A Novel Malware Analysis Framework for Malware Detection and Classification using Machine Learning Approach

Conference Paper · January 2018

DOI: 10.1145/3154273.3154326

4 authors, including:

Kamalakanta Sethi
Indian Institute of Technology Bhubaneswar
5 PUBLICATIONS **5** CITATIONS
SEE PROFILE

Shankar Chaudhary
Indian Institute of Technology Bhubaneswar
2 PUBLICATIONS **0** CITATIONS
SEE PROFILE

A Novel Malware Analysis Framework for Malware Detection and Classification using Machine Learning Approach

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, the digitization of the world is under a serious threat 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 become an obsolete method. The efficiency of the machine learning techniques in context to the detection of malwares has been proved by state-of-the-art research works. In this paper, we have proposed a framework 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.). Our solution uses Cuckoo Sandbox for generating static and dynamic analysis report by executing the sample files in the virtual environment. In addition, a novel feature extraction module has been developed which functions based on static, behavioral and network analysis using the reports generated by the Cuckoo Sandbox. Weka Framework is used to develop machine learning models by using training datasets. The experimental results using the proposed framework shows high detection rate and high classification rate using different machine learning algorithms

CCS CONCEPTS

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

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. 2018. A Novel Malware Analysis Framework for Malware Detection and Classification using Machine Learning Approach . In *ICDCN '18: 19th International Conference on Distributed Computing and Networking, January 4–7, 2018, Varanasi, India.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3154273.3154326

1 INTRODUCTION

Malicious software, also known as malware, is one of the major threats on the Internet today. Different types of computer applications are downloaded by users via Internet at large scale. Intruders are using the Internet for black marketing where they develop malwares for violating system security. This provides a strong incentive for the intruders to modify and increase the complexity of the malicious code in order to improve the confusion to decrease the chances of being detected by the anti-virus programs. As a result, the vulnerability of using the Internet is increasing day by day due to increasing threats of Malware[3], which get shipped within the software and files via Internet.

Malware is a malicious program which is used to breach the systems security policy with respect to confidentiality, integrity and availability of data. Malwares are of different types like virus, Trojan horse, spyware, rootkit,trapdoor, etc. according to their way of imposing threats to the system. According to AV-Test[3] in March 2017, the total number of malwares is rising 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 affect the systems security perimeter. As per the researches, the malware detection system consists of two tasks [4], i.e., (i) Malware Analysis and, (ii) Malware Detection.
(i)Malware Analysis:- It deals with static analysis, dynamic analysis and hybrid analysis.
(ii) Malware Detection:- Two well known detection techniques, i.e., Signature based and Behavioral based techniques are used. However, signature based techniques fail to detect Zero-Day attack. It is also not able to detect complex emerging malwares. On the other hand, using behaviour based techniques, it is very difficult to specify completely the entire set of valid behaviours a system should exhibit.

The above limitations from the existing techniques motivate us to build a malware analysis solution using machine learning technique in order to identify a file as benign or malware. In this paper, we have proposed an intelligent malware analysis framework to detect and classify the malwares effectively and efficiently.

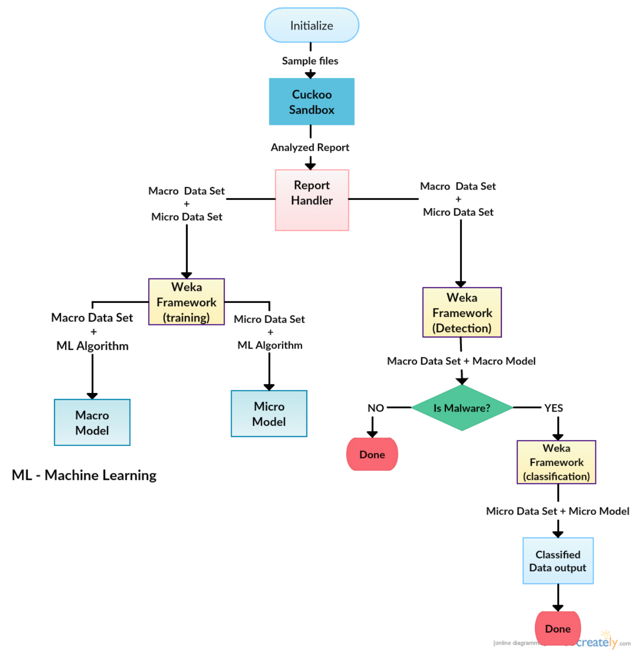
Initialize
Sample files
Cuckoo Sandbox
Analyzed Report
Report Handler
Macro Data Set + Micro Data Set
Macro Data Set + Micro Data Set
Weka Framework (training)
Macro Data Set + ML Algorithm
Micro Data Set + ML Algorithm
Macro Model
Micro Model
ML - Machine Learning
Weka Framework (Detection)
Macro Data Set + Macro Model
Is Malware?
NO
YES
Done
Weka Framework (classification)
Micro Data Set + Micro Model
Classified Data output
Done

Figure 1: Operational Flow Of Proposed Methodology

2 RELATED WORK

This section presents significant researches relevant to malware analysis. Since malware is becoming more complex to detect using the static or code analysis, researchers have attempted to extract the dynamic information by executing the files in virtual environment. This technique is also known as sandboxing [10]. Cuckoo [2] is a tool that allows malware analysis using sandboxing. It is used to automatically execute, analyze files, and collect comprehensive analysis results. These results outline the detailed log information of the flow path of the malwares inside an isolated operating system and retrieve traces of API calls, information related to registry and file modification and network traffic logs. Researchers [5, 7] have used Cuckoo Sandbox for malware analysis.

