Malware Detection

using Machine Learning

A Preliminary Master's Thesis
Department of Information Technology
Indian Institute of Information Technology Allahabad

SUBMITTED BY :

VIKAS VERMA (MIT2021077)

GUIDED BY :

Dr. SOUMYADEV MAITY SIR

Abstract :

.According to the recent studies, malicious software (malware) is increasing at an alarming rate, and some malware can hide in the system by using different obfuscation techniques. In order to protect computer systems and the Internet from the malware, the malware needs to be detected before it affects a large number of systems. Recently, there have been made several studies on malware detection approaches. However, the detection of malware still remains problematic.

Signature-based and heuristic-based detection approaches are fast and efficient to detect known malware, but especially signature-based detection approach has failed to detect unknown malware. On the other hand, behavior-based, model checking-based, and cloud-based approaches perform well for unknown and complicated malware; However, no approach can detect all malware in the wild.

- Any software which intentionally executes malicious payloads on victim machines (computers, smart phones, computer networks, etc.) is considered as malware.
- There are different types of malware including virus, worm, Trojan horse, rootkit, and ransomware. Each malware type and family is designed to affect original victim machine in different ways such as damaging the targeted system, allowing remote code execution, stealing confidential data, etc.
- According to scientific and business reports, approximately 1 million malware les are created every day, and cybercrime will damage the world economy by approximately $6 trillion annually by 2022 .

In this project we have worked on how machine learning can be combined with static and behavioural analysis in order detect malware in a system . various machine learning algorithms are performed and compared with each other and we found that , random forest gave us best efficiency for detecting malware .

Later on our main focus will be on applying behavioural analysis and comparision between both static and behavioural approach will be shown so that an effiecient malware detection system can be constructed .

Acknowledgement :

The semester project opportunity I had with Department of information and technology, IIIT , Allahabad was a great chance for learning and professional development. Therefore, I consider myself very lucky as I was provided with an opportunity to be a part of it.

I also express our deepest gratitude and special thanks to the DR . Soumyadev Maity sir , who in spite of being extraordinarily busy with his duties, took time out to hear, guide and keep us on the correct path and allowing us to carry out our project at his esteemed organization.

Finally, I would also like to thank IT lab Centre for providing us with the facilities and the necessary resources to complete our work.

Table of Contents

List of Figures

List of Tables

1. Introduction

The Internet is an essential part of our life, but its security is seriously threatened by malware. The technological revolution during the last decade has introduced a significant increase in individuals that rely on computers and the Internet in order to communicate, store information and conduct business. But an increase in the users has been accompanied by an increase in malicious attacks. Millions of computers have been compromised by various malware types and families which are used to launch all kinds of attacks and illicit activities such as remote encryption, spam, phishing and DDoS attacks. In order to cope with this, antivirus (AV) vendors are focussing all their efforts in making use of state-of-the-art methods to detect these malicious softwares with limited number of false positives.

Current AV detection methodologies rely, among other methods, on reverse-engineering the malware executable in order to extract code-block patterns that will point to various malware types and families, which is known as signature-based analysis. Hashes are then generated for the each malicious executable and added to a vast database that is bundled in AV software for detection on the client machine and has the purpose of fast-detecting the threat. As the number of malware types increase with new potential gains, their implementation evolves and adapts in order to take advantage of new vulnerabilities. This high number results often represents a significant increase of data that needs to be provided by the AV vendor and affects the performance and accuracy of the AV solutions. Since it is necessary that the hash is present in the database in order to detect the malware, zero-day detection becomes impossible.

An alternate to this is behavioral analysis, where a file is classified as malicious based on the features it depicts during its execution. When a new file is to be checked, it is executed in an isolated environment, known as a sandbox, and its executional behavior is then monitored. If the recorded behavior resembles that of some known malware, then the given file is classified as malicious.

In this project we have worked on how machine learning can be combined with static and behavioural analysis in order detect malware in a system . Various machine learning algorithms are performed and compared with each other and we found that , random forest gave us best efficiency for detecting malware .

Later on our main focus will be on applying behavioural analysis and comparision between both static and behavioural approach will be shown so that an effiecient malware detection system can be constructed .

