Malware Detection Based On Opcode Frequency

Thesis submitted in partial fulfillment of the requirements for the award of degree of

Masters of Engineering

in

Information Security

Submitted by

Abhijit Yewale

Roll no 801433030

Under the supervision of

Dr Maninder Singh

Associate Professor

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

THAPAR UNIVERSITY

PATIALA 147004

July 2016

CERTIFICATE

I hereby certify that the work which is being presented in the thesis entitled, *"Malware Detection Based on Opcode Frequency"*, in partial fulfillment of the requirements for the award of degree of Master of Engineering in Information Security submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of *Dr. Maninder Singh* and refers other researchers work which are duly listed in the reference section.

The matter presented in the thesis has not been submitted for award of any other degree of this or any other University.

(Abhijit Yewale)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

(Dr. Maninder Singh)
(Associate Professor, CSED)

Countersigned by:

(Dr. Maninder Singh)
Head and Associate Professor
Computer Science and Engineering Department
Thapar University
Patiala

(Dr. S. S. Bhatia)
Dean (Academic Affairs)
Thapar University
Patiala

ii

ACKNOWLEDGEMENT

Words are often too less to reveals one's deep regards. An understanding of the work like this is never the outcome of the efforts of a single person. I take this opportunity to express my profound sense of gratitude and respect to all those who helped me through the duration of thesis.

This work would not have been possible without the encouragement and able guidance of my supervisor, Dr. Maninder Singh, Associate Professor, CSED, Thapar University, Patiala. Most of the novel ideas and solutions found in this thesis are the result of our numerous stimulating discussions. Their feedback and editorial comments were also invaluable for the writing of this thesis.

I would also like to thank all the faculty members of the department and my friends who directly or indirectly helped me in completion of my thesis. No words of thanks are enough for my dear parents and sisters whose support and care makes me stay on earth. Thanks to be with me.

Above all, I would like to thank the Almighty God for his blessings and for driving me with faith, hope and courage in thinnest of the times.

Abhijit Yewale
801433030
Place: Thapar University, Patiala

ABSTRACT

Malware is a computer program or a piece of software that is designed to penetrate and detriment computers without owners permission. There are different malware types such as viruses, rootkits, keyloggers, worms, trojans, spywares, ransomware, backdoors, bots, logic bomb, etc. Volume, Variant and speed of propagation of malware is increasing every year. Antivirus companies are receiving thousands of malware on the daily basis, so detection of malware is complex and time consuming task.

Malware detection means detection of malware using different malware detection tools such as antivirus, Intrusion detection system, etc. Malware detection system means checking whether the software has malicious intent or not. There are many malware detection techniques like signature based, behavior based and machine learning based detection techniques, etc. The signatures based detection system fails for new unknown malware. In case of behavior based detection, if the antivirus program identify attempt to change or alter a file or communication over Internet then it will generate alarm signal, but still there is a chance of false positive rate. Also the obfuscation and polymorphism techniques are hinderers to the malware detection process.

In this research we introduce a method to detect malware using the concept of opcode frequency in the portable executable file format. This research applied machine learning algorithm to find True Positive Rate, Recall, Accuracy, False Positives, Specificity, False Negatives, True Negative Rate, True Positives, Sensitivity and True Negatives for malware and got 96.67 per cent success rate.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter provides overview of malware types and malware detection methods. It briefly describes research motivation for malware detection and structure for the rest of the thesis.

1.1 Malware

Malware terminology came from merging two words malicious and software. It a is computer software which is designed to harm or disrupt the operation of a computer system. There are different types of malware exist which ranges from stealing information to vandalism of a system.

1.2 Malware Types

Malwares are categorized into different types based on the functionality performed by malware. The characteristics related to malware includes: self replication; parasite behavior and population growth. The self replication of malware is increases by generating replica of self. The overall increase in malware instance due to self replication property of a malware is called as population growth. If malware does not replicate it self then it has zero population growth, i.e. malware is having a zero population rate. Parasitic malware are dependent malware, they are dependent on other executable

file to exist.

1.2.1 Logic Bomb

It is a malware code which contains two parts one is payload and other trigger. Trigger means condition on which payload will get executed and payload is nothing but an action that needs to perform. It is not a self replicating and having zero population growth, also it is possibly falls into parasitic [27].

1.2.2 Trojan Horse

Trojan horse is one type of malware that is frequently spread as legitimate software but its actual purpose is to steal information from user and send it to remote machine. The term Trojan horse came from ancient story of wooden horse that is used by Greek to enter into Troy city to win the war. It is not a self replicating and have zero population growth. It is a dependent malware [27].

1.2.3 Back Door

Back door is technique which bypasses the normal authentication in a computer system. Generally programmer create back doors in software for legitimate reason to perform fast maintenance work. These are not a self replicating and have zero population growth [27].

1.2.4 Virus

It is a one kind of malware, when run it will try to infect other executable code. After successful modification, the executable file is called as infected file. It is a self replicating and having positive population growth. Virus requires some action to get executed, that means it is parasitic [27].

1.2.5 Worm

Worms are standalone malware. They are not dependent on other executable file to get executed. They can get distribute from machine to machine through network.. The worm terminology had firstly used in 1975 by John Brunner in the novel of science fiction "the shockwave rider". It follows a self replicating property and have positive population growth [27]. To understand the problem created by computer worm you need to understand classification of it [29]

1.2.6 Rabbit

Rabbit is a one type of malware that multiplies rapidly. There are two kinds of rabbit malware one which try to exhaust system resource such as disk space, for e.g. fork bomb. The second type of Rabbit malware is standalone software which replicates across network, from one computer to another computer but deletes its actual replica after duplication. It is a self replicating and have zero population growth. It does not depend on other executable code to get executed [27].

1.2.7 Spyware

It is software which gathers information from personal computer and send it to the remote system. This software gathers information passively. It is not a self replicating and having zero population growth. It does not depend on other executable code to get executed [27].

1.2.8 Adware

Adwares collect information about the user behavior and their habits. Generally adwares are marketing focused, they redirect users browser to some other website in the hope of sale. It is not a self replicating and have zero population growth. It does not depend on other executable code to get executed [27].

1.2.9 Hybrids, Droppers and Blended Threats

Hybrid is a one kind of malware which has characteristics of different malware. Dropper is a malware which drops other malware. For example worm propagates itself by dropping Trojan horse onto the compromised machine. Technical vulnerabilities are exploited by blended threats [27].

1.2.10 Zombies

Zombie are computer system which are in compromised state, these can be used by attacker for their purpose like for sending spam mail. Mostly zombies are used in Distributed Denial of Service Attack(DDOS) [27].

1.3 Malware Detection Techniques

Malware detection is a technique that attempts to find whether a program has malicious intent or not [5]. There are different malware detection techniques such as signature based malware detection, specification based detection, anomaly based detection and machine leaning based detection.

1.3.1 Signature Based Malware Detection

In signature based malware detection, antivirus program looks for signature which is nothing but sequence of bytes in a particular file to declare the file as malicious [7]. It is difficult to detect polymorphic, metamorphic and unknown malware using signature based malware detection system. In Basic malware, Program entry point has changed to malicious payload. So detection is relatively easy as compare to other type of malware detection. Figure 1.1 shows Basic malware structure. Figure 1.1 shows that original code is separate from malicious code also entry block describes entry point. Malware authors modify the original entry point to entry point where malicious code has started.

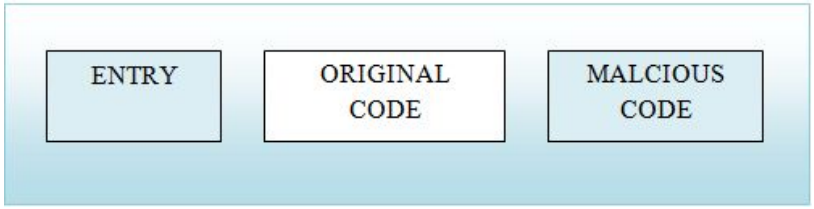


Figure 1.1: Simple Malware Structure

Polymorphic malware is an encrypted malware which changes the decryptor loop on each infection without changing actual code for malware. It has infinite number of decryptor loop variation. Polymorphic virus contains encrypted virus code along with decryptor module. The polymorphic engine creates new variant each time it is executed [7]. So it is difficult for antivirus companies to create signature because on every infection, decryptor loop changes. Figure 1.2 shows polymorphic malware structure.

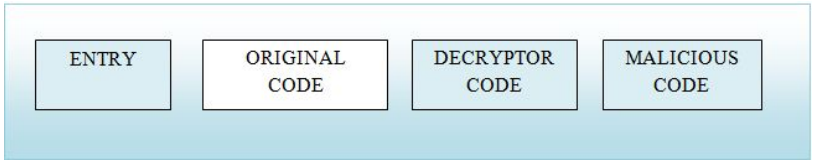


Figure 1.2: Polymorphic Malware Structure

Metamorphic virus is polymorphic in virus body and on each infection it changes virus body. It is not an encrypted virus hence no decryptor loop. Metamorphic virus can recode itself using different obfuscation techniques, so that the children never look similar to parent. These kinds of malware evade malware detection system because each new instance has different signatures. So it is difficult to create signature for such kind of malware. Figure 1.3 shows metamorphic malware having different signature for different variants.

Figure 1.3 shows that how the actual Virus A changes to FORM A and from FORM A to FORM B, and so on. After every infection virus changes its code, hence each form of virus will have different signature such as S1, S,.,.,Sn.

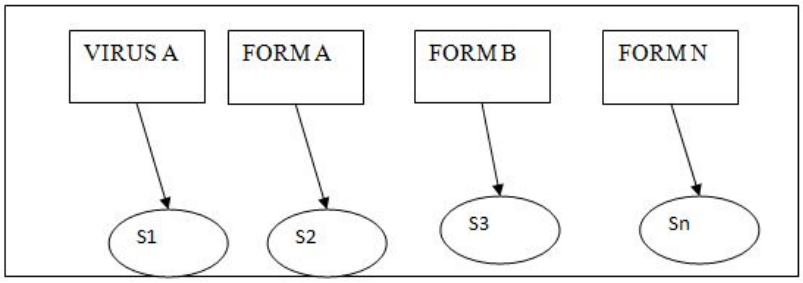


Figure 1.3: Metamorphic Malware Process

1.3.2 Anomaly Based Malware Detection

Anomaly based system detects any misuse of computer that falls out of the regular activity of a computer [8]. It monitors the computer program or software activity and classifies it as either normal or anomalous. Anomaly Based Malware Detection works in two phases is Training and Monitoring (Detection) Phase. During training phase, the malware detector is learning host machines normal behavior. Major advantage of anomaly based detection is that it is capable to detect Zero-Day attack. The loop hole in the software is exploited by the hacker and it is unknown to a vendor is called as Zero-Day attack. There is no signature available in the market for such exploits hence it is called as Zero-Day Exploit. The limitation of this method is high false positive rate.

The Figure 1.4 shows that why anomaly detection alone is not sufficient to detect malware. In Figure 1.4, A represents set of all behavior of host machine, V represents set of all valid behavior, V' represents set of all invalid behavior and Vapprox represents set of approximation of valid behavior. As often the implementation covers relative requirement and anomaly based detection proximate the implementation. Vapprox is an approximation of valid behavior of host machine that means some valid behavior might be flagged as malicious by anomaly based detection system [28].

1.3.3 Specification Based Malware Detection

It is one type of an anomaly based detection, which mostly focused to reduce the high false positive rate of anomaly based detection. It follows Figure 1.4, because

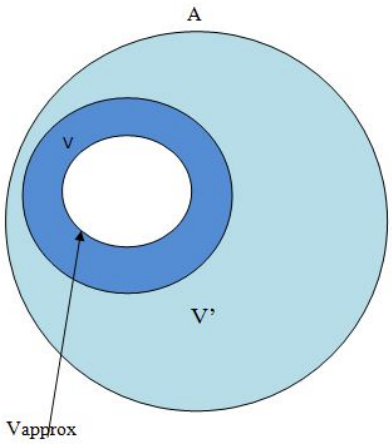


Figure 1.4: Behavior Characteristics for Anomaly Based Detection

it is also one type of anomaly based detection. Instead of trying to proximate the implementation of system or application, specification based detection system trying to proximate the requirements of system or application [28].

1.3.4 Machine learning Based Malware Detection

The detection of unknown malware is a big challenge for malware detection because daily new malware have been coming to antivirus vendor. Data mining approach generally depends upon the machine learning algorithm that uses the data of malicious as well as goodware files to classify the PE file into goodware or malware. Machine learning algorithms are classified into three categories such as supervised, unsupervised and semi-supervised learning.

