also can get the identity without user consent. According to AV-Test[5] in March 2017, the total number of malwares is increasing exponentially since 2008 and has reached more than 583 million. Due to such increasing number of malwares, it is necessary to detect these files before they harm.

As per the researches and studies, the malware detection system comprises of two tasks [6]:

(1) Malware Analysis:-where one deal with static analysis (also called code analysis which finds API call sequence in binary), dynamic analysis (behavioral analysis by running samples in emulation environment) and hybrid analysis (mix of static and dynamic analysis).
(2) Malware Detection:-where two well known detection techniques, i.e., Signature based (now its obsolete due to complexity of malware) and Behavioral based(also known as heuristic detection technique which uses learning algorithm for effective training and detection) techniques are used.

In this paper, we have built a framework to detect and classify the malware effectively and efficiently. For malware analysis, we have used Cuckoo Sandbox [3] (which runs the file in emulation environment to get broad analysis of the file) along with ReportHandler (which extracts the features from the report generated by Cuckoo). For the malware detection, we have used Weka Framework [14] and have tested the resulting dataset with different Machine learning (ML) algorithms.

The rest of the paper is organized as follows. Section 2 describes the related work. In Section 3, our proposed malware analysis framework has been presented. The efficacy of our proposed framework with experimental results has been presented in Section 4. Finally, we concluded in Section 5.

2 RELATED WORK

In this section, we discuss the related work used for malware analysis. Comar et al.[1] have proposed a framework which can detect known as well as newly emerging malwares at high precision using layer 3 and layer 4 network traffic features. In addition, they only focused on network analysis to detect and classify a file. Gavrilu et al.[4], used cascaded one-sided perceptrons and cascaded kernelized one-sided perceptrons to develop a framework for detecting a file as benign or malware with the aim to minimize the number of false positives. They have used a set of the features from binary files.

As malware is becoming more complex to detect using the static or code analysis, researchers have tried to get the dynamic information by executing the file in virtual or simulation environment. This is also known as sandboxing [13]. For dynamic and runtime analysis, different sandboxing technique [3] are used. This approach has merits and demerits, but it is a valuable technique for obtaining additional information related to network and other behaviour. Therefore, it is a good practice to perform both static and dynamic analysis while inspecting a malware, in order to gain a deeper understanding of it.

Cuckoo [3] is a tool that allows malware analysis using sandboxing technique. It is used to automatically run, analyze files and collect comprehensive analysis results. These results outline what the malware does while running inside an isolated operating system and retrieve traces of API (Application Programming Interface) calls, information related to registry modification, file modification and network traffics. Researchers [7, 9] have used Cuckoo Sandbox for malware analysis.

Another work [7] focused on the memory images to extract features like registry activity, API calls, and imported libraries. In addition, it reported the performance of different machine learning algorithms and found SVM (support vector machine) performed well than others. On the other hand, Salehi et al. [11] used API calls with deep analysis using passed argument. In addition, they tried to analyze and classify high amount of malware. Pirscoveanu et al. [9], used a combination of features to achieve high classification rate.

Limitations of existing techniques: As malware becomes complex, it is difficult to detect a given sample using previous techniques as benign or malicious. Limitations of different techniques are:-

- *Signature based approach:-* It fails to detect Zero-Day attack and threats with evolving capabilities such as metamorphic and polymorphic malwares. It is also not able to detect emerged malwares.
- *Anomaly-based approach:-* It produces high false alarm rate.
- *Specification-based approach:-* It is often difficult to specify completely and accurately the entire set of valid behaviours a system should exhibit.

The above limitations from the existing works motivate us to built new malware analysis framework using machine learning technique which can detect and classify a file as benign or malware efficiently and effectively.

In the next section, we discuss our proposed framework for malware analysis in details.