2. Literature Survey

Most traditional antivirus vendors use signature based malware detection. Shanklin S. D. et al (2002) [1] have patented an intrusion detection system that works on malware signatures stored in a high level syntax for future reference ease. Schmidt A.D. et al (2009) [2] proposed a two-fold model that extracted features from static analysis and then used those to classify android malware. Recently, Chua Z.L. et al (2017) [3] have proposed a new system called EKLAVYA in which they have trained an RNN to recover function type signatures from disassembled binary code. However, static analysis have its own disadvantages such as code obfuscation and polymorphism. Moser et al (2007) [4] have discussed these in detail.

Therefore, in recent years, research efforts have gone towards shifting from signature analysis to behavioral analysis, where the execution of a malicious sample is conducted and its behavior recorded in an isolated sandbox environment. Rieck K. et al (2011) [5] have developed an automated behavioral model capable of analysing and classifying more than 3,000 samples a day. Tian et al (2010) [6] attempted to use behavioral features to differentiate between malware and cleanware. Pirscoveanu et al (2016) [27] came up with a behavioral clustering model to cope with antivirus label inconsistencies. Chen S. et al (2016) [7] used a combination of behavioral features such as API calls and permissions to develop a machine learning based android malware detection framework. Bayer et al (2010) [8] proposed to improve the efficiency of dynamic analysis techniques by searching for modified API calls instead of running the whole analysis again for polymorphic viruses.

Nowadays, many antivirus vendors are using both static and dynamic analysis combined with machine learning for efficient detection of malware. Wang J. et al (2015) [9] proposed a model that combines static and dynamic analysis in order to detect javascript files. Graziano M. et al (2015) [10] have proposed a novel methodology which makes use of both static and dynamic analysis to automatically identify malware development cases.

But using machine learning for malware detection has certain drawbacks too. One major problem is the aging of classifiers. Jordaney R. et al (2017) [11] have addressed this issue by showing how a concept drift in malware can render these machine learning frameworks useless. Another drawback is that, malware creators often manipulate malwares to detect virtual machine environment to evade detection. Miramirkhani N. et al (2017) [12] have shown that features related to wear and tear can help in effectively detecting a sandbox environment which can help malware to easily evade detection. This takes us to the most critical drawback: the issue of evasion of the classifiers by adversaries. Xu W. et al (2016) [13] have proposed a method to evaluate robustness of classifiers under attack. Their method involves stochastically manipulating a malicious sample (pdf malware) to find a variant that preserves the malicious behavior but is classified as benign by the classifier. Dhilung K. and Giovanni V. (2015) [14] have proposed a novel technique called MalGene which automates the extraction of evasive signatures after performing evasive malware analysis. While Smutz C. and Stavrou A. (2016) [15] have proposed a method, ensemble classifier mutual agreement analysis, that makes use of ensemble classifiers is detect many forms of classifier evasion without additional external ground truth.

Due to the existing drawbacks of malware detection techniques, Meng X. and Taesoo K. (2017) [16] have proposed a new perspective for detection of malicious documents called platform diversity. Making use of this concept, they came up with PlatPal which can hook to Adobe Reader to detect malicious documents. Some other techniques include malicious server identification through fingerprint generation as discussed by Xu Z. et al (2014) [17] and malicious network traffic identification to detect new or unseen variants of malware as discussed by Bartos K. et al [18].

3. Problem Identification and Discussion

This section investigates the problem of malware and possibility of detection. It can be said that it is impossible to design an algorithm which can detect all malware. This is because the problem of detecting the malware has shown *NP-complete* in many studies.

This is important because before starting to build an effective detection system, it is a good practice and experience for researcher to understand the scope, limitation, and possibility of malware detector. The possibility of detection malware is remaining problematic because theoretically it is a hard problem, and practically malware creators using complicated techniques such as obfuscation to make detecting process very challenging.

The recent surge in malware attacks all around the world has made the issue of efficient malware detection a hot topic of discussion. Our project deals with how analysing malware on behavior based features and not only on features extracted from static analysis and comparing both static as well as behavioural analysis. In this section, we'll discuss what is malware and what are its types. We will also discuss the various labelling techniques used antivirus vendors and finally carry out a detailed discussion about the problem we have tried to solve through our project.