In supervised machine-learning technique requires training data, should be properly labeled in order to build a model. Algorithms which are under this category are decision tree, SVM, random forest, boosting, etc [26]. Unsupervised machine-learning technique does not require labeled data, in this first clusters of similar data are created using clustering algorithms. Clustering algorithm such as K-means, hierarchical clustering, etc. are belongs to this category. The semi-supervised machine learning is a mixture of labeled and unlabeled data to build a model. The following machine learning algorithms are applied in this research.

1.3.4.1 Decision Tree

Decision tree is a type of machine learning classifier which can be graphically represented as in figure 1.5. Decision tree starts with root node that split into multiple branches. In this each non leaf node represents testing condition that determines which branch to follow. Leaf node contains decision. Each path from root to leaf node represents one rule for a decision tree [25].

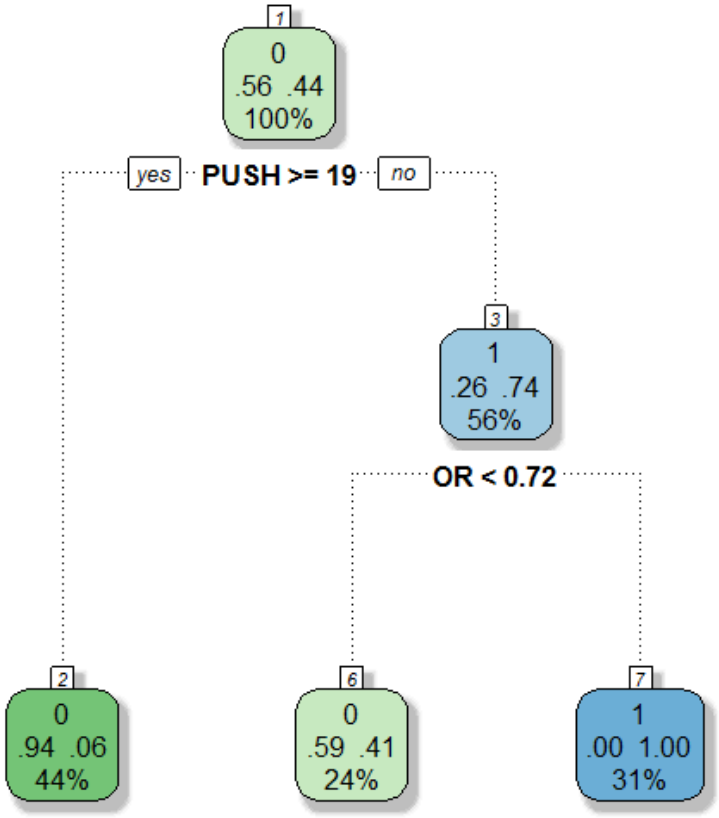


Figure 1.5: Decision Tree

1.3.4.2 Support Vector Machine

"A support vector machine (SVM) searches for so-called support vectors which are observations that are found to lie at the edge of an area in space which presents a boundary between one of these classes of observations (for example squares) and another class of observation (for example circle)"[25]. The space between two classes is

called as Margin. Each region contains observation with same value for target variable. Support vectors are used to identify hyperplane (that is a straight line) which separates the classes. Figure 1.6 depicts a working of a Support Vector Machine model. The x-axis represents WindGustSpeed attribute and y-axis reprsents Pressure9am attributes of weather dataset.The maximum margin between classes should be found, this will represent the model.

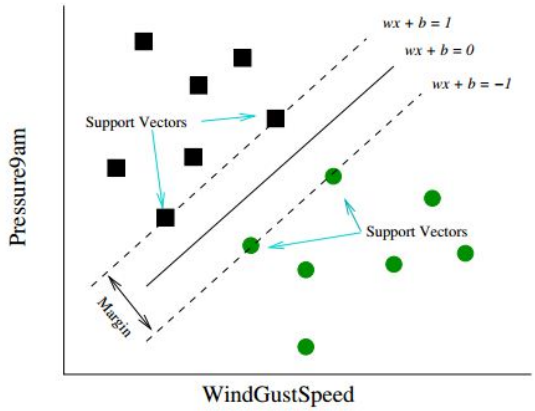


Figure 1.6: Working of Support Vector Machine

1.3.4.3 Random Forest

A single decision tree is a simple model. In data mining, it is clear that if you combine multiple models together it always give better result than single model. The random forest algorithm builds hundreds of decision trees and combines them into a single model. Random forest provides robustness to noise, outlier and over fitting problems as compare to single tree classifier. In this final decision of model will be the decision of majority of constituent trees [25].

1.3.4.4 Boost

The popular variant called as adaptive boosting has been depicted as best of the shelf classifier in the world. Boosting assigns weights to the observation in the dataset and increases the weight for those observations which are harder to classify. In this trees are developed one after another in order to make refinement in previously developed

model. The key concept is that after building one model any observation that wrongly classified by that model were boosted [25].

1.4 Research Motivation

Malware is malicious software which is designed to damage or steal information from computer system. In the last few years, motivation behind creation of malware has changed drastically. Most malware author was financially motivated [6]. There are different types of malware such as Trojan horse, Logic Bomb, Virus, Backdoor, Worms, Rabbit, Droppers, etc. The Intention of malicious software could be to obtain root access or steal confidential information. The malware has been spreading worldwide like some epidemic. Antivirus companies have been receiving thousands of malware on daily basis.

The creation of signature is very difficult task for antivirus vendor because of polymorphic malware. In the era of malware detection one more challenge has increased which is zero-day malware. It is one type of malware whose signature is not available in the market. Researcher have been using reverse engineering techniques, to create signature for malware but still they are facing problem because of encryption and obfuscation techniques. There are different obfuscation techniques namely Instruction Substitution, Dead Code Insertion, Code Transportation and Register Renaming. The malware author uses obfuscation techniques to modify the malware code so that it would be difficult for malware analyzer to understand the malicious intent.

There are different methods to detect malware such as signature based malware detection, behavioral based malware detection, anomaly based malware detection and signature based malware detection. The signature based detection system fails for new unknown malware. In case of behavior based detection, if the antivirus program identify attempt to change or alter a file or communication over internet then it will generate alarm signal, but still there is a chance of false positive rate. Also in case of Anomaly malware detection system set of rules are designed to detect anomalous

behavior of a file. In both behavior and anomaly based detection system requires malware to be executed in the system hence it might affect the other part of the system.

After studying current problems in the area of malware detection, we propose new method to detect malware based on the opcodes frequency. We used machine learning classifier to classify the malware into two category either malware or goodware.

1.4.1 Thesis Outline

Rest of the chapters in this dissertation organized as follows.

Chapter 2- This chapter explain malware detection based on machine learning classifier in sequential manner from the year 2007 to 2015.

Chapter 3- This chapter explains problem statement and objective for this research.

Chapter 4- This chapter explains proposed framework for malware detection. It comprises of malware or goodware selection, sample disassemble, fetch statistics of opcodes, feature selection, model creation and validate Model.

Chapter 5- This chapter explains performance evaluation of models namely Decision Tree, Support Vector Machine, Random Forest, and Ada Boost. It briefly explains about Error Matrix and performance evaluation parameters such as True Positives, True Negatives, False Positives, True Positive Rate, False Positive Rate and Accuracy.

Chapter 6 This chapter presents conclusion and future work for this research work.

Chapter 2

Literature Review

This chapter provides systematic review in the field of malware detection using machine learning classifier. It briefly explains about various malware detection techniques.

2.1 Systematic Review

Malware detection is very important topic over the World Wide Web. Antivirus vendor has been receiving thousands of malware on the daily basis. The research in this area is unstoppable. The survey here stated in the area of malware detection from year 2007 to 2015.

The author discusses malware detection technique for malware through statistical analysis of Opcode frequency distribution. Total 67 malware file was disassembled statically. The Opcode frequency distribution of all malicious file is compared with the 20 non malicious files Opcode frequency distribution. They performed statistical analysis of opcodes distribution using statistical method such as Pearsons chi square procedure, post-hoc standardized testing and Crammers'V. They found malware Opcode frequency distribution differ statistically significant from non-malicious file and rare opcodes are predictor for malware detection [9].

They proposed static malcode detection using decision tree classification algorithm

in chronological point of view. The dataset includes malware from year 2000 to 2007. Training and testing of classifier with malware up to respective year has done and performance has evaluated. They trained classifier with Malicious File Percentage (MFP) 50 or 16 per cent. If the training dataset contains 50 per cent MFP then almost in all years accuracy is below 0.9. The training dataset contains 16 per cent MFP then performance curve is better than 50 per cent MFP. The author concluded that as training data set is updated it will give better result [10].

The concept of text categorization used for detection of malware. They investigated imbalance problem about malicious and benign file. In this binary file parsed and n-gram terms were extracted. The extracted n-gram length was 3-gram, 4-gram, 5-gram and 6-gram respectively. The machine learning classifier used namely Decision Tree, Artificial Neural Network (ANN), Naive Bayes (NB) and Support Vector Machine (SVM) with 3 kernel function. They got accuracy of 95 per cent when the percentage of malicious file is below 20 per cent in training data set [11]. They used opcodes generated by disassembling the executable file, then uses n gram of the opcodes as a feature vector for classification process. Also they used concept of text categorization for detection of unknown malware [12].

The manual malware analysis was not effective and they proposed automated behavior based malware detection using machine learning techniques. The behavior of each malware analyzed in sandbox and its behavior report is generated. This report was preprocessed using sparse vector model. The preprocessed report was given to machine learning classifier for classification. The classifier trained namely Multilayer Perceptron Neural Network, J48 Decision Tree (DT), Support Vector Machine, Nave Bayes and K-Nearest Neighbors (KNN). Out of all classifier J48 Decision Tree gave best performance with accuracy 96.8 per cent [13].

The performance of machine learning is influenced by the feature vector and the algorithm used to generate classifier. Also they combined content based and behavior

based feature. They proposed ensemble learning method called SVM-AR, it combines Support Vector Machine and Association Rules [14].

The supervised machine learning approach was used to detect unknown malware but it require large amount of malicious executables and benign softwares. It required large amount of labeled data. They proposed semi-supervised machine learning approach to detect unknown malware. The method was based on analyzing the appearance of frequency of opcode sequences to create semi-supervised machine learning classifier using a set of labeled and unlabeled data. This method was based on appearance of Opcode sequence frequency and they proved, this approach required less labeled data as compare to supervised machine learning approach [15].

They proposed a framework to deal with Noise in a large training dataset for detection of malware. The Noise in a dataset means incorrectly labeling of data in the dataset. Firstly they established different distance based filtering rules that used to identify various levels of potential noise in training dataset. Secondly they analyzed effect on the performance of classifiers after removing potential noised records. They proved that careful use of distance based filtering rules leads to improvement in the result of malware detection [16].

The author proposed graph feature based method which could be used to train the machine learning classifier. The control flow graph was created for portable executable file. From this control flow graph features were extracted and using this features training dataset created. They build machine learning classifier namely J48 Decision Tree, Random Forest (RF) and Bagging. With the proposed framework they achieved 95.9 per cent malware detection rate and 5.9 per cent false positive rate [17].

They designed modified version of perceptron algorithm while training at low rate still getting low false positive rate. The very low false positive rate was very important as concern with operating system because it would treat clean file as malware.

They also provided a method of optimizing the training speed of an algorithm also maintaining same accuracy. The resultant algorithm was used in ensemble system to increase detection and reduce the false positive rate [18].

The gap between malware coming on users system and development of signature to detect malware could prove cataclysmic for user. They identified seven key feature within Microsoft portable executable file format that could given to machine learning classifier to classify malware or benign file. The identified seven features such as DebugSize, ImageVersion, IatRVA, ExportSize, ResourceSize, VirtualSize2 and NumberOfSections of Microsoft portable executable file format that could given to machine learning algorithm to classify the PE file into malware or clean file [19].

The Runtime behavior of processes was actively used for detection of malware. Mostly detection techniques build model of runtime behavior of process based on data flow or sequence of system calls. They explained these malware detection must meet the following performance metrics: (1) High accuracy for malware detection, (2) Low false positive rate, (3) Little detection time and (4) The detection techniques must be flexible to run-time bypass attempts. To meet these performance metrics author has proposed framework to detect runtime malware by excavate information in kernel Process Control Block (PCB). The parameter maintained inside PCB of a kernel for each running process defines behavior and semantics for each running process.
A systematic forensic study of the execution path of malware and benign processes was done to identify different parameters of a PCB. As a result 16 out of 118 parameter of PCB were shortlisted using time series analysis. As statistic analysis was done to confirm the feature retrieved from kernel PCB of processes and to determine proper machine learning classifier to detect malware. The whole framework was evaluated based on dataset consists of 114 malware and 105 benign processes for Linux. They achieved 96 per cent malware detection accuracy and 0 per cent false positive rate [20].