3 PROPOSED MALWARE ANALYSIS TECHNIQUE

In this section, we present our proposed methodology for detecting and classifying a given sample of a file. Cuckoo Sandbox [3] is used to generate static and dynamic analysis report. We have developed ReportHandler module in Java for extraction of relevant features and for generation of training and testing dataset. The dataset comprises of a Macro dataset for detection and a Micro dataset for classification. Weka framework is used for detection and classification of given dataset based on different machine learning algorithms. The flow chart illustrated in Figure 1 indicates the operational flow of the proposed methodology.
It includes the following four phases:

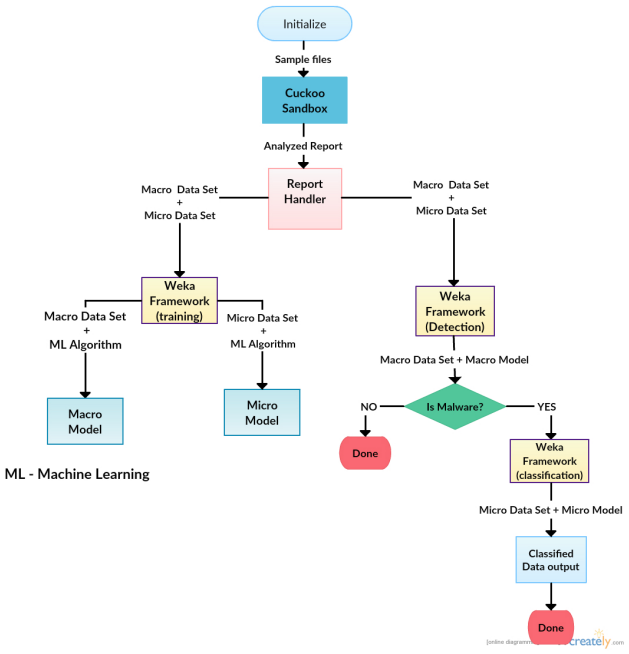

Figure 1: Operational Flow Of Proposed Methodology

- *Data Collection:* In this phase, clean files and malware files are scrapped from the Internet using Python Script.
- *Analysis Phase:* Here, Cuckoo Sandbox is used for analyzing the files.
- *Report Handling:* ReportHandler module has been developed in Java for extraction of relevant features and generation of training and testing dataset.
- *Detection and Classification:* In this phase, Weka Framework is used for generating a model based on machine learning algorithms to test the new files.

The detailed description of these phases are presented in the following subsections.

3.1 Data Collection

One of the most challenging tasks during machine learning process is to define a comprehensive training dataset. In order to train the model, one needs classified dataset along with a well-known learning algorithm and a periodic learning supervised process. Due to unavailability of a comprehensive dataset for malware analysis, we have built a dataset from scratch which will serve as a base point for malware analysis using ML algorithms.

Online site OpenMalware [10] is used to extract existing malware based on their classification like rootkit, adware, spyware, virus, Trojan, worm, backdoor, hijacker, etc. Clean files, e.g., pdf, exe, msi, php, etc., were downloaded from Softonic sites [2]. In order to make the collection of data process automated, a python script was developed to scrap malware files from sites automatically. The malware files downloaded were named with a proper format to make it descriptive so that this information can be used during the training phase for generating comprehensive data. Format for malware files is like *[malwaretype][index].extension* (e.g., spyware1.exe) and for clean files format is like *[clean_][filename].extension* (e.g clean_codeblocks.exe).

3.2 Analysis Phase

We have used Cuckoo Sandbox for the analysis of a file. It automatically runs, analyzes files and collects comprehensive analysis results that outline what the malware does while running inside an isolated operating system. Cuckoo Sandbox consists of a central management software which handles sample execution and analysis. Each analysis is launched in a fresh and isolated virtual or physical machine. The main components of Cuckoo's infrastructure are a Host machine (the management software) and a number of Guest machines (virtual or physical machines for analysis). The Host runs the core component of the sandbox that manages the whole analysis process, while the Guests are the isolated environments where the malware samples get actually safely executed and analyzed.

