Machine Learning Approach to Malware Analysis and Reporting

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

Degree of

Master of Engineering
In
Information Security

Submitted By
Arshi Dhammi
801333002

Under the supervision of:
Dr. Maninder Singh
Associate Professor

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR UNIVERSITY
PATIALA – 147004

July 2015

CERTIFICATE

I hereby certify that the work which is being presented in the thesis entitled, *"Machine Learning Approach to Malware Analysis and Reporting"* 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 researcher's 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.

(Arshi Dhammi)

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,
Computer Science and Engineering Department

Countersigned by:

(Dr. Deepak Garg)
Head
Computer Science and Engineering Department
Thapar University
Patiala

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

Acknowledgment

I would like to thank a number of people for their help and unending support over the year. Most of all, I would like to thank my supervisor, **Dr. Maninder Singh**, not only for his paramount mentorship but also for his guidance, patience and kindness to me. I owe him a heartfelt gratitude for galvanizing my interest in the area of network security.

I am also thankful to **Dr. Deepak Garg**, Head of Department, CSED and **Ms. Jhilik Bhattacharya**, P.G. Coordinator, for the motivation and inspiration that triggered me for the thesis work.

Most importantly, I would like to thank my **parents**, **friends** and the **Almighty** for showing me the right direction out of the blue, to help me stay calm in the oddest of the times and keep moving even at times when there was no hope.

Arshi Dhammi

(801333002)

Abstract

In today‘s scenario, cyber security is one of the major concerns in network security and malware pose a serious threat to cyber security. The foremost step to guard the cyber system is to have an in-depth knowledge of the existing malware, various types of malware, methods of detecting and bypassing the adverse effects of malware. In this work, machine learning approach to the fore-going static and dynamic analysis techniques is investigated and reported to discuss the most recent trends in cyber security.

This study captures 1230 samples of recent binaries from various sources. The peculiar details about the malware such as file details, signatures, and hosts involved, affected files, registry keys, mutexes, section details, imports, strings and results from different antivirus have been deeply analyzed to conclude origin and functionality of malware. This approach contributes to vital cyber situation awareness by combining different malware discovery techniques, for example, static examination, to alter the session of malware triage for cyber defense. This technique for triage decreases the count of false alarms from automatic investigation that permits high workload deduction over utilizing a static technique alone.

Table of Contents

List of Figures

List of Tables

Abbreviations

AV	Anti-Virus
API	Application Programming Interface
ARFF	Attribute Relation File Format
AMCS	Automatic Malware Categorization System
BSD	Berkeley Software Distribution
COFF	Common Object File Format
CFG	Control Flow Graph
CRC	Cyclic Redundancy Check
DLL	Dynamic Link Library
MS DOS	Microsoft Disk Operating System
GNU	General Public License
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
IDE	Integrated Development Environment
JAR	Java Archive
JSON	Java Script Object Notation
LMT	Logistic Model Tree
MECS	Malicious Executable Classification System
MD	Message Digest
MLP	Multilayer Perceptron
PCAP	Packet Capture

PC	Personal Computer
PDF	Portable Document Format
PE	Portable Executable
SHA	Secure Hash Algorithm
SVM	Support Vector Machine
TCP/IP	Transmission Control Protocol/ Internet Protocol
URL	Uniform Resource Locator
VB	Visual Basic
WEKA	Waikato Environment for Knowledge

Chapter 1

Introduction

1.1 Background

The paced advancement in computer technology makes people worried about the security of the computer systems. As the technology grows it makes the application environment more and more complex, leaving behind more loopholes in the system which results in increased threats. In such a case it is very difficult to safeguard the computer systems due to networked connectivity. Computers systems in a connected environment are not secure due to wide connectivity of Internet, absence of prime control and global sight.

Software that is intentionally aimed at accomplishing the adverse intents of harming a computer system is termed as malicious software or simply malware [1]. These are aimed at gaining unauthorized entry to the automated machines such as computer systems and various other network resources. The purpose of this unauthorized entry is nabbing peculiar information, without prior permission of the author, thereby creating a threat to the accessibility of Internet, confidentiality and probity of its users. Malware exist in different forms such as Worm, Rootkit, Virus, Ad ware, Spyware, Backdoor, Trojan-horse, Botnet, etc. This classification of malwares is not mutually exclusive which means that no class of malware is unique and one class of malware can exhibit the properties of one or multiple other classes at the same instance.

1.2 Growth of Malware

Malware pose as one of the most dreadful and dominant security threat Internet is facing in today‘s world. Being specific, FireEye [2] conducted a survey in June 2013 by which it was revealed that in the past one year almost 47% of the organizations combat malware security events /network fissures. Statics have demonstrated that malwares accounts for 66% of the total Internet security threats as shown in Figure 1.1. Everyday hundreds and thousands of new malwares come in market and are

reported and handled by malware writers. The malwares are growing endlessly in speed (rapidity of threats), number (growing threat land-scape) and discrepancy (collection of new innovative methods) [3].

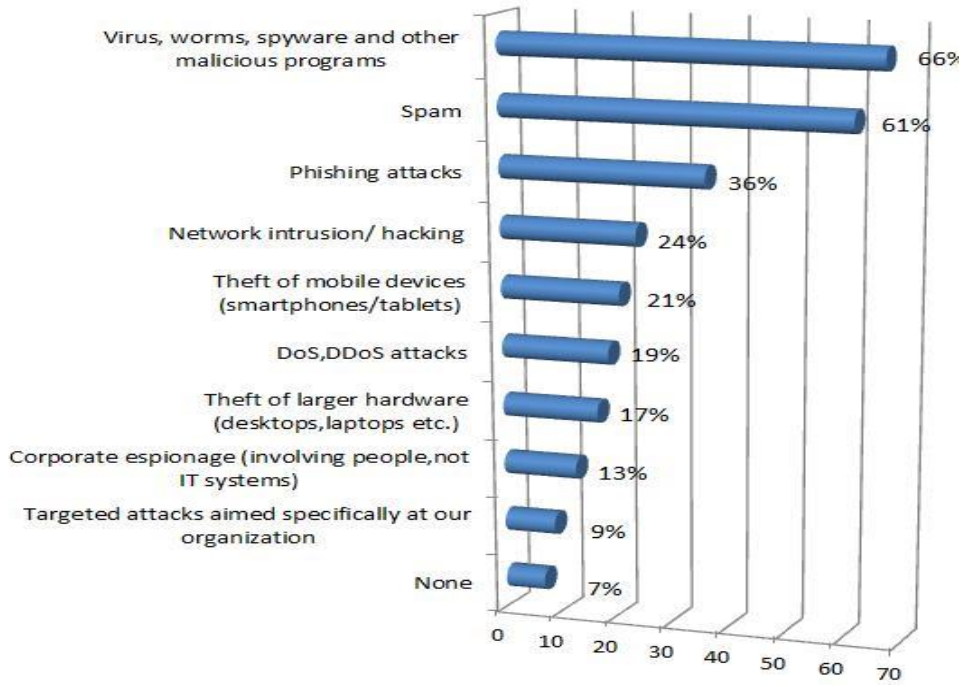

Figure 1.1: Types of attacks on network

Malware is growing at a very high pace with new ideas that can cover a large landscape with threats. In order to spread malwares in large number, various loop-holes in operating systems and web-browsers are ventured by attackers and novice users are made to run the malicious code [4].

1.3 Types of Malware

The different sorts of malware incorporate worms, viruses, trojan horses, bots, spyware and adware. They all display different kind of malicious behavior on their targets and it is crucial to detect and prevent their activities in the system**.** Malware can be classified broadly into the following major classes:

Adware: Adware also known as advertisement-supported malware is the most common form of malware that is aimed at making revenue for its author by delivering advertisements according. Various free versions of software come packed with a wide variety of adware. Mostly the authors of adware are advertisers. Examples of adware include pop ads on websites. In worst cases, adware may also come bundled with spyware which keep account of user activities and may also steal information.

Bot: Bots are simply known as web robots. They are specifically used for executing continual tasks with much higher speed and accuracy. Initially bots were used for innocuous purposes but with passage of time they are being used for malicious purposes. Many such bots are combined to form botnets with is nothing but bunch of computers guarded by third parties and are mainly used for performing DOS attacks. CAPTCHA is used to safeguard against attacks caused by botnets.

Spyware: Spyware actively monitors user activities such as email addresses, most frequently visited pages, keystrokes without their prior knowledge. In addition to these spyware can also modify the security settings of the system which can in turn hamper the network settings [5].

Worm: Worm is a computer malware program that takes the advantage of security flaws in the target system to gain access of it. It spreads rapidly over a network of computers. Initially worms were created with the intent of spreading without causing any harm to the system but nowadays they used as intermediate to spread among networks. The harmful intentions of the worm came out with the Morris Worm which increased the network traffic, deleted malicious files from the host system, encrypt files and send them via e-mail. [5]

Backdoors: Backdoor is a computer program that is used to circumvent the usual authentication process and guarding so as to gain unauthorized entry to the system, gaining access to clear text and so on, at the same time surviving detection by host system. Default password left unchanged for a long time can act as backdoors.

Logic Bomb: Logic bomb is a fragment of code deliberately placed inside software that executes or explodes when certain conditions are met. The conditions may include specific time interval when user fails to respond back to the system or some

other specific condition is met. They perform their task surreptitiously and have chances of carrying off their malicious tasks at a high pace without getting noticed. As these types of malware are active on certain dates such as 1st April or 25th December, they are also known as time bombs [6].

Rootkit: Rootkit is another type of malware with the primary goal of masking the existence of various malware on the system. Rootkit installation may by automized or they install by venturing certain vulnerabilities in the system. Once installed they can have root level access. These make use of peculiar API calls and functions of the operating system to hinder themselves from most of the anti- malware tools. These can be malicious or benign. Their most common entry points to the system are spam emails, Trojan horse or surfing which leads to installations of plugins which seem to be legitimate.

Virus: Virus is a software program which replicates by embedding its copies into other data files thus affecting the areas where the files have been replicated. These require human intervention for executing and spreading. A major target of viruses is Windows Microsoft operating systems. These are created for gaining monetary profit, denial of service, to demonstrate or explode a loophole and to infect network of computers. Virus can affect the performance of the computer by corrupting the memory space and in worst cases it may lead to system crashes [7].

1.4 Time taken in malware analysis

The study highlights the requirement for automated malware analysis techniques, for example, sandboxes, in the market. Analysis of a malware specimen can be done in a matter of minutes with computerized malware analysis techniques. Then again, it can take days to explore particularly complex specimens for malware investigators to break down the code. Figure 1.2 shows the time taken to analyze malware as reported by malware writers. The absence of an automated malware analysis system was referred to as an issue of worry by 35% of respondents, concerning protection of their organization from advanced malware attacks.