Signature based detection based system fails consistently to detect malware hence

they proposed method to detect unknown malware. This method was based on frequency of the appearance of opcode sequence. They described technique to excavate importance of each opcode and determine the frequency of each opcode sequence. They used following step to find the relevance of opcodes (1) Disassembling the executable file, (2) Generate opcode profile file, (3) Computation of opcode profile [21].

In machine learning based malware detection utilized file content extracted from file sample to detect malware. Beside file contents, relationship among file samples provided valuable information about properties of file samples. It used to improve malware detection efficiency. Social graph is a popular way to represent set of socially connected nodes by one or more relation. A social graph has used to study relation between portable executable files and the relation between files used as a feature vector for machine learning classifier to detect the malware [22].

The new malware contained pattern which was similar to observed malware, machine learning techniques used to find new malware. They presented a comparative study of different feature selection methods with different machine learning classifiers in the relation of static malware detection based on the n-grams analysis. They made comparison between several feature selection methods such as correlation based feature selection, principal component analysis, InfoGain Attribute, etc with different machine learning classifier. They demonstrated that the use of Support Vector Machines (SVM) classifier and Principal Component Analysis feature selection method gives the best classification accuracy with the help of a minimum number of features [23].

Cyber criminals and malware author had adapted code obfuscation techniques which subvert the effectiveness of malware defense mechanism. Hence they proposed a system which mainly focused on static analysis in extension with automated behavior analysis. The proposed technique used program as opcode density histograms and decreases total number of features. They employed eigen vector subspace to

refine and reduce the misclassification and conflict of features. The system used a hybrid approach to detect malware based on support vector machine classifier, hence potential of malware detection system can be increased to fight with different form of malware and attaining high accuracy and low false positive rate [24].

The below Table 2.1 shows that which machine learning technique used and which feature vector used from in the research paper from year 2007 to 2015.

Paper Name	Techniques	Feature Vector
[9]	Pearson's Chi Square, Post-hoc Standardize Testing, Crammer's V.	Opcode Frequency
[10]	DT	3,4,5-gram Term
[11, 12]	Text Categorization, ANN, NB, DT, SVM, KNN	Opcode n-grams
[13]	ANN, NB, DT, SVM, KNN	Behavior Report of Malware
[14]	SVM-AR	Content and Behavior Based Feature of PE file
[15]	Learning with Local and Global Consistency	Opcode Sequence
[16]	Perceptron	Static and Dynamic Feature of PE file
[17]	J48, DT, RF, Bagging	Control Flow Graph Feature of PE file
[19]	DT	PE File Features
[20]	J48, J-Rip	Process Control Block Parameter
[21]	DT, KNN, SVM, Bayesian Network	Opcode Sequence
[22]	SVM, C4.5, NBC, KNN	Social Graph Relation for PE file
[23]	J48, SVM, DT, NB, Neural Network	Opcod n-grams
[24]	SVM	Opcode Sequence

Table 2.1: Comparative Study of Malware Detection Using Machine Learning Methods

Chapter 3

Problem Statement and Objectives

3.1 Problem Statement

Malware is a computer program or a piece of software that is designed to penetrate and detriment computers without owners permission. There are different malware types such as viruses, rootkits, keyloggers, worms, trojans, spywares, ransomware, backdoors, bots, logic bomb, etc. Antivirus companies are receiving thousands of malware on the daily basis, so detection of malware is complex and time consuming task.

There are many malware detection techniques like signature based detection, behavior based detection and machine learning based techniques, etc. The signatures based detection system fails for new unknown malware. In case of behavior based detection, if the antivirus program identify attempt to change or alter a file or communication over internet then it will generate alarm signal, but still there is a chance of false positive rate. Also the obfuscation and polymorphism techniques are hinderers the malware detection process.

A comprehensive literature survey has been carried out and following two crisp research gaps have been identified:

- Signature based Detection system fails because of unknown and polymorphic malware.

- Behavior and anomaly based detection system requires malware to be executed.

This research proposes design and development of Opcode based frequency predictor to bridge these gaps without executing the malware and thus inhibiting the infection of development environment.

3.2 Objectives

The main objective of this work is to analyze the opcodes of portable executable file, find unique opcodes which can be used to detect unknown malware. It is totally based on machine learning based malware detection technique. This detection technique helps antivirus vendor to detect unknown and polymorphic malware. Also it will reduce time required to create a signature for malware. The following are prime objective of this work:

- To explore and analyze different malware detection techniques.
- To propose and develop Opcode frequency based malware detection framework which can be used to detect unknown and polymorphic malware.
- To verify and validate proposed framework.

Chapter 4

Proposed Framework for Malware Detection

This chapter demonstrates design of proposed solution through "Malware Detection Framework Based on Opcode Frequency ". It briefly explains how the malware detection is carried out using Machine Learning Classifier.

4.1 Framework for Malware Detection

This framework comprises of Malware or Goodware File Selection, Sample Disassemble, Fetch Statistics of Opcodes, Feature Selection, Model Creation and Validate Model. Model creation includes Decision Tree, Boost, Random Forest and Support Vector Machine. Figure 4.1 shows Framework for malware detection

4.1.1 Malware or Goodware File Selection

In Malware File Selection, we have downloaded malware from malware website malwr. It is a free malware analysis community launched in January 2011. User are able to submit files to it and get the result of a complete analysis [1]. All downloaded malware are in portable executable file format. These malware can execute on windows machine only. We have saved all the malware in Window XP virtual machine. The

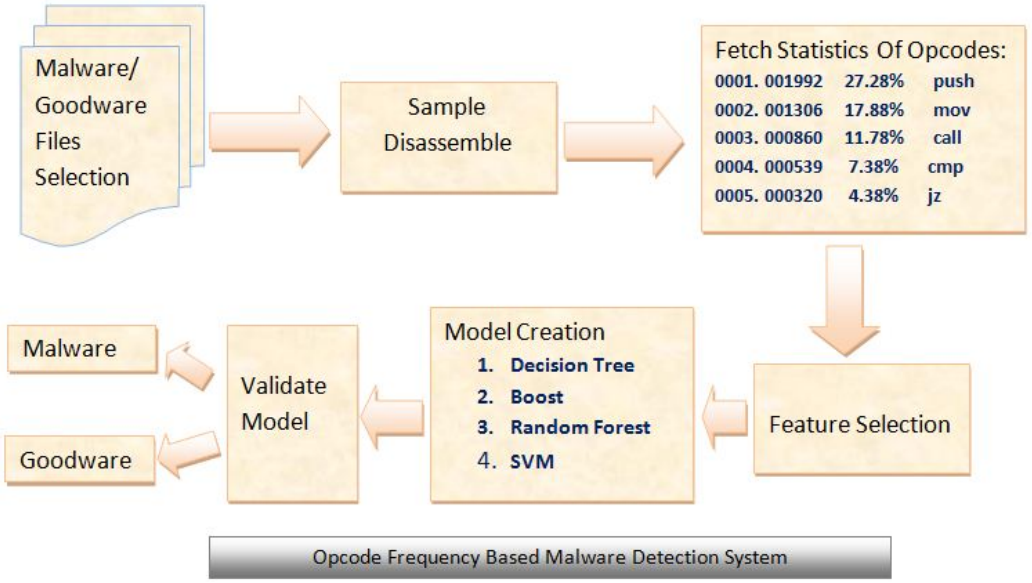


Figure 4.1: System Block Diagram

whole experiment has been performed in virtual environment.

This research work focus is on goodware and malware files. Generally there are different categories of malware such as Worms, Viruses, Rootkits, Trojan horse, Back door, Spyware, Adware, etc. All malware were downloaded from malwr site, we tested and categorized them using virus total website to check malware type. Virustotal website is owned by Google. It is a free online service that can be used to upload any file and check whether it malicious or not. Virustotal website contains database of different antiviruses. If we upload any file in Virustotal website, all antiviruse program scan that file and search for signature. If they get signature then they will display file as malware. After uploading file Lab11-3.exe, Virustotal website shows the output of all antiviruse software as shown in figure 4.2 and figure 4.3.

Figure 4.2: Virustotal Output 1 for Lab11-03.exe

It was observed that output of all antivirus software, mostly says that Lab11-3.exe

Antivirus	Result	Update
AVG	Generic36.ALUH	20160607
AVware	Trojan.Win32.Generic!BT	20160607
AegisLab	Troj.Dldr.Waski!c	20160606
Avast	Win32:Malware-gen	20160607
Avira (no cloud)	TR/Dldr.Waski.49152.5	20160607
Bkav	W32.Clod32f.Trojan.760a	20160606
Comodo	UnclassifiedMalware	20160607
ESET-NOD32	a variant of Generik.MDIXJSP	20160607
Ikarus	Trojan.SuspectCRC	20160607
McAfee	RDN/Generic.dx	20160607
McAfee-GW-Edition	RDN/Generic.dx	20160607
Microsoft	Trojan:Win32/Dynamer!ac	20160607
NANO-Antivirus	Trojan.Win32.Waski.dyxfri	20160607
Qihoo-360	Win32/Trojan.6ac	20160607
Sophos	Mal/Generic-S	20160607
Symantec	Trojan.Gen.2	20160607
VIPRE	Trojan.Win32.Generic!BT	20160607
Yandex	Trojan.Agent!Ynx+rim+6aQ	20160606
Zillya	Adware.Otezinu.Win32.336	20160606
ALYac		20160607

Figure 4.3: Virustotal Output 2 for Lab11-3.exe

is a Trojan, like this we did in each and every malware file and we got the same result. Hence we have not categorized the malware into different category. In goodware file selection, all these files were collected from system32 directory of windows operating system. These files were in portable executable file format.

4.1.2 Sample Disassemble

It is composed of two processes one is unpacking and another is disassembling. In Unpacking process, we identify that Malware or Goodware file is packed or unpacked. If the file is packed then we have used different unpacking tools and techniques to unpack it. After this we can provide these file to disassemble process that is to convert it into assembly language. In case of unpacked file, we directly passed that file for disassemble process. The below subsection will explain unpacking and disassemble

process in detail.

4.1.2.1 Unpacking

In this process, we have to check the Malware or Goodware file is packed or not. Following tools and techniques were used to detect whether a file is packed or unpacked.

- **PEiD**

  It detects most common packers and compilers for portable executable files. It contains database of more than 470 signatures of packers [31]. The Figure 4.4 shows packer detection using PEiD [4]. By using PEiD, we would know about Entrypoint point address, Entry Point Section name, File Offset, Linker Info and Subsystem for particular executable file. Entry point means from where the actual execution of program will start. The subsystem indicates does given file has Graphical User Interface or not. The Linker Info indicates that which version of linker is used while creating executable file. EP section is means section name where the entry point lies.

  When we open Lab18-03.exe into PEiD, it shows Entrypoint as 00005130, EP Section as pec2, Linker Info 6.0 and Subsystem as Win32 GUI. The Lab18-03.exe is packed using PECompact 1.68-1.84 packer as shown in Figure 4.4.

Figure 4.4: Packer Detection Using PEiD

- **Exeinfo PE**

  Exeinfo PE can be used to detect packer. It will complement PEiD packer detection tool. It provides information about unpacking tool name and it's website. It also provides information such as Entry Point, EP Section, Linker Info, File Size, Overlay, etc. The Figure 4.5 shows after opening Lab18-03.exe file into Exeinfo PE packer detection tool it will show below information about file.