The analysis phase involves two sub phases as follows:

(1) Sandbox configuration
(2) Malware analysis lab set up

These sub phases are discussed in details as follows

3.2.1 Sandbox Configuration.

To get the malware behavioural reports and to ensure that malware samples runs correctly, including all of its functionality, it is important to configure Cuckoo Sandbox. In the real world, different malware samples exploit different vulnerabilities that might be part of certain software products. Therefore, it is important to include a broad range of services in the virtual machines created by the sandbox. The hypervisor used for the virtual machines for Cuckoo is Virtualbox. The specification of a Virtual machine is:-

- 1 CPU core 3.2 Ghz
- 1 GB RAM
- Internet connection

Installed softwares on Virtual machine are:-

- Windows 7 Professional 64bit without any updates.
- Adobe PDF Reader
- python

3.2.2 Malware analysis lab set up.

The malware analysis environment as shown in *Figure 2* was created. The Cuckoo agent is installed in the guest machine in the startup menu. The Cuckoo host is installed on the host machine. Configuration setup for Cuckoo in the host is done accordingly to the Virtual machine which will be used for execution of sample. For communication between Virtual machine and Cuckoo host, Virtual Box only Adapter (Vboxnet0) is used where as for communication between Cuckoo guest (XP Virtual machine) and the internet, NAT

adapter is used. The initial state of the Virtual machine(clean state not infected by any malware) is saved as a snapshot. For starting analysis on any file, Python script cuckoo.py (cuckoo host) is executed with root privilege. Once Cuckoo host started, we can submit files for analysis in a Virtual machine as specified in Cuckoo configurations. When a sample is submitted, Cuckoo sandbox executes the files in the Virtual machine in the clean state and monitor each action happening in the virtual environment to the Cuckoo host and generates a report for each sample. The Analyzed Report "AR" can be retrieved using web interface as well as using API Calls. ReportHandler is used for retrieving the AR as discussed in the following subsection to generate Macro and Micro datasets.

Useful Commands used during the analysis phase are:

- $ sudo python cuckoo.py - (1)
- $ sudo python submit.py /home/cuckoo/Malware -(2)
- $ sudo python api.py -(3)

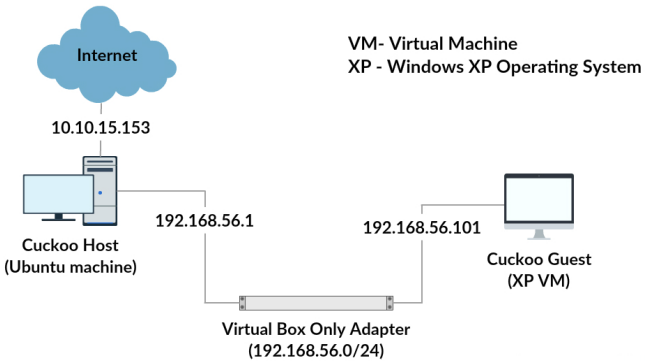

Figure 2: Malware Analysis: Environment Design and Architecture

3.3 ReportHandler

In our proposed solution, a Java-based module was developed for feature extraction and generation of training as well as testing datasets for the experimentation using the analysis report which was generated by the Cuckoo Sandbox. ReportHandler is a maven project in Java that uses HTTP request to serve the required task. The analysis report "AR", generated for the individual sample is submitted to ReportHandler using HTTP request with AR Id. AR is basically JSON (JavaScript Object Notation) format document which has all analysis information. Reporthandler contains a parsing algorithm which parses the AR. In addition, it extracts the relevant features from AR which is useful for detection and classification. These features are stored in a global variable that is Currentfeatures. Currentfeatures is nothing but a vector/list/array of FeatureDetails. Listing 1 shows the definition of FeatureDetails class.

Our proposed ReportHandler module runs in three steps, i.e.,

(1) Feature Extraction
(2) Feature Representation
(3) Generation of training and testing datasets

These steps are explained in details as follows.

Listing 1: Feature Details Class

```
class FDetail{
    int id;
    double count;
    String type;//countable or type
}
```

3.3.1 Feature Extraction.

In this step, we focused on extracting the features from analysis report "AR", which is generated by Cuckoo Sandbox in the analysis phase. API call states the exact action which is executed in the machine like creation, deletion, access, and modification of files as well as registry key related action and internet protocol level information that is used for connecting to the Internet. API call which was invoked during execution of samples with few details and existing signature information as detected by the Cuckoo Sandbox is used for the creation of feature sets. The AR report is of JSON format. A parsing algorithm is used to parse AR report to get feature sets as call|(category)|(API name) format (e.g., call|registry|RegOpenKeyExA). During execution, some sample may imports library for which feature is extracted as Modules|(basename) format (e.g., modules|cryptdll.dll).