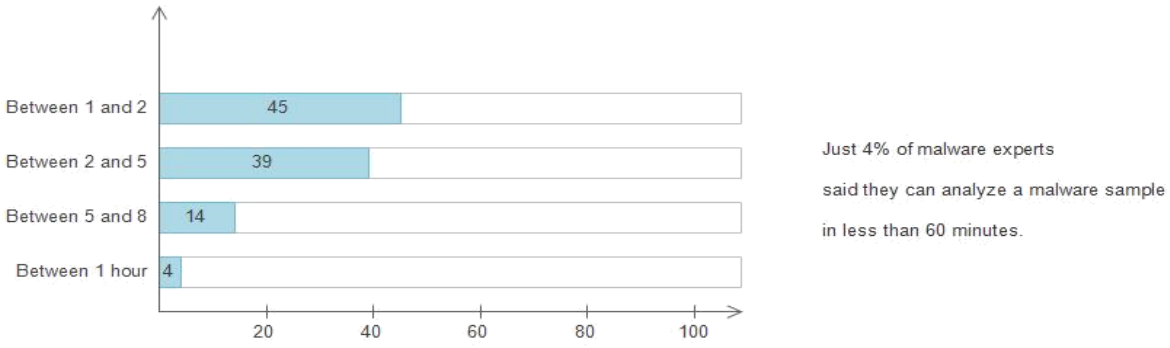

Figure1.2: Time required in malware analysis

1.5 Network Security

Network Security is separated into three D's

- **Defense**

Defense is most essential type of system security. At the point when the assaults are shielded, then there are fewer chances of system execution getting deteriorated. Alongside that it likewise diminishes the chances of different resources getting compromised. Network safeguards implements strategies, for example, spam or virus percolators, firewalls and access rights on specific files.

- **Deterrence**

Deterrence is the second method of security. Deterrence in system security can be defined as the measures taken to oppose the attacks ahead of time. It requests laws for the clients so that association's security strategies may not be damaged. Deterrence minimizes the dangers to the data through suspicion.

- **Detection**

With the assistance of detection, security bargains, diminishes and harm can happen just for restricted length of time if legitimate efforts to establish safety are taken in the ordered and timely manner. It incorporates IDS, log documents, and so on.

1.6 Malware analysis as part of network security

Information security is a pursuit of ensuring data and taking care of hazards identified by processing, utilization and transmission of data. It is an expansive term which incorporates security of information in the terms of authenticity, confidentiality, accessibility, non-repudiation and integrity.

- **Integrity**

Integrity is keeping up reliability and exactness of information. To keep up the integrity, information must not be subjected to changes by unauthorized individuals. This can be possible with the assistance of record authorizations and access controls. Data integrity is not constrained to sparing the information from getting changed by non-trusted parties; however it additionally incorporates sparing the information from mistakes incurred by degradation.

- **Confidentiality**

Confidentiality is similar to privacy in one or the other way. Confidentiality is to make sure that data is accessible to approved clients only. It is a measure to avoid access to delicate data by the wrong individuals. In system security confidentiality is basically achieved with the assistance of cryptography.

- **Accessibility**

Information is only helpful in the event that it is accessible when needed by the approved and authenticated client. It is vital to give satisfactory data transfer capacity so as to avoid occurrence of bottlenecks.

- **Non-Repudiation**

Non-disavowal means every individual in a particular must be receptive of all the activities occurring within a group. Either sender or receiver can't deny its activity once the information is sent or got.

- **Authenticity**

Authenticity is guaranteeing that archives, information, exchanges or communications are factual. It is approving that gatherings included are the same who they claim to be. This should be possible with the assistance of "digital signatures". Computerized signatures guarantee that information is sent by some individual having signing key and thus is genuine. It prohibits the client or data from being altered .

An adequate example of the elements above would be an online exchange. In this sample, integrity is accomplished by TCP's checksum and re-transmission techniques, confidentiality through RSA cryptography and validation with SSL endorsements. It is made possible through the utilization of HTTPS convention. Availability is overseen by the foundation of organization and non-repudiation is taken care of at application layer.

1.7 Detection Techniques:

Malware writers are developing malware codes at an alarming rate which are spread all over internet. This requires an increased effort in detection and mitigation of malwares. Earlier viruses were simpler ones that could be easily detected by anti-viruses.

1.7.1 Static Analysis

Static malware analysis is a procedure that analyzes malware programs without executing it. Malware analysts make use of static analysis methods to figure out

minute details regarding malware. To serve the purpose of analysis on the basis of malware structure various disassemble tools, source code analyzers, decompile tools such as IDA pro and Ollydbg are used. Static malware analysis is preferred because of its ability to unearth the objective and functionality of malware. However, this technique requires time to understand the malware working by breaking down malware structure.

Most of the antivirus vendors make use of signature based detection techniques in which they use pre-defined signature sets. Signature is a unique hex code string representation of malign files or executable. Signature based detection a technique is explained below:

1. A novel malware begins spreading on a more extensive scale
2. Segment of this malware is concentrated on by hostile to malware scholars to have an intensive investigation of its direct.
3. It is then spoken to by a one of a kind hex code string known as its mark.
4. This mark is added to the store of marks.
5. Vendors solicit clients to redesign their database from marks
6. Now clients are protected from this malware by redesigning the mark set.

1.7.2 Dynamic Analysis:

Automated, dynamic investigation of malware works by observing program execution and creating an investigation report compressing the conduct of the program. These examination reports ordinarily cover file exercises (e.g., what files were created recently), Windows registry exercises (e.g., what registry assessments were situated), system exercises (e.g., what files were downloaded, what adventure were sent over the wire), process exercises, for example, at the point when a procedure was made or ended and Windows service exercises, for example, when the service was introduced in the framework and so on. A few of them are freely accessible on the Internet (Anubis, CWSandbox, Cuckoo, Joebox and Norman Sandbox). The major thing to

note about such frameworks for dynamic analysis is that they execute the binary executable for a restrained time period. Since malevolent binaries don't uncover their conduct when executed for a few seconds, dynamic frameworks are obliged to screen the execution of binaries for a more extended time period. Therefore this analysis is resource constrained in terms of important equipment and time. There is additionally the issue of multiple execution paths which are difficult to be analyzed individually in a sandbox environment.

1.8 Thesis Overview

This work is add-on to earlier malware detection techniques. It detects complex malware with high accuracy. Entire thesis work is divided into 6 chapters which are portrayed as follows:

There are 6 chapters in this thesis which are described as follows:

Chapter 1 describes the need of malware analysis to enhance the security of the system. A brief classification of malware and the prevailing analysis techniques is also illustrated.

Chapter 2 discusses the previous work done in this field. It covers the various static, dynamic and machine learning techniques used for malware analysis.

Chapter 3 states the current problem, explains the need for this investigation and major objectives of the research.

Chapter 4 illustrates the data acquisition techniques and detailed methodology along with the experimental setup. Various machine learning algorithm are also discussed.

Chapter 5 explains the various patterns observed and the relationship between them. It then compares the results of different classifiers to select the most efficient classifier.

Finally conclusions are derived from the experimentation work and the need for future work is discussed.

Chapter 2

Literature Review

PC infection recognition has advanced into malware discovery since Cohen initially formalized the term PC infection in 1983[9]. Malware detection techniques are broadly classified into two major types:

2.1 Static and Dynamic Analysis

Static analysis is nothing but studying the malware without executing it. Malware first needs to be unpacked and disassembled for static investigation.

Alazab M. *et al.* [10] made an attempt to design a system which can unpack, de-obfuscate and analyze malware programs automatically so that features can be selected easily and effectively. Authors have effectively utilized the framework to extract behavioral components of API calls that can be related with:

- linking of system applications
- creation or alteration of files
- getting necessary information from the file for changing data about DLLs stacked by malware

They have connected n-gram factual model to acquire the distribution of executable for n-gram values varying between 1 and 5. They have measured the model in light of factors, for example, accuracy, false positive and false negatives. Their method explains the need of machine learning techniques for accurate identification of malicious code as against benign code. A major drawback of the technique is its dependency on the pre-existing unpacking tools. In case the tool fails to unpack the malware the analysis can‘t be done.

Eskandari H. *et al.* consolidates a control flow graph (CFG) from disassembled data with an application programming interface (API) set to achieve 97.77% exactness on a dataset of 2,140 benevolent records from Microsoft Windows XP SP3 and 2,305 "network worms" from a database of Shiraz University [11]. A CFG shows the list of possible activities that the system could perform upon execution. They made a

comparison of the 97.77% accuracy from CFG tests to 92.19% exactness utilizing static n-grams as features. Both trials utilized random forest classifier on the same sample set. To cope with the high processing time from the huge diagrams of CFG investigation, they simplified every graph into a feature vector utilizing a sparse matrix layout. Information based on disassembly acquires fine-grained data; however it still contains loopholes due to obfuscation. The paper does not report the amount of time the technique takes for gathering or investigation.

Moser *et al.* [12], investigated the downsides of static analysis system. In their work, they presented a plan in light of code confusion uncovering the way that the static investigation alone is insufficient to distinguish or group malwares. Further, they suggested that dynamic investigation is an essential supplement to static examination as it is less defenseless against code obfuscation and transformation.

In this part, we will take a look at a portion of the related work done in the conduct based malware analysis and classification. In malware analysis issue, there is not only one standard dataset in all past research. Most malware datasets are gathered from a honey net setup and they include PE executable, which are aimed at assaulting Window based frameworks. The dynamic analysis methods are preferred static examination methods due to confinement present in static methods. Moser *et al.* [12] proposed a strategy where the ordinary model of programs was designed utilizing groupings of six system calls and any deviations from this was abated as irregularity or a security threat. This was one of the first methodologies of utilizing behavior to differentiate malware from benign programs.

Bailey *et al.* [13] followed more dynamic components like system state changes as opposed to system call successions for malware classification. The clustering technique they have utilized takes a look at general system state and produces a behavioral fingerprint for each malware investigated. This is one of the prior works that tended to the issue of anti-virus naming inconsistencies. The Normalized Compression Distance is figured for each pair of malware tests and a simple leveled clustering strategy (utilizing single linkage) was done on a dataset of malware gathered in a sandbox framework running Windows XP. There were 3360 malware that were studied and resulting cluster count sums up to 403 groups.

In correlation with labeling by prominent anti-malware systems like Symantec, Norton, McAfee, Avira and ClamAV, the conduct based strategy gave vastly improved identification exactness of 91% and reliable naming for a given profile. The creators however mention that the behavior profiles of malwares can be looked at a deeper level (instead of state changes alone) with advanced framework review for example, CWSandbox, for better execution. Additionally the length of malware conduct signatures differ a considerable measure and this affects the clustering precision.