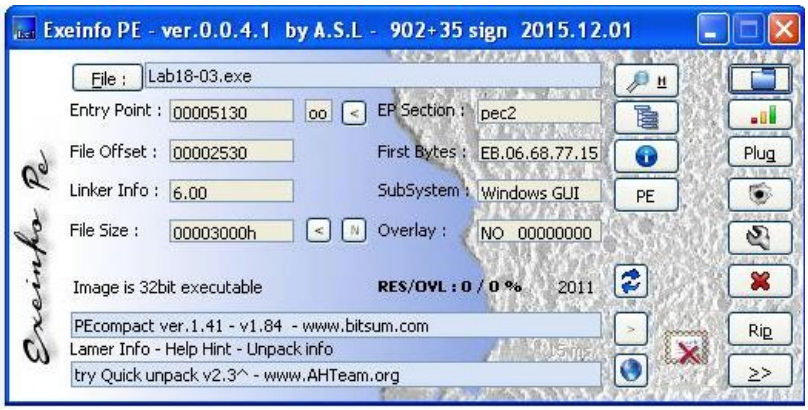


Figure 4.5: Packer Detection Using Exeinfo PE

- **OllyDbg Memory Map**

  Mostly malware author use obfuscation techniques to hide the existence of malicious intent. They used different packer to compress the code and use obfuscation techniques such as Register Renaming, Dead Code Insertion, Code Transposition and Instruction Substitution. OllyDbg [30], is an X86 debugger which mainly focuses on binary code analysis which is important when source code is not available. Mainly it has four windows namely: text area where actually code in assembly language is shown. Register window contain different registers such as EAX, EBX, ECX, EDX, EBP, ESI, EIP, EDI and EBP. These registers are used by program for temporary data manipulation. These changes we can seen in register section. The Hex Dump section contains code in hexadecimal format. The last is a Stack section. While program is executing, the function parameter, return address and local variable are stored on the stack.

When we clicked on M tab of OllyDbg, then Memory Map for particular executable will get open. The Figure 4.6 shows Memory Map for an executable file LAB.exe to the highlighted part is a PE header for executable file. If you click on the PE header then you will get the whole structure of portable executable. In any exe file, it will have mainly four sections code, data, stack, and resource section. There are other sections such as read only data (rdata), export data (edata). The actual text of a whole program resides in code section. Data section contains global variables. The resource section contains all resources which are used by executable such as strings, icon, images and menus. The idata section contains import function information and edata contains export function information.

If we observe the Memory Map for LAB.exe in below Figure 4.6, we get to know about name 'UPX'. It is one type of symptom that LAB.exe is packed using UPX packer.

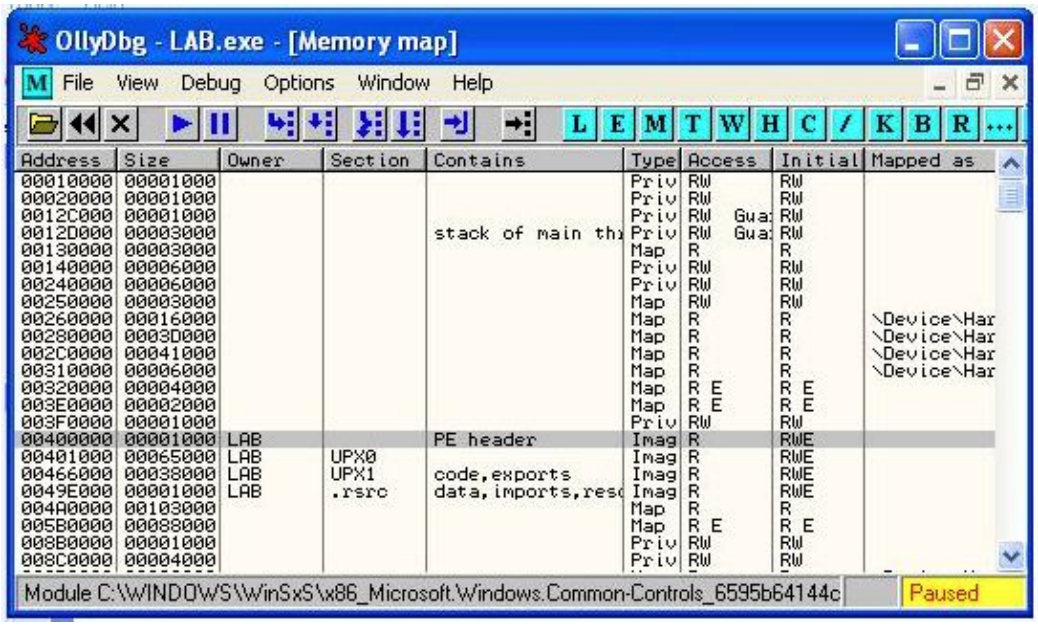


Figure 4.6: Lab.exe Packed with UPX packer

There is one more way that can be used to identify whether given executable file is packed or not. In Figure 4.7 Virtual Size of UPX0 section is 65000 byte in hex format and SizeOfRawData is 0 byte. SizeOfRawData means size of a

program in hard disk and Virtual Size means size of a program in main memory.

Suppose Virtual Size of a Section is very much larger than Raw Size of data then we can say the given executable is packed. In the Figure 4.7 Virtual Size is so much greater than Raw Size, hence it is one sign to say that LAB.exe file is packed. Also if we saw the Raw Size of a section is 0 then also it is a sign to say that program is packed. When we open exe file in OllyDbg then one dialog box is gets popped up and it is a warning to user that the given executable may be packed.

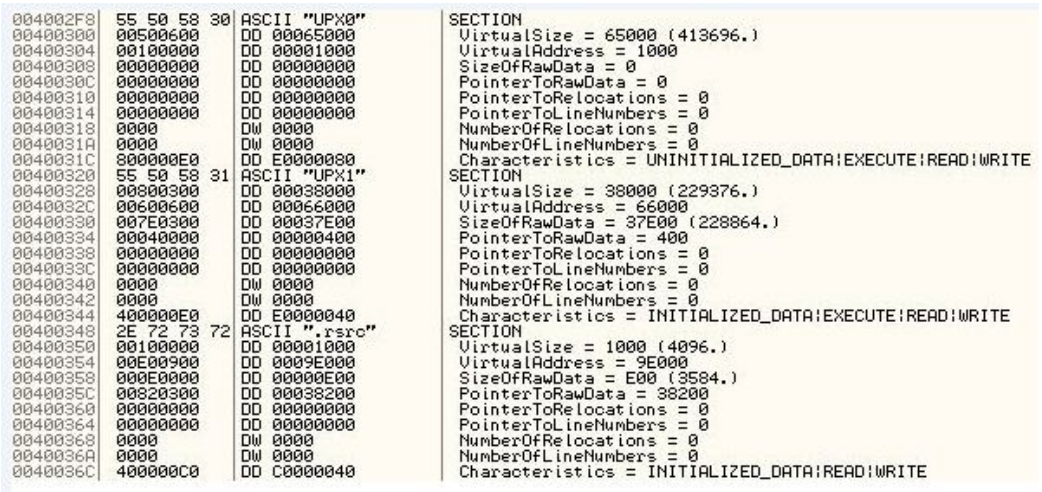


Figure 4.7: Indication of Packed Executable

- **Manual Unpacking**

  We can unpack the executable in two ways one is Manual Unpacking and other way is by using different tools like UPX, PE explorer.There are two ways to perform Manual Unpacking:

  1. Find the packing algorithm then write a code to execute it in reverse direction. By running the algorithm in reverse, program undoes every steps of the packed program. Currently, there are different automation tools available to perform same process. Still this method is not useful because code written for unpacking the malware would be for a single

packer.

2. Open the packed executable in debugger then click on the unpacking stub and perform necessary changes in PE header. This is a more efficient approach.

Manual Unpacking Process:

Load the executable file in OllyDbg. Find the Original Entry Point (OEP). It is the first instruction before program is packed. In manual unpacking, locating OEP is one of the difficult tasks. In OllyDbg, click on plug-in tab, select Olly-Dump, select Find OEP by section Hop. This program will come to halt after hitting the breakpoint just before Original Entry Point executes. As breakpoint gets hit, all the code has been unpacked into main memory and original code is available for analysis.

Last thing is to modify the PE header so that analysis tools can read the code properly. Note down the original entry point. In OllyDump plug-in, select Dump Debugged Process to dump the executable from main memory to hard disk. After Finding actual entry point, we required to do changes into PE header of packed executable. The required changes into PE header are as follows:

1. The import table of exe must be restructured.
2. The entry point in PE header must be changed to the OEP.

If no changes are made in PE header then OllyDump will perform changes on its own [31].

The Figure 4.8 shows OllyDump plug-in option. The OllyDump Plug-in contains following option are: Dunp debugged process, Find OEP by Section Hop (Trace into), Find OEP by Section Hop (Trace over). The Dump debugged process means dumping the process from main memory to hard disk. Find OEP by Section Hop (Trace into) means finding the OEP using F7. Find OEP by Section Hop means (Trace Over) means finding OEP by using F8. In Figure

4.8, we have opened Lab10-3.exe then to find OEP , clicked on Plug-in tab and select OllyDump Plug-in. Select menu item Find OEP by section Hop (Trace Over). After clicking on Find OEP by section hop then program will halt before entry point and new entry point is 0x004271B0 as shown in Figure 4.9.

After getting OEP, we need to modify entry point in PE header of packed executable. Also we need to reconstruct import table. After clicking on Dump Debugged process new window will get open as shown in Figure 4.10. In Figure 4.10 shows entry point is changing from 650BE to 271B0 also Rebuilt import option is checked that means the OllyDump plug-in will automatically reconstruct import table for unpacked file Lab10-3.exe. Then user needs to click on Dump button to save the unpacked program into hard disk after performing all necessary changes.

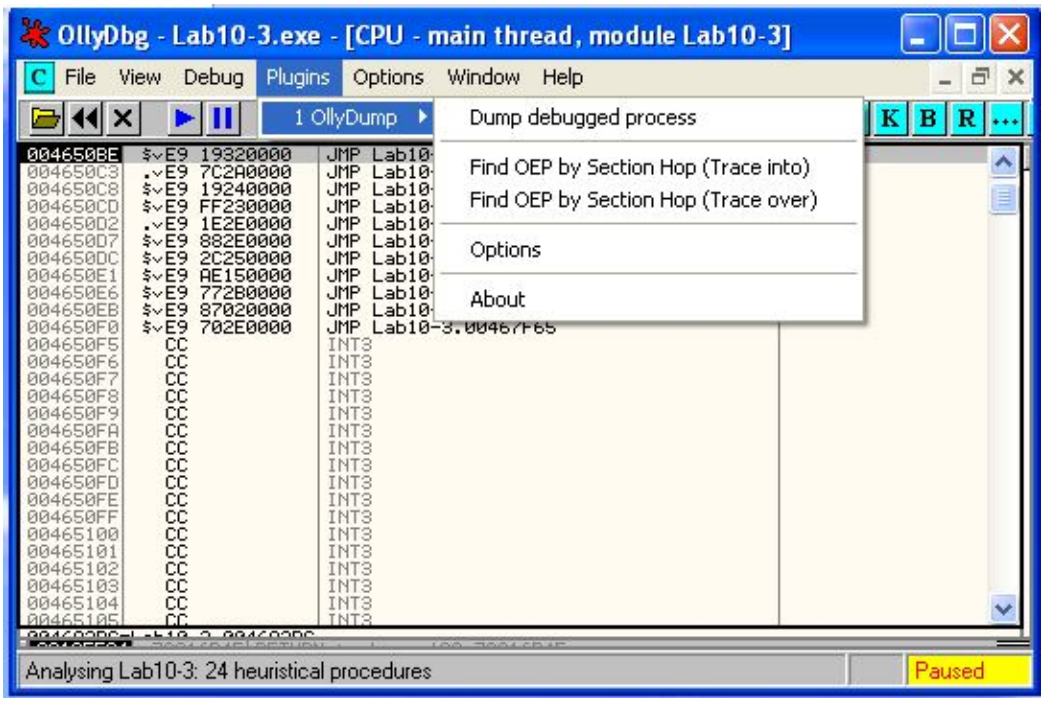


Figure 4.8: OllyDump Plug-in Option

- **Automated Static Unpacking**

  Automated static unpacking program decrypt and or decompresses the executable. It is a fastest method because it does not run the executable and finally

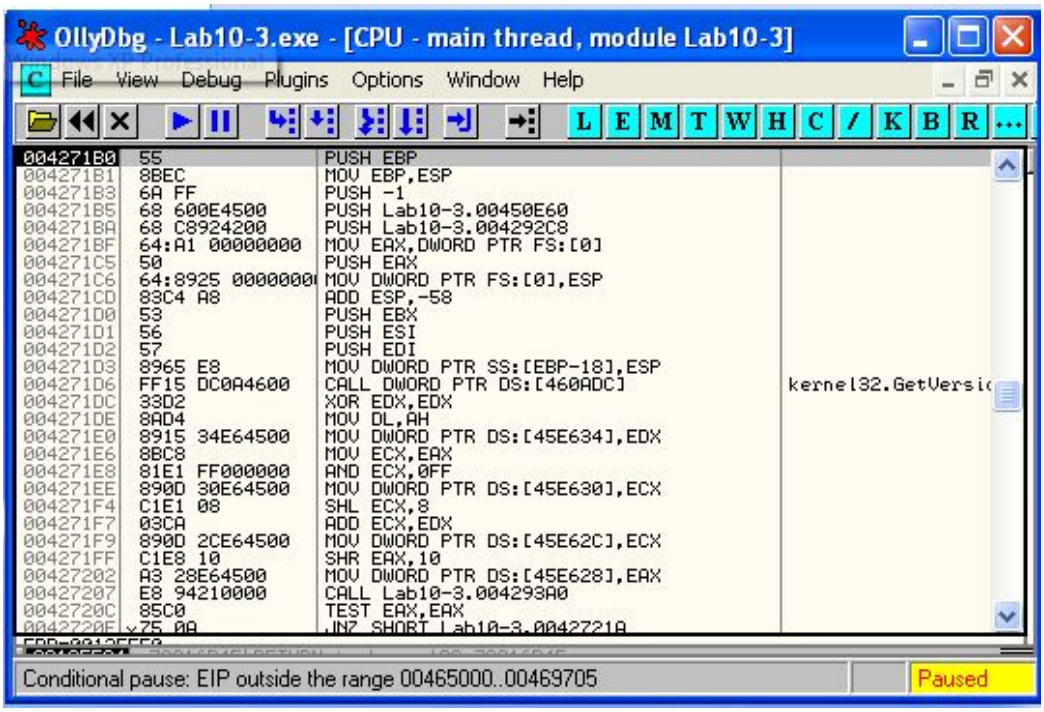


Figure 4.9: After Changing Entry Point

it will convert it into original form. These are designed for specific packer. PE Explorer, a free program that works with exe and dll file comes with several static unpacking plug-ins. The default plug-in supports NSPack, UPack, and UPX. Unpacking of exe with PE Explorer is very simple.