Some of the useful information which are retrieved as a feature is CuckooScore, CuckooIsMalware, CuckooMalwareType. CuckooScore is a score which Cuckoo Sandbox gives to a file by checking the severity of different actions performed during executions. By observation, a threshold on the score is defined to classify a file as malicious or benign according to the Cuckoo Sandbox. CuckooIsMalware is set as 'yes' if the score is greater than 1.0 else it is set as 'no'. Cuckoo Sandbox uses a database of signature from Virustotal to generate comprehensive information (e.g., which antivirus marks the file as benign or malicious and the type of malware) about executed sample. This Virustotal information is also available in AR report. CuckooMalwareType is set to the malware type whose frequency in Virustotal information available in AR report is maximum. The malware type contains adware, spyware, virus, worm, trojan, rootkit, backdoor, keylogger, ransomware, hijacker, spam, and unknown. As the parsing algorithm proceeds, it updates the CurrentFeatures (Global variable) which is later used for the creation of training or testing dataset.

3.3.2 Feature representations.

For each sample, it maintains a vector of different features containing the frequency of each API call, showing the number of times it triggers. The representation of feature is illustrated in Table 1.

$$\begin{array}{c} \begin{array}{ccccccc} API_1 & API_2 & API_3 & -- & API_{\mathrm{i}} & -- & API_{\mathrm{n}} \end{array} \\ \begin{pmatrix} 100 & 1 & 76 & & | & & 234 \\ 20 & 29 & 0 & & | & & 143 \\ 45 & 0 & 19 & & | & & 10 \\ -- & -- & -- & & -- & & -- \\ 0 & 40 & 70 & & | & & 16 \end{pmatrix} \begin{array}{c} S_1 \\ S_2 \\ S_3 \\ S_{\mathrm{j}} \\ S_{\mathrm{m}} \end{array} \end{array}$$

Table 1: Feature Representations

In Table 1, the horizontal axis represents the samples and the vertical axis represents the API call, where each number represents a number of times the API call was triggered. This approach clearly provides more details resulting in better accuracy.

3.3.3 **Generation Of training and testing datasets**. Training and testing dataset is developed using the Current-Features in ReportHandler. Each instance of dataset contains features in a vector form with their frequency or type. We developed two types of datasets in our approach, i.e.,

(1) *Macro Dataset:-* Macro dataset is used for the detection of a sample as a malware or benign. It is used for developing Macro Model using machine learning algorithm in training phase. Macro dataset of a sample is shown in Table 2.
(2) *Micro Dataset:-* Micro dataset is subset of Macro dataset which is used for the classification of a sample as a different form of malware. It is used for developing Micro Model using machine learning algorithm in training phase. Macro dataset of a sample is shown in Table 3.

3.4 Detection and Classification

In our proposed methodology, we have used Weka Framework [8] for detection and classification of malwares. Two datasets namely Macro and Micro (.ARFF, i.e., Attribute-Relation File Format data) are created using ReportHandler using the comprehensive feature for the development of machine learning model in Weka. The training datasets were imported in Weka Framework to produce model using different machine learning algorithms (e.g., SMO, J48 Decision tree, Random Forest tree).

Two models developed in Weka are:-

(1) *Macro Model:-* Macro model is used for detecting a given sample as malware or benign using its feature vector or dataset.
(2) *Micro Model:-* Micro Model is used for classifying a given sample in different forms of malware using its feature vector or dataset.

In the next section, we experimentally evaluate our proposed framework with sufficient number of sample files and report the accuracy and efficiency of the propose solution.

4 EXPERIMENTAL RESULTS AND DISCUSSIONS

This section presents the efficacy of our proposed framework with experimental results. We have used 220 samples of data containing malicious as well as benign files for our experiment. For malware analysis, we divide them into training and testing set. Training set contains 60% of malware samples and testing set contains 40% of malware samples. We have observed results for classification and detection problem using the two models developed for each of the machine learning algorithms: *(J48 decision tree, Random Forest, SMO)*. Results generated by Weka Framework contains detailed accuracy of each class and confusion matrix.