During the time spent on malware clustering, different separation measures have been utilized to find the closeness between every pair of files crosswise over different malware families. Some of these measures are suitable for the investigation of metamorphic variations though some are not, especially when the request of the activities in the conduct isn't taken into thought.

Lee *et al.* proposed a malware clustering methodology where a modified Levenshtein separation is utilized and a k-medoid partitioned clustering [14]. The many-sided quality of figuring separations between malware in their technique is quadratic in the quantity of framework calls and may not scale for expansive number of files with a great deal of variability.

In another work[15], Bayer *et al.* have implemented speedier approximated closest neighbor search utilizing Locality Sensitive Hashing for correlation of the investigation reports with known behavior profiles that they have made (utilizing data mining techniques to track framework call dependencies). The conduct reports are then grouped utilizing progressive grouping algorithm. Contrasting the clusters with the genuine malware groups gave them 0.98 and 0.93, exactness and recall values.

2.2 Machine Learning Approach

Schultz *et al.* [16] was first one to apply data mining for malware detection. Their analysis was based on three different static features i.e. Portable Executable (PE), byte sequence and strings. Strings were selected from executable using GNU strings program, which collects sequential printable characters from any file. Byte sequence

approach was mainly focused at extracting sequence of n bytes from an executable. Features used in PE approach were number of DLL functions calls and count of unique calls made within every DLL. A comparison of learning algorithms and conventional signature based techniques was made and learning algorithms had accuracy twice that of signatures based techniques. This technique failed to make a good use of byte-sequences and moreover the algorithms used were not time efficient. Algorithms were not tested on a larger scale.

Kolter *et al.* [17] addressed the problem of malware analysis by devising a field application named Malicious Executable Classification System (MECS) using machine learning and data mining techniques. The dataset consists of variety of malwares including backdoors and key loggers. MECS successfully detected malware in the wild and from obfuscated binaries as well, without the need of de-obfuscation. Authors gathered byte sequences from binaries and transformed them into n-grams and finally devised various classifiers. Text classification was also employed on executable. A major drawback was that it was unable to remove obfuscation, analyze functional characteristics and resulted in several misclassifications.

Moscovitch *et al.* [18] focused on employing text categorization and addressed the imbalance problem in the number of malicious and benign files in practice. As malicious files are only 10% of the total files, it becomes very difficult to detect such files with higher accuracy. According to the experiments, accuracy of more than 95% was hit when malicious content was less than 20% in the training data set. The different classifications used were Naïve Bayes, Artificial Neural Networks, Support Vector Machine and Decision Trees.

Santos *et al.* [19] made an attempt to classify unknown malicious files on the basis of static methods i.e. using signature based techniques. The three primary parameters used in the investigation were:

- Extent of the n-grams: The span of the n-grams, or n, permits to choose to what extent in bytes the n-gram will be.
- Quantity of closest neighbors for the kNN calculation, where the choice of k relies mostly on the framework‘s conduct.

- Breaking point that checks the way of the obscure occurrence as malware: The main motive here is to reduce the number of false positives

Here, n-grams of both the known records and the obscure ones were removed. For any example of the obscure set, it was grouped into malware or benevolent programming, based upon the appearance of n-grams between the arrangement of n-grams of the known records and the arrangement of n-grams of the obscure ones. The different abilities of n-gram could have been applied and the scope of the research could have been practiced on bigger malware accumulations with a blend of behavioral examination.

The experimentation work presented by Zhou *et al.* [20] exhibits a malware recognition strategy that finds malware by method for a learning motor prepared on an arrangement of malware occasions and an arrangement of kind code cases. The learning motor uses a versatile information pressure model— expectation by fractional coordinating Prediction by Partial Matching (PPM) — to manufacture two pressure models, one from the malware occasions and the other from the favorable code occurrences. A code occasion is arranged, either as "malware" or "kind", by minimizing its evaluated cross entropy. Results moreover show that this system can successfully identify obscure and jumbled or obfuscated malware.

This method failed to figure out what sort of systems a potential enemy could use to modify their malware with a specific end goal to trap the pressure based classifier into allotting false-negatives. Assaults utilizing words known not considered great by a classifier are regular in spam sifting what's more, pressure based spam channels are known not defenseless these assaults [20]. Feature selection was not applied in selecting the decision boundary and a large number of features were used for the same.

Yanfang *et al.* [21] presented a paper in which they used instruction recurrence and function-based instruction successions to devise a framed named as Automatic Malware Categorization System(AMCS) for consequently gathering malware tests into families that share some normal qualities utilizing a group troupe by amassing the bunching arrangements created by diverse base bunching calculations. Particular clustering algorithms were merged to propose a cluster ensemble framework which

combines the benefits of level bunching, K-mediods calculations and weighted subspace K-mediods algorithms to create base clustering‘s. The special features of AMCS are:

- Well- picked component representations
- Deliberately Planned Base Clustering
- A noble cluster ensemble method
- Signature generation by natural methods
- Inclusion of human expertise

Siddiqui *et al.* [22] shifted from traditional signature based detections to recognition of malware behavior for analysis. This paper displays a clever thought of separating variable length direction arrangements that can distinguish worms from clean projects utilizing information mining procedures. The investigation is encouraged by the program control stream data contained in the instruction flow. Based upon measurements accumulated from these guideline groupings we figured the issue as a binary characterization issue and constructed tree based classifiers including Random Forest, Packing and Decision tree. Data mining methods were used to consequently extract conduct from worms and clean projects. Secondly, all elements were groupings of guidelines separated by the dismantling as opposed to utilizing settled length of bytes, for example, n-gram. The favorable circumstances are:

- The guideline successions incorporate program control flow data, not introduce in n-grams.
- The guideline groupings catch data from the system at a semantic level instead of syntactic level.
- These guideline groupings can be followed back to their unique area in the system for further examination of their related operations.
- These elements can be gathered together to frame extra inferred components to expand classification exactness.
- A noteworthy number of groupings that showed up in just clean program or worms can be wiped out to accelerate the demonstrating procedure.

The feature used here lacked the inclusion of executable section of PE file, Import Address Table and PE header and API calls, which act as important tools in malware detection [23].

In another information mining methodology, [24] utilized three diverse sorts of components and a mixed bag of classifiers to identify pernicious programs. Their essential dataset contained 3265 malignant and 1001 clean files. They connected RIPPER (a standard rule based framework) to the DLL dataset. Strings information was utilized to apply Naive Bayes classifier while n-grams were utilized to prepare a Multi-Naive Bayes classifier with a deciding technique. No n-gram diminishment calculation was accounted for to be utilized. Rather information set parceling was utilized what's more, 6 Naive Bayes classifiers were prepared on every allotment of the information. They utilized diverse elements to constructed distinctive classifiers that don't represent a reasonable examination among the classifiers. Naive Bayes utilizing strings gave the best precision in their model.

Malware essayists persistently plan and actualize new techniques to sidestep present filters and anti-malware frameworks. One of the approaches to accomplish this is to change the estimations of certain components that have been viewed as pertinent to group samples for quite a while. Entropy is a measure usually used to figure out if an executable is packed or unpacked. This kind of document characterization can be a past separating stride to a time consuming dynamic unpacking procedure.

Ugarte *et al.* [25] designed a system to measure executable entropy which diminishes randomness. They report a portion of the assaults found on genuine Zeus family malware samples. They tested the technique by setting up an edge for entropy measure as an option, acquiring preferred results over exemplary record entropy measure.

However, they failed to detect particular parts of the executable which contains attack patterns. This finding will make sure which parts of the program obscure the real code. Entropy is not the only feature which classifies malware and can be used in combination with other features for better accuracy.

Another paper [26] talks about the idea of structural for detection of metamorphic malware. The procedure proposed has two stages, namely,

- file division or segmentation
- Sequence comparison.

In the first stage, segmentation was done on the basis of entropy estimations and wavelet analysis. In the next stage, edit distance between the segment sequences obtained from the first step was compared. The likeness measured was comparable to an especially difficult gathering of metamorphic malware.

Chapter 3

Problem Statement

3.1 Gap Analysis

Though various techniques and tools for malware detection are prevailing from decades, yet malware analysis is considered as a milestone in system security. The various obstacles to malware analysis are:

- Large volume: Malware is growing at a very fast pace and it is very difficult to analyze this huge volume.
- Obfuscation: To avoid detection by anti-malwares, malwares authors make use of sophisticated obfuscation techniques [5] like code interchange, code amalgamation, register reassignment, null value insertion and reordering of subroutines.
- False Positives: Though anti-malware systems are prevailing from decades but they detect malware with high false positive rate, which in turn affects the accuracy.
- Detection of virtual environment: Some malware detects that they are executed in virtual environment, resulting the malwares to act in an artificial manner or stop execution.
- Detection speed: Due to different evasion techniques used by malware, it takes time to detect malware which in turn can be used by malware to impair the system. The detection speed may vary from minutes to hours depending upon the type of malware.
- API calls: There are certain API calls which are used by both malware and benign files to accomplish their tasks. It becomes cumbersome as which calls are used by malwares and which are used by benign files.
- System degradation: Installation of various types of anti-malware in one way or the other slows down the system.

3.2 Problem Statement

The ever growing volume of malware can cause severe damage to the internet security. The damage can range from gaining of personal information, stealing of credit card passwords. In case the critical information is not protected it may affect personal and networked computers. The need of the hour is to automatically analyze this large bunch of newly growing malware so that one can make his/her system more secure. Although static and dynamic analysis are prevailing in the market from decades, malware writer are smart enough to devise techniques to evade detection by these. Moreover, to deal with advanced malware an efficient and accurate technique needs to be devised. Machine learning approach embedded with static and dynamic analysis is one such approach. The machine learning approach uses different classifying and clustering algorithms for ascertaining and correlating undiscovered malware samples into either already discovered malware families or highlighting those samples that illustrated somewhat different behavior.

3.3 Objectives of Research

- To capture and analyze the latest malware traffic using static analysis.
- To perform dynamic analysis using sandboxing.
- To perform malware categorization using machine learning approach.
- To validate and verify the results

Chapter 4

Implementation Details

4.1 Implementation

In this chapter, implementation details of proposed method are given. For the implementation, machine learning approach is used, which will be explained subsequently.

4.2 PE file format

The PE file format [35] was devised after the invention of Common Object File Format (COFF). The essential objective behind designing of PE file format was to standardize the executable format for all operating systems based on Window using different processors. PE file format is utilized by every windows32 executable except win64 dlls. The auxiliary objective was to give the ingenious and simple route for the windows operating system to execute programs and store the data required for execution of specific programs such as resource files, register values and so on [36]. PE format is shown in the Figure 4.2.