After opening the exe file in PE Explorer the plug-in of unpacking performs checking, whether the given file is packed with NSPack, UPack and UPX. If it is found packed then automatically file will get unpacked [31]. The figure 4.11 shows that LAB.exe is packed with UPX packer, it is gets detected with PE Explorer and automatically decompressed. In Figure 4.11, text in green color shows that LAB.exe is packed with UPX.

- **Automated Dynamic Unpacking**

  Automated dynamic unpacker requires executable file to run. It allows unpacking stub to unpack the original exe code. Once the original exe code is unpacked, then the program is written to disk, finally unpacker reconstructs the original

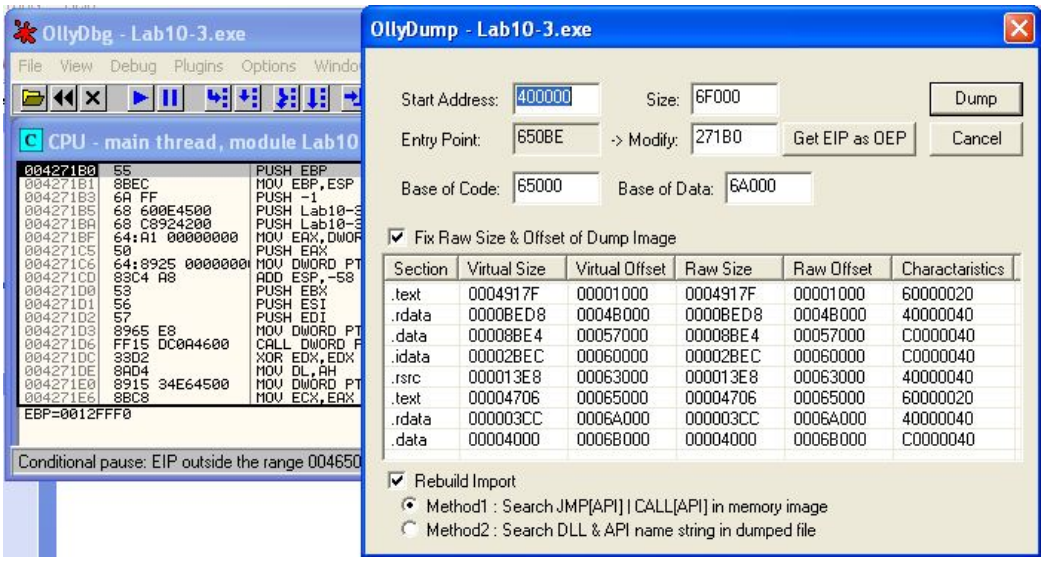


Figure 4.10: Dumping the executable file

import table. The automated unpacking program must find where the unpacking stub ends and actual executable begins. It is difficult to identify the end of unpacking stub. When unpacker fails to identify the end of unpacking stub correctly, unpacking fails.

Unfortunately, there are no good publicly available dynamic unpackers. Many publicly available tools will perform adequate job on some packers, but none of them is ready for serious usage [31].

4.1.2.2 Disassemble process

In above section Packing/Unpacking of exe was demonstrated. After unpacking the executable file we to have disassemble the exe file using IDA pro. Ida pro is an Interactive Disassembler for computer software to convert executable file into assembly language [2]. The conversion of executable into assembling language is as shown in Figure 4.12.

4.1.3 Fetch Statistics of Opcodes

To get the statistics of opcodes, we have used IDA pro plug-in Instruction Counter. But if you used directly this plug-in to IDA pro free-ware version then it will not

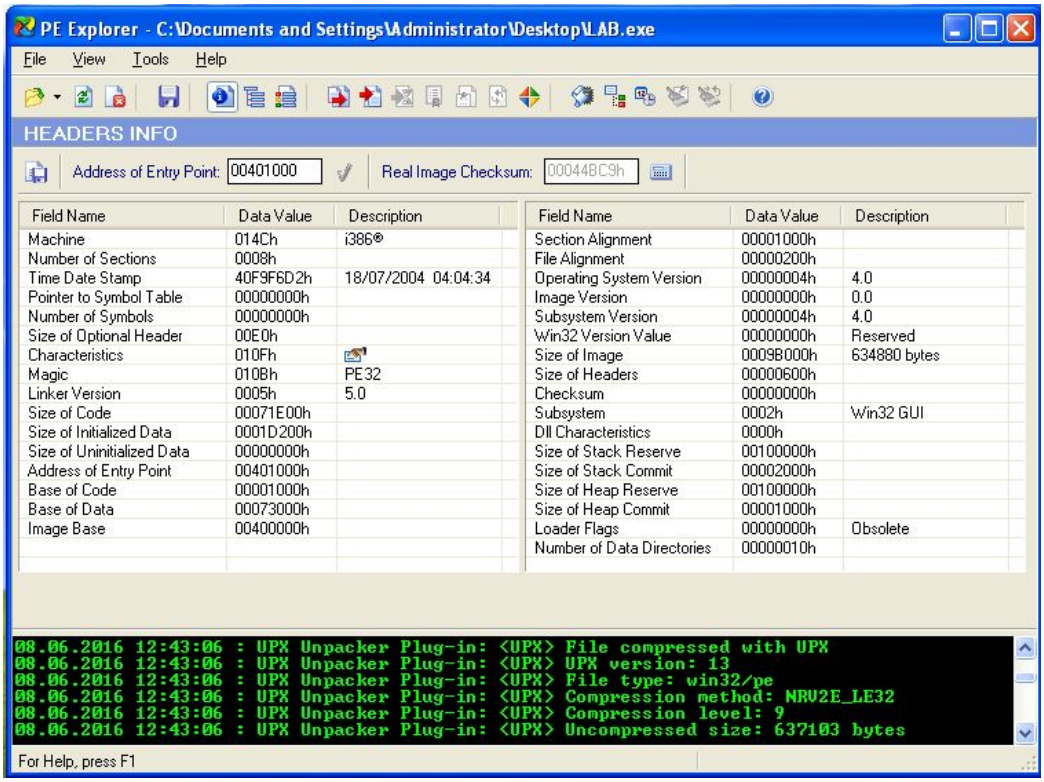


Figure 4.11: Automated Static Unpacking

work. Hence we modified the Instruction Counter plug-in according to requirement of this research. After this open the executable file into IDA pro then click on the view tab and select plug-in Menu item. After clicking on Instruction Counter plug-in tab then new window will get open to save the statistics of opcodes of an executable in textual format. The Figure 4.13 contains four columns:

1. Sequence Number
2. Count for an Operational Code
3. Percentage for an Operational Code
4. Opcode Name

The Count for an operational code means number of times particular opcode is repeated in exe file. Percentage for an Opcode means in the whole exe file how much

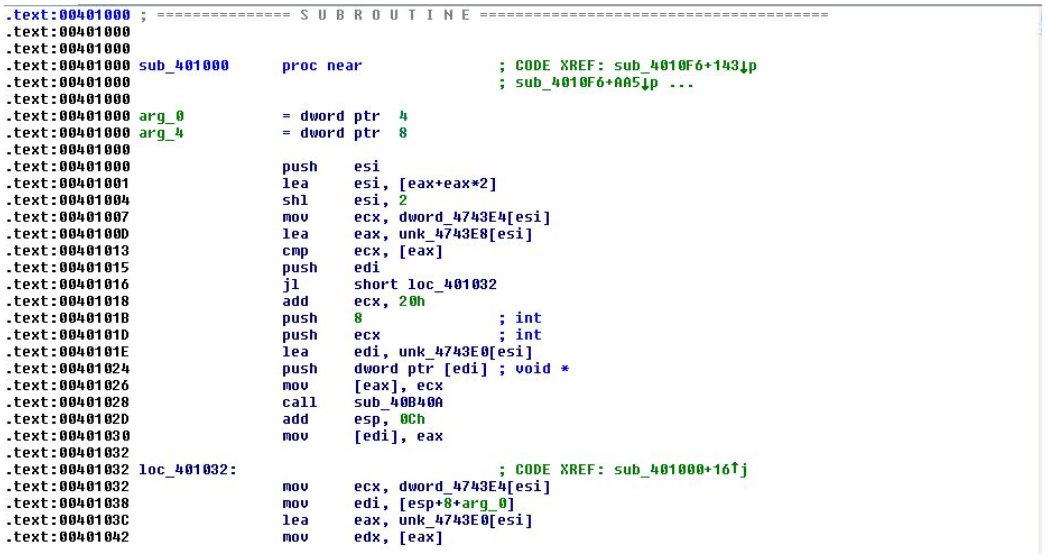


Figure 4.12: Disassembly of executable file

percent particular opcode is? The Instruction Counter Plug-in also displays the total number of opcodes in executable file. Figure 4.13 shows that at sequence number 1, 26552 is number of times PUSH opcode has repeated in the exe file, 22.42 per cent times PUSH opcode out of total number of opcodes. The total number of opcodes in the executable is 118413.

4.1.4 Feature Selection

Feature Selection means selecting subset of feature from whole. Feature vector plays major role in constructing machine learning model. Principal Component Analysis (PCA) and Feature Rank algorithm are used to extract most important features. Goodware files are not malicious (benign) files. We have collected total 100 malware from the malware website [1] and goodwares from the windows Operating Systems, system32 directory.

Malware is unpacked and disassemble using UPX Unpacker and IDA pro respectively. We have used the Instruction Counter plug-in to get the statistics of Opcodes in text format. After Observing 100 goodware and malware samples, 20 Opcodes are frequently repeated, hence considered for further analysis. Twenty most frequent Opcodes identified such as MOV, PUSH, CALL, CMP, POP, JZ, TEST, JNZ, JMP,

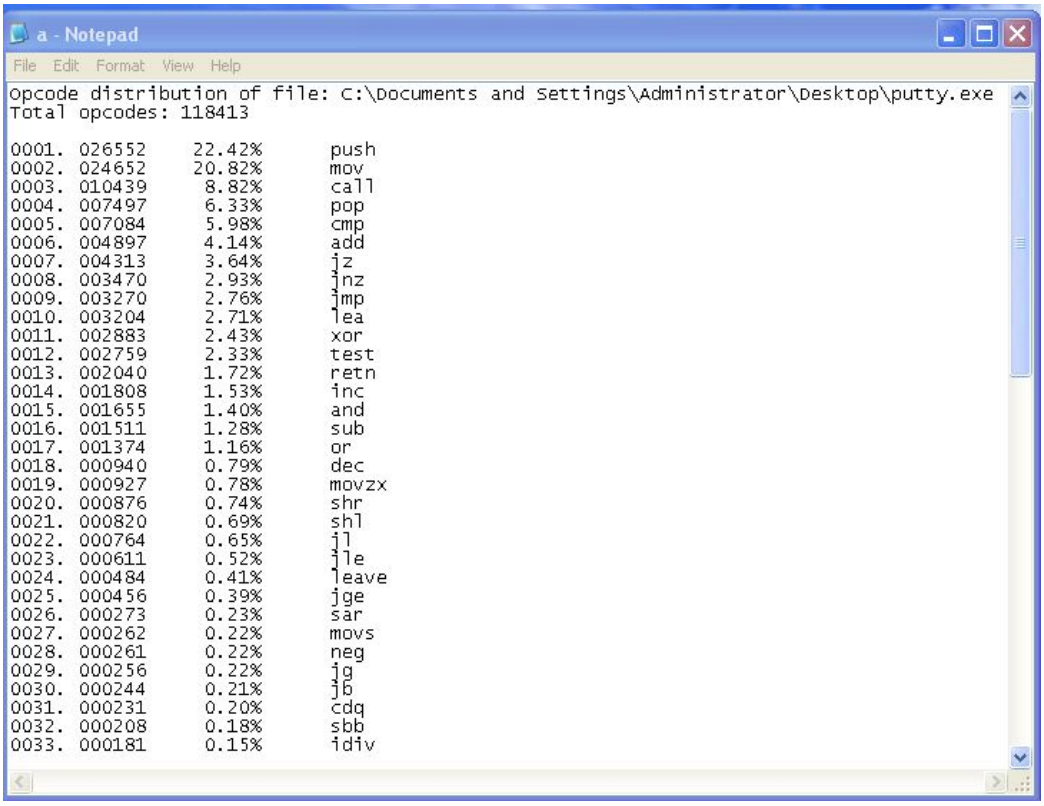


Figure 4.13: Statistics of Opcodes of an Executable File

ADD, LEA, XOR, etc.