<u>DIFFICULTY OF PROBLEM IN PRACTICE </u>:

- The new generation malware uses the common obfuscation techniques such as encryption, oligomorphic, polymorphic, metamorphic, stealth, and packing methods to make detection process more difficult. This kind of malware can easily bypass protection software that is running in kernel mode such as firewalls, antivirus software, etc. and some malware instances can also present the characteristics of multiple classes at the same time. This makes practically almost impossible to detect all malware with single detection approach .

3.1. Malware and its types

Malware is an abbreviated term meaning “malicious software”. This is software that is specifically designed to gain access or damage a computer without the knowledge of the owner. There are various types of malware including spyware, keyloggers, true viruses, worms, or any type of malicious code that infiltrates a computer. [Norton antivirus]

Generally, software is considered malware based on the intent of the creator rather than its actual features. Malware creation is on the rise due to the sheer volume of new types created daily and the lure of money that can be made through organized Internet crime. Malware was originally created as experiments and pranks, but eventually led to vandalism and destruction of targeted machines. Today, much of malware is created for profit through forced advertising (adware), stealing sensitive information (spyware), spreading email spam or child pornography (zombie computers), or to extort money (ransomware). Various types of malware exist in the wild. Below we have discussed about some types and families found in antivirus labels.

- **Adware** represents one of the least dangerous types as its only purpose is to display ads to the user. In order to provide the infected machine with ads that the user might be interested in, it logs information like browser history, search engines history or history of installed programs. Depending of the severity of the logging, Adware may be labeled by AV vendors as Spyware.

- **Spyware** represents a type of malware which installs itself without the permission of the user. Used to collect browsing history and tracking information it usually bundles with free software (Symantec, 2009) [19]. AV vendors also name this type, PUP, just because of the bundling with freeware.

- **Worm** represents a similar type as a Virus, being able to do the same amount of damage to an infected machine. The main difference is represented by its independence from other software as it does not require a host program to attach itself to. A worm usually infects its target via exploits or vulnerabilities and it uses different transport protocols to spread and infect other machines (Cisco, 2015) [20].

- **Bot** represents a malware type that grants access of the infected machine to its master. This type can spread using Backdoors opened on the target by a Virus or a Worm and it is mostly known for using Internet Relay Chat (IRC) to communicate with its master. With multiple bots, Distributed Denial of Service (DDoS) attacks can be initiated that could block the services of the target by overwhelming it with requests.

- **Ransomware** represents a more sparse type of malware that takes control of the graphical interface and blocks the user from accessing its machine until a certain amount of money is paid. Most commonly, these types infect their targets via Trojan Horse.

- **Trojan Horse** is presented as software that the user might find useful, just like any other legitimate program. By opening the package, this malware releases other types of malware that will infect the machine, including key-loggers, account stealers etc. Compared with Viruses and Worms, Trojans do not replicate on their own but instead they require user interaction to do so. For this reason, this type is one of the most dangerous out there as it is usually detected when it has already infected the machine (Cisco, 2015) [20].

- **Virus** represents a malware type that can exhibit actions ranging from just showing random errors to taking the system in a Denial of Service (DoS) state. The main difference between a Trojan and a Virus represents the ability to self-replicate by becoming part of other legitimate software. These types are commonly spread by sharing files, disks or e-mails to which the virus has attached on (Cisco, 2015) [20].

Some companies follow the CARO convention which presents a standard format for malware labelling. In the next section we will be discussing the various techniques adopted by antivirus companies to come up with these labels.

3.2. Labelling techniques adopted by AV vendors

Malicious software has advanced in both complexity of coding and methods of obfuscation and polymorphism the detection more and more in order to cover all possible use cases. As technology evolved, the detection methods that AV vendors used have changed as well, improving their accuracy for new-released malware. Methods currently used are:

- **Signature Based** - Most commonly used by AV vendors to fast detect known malicious software. There exist two different signature-based methods:

  Hash *Signatures* are used by AV vendors to compare the hash of the file with known malware hashed signatures. This provides a fast detection rate only if the malware exists in the AV database or if the malware has not changed. AV vendors commonly use MD5 message-digest algorithm that provides a very accurate result. For example if a single character is changed in a file-name, the MD5 will return a totally different hash [21].

  Byte *Signatures* represent a sequence of bytes that are present in executable files or data streams. This method can accurately detect malware families by analysing sequences in data streams of an executable file [22].

- **Heuristic Based** - The method attempts to detect malware by simulating the run-time of the suspected executable and determine its intent. This approach is useful when, for example, an updated version of a known malware is released and it is mostly used alongside signature based detection to form a completed real-time protection in terms of known and new malware threats.