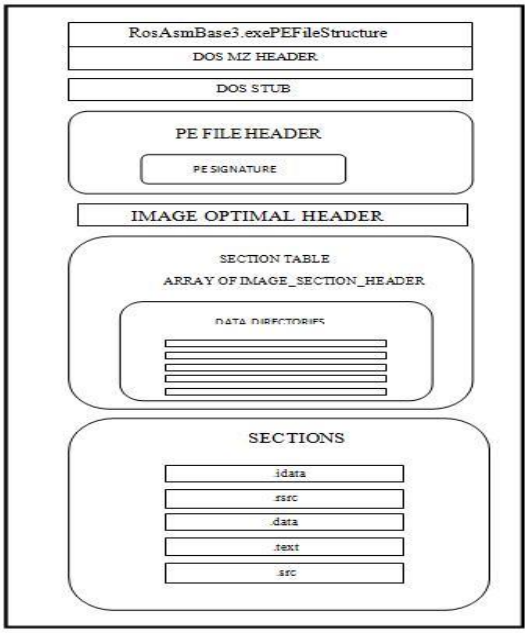

Figure 4.1 PE file format

4.2.1 Detailed explanation

A detailed explanation of section is given as under:

- **MZ DoS Header:** It is the starting point of every PE executable and it takes up to first 64 bytes of the file. It is there to detect the presence of a valid executable that is run from DOS. If it is a valid executable DOS Stub, stored immediately after Dos Header starts execution by displaying some message on screen such as ―This program must be run under Microsoft Windows‖. Layout of DOS header is saved in winnt.h or windows.inc files. It comprises of 19 members. DOS stub execution is usually skipped by PE loader for normal executable.

- **PE header:** PE comprises the necessary information required by the loader. It is usually represented by a structure named as IMAGE_INT_HEADERS which includes 3 members:

  - Signature: Signature is a DWORD with the combination 50h, 45h, 00h, 00h ("PE" trailed by two ending zeroes).
  - File Header: Information regarding the physical structure and properties of file is contained in the file header which extends up to next 20 bytes.
  - Optional Header: Next 224 bytes contains information about the logical structure of PE files such as entry point of the program.

- **Section Table**: Section table lies amidst the raw data and PE header. The sections include the major content of the file, including code, resources, data and other executable data. Group of data having similar attributes is put under one section. Every section is divided into two parts- a header and a body (the raw information). The section headers are present in the section table however the structure of raw data or section body is not fixed. They can be sorted out the way linker wishes to arrange them.

An application for Windows NT normally has the nine predefined areas listed as below:

- .text
- .debug
- .rdata
- .data
- .pdata
- .edata
- .idata
- .rsrc
- .bss.

A few applications needn't bother with all of these areas, while others may characterize still more areas to suit their particular needs.

Another critical point is that the structure of document on disk is precisely the same as when it is stacked into memory so that once the information in the file is located on disk, the same can be used to discover it when the document is stacked into memory as shown in Figure 4.2. Despite it is not replicated precisely into memory. The windows loader chooses which parts need mapping in and overlooks any others. Information that is not mapped in is set toward the end of the record past any parts that will be mapped in e.g. debug information. Additionally the location of an element in the record on disk will often vary from its location once stacked into memory as a result of the page-based virtual memory administration that windows employs. At the point when the sections are stacked into RAM they are adjusted to fit to 4Kb memory pages, every section beginning on a new page.

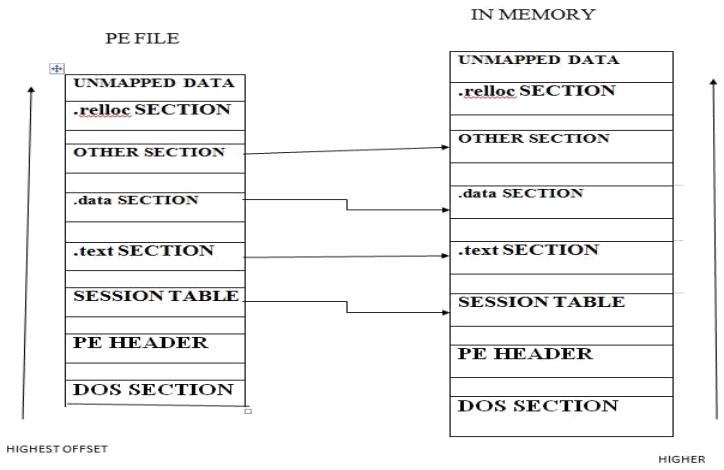

Figure 4.2 Memory view of PE file

The idea of virtual memory is that rather than allowing program straight away get to physical memory, the processor and OS make an imperceptible layer between the two. Each time an endeavor is made to get to memory, the processor counsels a "page table" that tells the procedure which physical memory location to really utilize. It wouldn't be functional to have a table entry for every byte of memory (the page table would be bigger than the aggregate physical memory), so processors separate memory into pages.

4.2.2 PE code cave injection

In some cases it becomes vital to add code to a software or program keeping in mind the end goal to either exploit a security loophole or to enhance its usefulness.

The three methods used to add code to an executable are:

- Modify or increase the size of a prevailing section when the space is not sufficed.
- Add to an existing section a when there is sufficient space for code.
- Add a completely new section.

4.3 Static Analysis

Static malware analysis is a procedure that analyzes malware program without executing it. Malware analysts make use of static analysis methods to figure out minute details regarding malware. To serve the purpose of analysis on the basis of malware structure various disassemble tools, source code analyzers, decompile tools such as IDA pro and Ollydbg are used. Static malware analysis is preferred because of its ability to unearth the objective and functionality of malware. However, this technique requires time to understand the malware working by breaking down malware structure.

Moreover, these days‘ malwares use packers to change themselves. Packer is a program that is used to wrap malware in order to compress, change or encrypt its content. At the point when a packer packs, scrambles, or modifies an executable system, the program looks distinctive. There arises a need to unpack malware before analysis by above mentioned tools namely IDA pro and Ollydbg. PEiD is a program that is used to detect the presence of packers or cryptors. Figure 4.3 shows the analysis done by PEiD.

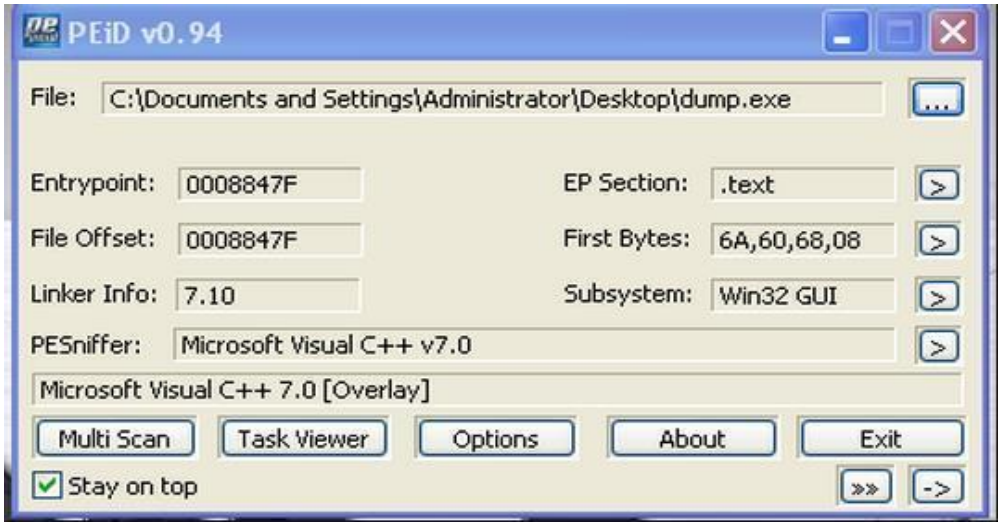

Figure4.3: PEiD to reveal packers

IDA pro is a disassembler which converts machine code to assembly language code, which helps in automatic code analysis by making use of API calls and control flow between different code sections. It is widely used by reverse engineers as it

supports multiple formats and available for different operating systems. Figure 4.4 shows a sample output of IDA pro.

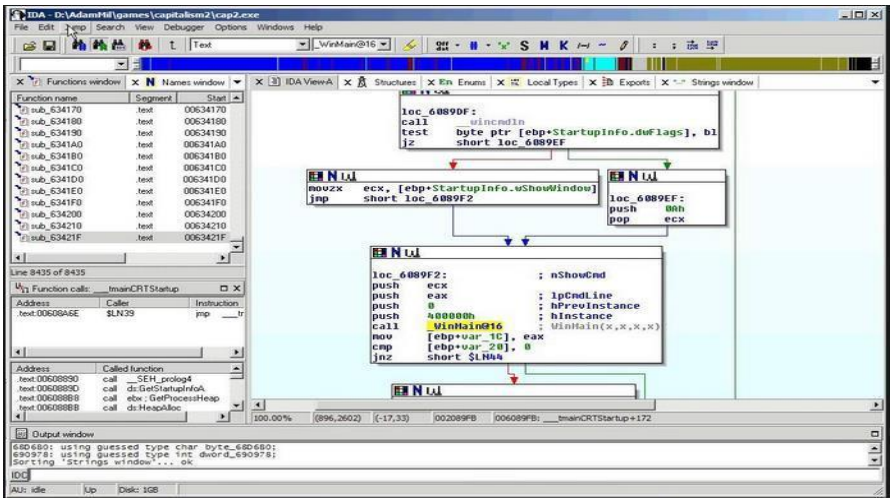

Figure4.4: IDA Pro Output

OllyDbg is an x86 debugger that is mainly used to analyze binary codes, which is helpful when source code is not accessible. It analyzes registers, identifies procedures, switches, tables, constants, API calls and strings, and also finds local routines from various object files. It has a user friendly interface, and its usefulness can be extended by installation of plugins.

OllyDbg is frequently utilized for figuring out of programs [8]. It is regularly utilized by crackers to break down software made by other programmers. It is the common tool used for reverse engineering and cracking as its open source and easy to use. With the help of OllyDbg any 32-bit executable can be converted to assembly code in no time. It is likewise valuable for software engineers to guarantee that their project is running as expected, and for malware analysis purposes. There are four principle windows in OllyDbg debugger as shown in Figure 4.5 the code window in the middle, the register window on the upper left corner, the stack window in the base right hand corner and the memory dump window in the base left corner.