- **Data set Creation procedure**

  The data set creation procedure comprises of Fetching Statistic of Opcodes, Preprocessing, and Structural Output. In Fetch Statistics of Opcodes, we have collected stat of opcodes for each goodware or malware file using Instruction Counter plug-in of IDA pro. The preprocessing of text document of opcodes stats completed using cut command in Linux and C-program. cut command used to cut two columns from text document of stats, which are opcode frequency an opcode name. After cutting these two columns, we saved them into comma separable file format (CSV).

  The order of opcodes in text document is vary with different input file. Hence we have developed c-program. After getting output from cut command, we have

given it to C-program to arrange in proper format as shown in Figure 4.16. The code snippet to make structural output is as shown Figure in 4.14 and 4.15.

```
fp = fopen("sample.csv", "r");
while( (ch = getc(fp)) != EOF)
{   if(ch!='\n'&&ch!=',')
     {
      s[k].arr[i]=ch;
      i++;
     }
    if(ch==','||ch=='\n')
    {
     s[k].arr[i]='\0';
     k++;
     i=0;
    }
}
```

Figure 4.14: Code Snippet for Tokenization

In Figure 4.14, input is sample.csv file, the sample.csv contains two field one is opcode name and other is opcode frequency. Both parameters are not in order as it required. Hence we have applied the concept of tokenization over this CSV file and save the file into array as shown in Figure 4.14. After tokenization, required searching of opcode name into array, if found the opcode name then display corresponding opcode frequency on output screen else display null. The code snippet for searching the opcode name in array is as shown in Figure 4.15.

These 20 Opcodes are given input to the machine learning classifier namely BOOSTING, Support Vector Machine, Random Forest and Decision Tree to classify the portable executable file into malware or goodware.

4.1.5 Model Creation

These top 20 Opcodes are selected as feature vector for machine learning classifier. Machine learning algorithms are classified into three categories such as supervised, unsupervised and semi-supervised learning. In supervised machine-learning technique

```
while(count<40)
{
  r=strcmp(s[count].arr,"mov");
  if(r==0)
  {
    printf("%s\n",s[count+1].arr);
    count=0;
    r=1;
    break;
  }
  if(count==38)
  {
    printf("%d\n",t);
    count=0;
    r=1;
    break;
  }
  count=count+2;
}
```

Figure 4.15: Code Snippet for Searching Opcode

requires training data should be properly labeled in order to build a model. Algorithms which belongs to this category are decision tree, SVM, random forest, boosting, etc [26].

In unsupervised machine-learning technique does not require labeled data, in this first clusters of similar data are created using clustering algorithms. Clustering algorithm such as K-means, hierarchical clustering, etc are come into this category. The semi-supervised machine learning is a mixture of labeled and unlabeled data to build a model. In our case data has properly labeled, we have used supervised machine learning algorithm to build a model.

In model creation, we have used Rattle as Graphical User Interface (GUI) tool with R. Firstly you need to install R-Studio. When you open the R-studio you will get R-Console where you need to type following command to get rattle GUI.
Library ('rattle').
rattle ().
After typing above commands you will get Rattle GUI as shown in Figure 4.17.

The Rattle GUI consist of various tabs namely are Execute, New, Open, Save, Data, Explore, Test, Transform, Cluster, Associate, Model, Evaluate, Log. Filename

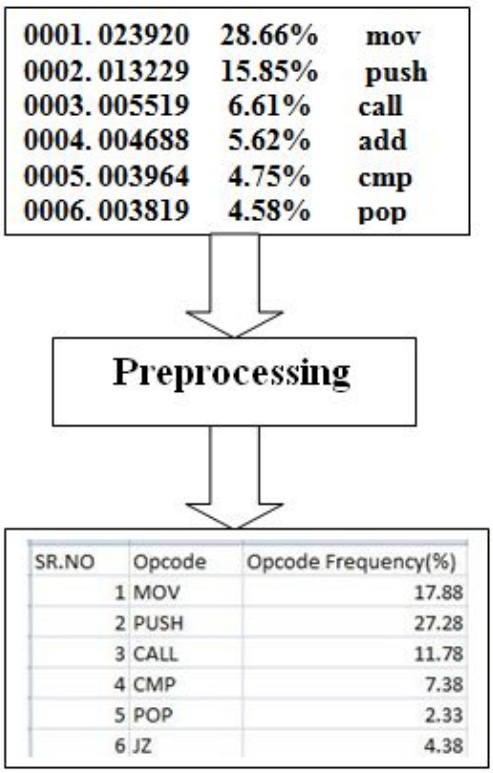


Figure 4.16: Dataset Creation Process

used as to choose the file from any device into Rattle. The partition check box used to partition the whole dataset by default it shows three partition as 70/15/15 per cent. The First partition 70 per cent indicates 70 per cent data for training the model, second partition 15 per cent indicates 15 per cent data to test different parameter setting or different variable while doing data mining and last 15 per cent to estimate the performance of a model.

Select file from Filename file chooser, then click on execute tab to get data loaded into Rattle. After clicking on anything tab from rattle, click on execute tab to perform the operation. By using New tab, we can clear the current work and can able to perform new work from initially. After clicking on open tab, user can able to open existing project. Using Save tab, we can save the current project. The starting point for Rattle is Data tab where we can load our dataset. Rattle provides functionality to load data from different sources namely spreadsheet, ARFF files, From ODBC connection, etc [32]. In dataset rows are called as observation and columns are called

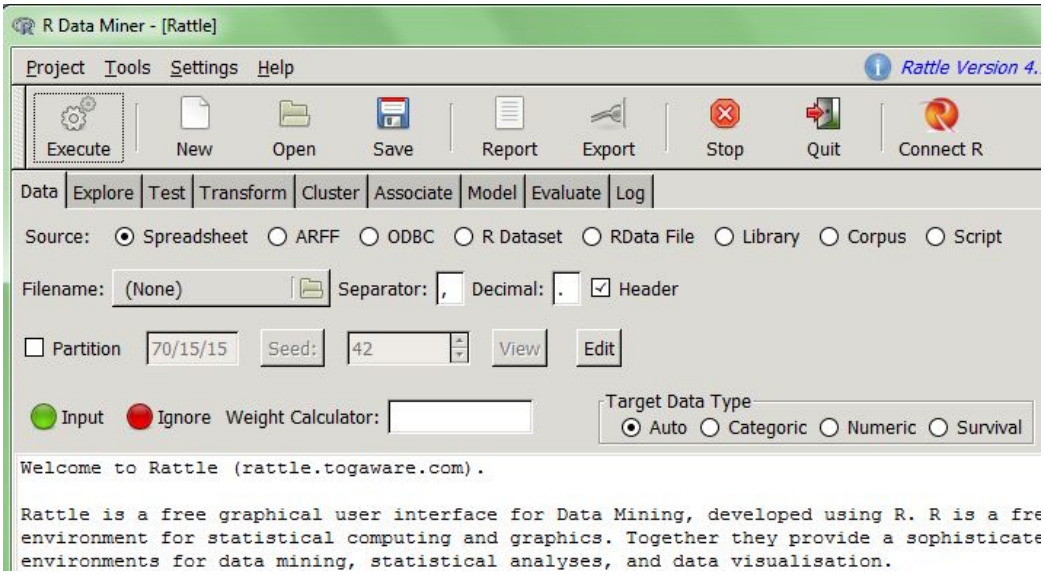


Figure 4.17: Rattle GUI

as variables. Variables can be serves as Input or Output variable. Input variables are also known as predictors, independent variables, covariates, observed variables and observed variables. Output variable are also known as dependent variable, target, or response variable. In data mining, prediction of target variable is calculated in terms of input variable.

Some observation are used to uniquely identify observation of a dataset are called Identifiers. They are not used while building model. Variable can store various types of data. There are mainly two types of data. A Categoric variable takes single value for a specific observation, from a fixed set of probable values. They are always discrete and also known as factors, qualitative variables, or nominal variables. Numerical variables take values that are real numbers or integers. They can be discrete or continuous [25].

Malware dataset loaded into rattle as shown in the Figure 4.18. Rattle shows malware dataset contains 100 Observations, 20 Input Variables and 1 Output Variable is class. Total malware dataset has partition in 70 and 30. 70 per cent is for training the models and 30 per cent for testing the performance of models. The 30 per cent dataset is totally new for models. The data type of all variables is Numeric. To build a model, we need to click on model tab of Rattle. In that, there are different machine

learning models are available namely Decision Tree, Random Forest, Support Vector Machine, Boost, etc as shown in the Figure 4.19. After training the all models, click on evaluate tab to check the performance of all models. In this select testing radio button to check the model performance based on dataset which is totally unknown to all models. Because of this rattle provides unbiased results. Error matrix used to check the decision made by model with actual decision. The Figure 4.20 shows error matrix inside evaluate tab used to test the performance of models.