The results of the experimental evaluation are explained in the following subsections.

4.1 Weka Data Analysis

The analysis results of Weka Framework are as follows. Weka results output:

- TP (True Positive): number of examples predicted positive that are actually positive
- FP (False Positives): number of examples predicted positive that are actually negative
- TN (True Negatives): number of examples predicted negative that are actually negative
- FN (False Negatives): number of examples predicted negative that are actually positive
- Recall is the TP rate (also referred to as sensitivity) i.e.,the fraction of those that are actually positive were predicted positive :

$$recall = \frac{TP}{TP + FN}$$

- Precision is defined as what fraction of those predicted positive are actually positive.

$$Precision = \frac{TP}{TP + FP}$$

- Accuracy is defined as:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

- F Measure: It is a measure that combines precision and recall is the harmonic mean of precision and recall, the traditional F-measure or balanced F-score. It is defined as:

$$F = 2 * \frac{precision * recall}{precision + recall}$$

- Confusion Matrix : A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known.

4.2 Malware Detection Results

The detail assessment for malware detection technique using proposed methodology is illustrated in *Table 4*. High detection rate with an accuracy of 100% using J48 decision tree model, 99% using SMO and 97% using Random Forest tree has been

Table 2: Macro Dataset: Instances of Sample Feature

Feature ID (type)	CuckooScore (number)	CuckooIsMalware (bool)	ResultIsMalware (bool)	call—file—NtReadFile (countable)	- -	modules—USP10.dll (type)
Instance1	2.4	yes	yes	16	- -	2
Instance2	0.8	no	no	1	- -	1
Instance3	4.0	yes	yes	644	- -	1

Table 3: Micro Dataset: Instances of Sample Feature

Feature ID (type)	CuckooScore (number)	CuckooMalwareType (bool)	ResultMalwareType (bool)	call—file—NtReadFile (countable)	- -	modules—USP10.dll (type)
Instance1	2.4	trojan	trojan	16	- -	2
Instance2	4.0	adware	adware	644	- -	1

Table 4: Malware detection results using our proposed method

Weka with ML Algorithms used	TP Rate	FP Rate	Accuracy	Precision	Recall	F-measure
Random Forest tree	0.977	0.130	97.6852%	0.977	0.977	0.976
J48 Decision tree	1.000	0.000	100%	1.000	1.000	1.000
SMO	0.995	0.001	99.537%	0.996	0.995	0.995

Table 5: Malware classification Results using our proposed method

Weka with ML Algorithms used	TP Rate	FP Rate	Accuracy	Precision	Recall	F-measure
Random Forest tree	0.667	0.146	66.6667%	0.641	0.667	0.624
J48 Decision tree	1.000	0.000	100%	1.000	1.000	1.000
SMO	0.910	0.027	91.0053%	0.891	0.910	0.900

Table 6: Comparative Performance Study in Different Applications

Types of Cloud Application Level Required	Time Complexity	Accuracy	Precision
Enterprise Cloud	✓	✗	✗
E-government cloud	✓	✓	✗
Business Oriented Cloud	✗	✓	✓

observed during the experiment. Accuracy of decision tree is 100% due to the fact that it is making decision based on the feature CuckooIsMalware and CuckooMalwareType which is generated by Cuckoo sandbox.

4.3 Malware Classification results

The detail assessment for malware classification technique using proposed methodology is illustrated in *Table 5*. Effective classification rate with an accuracy of 100% using J48 decision tree model, 91% using SMO and 66.67% using Random Forest tree has been observed during the experiment. Accuracy of decision tree is 100% due to the fact that sample has not much data.