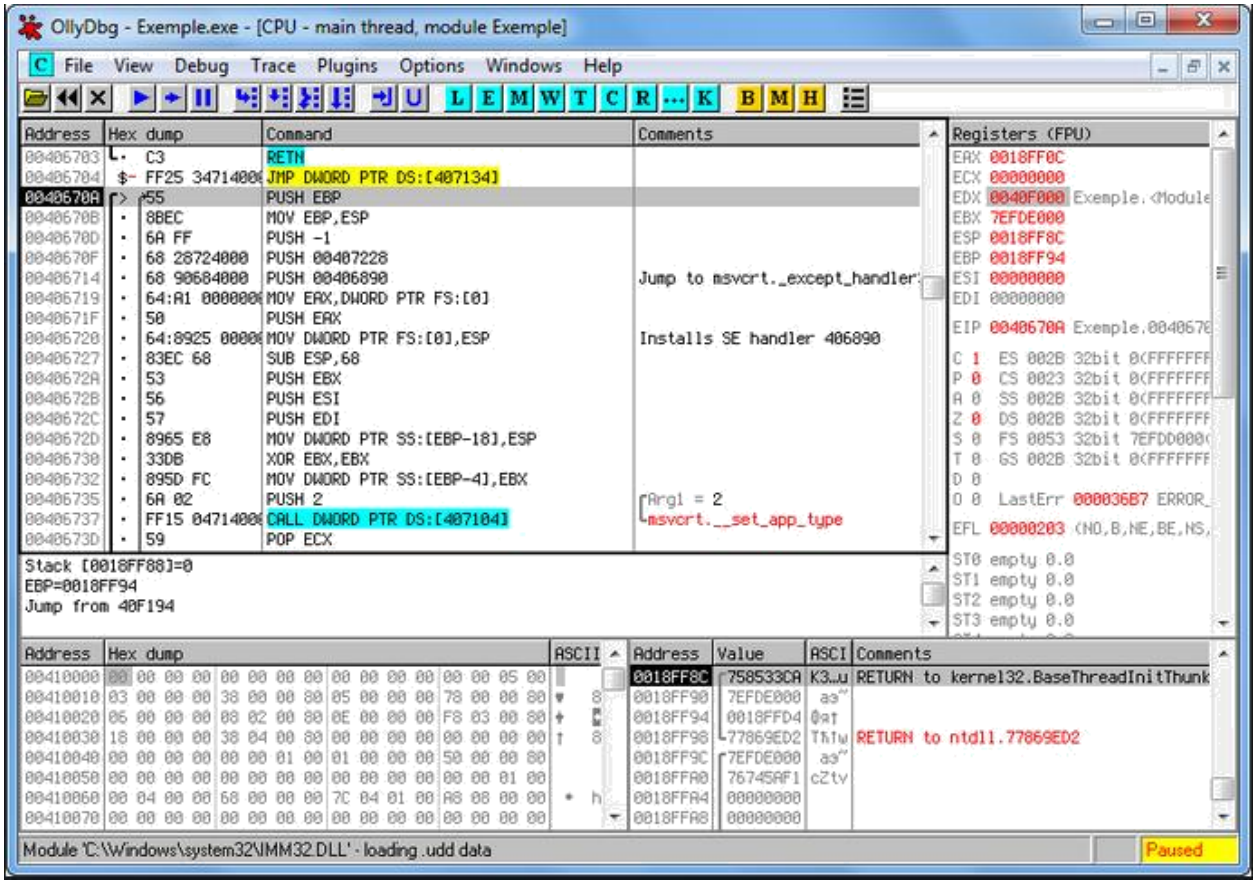

Figure4.5: Overview of OllyDbg

Most of the antivirus vendors make use of signature based detection techniques in which they used pre-defined signature sets. Signature is a unique hex code string representation of malign files or executable.

4.4 Dynamic Analysis

Automated, dynamic investigation of malware works by observing a programs execution and creating an investigation report compressing the conduct of the program. These examination reports ordinarily cover file exercises (e.g., what files were created recently), Windows registry exercises (e.g., what registry assessments were situated), system exercises (e.g., what files were downloaded, what adventure were sent over the wire), process exercises, for example, at the point when a procedure was made or ended and Windows service exercises, for example, when the service was introduced or began in the framework and so on. A few of them are freely

accessible on the Internet (Anubis, CWSandbox , Cuckoo, Joebox , Norman Sandbox). The primary thing to note about such frameworks for dynamic analysis is that they execute the binary executable for a restrained time period. Since malevolent projects don't uncover their conduct at the point when executed for a few seconds, dynamic frameworks are obliged to screen the execution of binaries for a more drawn out time. Therefore this analysis is resource constrained in terms of important equipment and time. There is additionally the issue of multiple execution paths which are difficult to be analyzed individually in a sandbox environment. Sample output of cuckoo sandbox is shown in Figure

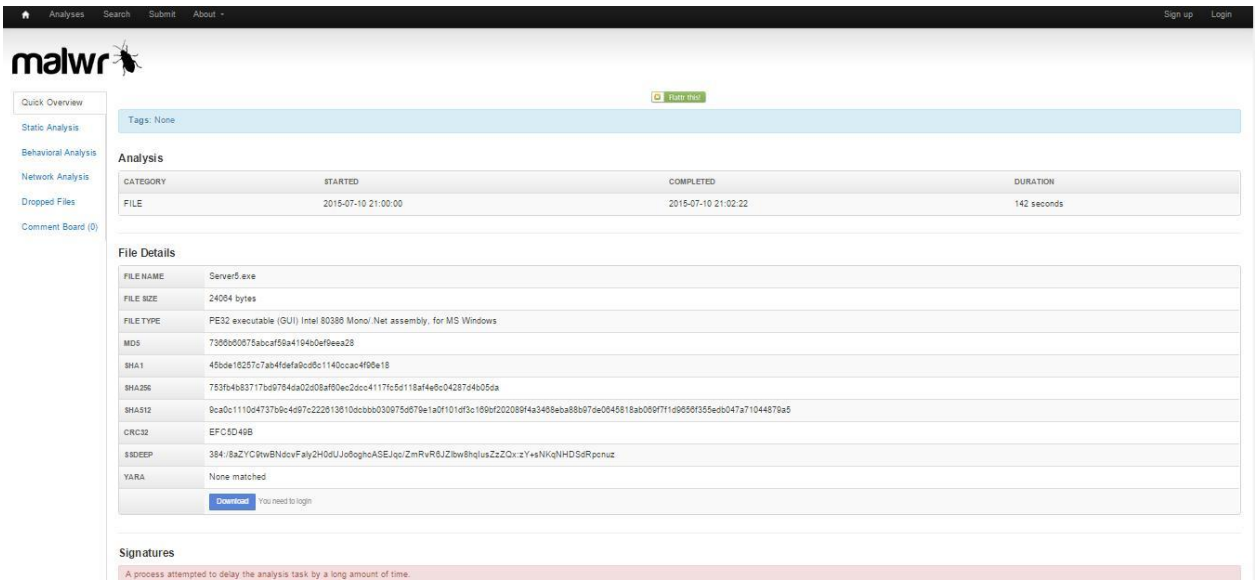

Figure 4.6 Sample Cuckoo Output

4.5 Machine Learning

Machine Learning is generally another field when contrasted with brain research. Comprehensively machine learning can be ordered into three fundamental classes as shown in Table 4.1:

i. Supervised Machine Learning.

ii. Unsupervised Machine Learning.

iii. Ensemble Learning

Table 4.1 Types of Machine Learning

Supervised	Learning algorithm is fed with some labeled training examples and yields a hypothesis function h(x) which is later utilized for forecasting unseen data e.g. SVM,LR,DT.
Unsupervised	Algorithm discovers hidden structure in unlabelled data e.g. k means, hierarchical clustering, GMM.
Ensemble	Learning from a combination of two or more models.

Machine Learning is the procedure which is utilized as a superset while applying information mining calculations.

4.5.1. Supervised Machine Learning

Supervised Machine learning is known to be administered as machine is given the essential information from the client end. Information may be in diverse structures, for example, extending expectations sheet, csv sheet, mat lab record or some other organization. Idea driving supervised learning is that machine will take in the cases from the information included and do expectations in light of that. Naive Bayes technique is the best case for indicating how this work is finished.

There are two cases in supervised machine learning and those are:

- Regression Strategy
- Classification Strategy

Regression algorithms deal with normal values and Classification algorithms take a shot of entire numbers. This explanation is the sole informative articulation for reasonable execution of the directed machine learning classification.

4.5.2. Unsupervised Machine Learning

Unsupervised Learning, as the name proposes is identified with machine adapting capabilities from raw data with a few calculations without impedance of the client giving any information. Most prominent unsupervised machine learning is Clustering.

Clustering: is the assignment of a set of observations into subsets (called clusters) so that observations in the same cluster are similar in some sense. Clustering is a method of unsupervised learning, and a common technique for statistical data analysis used in many fields.

4.5.3. Ensemble Learning

Ensemble learning is the combination of two or more machine learning and information mining techniques which fill in as joined unit to give out the best result conceivable from either calculation. This system is broadly utilized nowadays as a part of exploration in diverse fields.

Bagging and Boosting are the two methods that are utilized as a part of diverse machine learning strategies. Bagging can be characterized as Train-Test-Repeat strategy, utilizing the K-Fold Validation Technique. K-Fold Validation is considered best for estimation results in Data mining. In this approach particularly 10 fold validation the dataset is divided into 10 equal parts in size, from which 9 parts are used for validation and 1 part for testing. The process is then re-iterated 10 times so that each subpart can be used as test data. The results of every outcome are averaged to showcase how good the algorithm for the entire dataset is. Though, Boosting applies to the procedure of bolstering the information gradually and making the machine learn by expanding the weight step by step and making machine shrewder as far as computerized reasoning [30].

Cluster models join more than one machine learning model making it more powerful than one single base model results [27, 28, 29]. Classification machine learning models have been considered on the grounds that categorization of a file is the twofold class forecast i.e. either the file is malicious (class 1) or the file is benign

(class 0). General examination of the models will indicate what percent of files are malicious out of the complete dataset of 1230 samples. Deduction will be made about what attributes ought to be considered for categorization of a file.

4.6 Classification Models

In our work, six data mining algorithms are used which have been explained below:

1. LMT
2. Decision Tree
3. Naïve Bayes
4. Support Vector Machine
5. kNN
6. Neural Network

4.6.1 LMT

In machine learning, a more regular approach to manage classification tasks is to utilize a blend of a tree structure and logistic regression models resulting in a solitary tree. Another point of interest of utilizing logistic regression is that explicit class probability assessments are created as opposed to only clustering. Classifier for building logistic model trees, which are order trees with logistic regression capacities at the ends or leaves. The calculation can manage two fold and multi-class target variables, numeric and nominal properties and missing fields.

4.6.2 Decision Tree

For the most part, a Decision tree is a choice bolster device utilizing tree like diagram of choices and the conceivable results. What's more, choice tree realizing uses the above clarified choice tree as the prescient model, which utilizes the perceptions as

the traits for anticipating the objective worth. The tree models, those can take just a limited arrangement of qualities are called characterization trees. Also, where the tree variable can take esteem that is nonstop in nature i.e. the genuine numbers, are known as the relapse trees. In particular, J48 decision tree classifier is used.