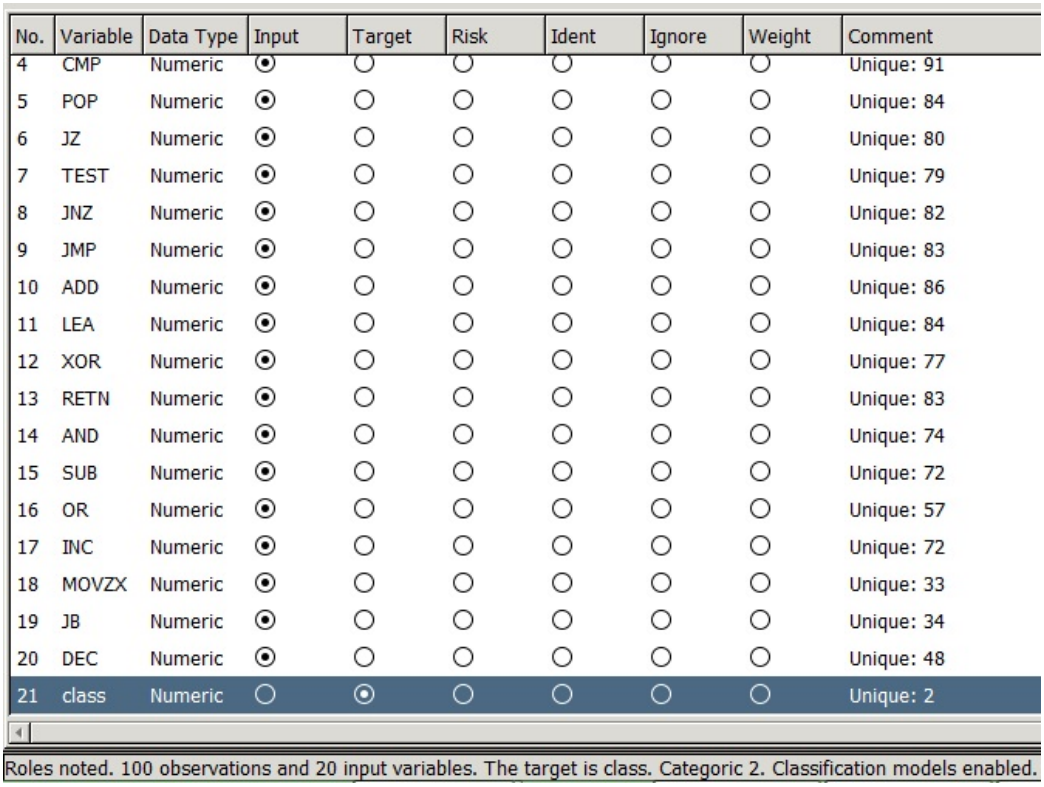


Figure 4.18: Malware Dataset Loaded into Rattle

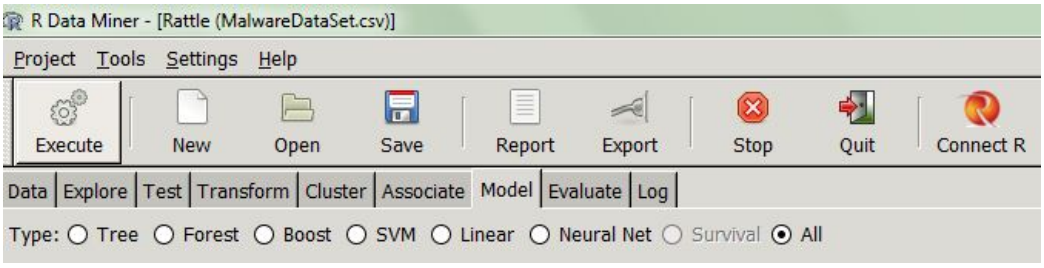


Figure 4.19: Machine Learning Models

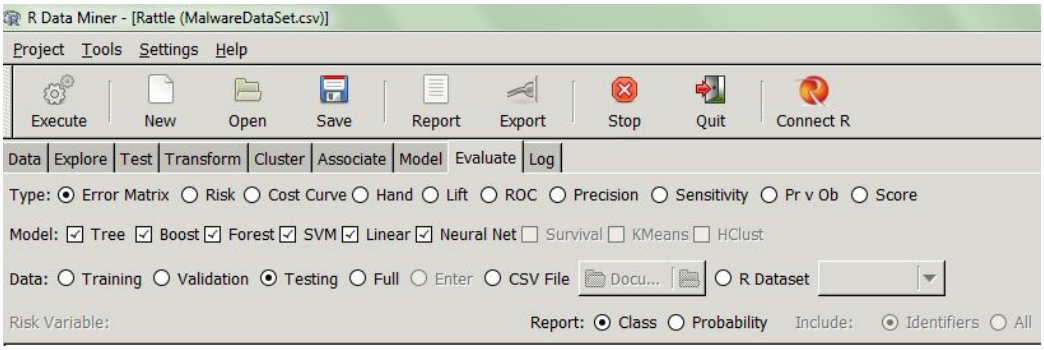


Figure 4.20: Evaluate tab of Rattle

4.1.6 Validate Models

The validation of models has done based on 30 per cent of testing dataset. The models are evaluated based on Error matrix. Error matrix used to check the decision made by model with actual decision. It determines how much accurately model predicts goodware or malware. The models are validated based on True positives, False Negatives, True Negatives, False Positives, True Positive Rate, False Positive Rate and Accuracy. All these terms and Error matrix we will be explained in next chapter Results and Discussions.

Chapter 5

Results and Discussion

This research aims to detect unknown malware using machine learning techniques. We analyzed 100 goodwares and malware files each containing opcodes approximately 30 to 40 thousands. All malware were verified from virus total website. The virus total website contains database of approximately 50 antivirus software. After testing malware on virus total, mostly antivirus identified malware as Trojan hence we have not done further categorization of malware. Our focus in this research is opcodes and their frequency.

We figured out 20 most frequent opcodes and theses opcodes were extracted from assembly file of each malware and goodware file. These 20 opcodes were used as feature vector for machine learning classifier. The machine learning classifier which used for this research are support vector machine, random forest, decision tree and boosting. Next, we divided the dataset in two parts are training and testing dataset. Training and testing dataset percentage are 70 and 30 per cent respectively. In specifically we measured True Positives (TP), True Negatives (TN), False Positives (FP), False Negatives (FN), True Positive Ratio (TPR), False Positive Ratio (FPR) and Accuracy (AC), True Negative Rate (TNR), Recall, Sensitivity and Specificity. Let Assume x=TP, y=FN, z=TN, and w=FP. True Positive Ratio is nothing but number of malware correctly classified divided by total number of malware. The formula for TPR as shown in 5.1 :

$$TPR = \frac{x}{x+y} \tag{5.1}$$

Where TP is the number of malware correctly classified and FN is the number of malware misclassified. The TPR is also called as Sensitivity and recall of the model. False Positive Ratio is nothing but number of goodware files misclassified as malware divided by total number of goodware files. The formula for FPR as shown in 5.2:

$$FPR = \frac{w}{w+z} \tag{5.2}$$

Where FP is the number of goodware files misclassified as malware and TN is nothing but number of correctly classified goodware files. True Negative ratio is nothing but number of goodware file correctly classified divided by total number of goodware file. It is also called Specificity of the model. The formula for True Negative Ratio as shown in 5.3:

$$TNR = \frac{z}{w+z} \tag{5.3}$$

Accuracy is nothing but total number of correctly classified observation is divided by total number of observations in whole dataset [21]. The formula for Accuracy as shown in 5.4:

$$Accuracy = \frac{x+w}{x+y+z+w} \tag{5.4}$$

Error matrix used to check the decision made by model with actual decision. The validation of all models has been done with the help of an Error matrix. The Figure 5.1 shows the Error matrix for the decision tree model. In this research, total dataset size was of 100 observations which was partitioned into 70 percent and 30 percent, hence 70 observations are for training the Decision Tree Model and 30 Observation for testing the Decision Tree model. Figure 5.1 regards to observations count which contains rows is actual observations and column is predicted observations. In Error matrix '0' represents goodware and '1' represents malware. In first matrix out of 11 goodware 9 are correctly classified as goodware and 2 are misclassified as malware,

```
Error matrix for the Decision Tree model on MalwareDataSet.csv [test] (counts):

      Predicted
Actual  0  1
     0  9  2
     1  4 15

Error matrix for the Decision Tree model on MalwareDataSet.csv [test] (proportions):

      Predicted
Actual    0    1 Error
     0 0.30 0.07  0.18
     1 0.13 0.50  0.21

Overall error: 20%, Averaged class error: 20%

Rattle timestamp: 2016-06-11 23:32:52 Abhijit
```

Figure 5.1: Error Matrix for Decision Tree

also out of 19 malware, 15 are classified correctly as malware and 4 are misclassified as goodware. The second matrix shows percentage representation of first matrix. In this each row represents a class, 'first' class for goodware and 'second' for malware. The 'first' class error means number of goodwares misclassified in that class. The 'second' class error means number of malware misclassified in that class. In first class 18 per cent goodware are misclassified while in second class 21 per cent malware are misclassified. The TP, FN, FP, TN, TPR, FPR, TNR, Recall, Sensitivity and Specificity for Decision Tree are shown in Table 5.1 .

Model	TP (%)	FN (%)	TN (%)	FP (%)	TPR/ Sensitivity/ Recall (%)	TNR/ Specificity (%)	FPR (%)	AC (%)
DT	50	13	30	7	78.94	81.81	18.18	80
BT	57	7	30	7	89.47	81.81	18.18	86.67
RF	63	0	33	3	100	90.90	9	96.67
SVM	60	3	33	3	94.73	90.90	9	93.33

Table 5.1: Performance Evaluation of Models

where DT- Decision Tree, BT- Boost, RF- Random Forest, SVM- Support Vector Machine

The Figure 5.2 shows the Error matrix for the Random Forest model. In this First matrix, out of 11 goodware 10 are correctly classified as goodware and 1 are misclassified as malware, also out of 19 malware 19 are classified correctly as malware

and 0 are misclassified as goodware. In the Second matrix, 'first' class shows 9 per cent goodware are misclassified while in 'second' class 0 per cent malware are misclassified. The TP, FN, FP, TN, TPR, FPR, TNR, Recall, Sensitivity and Specificity for Random Forest are shown in table 5.1.

```
Error matrix for the Random Forest model on MalwareDataSet.csv [test] (counts):

      Predicted
Actual  0  1
     0 10  1
     1  0 19

Error matrix for the Random Forest model on MalwareDataSet.csv [test] (proportions):

      Predicted
Actual    0    1 Error
     0 0.33 0.03  0.09
     1 0.00 0.63  0.00

Overall error: 3%, Averaged class error: 4%

Rattle timestamp: 2016-06-11 23:32:52 Abhijit
```

Figure 5.2: Error Matrix for Random Forest

The figure 5.3 shows the Error matrix for the Ada Boost model. In this First matrix, out of 11 goodware 9 are correctly classified as goodware and 2 are misclassified as malware, also out of 19 malware 17 are classified correctly as malware and 2 are misclassified as goodware. In Second matrix 'first' class shows 18 per cent goodware are misclassified while in 'second' class 11 per cent malware are misclassified. The TP, FN, FP, TN, TPR, FPR, TNR, Recall, Sensitivity and Specificity for Ada Boost are shown in Table 5.1.

The figure 5.4 shows the Error matrix for the Support Vector Machine model. In this First matrix, out of 11 goodware 10 are correctly classified as goodware and 1 are misclassified as malware, also out of 19 malware 18 are classified correctly as malware and 1 are misclassified as goodware. In Second Matrix 'first' class shows 9 per cent goodware are misclassified while in 'second' class 5 per cent malware are misclassified. The TP, FN, FP, TN, TPR, FPR, TNR, Recall, Sensitivity and Specificity for Support Vector Machine are shown in Table 5.1.

The Table 5.1 represents the how TP, TN, FN, FP, TPR, TNR, Recall, Sensitivity, Specificity, FPR and AC vary with respect to classifiers. It represent the performance

```
Error matrix for the Ada Boost model on MalwareDataSet.csv [test] (counts):

      Predicted
Actual  0  1
     0  9  2
     1  2 17

Error matrix for the Ada Boost model on MalwareDataSet.csv [test] (proportions):

      Predicted
Actual    0    1 Error
     0 0.30 0.07  0.18
     1 0.07 0.57  0.11

Overall error: 13%, Averaged class error: 14%

Rattle timestamp: 2016-06-11 23:32:52 Abhijit
```

Figure 5.3: Error Matrix for Ada BOOST

```
Error matrix for the SVM model on MalwareDataSet.csv [test] (counts):

      Predicted
Actual  0  1
     0 10  1
     1  1 18

Error matrix for the SVM model on MalwareDataSet.csv [test] (proportions):

      Predicted
Actual    0    1 Error
     0 0.33 0.03  0.09
     1 0.03 0.60  0.05

Overall error: 7%, Averaged class error: 7%

Rattle timestamp: 2016-06-11 23:32:53 Abhijit
```

Figure 5.4: Error Matrix for SVM

evaluation of models based on testing dataset.

After analyzing the performance evaluation of all four models, we come to conclusion that Random Forest classifier provides highest accuracy about 97 per cent as compare to all the models. In table 5.2 shows how training dataset percentage affects the TPR, Accuracy and FPR in Random Forest Model.

We have shown how training dataset percentage affects FPR, TPR and Accuracy parameter for random forest by drawing graphs as shown in Figure 5.5 and Figure 5.6.

The Figure 5.5 represents graph between Training Dataset versus TPR and Accuracy. If we observe in the Figure 5.5 as Training Dataset percentage increases the

Training Dataset (%)	TPR (%)	Accuracy (%)	FPR (%)
10	64	76	13
20	54	74	3
30	71	81	6
40	90	92	7
50	96	96	4
60	100	98	5

Table 5.2: Performance Evaluation of Random Forest

True Positive Rate also increases. In case of Accuracy, as training data percentage increases the accuracy also increases. At 60 per cent training data we got highest accuracy of 98 per cent. The Figure 5.6 represents the graph between training dataset and false positive ratio. If we observed the graph that as Training dataset increase the False Positive Ratio become approaching to zero. In particular, at 60 per cent training dataset the FPR is 5 per cent. From both the graph we come to conclusion that if you train the classifier with more dataset then it will provide high accuracy and low false positive rate.