4.4 Comparative Performance Study in Different Applications

The comparative performance study in different cloud applications is illustrated in *Table 6*. As per time, Random Forest tree will take much time than SMO as it is multiple SMO itself. If we compare SMO and J48 with respect to time taken, SMO is faster than J48. The results show that our proposed framework with SMO approach works well in Enterprise cloud applications. This is because of its efficiency (time complexity). J48 or SMO is recommended for E-government cloud. This is due to the better accuracy and efficiency of these approach. On the other hand, J48 is for business critical applications.

5 CONCLUSION AND FUTURE WORKS

In this paper, a novel malware analysis framework has been developed for dynamic and static analysis of samples based on similarity in behaviour of the sample. API calls represent the whole behaviour of an application and its state transition. So API names, imported library and existing signature is utilized as a feature to generate the sample datasets.

In the next step, each sample is modeled as a vector of these features and various classification techniques have been applied on the datasets. Experimental results demonstrate acceptable performance of the proposed procedures in detecting and classifying malicious files using machine learning model in Weka. We have found that J48 Decision tree shows best performance.

In this paper, we have considered only 220 sample of file for analyzing which may be biased, because not all the features may have incorporated using this number of samples. In future, we will add more datasets so that we will get extensive set of features for visualizing the performance on broad spectrum. Feature Selection can be used in future using information contained by each feature so that unused features can be discarded to get fine set of features.

REFERENCES

[1] P. M. Comar, L. Liu, S. Saha, P. N. Tan, and A. Nucci. 2013. Combining supervised and unsupervised learning for zero-day malware detection. In *2013 Proceedings IEEE INFOCOM*. 2022–2030. https://doi.org/10.1109/INFCOM.2013.6567003

[2] Tomas Diago. 2014. Software and app discovery portal. Available at: https://en.softonic.com/. (2014). [Online].

[3] Cuckoo Foundation. 2014. Automated Malware Analysis - Cuckoo Sandbox. Available at: http://www.cuckoosandbox.org/. (Feb 2014). [Online].

[4] D. Gavrilut, M.Cimpoesu, D.Anton, and L.Ciortuz. 2009. Malware detection using machine learning. In *2009 International Multiconference on Computer Science and Information Technology*. 735–741. https://doi.org/10.1109/IMCSIT.2009.5352759

[5] The AV-TEST Institute. 2017. current malware statistics. Available at: https://www.av-test.org/en/statistics/malware/. (May 2017). [Online].

[6] Prof. M. P. Wankhade Jyoti Landage. 2013. Malware and Malware Detection Techniques : A Survey. Available at: http://www.ijert.org/view-pdf/6744/malware-and-malware-detection-techniques--a-survey/. (December 2013). [Online].

[7] R. Mosli, R. Li, B. Yuan, and Y. Pan. 2016. Automated malware detection using artifacts in forensic memory images. In *2016 IEEE Symposium on Technologies for Homeland Security (HST)*. 1–6. https://doi.org/10.1109/THS.2016.7568881

[8] University of Waikato. 2014. Weka : Data Mining Software in Java. Available at: http://www.cs.waikato.ac.nz/ml/weka/. (2014). [Online].

[9] R. S. Pirscoveanu, S. S. Hansen, T. M. T. Larsen, M. Stevanovic, J. M. Pedersen, and A. Czech. 2015. Analysis of Malware behavior: Type classification using machine learning. In *2015 International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA)*. 1–7. https://doi.org/10.1109/CyberSA.2015.7166115

[10] Danny Quist. 2014. malware sample with category. Available at: http://openmalware.org/. (2014). [Online].

[11] Z. Salehi, M. Ghiasi, and A. Sami. 2012. A miner for malware detection based on API function calls and their arguments. In *The 16th CSI International Symposium on Artificial Intelligence and Signal Processing (AISP 2012)*. 563–568. https://doi.org/10.1109/AISP.2012.6313810

[12] Symantec. 2017. Internet Security Threat Report. Available at: https://www.av-test.org/en/statistics/malware/. (2017). [Online].

[13] Wikipedia. 2017. Sandbox (computer security). Available at: https://en.wikipedia.org/wiki/Sandbox_(computer_security)/. (2017). [Online].

[14] Wikipedia. 2017. Weka (machine learning). Available at: https://en.wikipedia.org/wiki/Weka_(machine_learning)/. (2017). [Online].