4.6.3 Naïve Bayes

In machine learning, Naive Bayes classifiers are a group of straightforward probabilistic classifiers in view of applying Bayes' hypothesis with strong (naive) autonomy presumptions between the components**.** It has been in use since 1950s and is still prevalent for content categorization. In view of word frequencies it sorts the reports into different classes. Naive Bayes classifiers are straightforward and versatile. These make use of various calculations to prepare the classifier.

4.6.4 SVM

SVM or Support Vector Machine is the administered learning model which is generally utilized as a part of example acknowledgments, utilized as a part of relapse and additionally arrangement displaying.

Assume there are a few cases for preparing of the model , SVM will attempt to order it in like manner into as much as high accuracy of grouping.

SVM is of two sorts:

- Linear SVM Classifier.
- Kernel Based SVM Classifier. [Non - Linear]

4.6.5 kNN (k- Nearest Neighbour)

The kNN classifier is an example of Instance-based classifier that work on the premises that classification of unknown samples can be possibly done by relating the noval or unknown to the known with respect to some separation/similarity values. The

instinct is that two samples far separated in the instance space characterized by the distance function are less likely than two firmly arranged examples to have a place in the same class.

Classification (generalization) utilizing a instance based classifier can be a straightforward matter of finding the closest neighbor in instance space and naming the unknown instance with the same class as that of the found (known) neighbor. This methodology is frequently referred to as a closest neighbor classifier. The drawback of this basic methodology is the absence of robustness that group the subsequent classifiers. The high level of local sensitivity makes closest neighbor classifiers profoundly vulnerable to noise in the training data.

4.6.6 Neural Network Model

This model is taking into account neurons working in cerebrum. Sign starts its working with one neuron then onto the next till the yield is given by whole body. Same rationale is connected in neural systems administration in machine learning. Diverse weights are allocated to the inputs that are given to the machine. This is utilized as a part of unsupervised learning, supervised learning and reinforcement learning. Nowadays DNA reinventing is additionally being done in the most recent research in light of neural systems only.

4.7 WEKA

WEKA (Waikato Environment for Knowledge Analysis) [31] is a suite of machine learning programming written in Java and openly accessible under the GNU General Public License, created at the University of Waikato. It contains a combination of tools and techniques arranged to data analysis and predictive demonstrating. It additionally offers a helpful Graphic User Interface, which provides ease of use. It permits the execution of a few data mining functions, for example, data preprocessing, bunching, classification, regression, representation and feature importance.

Since the product is composed in Java, it can be effortlessly ported to the majority of advanced processing stages, it is uninhibitedly accessible, and contains an extensive gathering of algorithms, preprocessing strategies and modeling techniques.

We now portray the information organization connected with WEKA, which compares to the way we have to exhibit our information with a specific end goal to have the capacity to utilize the product's abilities for our goal of analysis.

4.7.1 ARFF information files

ARFF is a precise word for Attribute-Relation File Format. It is the typical input structure accepted by WEKA, comprising the explanation of the list of instances sharing an arrangement of traits or attributes. Each ARFF file has two particular segments:

- Header data comprises the name of the relation, and a table with the names and the description of the attributes;
- Information data comprises the basic information, organized by kind of attributes specified in the header area.

ARFF files take into account four sorts of attributes to be pronounced: numeric, nominal, string along with date. In ARFF files, every example is epitomized on a specific line, and attributes are separated by commas like csv (comma separated values) files. Generally the last example of every case is considered by WEKA as the particular case that must be anticipated for new cases.

4.8 Cuckoo Sandbox

Cuckoo [32] is an open source robotized malware investigation framework. To do as such it makes utilization of custom parts that screen the conduct of the noxious procedures while running in an isolated environment. Malware sandboxing is a reasonable use of the dynamical examination approach: rather than statically investigating the binary document, it gets executed and observed progressively. This

methodology clearly has advantages and disadvantages, yet it‘s a significant strategy to get extra subtle elements on the malware, for example, its system conduct. Consequently it‘s a decent practice to perform both static and element investigation while assessing a malware, so as to pick up a more profound comprehension of it.

It's utilized to consequently run and dissect files and gather exhaustive investigation conclusions about that layout what the malware does while running inside a confined Windows operating system.

It can recover the accompanying sort of results:

- Hints of win32 API calls performed by all procedures brought forth by the malware.
- Files or records being created erased and downloaded by the malware amid its execution.
- Memory dumps of the malware forms.
- Network traffic flow in form of PCAP file.
- Screenshots of Windows desktop taken amid the execution of the malware.
- Full memory dumps of the machines.

4.8.1 Custom Cases

Cuckoo is intended to be utilized both as a standalone application and also to be coordinated in bigger structures, with the help of its immense flexible design.

It can be utilized to examine:

- Universal Windows executable
- DLL documents
- PDF records
- URLs and HTML documents
- PHP scripts
- CPL documents
- Visual Basic (VB) scripts
- Java JAR

- Python documents
- And anything else given

Because of its adaptability and dominant scripting capacities, there's not farthest point to what you can accomplish with Cuckoo.

4.8.2 Cuckoo Design

Cuckoo Sandbox comprises of a principal management application which handles test execution and analysis.

Every examination is dispatched in a crisp and confined virtual machine. Cuckoo's framework is formed by a Host machine (the management application) and various Guest machines (virtual machines for examination). The Host runs the center part of the sandbox that deals with the entire investigation procedure, while the Guests are the separated situations where the malware tests get really securely executed and analyzed.

The accompanying picture clarifies Cuckoo's fundamental structural plan:

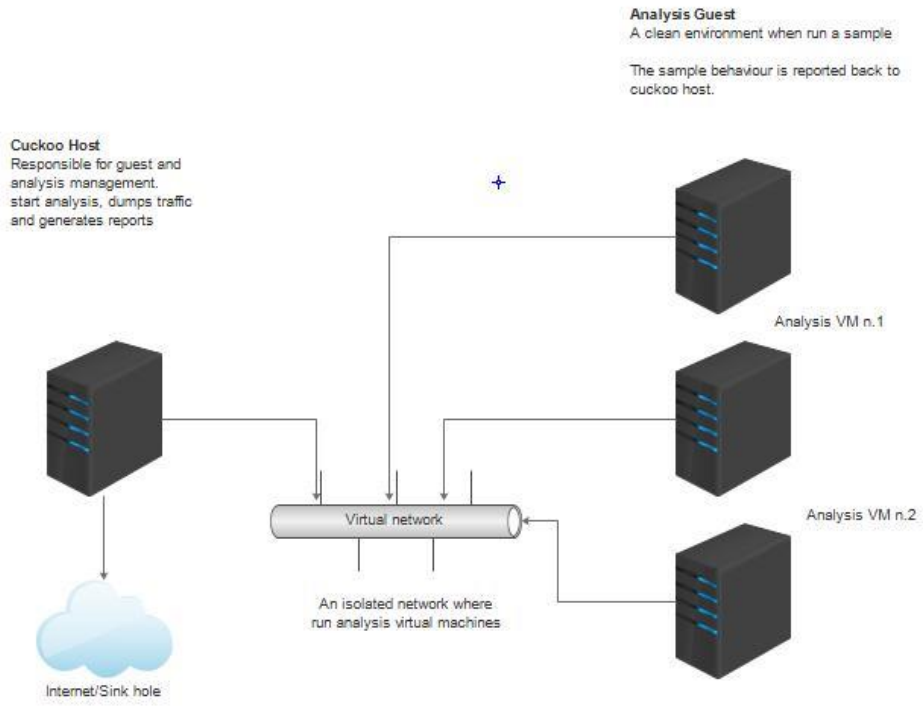

Figure 4.7: Cuckoo design

4.8.3 Other Requirements:

- **Python:** Cuckoo host parts are totally composed in Python, it becomes mandatory to install a suitable version of Python. It is a universally accepted high-level programming dialect. Its plan logic offers significance to code coherence. Contrasted with different dialects like Java and C++, Python permits the developers to compose the code in fewer lines. It empowers clear programs by giving necessary constructs. It has a few elements like programmed memory management and element sort framework and has far reaching and vast standard library. Python is a multi-paradigm programming dialect because of its full backing to organized programming and object oriented programming. A key normal for this dialect is dynamic name determination which is likewise called late binding. Because of this element, variable names and methods are limited at the point of program execution [33].

  Python is exceptionally extensible and can be used in combination with applications that require programmable interface. It is composed with the intension of having clear visual design. In python, normally English decisive words are utilized rather than punctuations. It likewise has lesser number of special cases and syntactic special cases or exceptions than other languages. As opposed to utilizing different supports, python utilizes whitespace space to delimit the squares. This component is known as off-side principle. There are three sorts of advancement situations for python:

  - Command Line: Python goes about as a shell in this creating environment. The greater part of the python executions can fill in as command line translator.
  - Integrated Advancement Environment (IDE): Different shells like IDLE and IPython include highlights past that in fundamental interpreter. These elements incorporate maintenance of session state, auto-fulfillment and syntax highlighting.
  - Browser Based IDEs: IDEs are not limited to the desktop form as well as there are program based IDEs too. A portion of the samples incorporate Sage.

Various other python libraries and dependencies proved to be useful such as pip, mongodb, XenAPI, Pydeep and Yara.

- **Tcpdump**: Tcpdump [34] is a standard packet analyzer that is usually run under the command line. It permits the client to show TCP/IP and different packets being transmitted or collected over a system to the computer is connected. It is an open source tool commonly used on Linux, Solaris, BSD and Android environments. In those frameworks, tcpdump utilizes the libpcap library to catch packets. The port of tcpdump for Windows is called WinDump; it utilizes WinPcap, the Windows port of libpcap.

  Tcpdump prints the details of network files. It can interpret packets from a system interface card or from a saved parcel file created earlier. Tcpdump can compose packets to typical output or a record.

  It is likewise conceivable to utilize tcpdump for the particular motivation behind capturing and showing the correspondences of another client or PC. A client with the higher level access rights on a system being used as a switch or gateway through which unencrypted activity, for example, Telnet or HTTP passes can utilize tcpdump to see login IDs, passwords, the URLs and contents of sites being seen, or whatever other decoded data.

4.9 Methodology

The methodology adopted for malware analysis of the following six steps:

4.9.1 Data Acquisition: The data set utilized as a part of this paper is essential one and comprises of latest malware, 2014 onwards. The mix of malicious and benign files aggregates up to 1270. The malware tests have been gathered from different online sources [37], [38], [39], and so on and benign data is gathered from System32 registry of a clean establishment of Windows 8.1 64 bit. The specimens are in different formats including Portable Executable (PE) position, HTML report format, executable, zipped organization, jpeg format and some more.