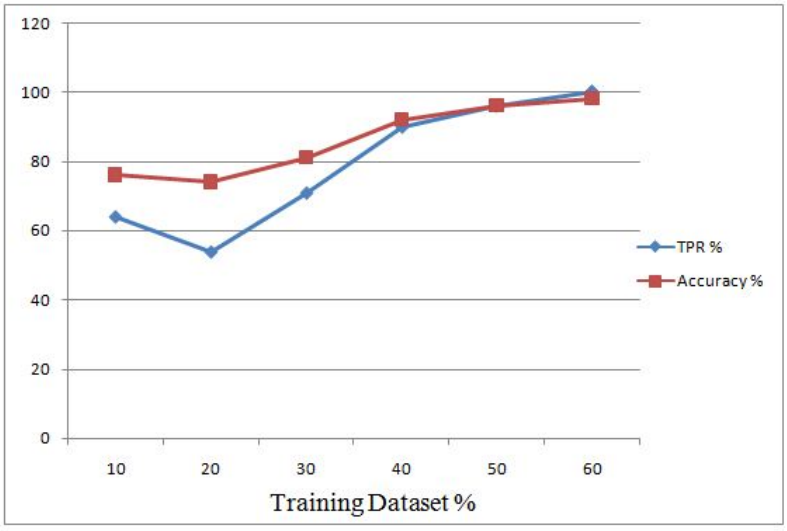


Figure 5.5: Graph for Training Dataset vs TPR and Accuracy

Figure 5.6: Graph for Training Dataset vs FPR

Chapter 6

Conclusions and Future Work

Malware detection is growing research area because of concern over increase of malware on the daily basis. Signature based antivirus system is not useful for unknown malware detection and they are facing difficulties because of polymorphic viruses and zero day attack. So signature based methods must be along with other approach that can able to detect unknown malware. The machine learning is a suitable approach to complement classical signature based malware detection system.

In this paper we collected dataset from 100 goodware and malware files for machine learning classifier. We identified that, Opcode frequency can be used to detect the unknown malware. We found 20 most frequent opcodes can be used as feature vector for machine learning classifier. The dataset for goodwares and malware are containing 20 most frequent Opcode with their frequency. By using our dataset we have constructed four models are SVM, RF, BOOST and Decision Tree. Out of four models Random Forest has provided 97 per cent accuracy and five per cent false positive ratio. In this research we have classified portable executable file into two categories are either goodware or malware.

In future, we plan to apply our dataset to more machine learning classifier, in this paper we have classified PE file in only two categories either goodware or malware but we are planning to classify malware into further categories like Trojan, Spyware, Backdoor, worm, etc.

References

[1] Sandbox, Cuckoo. "Malwr-Malware Analysis".

[2] Guilfanov, Ilfak. "The IDA Pro disassembler and debugger version 5.0", March 2006.

[3] ExEinfo PE. http://www.exeinfo.go.pl/.

[4] Jibz, Qwerton, XineohP Snaker, and PEiD BOB. "Peid," Available in: http://www. peid. info/, Accessed in 21 (2011).

[5] Christodorescu, Mihai, et al."Semantics-aware malware detection," *2005 IEEE Symposium on Security and Privacy* , IEEE, 2005.

[6] Ollmann, Gunter. "The evolution of commercial malware development kits and colour-by-numbers custom malware." *Computer Fraud and Security* 2008.9 (2008): 4-7.

[7] Vinod, P., et al. "Survey on malware detection methods." *Proceedings of the 3rd Hackers Workshop on computer and internet security (IITKHACK09).* 2009.

[8] Teng, Henry S., Kaihu Chen, and Stephen C. Lu. "Adaptive real-time anomaly detection using inductively generated sequential patterns." *Research in Security and Privacy, 1990. Proceedings., 1990 IEEE Computer Society Symposium on.* IEEE, 1990.

[9] Bilar, Daniel. "Opcodes as predictor for malware." *International Journal of Electronic Security and Digital Forensics* 1.2 (2007): 156-168.

[10] Moskovitch, Robert, Clint Feher, and Yuval Elovici. "Unknown malcode detectiona chronological evaluation." *Intelligence and Security Informatics, 2008. ISI 2008. IEEE International Conference on*. IEEE, 2008.

[11] Moskovitch, Robert, et al. "Unknown malcode detection via text categorization and the imbalance problem." *Intelligence and Security Informatics, 2008. ISI 2008. IEEE International Conference on*. IEEE, 2008.

[12] Moskovitch, Robert, et al. "Unknown malcode detection using opcode representation." *Intelligence and Security Informatics*. Springer Berlin Heidelberg, 2008. 204-215.

[13] Firdausi, Ivan, et al. "Analysis of machine learning techniques used in behavior-based malware detection." *Advances in Computing, Control and Telecommunication Technologies (ACT), 2010 Second International Conference on*. IEEE, 2010.

[14] Lu, Yi-Bin, Shu-Chang Din, Chao-Fu Zheng, and Bai-Jian Gao. "Using multi-feature and classifier ensembles to improve malware detection." *Journal of CCIT* 39, no. 2 (2010): 57-72.

[15] Santos, Igor, et al. "Opcode-sequence-based semi-supervised unknown malware detection." *Computational Intelligence in Security for Information Systems*. Springer Berlin Heidelberg, 2011. 50-57.

[16] Gavrilut, Dragos, and Liviu Ciortuz. "Dealing with Class Noise in Large Training Datasets for Malware Detection." *Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), 2011 13th International Symposium on*. IEEE, 2011.

[17] Zhao, Zongqu. "A virus detection scheme based on features of Control Flow Graph." *Artificial Intelligence, Management Science and Electronic Commerce (AIMSEC), 2011 2nd International Conference on*. IEEE, 2011.

[18] Gavrilut, Dragos, Razvan Benchea, and Cristina Vatamanu. "Optimized zero false positives perceptron training for malware detection." *Symbolic and Numeric*

Algorithms for Scientific Computing (SYNASC), 2012 14th International Symposium on. IEEE, 2012.

[19] Raman, Karthik. "Selecting features to classify malware." *InfoSec Southwest* 2012 (2012).

[20] Shahzad, Farrukh, Muhammad Shahzad, and Muddassar Farooq. "In-execution dynamic malware analysis and detection by mining information in process control blocks of Linux OS." *Information Sciences* 231 (2013): 45-63.

[21] Santos, Igor, et al. "Opcode sequences as representation of executables for data-mining-based unknown malware detection." *Information Sciences* 231 (2013): 64-82.

[22] Jiang, Qingshan, Nancheng Liu, and Wei Zhang. "A Feature Representation Method of Social Graph for Malware Detection." *2013 Fourth Global Congress on Intelligent Systems*. IEEE, 2013.

[23] Khammas, Ban Mohammed, et al. "Feature Selection and Machine Learning Classification for Malware Detection." *Jurnal Teknologi* 77.1 (2015).

[24] Ranveer, S., and Hiray, S. "SVM Based Effective Malware Detection System." *In: 2015 International Journal of Computer Science and Information Technologies*, Vol. 6 (4) , 2015, 3361-3365.

[25] Williams, Graham. *Data mining with rattle and R: the art of excavating data for knowledge discovery*. Springer Science and Business Media, 2011.

[26] S. Kotsiantis, "Supervised machine learning: a review of classification techniques", *In: Proceeding of the 2007 Conference on Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in eHealth, HCI, Information Retrieval and Pervasive Technologies*, 2007, pp. 3-24.

[27] Aycock, John. *Computer viruses and malware*. Vol. 22. Springer Science and Business Media, 2006.

[28] Idika, Nwokedi, and Aditya P. Mathur. "A survey of malware detection techniques." *Purdue University* 48 (2007).

[29] N. Weaver, V. Paxon, S. Staniford, and R. Cunningham. "A taxonomy of computer worms." *In Proceedings of the 2003 ACM Workship on Rapid Malcode*, pages 11-18, 2003.

[30] Yuschuk, Oleh. "Ollydbg." (2007).

[31] Sikorski, Michael, and Andrew Honig. *Practical malware analysis: the hands-on guide to dissecting malicious software*. no starch press, 2012.

[32] "Loading Data into Rattle", togaware, [Online]. Available:http://datamining.togaware.com/survivor/Loading_Data.html [Accessed 11 06 2016].

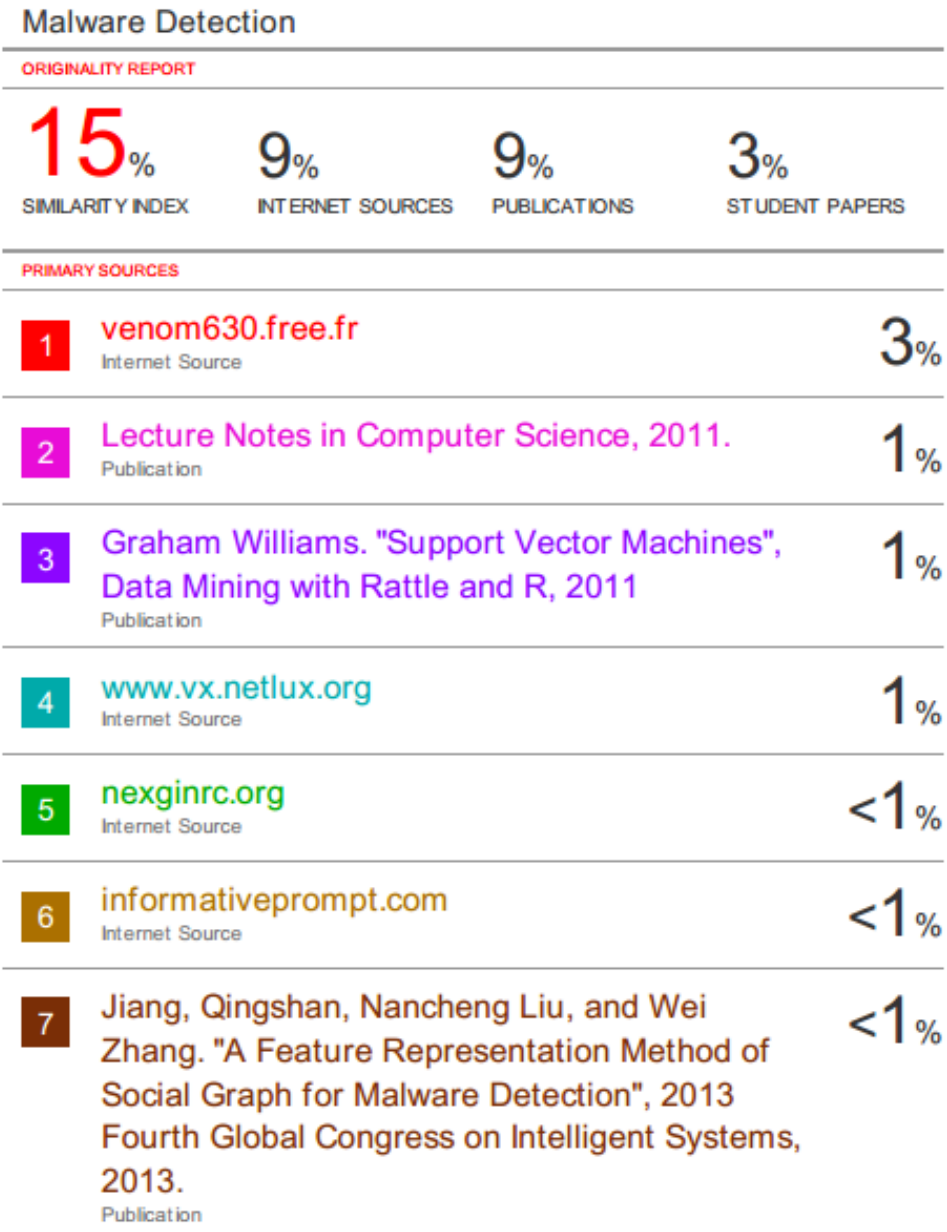
Publication

Abhijit Yewale, Maninder Singh, Malware Detection Based on Opcode Frequency, *In: International Conference on Advanced Communication Control and Computing Technologies (ICACCCT 2016)* [Accepted], pp.754-757, 2016.

Video Link

url “https://www.youtube.com/channel/UCgw0OM-mJ6BX0emSH79BX1Q”

Plagiarism Certificate



Malware Detection

ORIGINALITY REPORT

15%	9%	9%	3%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	venom630.free.fr Internet Source	3%
2	Lecture Notes in Computer Science, 2011. Publication	1%
3	Graham Williams. "Support Vector Machines", Data Mining with Rattle and R, 2011 Publication	1%
4	www.vx.netlux.org Internet Source	1%
5	nexginrc.org Internet Source	<1%
6	informativeprompt.com Internet Source	<1%
7	Jiang, Qingshan, Nancheng Liu, and Wei Zhang. "A Feature Representation Method of Social Graph for Malware Detection", 2013 Fourth Global Congress on Intelligent Systems, 2013. Publication	<1%