Another work [5] focused on the memory images in order to extract features like registry activities, 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. [9] implemented API calls with deep analysis using passed argument. In addition, they tried to analyze and classify large number of malwares. In another work, Pirscoveanu et al. [7], used a combination of features to achieve high classification rate. In the next section, we discuss our proposed solution for malware analysis in details.

3 MALWARE ANALYSIS FRAMEWORK

In this section, we present our proposed methodology for detecting and classifying a given sample of a file. The flow chart illustrated in Figure 1 indicates the operational flow of the proposed methodology. It includes the following four phases, (i)Data Collection,(ii) Analysis Phase,(iii) Report Handling, and (iv) Detection and Classification. 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 sites OpenMalware [8] and Softonic [1] are used to download malwares and clean files respectively.

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, i.e.(i) Sandbox configuration,and (ii) Sandbox configuration. 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 and internet connection. Installed softwares on Virtual machine are Windows 7 Professional(64 bit) ,Adobe PDF reader and pyton.

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.

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. Our proposed ReportHandler module runs in three steps, i.e.,(i) Feature Extraction, (ii) Feature Representation and (iii) Generation of training and testing datasets.These steps are explained in details as follows.

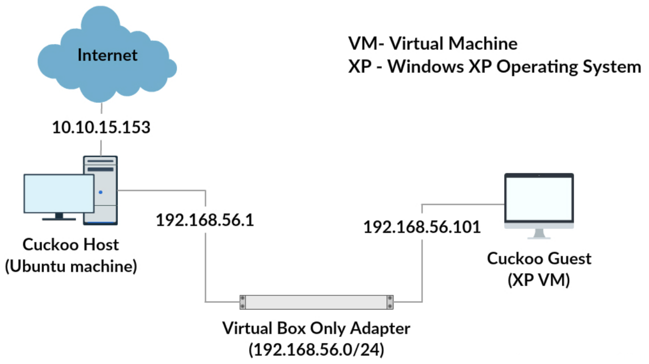

Figure 2: Malware Analysis: Environment Design and Architecture

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. 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.

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.

3.3.3 ***Generation Of training and testing datasets.*** Training and testing dataset is developed using the CurrentFeatures 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.,(i) Macro Dataset and (ii) Micro Dataset.
(i)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 1.
(ii)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 2.

3.4 Detection and Classification

In our proposed methodology, we have used Weka Framework [6] for detection and classification of malwares. Two datasets namely Macro and Micro 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.

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

In this section, the efficacy of our proposed framework is discussed with adequate experimental results. We have considered 220 samples of data containing malicious as well as benign files. For malware analysis, the datasets are divided them into training and testing set. Training and testing sets contain 60% and 40% of malware samples respectively. 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*). 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 samples predicted positive that are actually positive

Table 1: 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

Table 2: 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 3: 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 4: 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

- FP (False Positives): number of samples predicted positive that are actually negative
- TN (True Negatives): number of samples predicted negative that are actually negative
- FN (False Negatives): number of samples predicted negative that are actually positive
- Recall: the TP rate (also referred to as sensitivity), i.e., the fraction of those that are actually positive were predicted positive
- Precision: the fraction of those predicted positive are actually positive
- Accuracy: the number of samples predicted correctly (i.e., TP+NP) over total number of samples taken for testing(i.e., TP + TN + FP + FN)
- F-Measure: the harmonic mean of precision and recall

4.2 Malware Detection Results

The detailed assessment of malware detection technique using proposed methodology is illustrated in *Table 3*. High detection rate with an accuracy of 100% using J48 decision tree model, 99% using SMO and 97% using Random Forest tree has been observed in the experiment. Accuracy of decision tree is reported as 100% due to the fact that the decision is made based on the feature CuckooIsMalware and CuckooMalwareType which is generated by Cuckoo sandbox.

4.3 Malware Classification results

Table 4 illustrates the detailed assessment of malware classification technique using proposed methodology. Effective classification rate with an accuracy of 100%, 91% and 66.67% using J48, SMO and Random Forest tree respectively has been observed in the experiment. It is reported that decision tree yields accuracy of 100% because of small sample size.

5 CONCLUSION AND FUTURE WORKS

In this paper, a novel intelligent malware analysis framework has been developed for dynamic and static analysis of malware samples based on similarity in their behaviour. Experimental results demonstrate acceptable performance of the proposed procedures in detecting and classifying malicious files using machine learning models in Weka. We have observed that J48 Decision tree shows the best performance in terms of accuracy and precision. We have considered only 220 samples of files for analysis which may be biased, because not all the features may have incorporated using these 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.

REFERENCES

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

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

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

[4] 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].

[5] 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

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

[7] 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

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

[9] 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

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