4.9.2 Behavior Auditing and Report Generation: Data acquisition is followed by automatic behavior monitoring of both malware and benign files also known as dynamic analysis. The purpose of behavior monitoring is resolved by submitting every sample to Cuckoo sandbox [32], which is pre-installed. Cuckoo generates a set of raw data which includes results from http://virustotal.com, a well-known malware detecting site, copies of malwares either created or removed from the file system, snapshots of the system taken during the execution of the malware, windows API call traces, memory dump and network dump of the machine. A peculiar feature of Cuckoo is that it presents its output in various formats such as JSON report, MAEC report, HPfeeds interface, HTML report and MongoDB interface. After submitting each sample the output report is generated in HTML format.

4.9.3 Data Preprocessing

Data processing is the next step which is explained here. The entire HTML report files were disintegrated and the most relevant attributes were selected (feature selection).

The features initially fed into the .csv file include File name, File size, File type, CRC32, MD5, SHA1, SHA256, SHA512, Ssdeep , Section name, Virtual address, Virtual size, Size of raw data, Entropy, Registry keys, Imports and IP address in the same order. Though on applying feature selection, importance of the features such as CRC32, MD5, SHA1, SHA256, SHA512 was found to be least. Registry keys, IP addresses, Section names have utmost importance in the analysis part.

After selection of appropriate features, Attribute-Relation File Format (ARFF) files were created. ARFF is a standard file format accepted by WEKA. In this thesis, Window‘s OS version of WEKA 3.2 is utilized.

4.9.4 Classification and Learning using WEKA

Resulting ARFF file is fed into WEKA [31]. Weka (Waikato Environment for Knowledge Analysis) is a free machine learning software written in Java and

is developed at the University of Waikato, New Zealand. It contains set of algorithms for classification and clustering which can be used for analysis.

Different classifiers are applied for classifying the various malwares and their accuracy for correctly classifying malware instances is noted. We have used 10-fold cross validation technique for estimation as it has been considered best for estimation results in Data mining. In this approach the dataset is divided into 10 equal parts in size, from which 9 parts are used for validation and 1 part for testing. The process is then re-iterated 10 times so that each subpart can be used as test data. The results of every outcome are averaged to showcase how good the algorithm for the entire dataset is.

4.9.4 **Output**: Dataset so obtained in ARFF format is applied to 5 different classifiers namely Logistic Model Trees (LMT) , Naïve Bayes, Instance based learning using k-nearest neighbors, (IBk), Ridor and SVM. Detection rate is calculated on the basis of five metrics which include True Positive Rate (TPR), False Positive Rate (FPR), Precision, Recall and F-measure .The output is finally verified from Virustotal results.

Chapter 5

Results and Discussion

In this chapter, result for malware analysis on dataset 1230 samples consisting of both malware and benign files is shown and further discussed to have a deep knowledge of functionality of malware.

5.1 Classification Evaluation Metrics

The binary classification issue in machine learning can be expressed in the following way. Given an example of n training items (x; y), the learning technique regularly gives back a model h(x), that minimizes the normal error in the outcome in comparison to the actual joint distribution D over the input and outcome variables. Scientific proof for chosen methodology and Evaluation Metrics are known as the Type I and Type II blunders to gauge the execution of a classifier.

The terms true negatives, true positives, false negatives and false positives, and contrast the outcome of the classifier under test with trusted outside naming. The terms positive and negative indicate the classifier's expectation or prediction, and the terms true and false alludes whether that expectation is same as that of the outside judgment.

Consider a study assessing a test that screens individuals for a disease. Every individual taking the test either has or does not have the disease. The test result can be sure (foreseeing that the individual has the disease) or negative (anticipating that the individual does not have the disease). The test results for every subject could possibly coordinate the subject's genuine status. In that setting:

True positive: Sick individuals accurately analyzed as sick

False positive: Healthy individuals mistakenly recognized as sick

True negative: Healthy individuals effectively recognized as healthy

False negative: Sick individuals inaccurately recognized as healthy

On a general basis:

True positive(tp): True positive is the number of correctly identified instances.

True negative(tn): True negative refers to the number of correctly rejected instances.

False positive(fp): False positive is the number of incorrectly identified instances.

False negative(fn): False negative indicates the number of incorrectly rejected instances.

1. True positive rate: The True Positive (TP) rate is the portion of illustrations which were named class x, among all samples which genuinely have class x, i.e. it indicates the captured part of the class. Mathematically,

$$\text{True positive rate} = \frac{tp}{fn + tn}$$

2. False Positive rate: The False Positive (FP) rate is the portion of illustrations which were identified as class x, though they are a part of other class, among all samples which are not of class x. Mathematically,

$$\text{False positive rate} = \frac{fp}{fp + tn}$$

3. Precision: Positive predictive value is another term for precision. It is the proportion of true positives or the quantity of instances classified effectively by the classifier to the aggregate number of positive instances in the experiment (true positives + false positives). Hence,

$$\text{Precision} = \frac{tp}{tp + fp}$$

4. Recall: Recall is also known as sensitivity or true positive rate. It is the number of true positives or number of instances identified by the classifier divided by the total number of positive instances in the output (true positives + false negatives).

$$\text{Recall} = \frac{tp}{tp + fn}$$

5. F-measure: A combined measure for precision and recall is known as F-score or F- measure. It is nothing but harmonic mean of recall and precision and is calculated by using the formula:

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

6. Accuracy: Accuracy is the most important metric in classification and is suffice in itself. Accuracy is defined as the total number of true positive and true negatives divided by the sum total of all the instances in the experiment. Thus, mathematically accuracy is given by the below formula:

$$\text{Accuracy} = \frac{tp + tn}{tp + tn + fp + fn}$$

5.2 Classification before feature selection

Initially all the features shown by Cuckoo sandbox are considered for classifying malwares. The feature set contains the static features, section details, resources, results of various anti-viruses and network analysis. A detailed list of all the features extracted from WEKA is shown in Figure 5.1.

S.No.	Features	S. No.	Features	S.No.	Features
1	File Name	15	Internalname	28	Size
2	File Size	16	File Version	29	Language
3	File Type	17	Comments	30	Sub Language
4	MD5	18	Product Version	31	File Type
5	SHA1	19	File Description	32	Imports
6	SHA256	20	Original Filename	33	Strings
7	SHA512		**Section**	34	Anti Virus
8	CRC32	21	Section Name	35	API
9	Ssdeep	22	Virtual Address		**Network Analysis**
10	Yara	23	Virtual Size	36	Domains
	Static Analysis	24	Size of Raw Data	37	Hosts
11	Version Info	25	Entropy	38	HTTP
12	Translation		**Resources**	39	IRC
13	Legal Copyright	26	Resource Name	40	SMTP
14	Assembly Version	27	Offset	41	Registry Keys

Figure 5.1 Features extracted from Cuckoo sandbox

After applying 5 different classifiers namely LMT, Naïve Bayes, IBk, Ridor and kNN on the data set, the following results were obtained. The performance of the classifier was tested on the basis of TP rate, FP rate, Precision, Recall and F-measure as shown in Table 5.1.

Table 5.1: Classifier performance before feature selection

Classifier	**TP Rate**	**FP Rate**	**Precision**	**Recall**	**Accuracy**
LMT	73.393	11.7	73.389	73.389	73.3857
Naïve Bayes	68.2	16.9	68.19	68.20	68.2315
SVM	71.18	14.31	71.04	71.04	71.0131
Ridor	53.47	18.88	54.19	54.20	54.2692
kNN	61.37	22.82	61.38	61.38	61.3697

5.3 Classification after feature selection

After analyzing the above results, there was a need to apply feature selection to improve the TP rate. Features were selected by manually by checking the output of classifier on application of appropriate features. The features selected were as shown in Figure 5.2.

S.No.	Features
1	FileType
2	Section Name
3	Virtual Address
4	Entropy
5	Resource Name
6	Imports
7	Antivirus
8	Domains
9	Registry Keys

Figure 5.2 Selected features

Features namely File Type, Section Name, Virtual Address, Entropy, Imports, Domains were having high contributions. The feature selection was done so as to include one feature was included from each part of static analysis, dynamic analysis and network analysis.

On applying the previously used classifiers on this feature set, relatively higher performance was achieved which is shown by Table 5.2. Among the 5 chosen classifiers higher TP rate is shown by LMT classifier with least number of false positives. LMT classifier is best suited for malware detection with high accuracy.

Table 5.2 Performance after feature selection

Classifier	TP Rate	FP Rate	Precision	Recall	Accuracy
LMT	98.3	7.7	98.3	98.3	98.2857
Naïve Bayes	96	13	95.9	96	96
SVM	97.7	7.8	97.7	97.7	97.7143
Ridor	94.3	15.8	94.2	94.3	94.2857
kNN	81.7	8.19	81.8	81.8	81.6379

5.4 Major Attackers

The list of major attacker with their location as reported by nslookup is shown in Table 5.3. As per the table, one the major source of all attacks is USA which alone accounts for about 66.95% of the attacks. Following USA, Germany is the second major source of attacks and account to 10.17% of the total attacks. Among all the attacks, frequency of the attack by 208.113.163.195 is highest. This particular IP belongs to USA and was involved in 18 major attacks. According to heuristics, this may be an attempt of DOS attack.

Due to security reasons, complete private IPs are not revealed which public IPs shown are complete. Some of the IPs that are used to launch attacks are blacklisted IPs. A wide majority of attacks are launched by private IPs while one or two public IPs are involved in the attacks.

Table 5.3 Location of major attackers

IP address	Location reported by whois	Number of Hits
72.52.224.*,208.113.163.*,23.10.23.*,52.6.1.*,104.18.62.*,23.107.23.*,108.171.164.*,74.125.129.*,23.253.254.*,91.211.17.*,38.124.72.*,104.131.25.*,173.194.76.*,23.21.142.*,198.41.182.*	USA	8234
84.45.8.*, 188.165.164.*, 44.50.2.*, 195.8.66.*, 37.59.212.*	France	125
91.211.17.*	Ukraine	41
95.213.4.*, 62.76.179.*, 193.93.193.*	Russian Federation	41
78.46.76.*,5.9.80.*,192.168.226.206	Germany	41
105.101.69.*, 41.107.139.*	Algeria	26
62.76.179.*	Israel	20
5.79.6.*	UK	11
177.55.109.*	Brazil	11
188.49.15.*	Saudi Arabia	11
222.236.47.*	Korea	11
105.189.63.*	Morocco	11
81.17.250.*	Ireland	4
46.246.12.*	Sweden	4

A pictorial representation showing the distribution of major attacks is shown in Figure 5.3. A vast area is covered by USA, followed by France, Ukraine, Algeria and others.

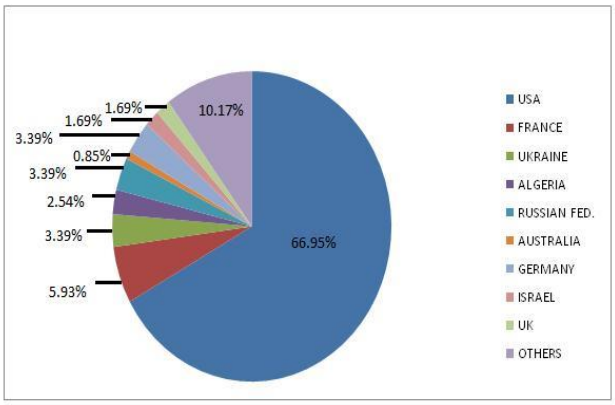

Figure5.3. Pie chart depicting major attacker

5.5 Registry keys

On a deeper analysis, malware were found be residing on some specific registry keys. These registry keys are collected by calculating the frequency of registry keys of the malicious binaries. The following is the list of registry keys having highest frequencies.

Table 5.4 Registry keys used by malware

S.No.	Registry Keys
1	HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
2	HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\WindowsNT\CurrentVersion\Winlogon\ Notify
3	HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Session Manager
4	HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\system
5	HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\explorer\User Shell Folders
6	HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunServices

5.6 Entropy

Entropy is nothing but the randomness gathered by an Operating system for utilization in cryptography or different uses that require irregular information. This randomness is frequently gathered from equipment sources, either previous ones, for example, mouse movements or some specific randomness generators.

It provides a rough guide as to what analysis methods to use and what to expect. For example, a file with low entropy values can mean an encrypted sample and static analysis can be straight way used. A file with high entropy, on the other hand, may mean that the malware is packed and that dynamic analysis may be more useful.

Mean of all the entropy value is calculated which comes out to be 5.323887333. According to this, all binaries having entropy value less than the mean needs to be

analyzed by static and for the other having higher entropy value dynamic analysis is used.

Table 5.5 Determination of type of analysis

Entropy	**Analysis**
<5.323887333	Static analysis
>5.323887333	Dynamic Analysis

Chapter 6

Conclusion and Future Scope

The outcome of our implementation has shown that LMT classifier is best suited for classifying the algorithms into malicious or benign classes in our case. LMT algorithm gave optimum results clustering malware samples into different classes. These results can be used by anti-malware writers for detecting whether a particular binary is malicious or benign within very less time. As behavior is the most important factor for deciding whether a file is malicious or benign, our results are optimum. Some more insights can also be drawn on the behavior of the malware including the detection of code cave injection, API calls made, number and name of the files being affected.

One special intriguing field for further research would be examining the location of various infections on a PC utilizing machine learning methods. This wouldn't be conceivable utilizing a solitary naive bayes classier, as it can just give back a single class for an occurrence of data. By getting more infections which spread effectively we could further fortify our conclusions. Another methodology for recognizing infections which would be fascinating to attempt, could include contrasting system calls made by viruses with those made by genuine programs. This could be performed utilizing the project `strace' which has the capacity to record system calls made. This data could be logged for both infections and ordinary projects. We also wish to expand our dataset so that analysis can be done at a larger scale with higher accuracy and efficiency. Methods for further improving classifier accuracy can be devised. To speed up the analysis process, combination of more than one algorithm will be applied. In such a way, zero days attacks will be detected and handled in a much more efficient way.

References

[1] Bayer M., Ulrich et al. ―Dynamic Analysis of Malicious Code‖. *J Computer Virology* 2.1 (2006): 67-77. Web.

[2] Ashar A., et al. (2013). The Need for Speed: 2013 Incident Response Survey, FireEye [Online] . http://www.inforisktoday.in/surveys/2013-incident response-survey-s-18/ (Accessed: May, 2015)

[3] Yhang Yu., et al. (2012). Addressing Big Data Security Challenges: The Right Tools for Smart Protection, http://www.trendmicro.com/cloudcontent/us/pdfs/business/whitepapers/wp_addressing-big-data-security-challenges [Online]. (Accessed: April, 2015)

[4] Ashar A., et al. (2013) Next Generation Threats. http://www.fireeye.com/threat-protection/ [Online] (Accessed 12 may, 2015)

[5] Young K., et al., ―Internet Abuse In The Workplace: New Trends In Risk Management‖. *CyberPsychology & Behavior* 7.1 (2004): 105-111. Web.

[6] Rehman R., et al., ―Malware Threats and Mitigation Strategies: A Survey‖. *Journal of Theoretical and Applied Information Technology* (2011). *29*(2), 69-73. Web.

[7] Imtithal, A., et al., ―A Survey on Malware And Malware Detection Systems‖. *International Journal of Computer Applications* 67.16 (2013): 25-31. Web.

[8] Chen, X., Nazario, J., et al. Towards an understanding of anti-virtualization and anti-debugging behavior in modern malware. In *Dependable Systems and Networks With FTCS and DCC, 2008. DSN 2008. IEEE International Conference on* (pp. 177-186). IEEE.

[9]Cohen, F. (1987). Computer viruses: theory and experiments. *Computers & security*, *6*(1), 22-35.

[10] Alazab, M., Layton, R., et al., ―Malware detection based on structural and behavioural features of api calls‖ IN J(2010).

[11] Eskandari, M. and Hashemi S. et al. ―A graph mining approach for detecting unknownmalwares‖. Journal of Visual Languages & Computing In Computer security applications conference, 2012. ACSAC 2012. Twenty-third annual (pp. 421-430). IEEE.

[12] Moser, A., Kruegel, C., et al.,― Limits of static analysis for malware detection.‖ In Computer security applications conference, 2007. ACSAC 2007. Twenty-third annual (pp. 421-430). IEEE.

[13] Bailey M., Oberheide J., et al., ―Automated classification and analysis of internet malware", In Proceedings of RAID (2007) (pp. 67-73) IEEE .

[14] Lee T., Mody J., Behavioral classification in Proc. of EICAR, 2006.

[15] Bayer, U., Kirda, E., et al., C.: Improving the efficiency of dynamic malware analysis. In: Proceedings of the 2010 ACM Symposium on Applied Computing, ser. SAC 10, pp. 18711878. ACM, New York (2010).

[16] Schultz M., Eskin G., et al., (2001). ―Data mining methods for detection of new malicious executables‖. In *Security and Privacy, 2001. S&P 2001. Proceedings. 2001 IEEE Symposium on* (pp. 38-49). IEEE.

[17] Kolter, J. Z., & Maloof, M. A. (2004, August). Learning to detect malicious executables in the wild. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 470-478). ACM.

[18] Moskovitch, R., Stopel, D., et al., (2008, June). Unknown malcode detection via text categorization and the imbalance problem. In *Intelligence and Security Informatics, 2008. ISI 2008. IEEE International Conference on* (pp. 156-161). IEEE.

[19] Santos, I., Penya, Y. K., Devesa, et al.,. (2009). N-grams-based File Signatures for Malware Detection. *ICEIS (2)*, *9*, 317-320.

[20] Zhou, Y., & Inge, W. M. (2008, October). Malware detection using adaptive data compression. In *Proceedings of the 1st ACM workshop on Workshop on AISec* (pp. 53-60). ACM.

[21] Yafang, Ye., Li, T., Chen, Y., & Jiang, Q. (2010, July). ―Automatic malware categorization using cluster ensemble.‖ In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 95-104). ACM.

[22] Siddiqui, M., Wang, M. C., & Lee, J. (2008). ―Detecting internet worms using data mining techniques.‖ *Journal of Systemics, Cybernetics and Informatics,6*(6), 48-53.

[23] Sung, A. H., Xu, J., et al. (2004, December). ―Static analyzer of vicious executable (save).‖ In *Computer Security Applications Conference, 2004. 20th Annual* (pp. 326-334). IEEE.

[24] Saini, A., Gandotra, E., et al., (2014, September). ―Classification of PE Files using Static Analysis.‖ In *Proceedings of the 7th International Conference on Security of Information and Networks* (p. 429). ACM.

[25] Ugarte-Pedrero, X., Santos, I., et al., P. G. (2012, January). Countering entropy measure attacks on packed software detection. In *Consumer Communications and Networking Conference (CCNC), 2012 IEEE*(pp. 164-168). IEEE.

[26] Willems, C., Holz, T., & Freiling, F. (2007). Toward automated dynamic malware analysis using cwsandbox. *IEEE Security & Privacy*, (2), 32-39.

[27] E. Alpayd, ―Introduction to Machine Learning‖, The MIT Press, February 2010.

[28] R. Polikar, ―Ensemble Based Systems in Decision Making‖, IEEE Circuits and systems Magazine, Third Quarter, 2006.

[29] Lei Xu, Adam K, Ching Y. Suen, ―Methods of Combining Multiple Classifiers and Their Applications to Handwriting Recognition‖, IEEE Transactions on Systems, Vol. 22, No. 3, May/June 1992.

[30] Kaur A., Kaur K., ―Performance Analysis of Ensemble Learning for Predicting Defects in Open Source Software‖, IEEE International Conference on Advances in Communications and Informatics (ICACCI-2014), New Delhi, Pages 219-225, 24-27 Sept, 2014.

[31] WEKA, the University of Waikato, and Available at: http://www.cs.waikato.ac.nz/ml/weka/, (Accessed: April, 2011).

[32] Claudio G., Alessandro T., et al.,(2007), Cuckoo[online].Available: http://cuckoosandbox.org/about.html (Accessed: March, 2015)

[33] Pedregosa, F., Varoquaux, G., et al., (2011). ‗Machine learning in Python'. *The Journal of Machine Learning Research*, *12*, 2825-2830.

[34] Jacobson, V., Leres, C., & McCanne, S. (1989). The tcpdump manual page. *Lawrence Berkeley Laboratory, Berkeley, CA*.

[35] Pietrek, M. (1994). Peering inside the PE: a tour of the win32 (R) portable executable file format. *Microsoft Systems Journal-US Edition*, 15-38.

[36] Lyda, R., & Hamrock, J. (2007). ―Using entropy analysis to find encrypted and packed malware‖. *IEEE Security & Privacy*, (2), 40-45.

[37]Al-Assad(2011,April). Syrian Malware Samples [online]. Available:http://www.syrianmalware.com/ (Accessed: June, 2015)

[38]Matteo C. et al. (2014) Malware archives download [online].Available: http://www.nothink.org/honeypots/malware-archives/ (Accessed: June, 2015)

[39]Lenny Zelster(2007). Malware Sample Sources for Researchers [online].Available: https://zeltser.com/malware-sample-sources (Accessed: June, 2015)

[40] Qiao, Y., et al. (2014). CBM: free, automatic malware analysis framework using API call sequences. In *Knowledge Engineering and Management* (pp. 225-236). Springer Berlin Heidelberg.

Publication

Communicated:

A. Dhammi and M. Singh, ―Machine Learning Approach to Malware Analysis and Reporting‖, in International Symposium on Advanced Computing and Communications, IEEE, 2015.

Video Link

https://www.youtube.com/watch?v=Jlm6RMg5VbY

Plagiarism Report