- **Behavioral Based** - A more advanced method of detection that requires the malware to run and infect a machine in order to record its actions. Behavioral based detection is done in a confined environment to prevent the spread of the malware. Information collected is usually used for classification or clustering of malware behavior [23].

Modern AV programs use all of the above methods to provide a more complete solution to the customer for a higher and more accurate detection of malware. AV vendors do not only detect malware but also label them according to patterns seen when reverse engineering the executable. We will now discuss the problem that we will be addressing in our project.

3.3. Problem Formulation

As discussed in the previous sections, many antivirus companies use signature based analysis to identify malware while some adopt behavioural analysis using a sandbox based environment. There are also researchers who use machine learning to automate the process of behavioural analysis and use antivirus labels to facilitate these sorts of models. In this project we'll be showing the inconsistencies associated with antivirus labels by performing and analyzing various experiments and will compare different machine learning algorithms used in static analysis and than we will further propose a behaviour based malware detection model automated using machine learning without using antivirus labels.

4. Data set generation

For static analysis , dataset is taken from kaggle , which is standard dataset from VirusShare and VirusTotal and than dataset was preprocessed to get better efficiency .

For dynamic analysis , To start with malware analysis, we'll need a dataset consisting of various malware samples and features associated with those samples. This section deals with generation of dataset which will consist of behavior based features and we will further use this dataset to perform our analysis.

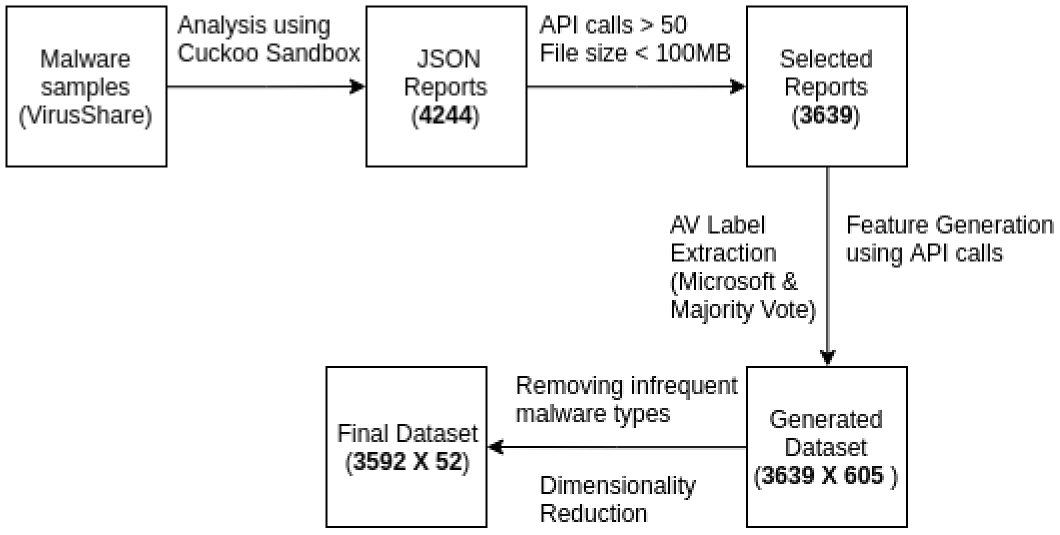

Malware samples (VirusShare)
Analysis using Cuckoo Sandbox
JSON Reports (4244)
API calls > 50
File size < 100MB
Selected Reports (3639)
AV Label Extraction (Microsoft & Majority Vote)
Feature Generation using API calls
Generated Dataset (3639 X 605)
Removing infrequent malware types
Dimensionality Reduction
Final Dataset (3592 X 52)

Figure 1: The workflow of the Data Generation module

4.1. Analyzing malware samples

For performing any machine learning task, it’s important to have a dataset. For this project we made our own dataset using malware samples from VirusShare. The samples primarily consisted of executables, html documents, javascript files and binaries. VirusShare is a repository of malware samples to provide security researchers, incident responders, forensic analysts, and the morbidly curious access to samples of live malicious code. Note that we needed a dataset consisting of behavioral features of various malwares, hence, acquiring only malware samples wouldn't have sufficed.

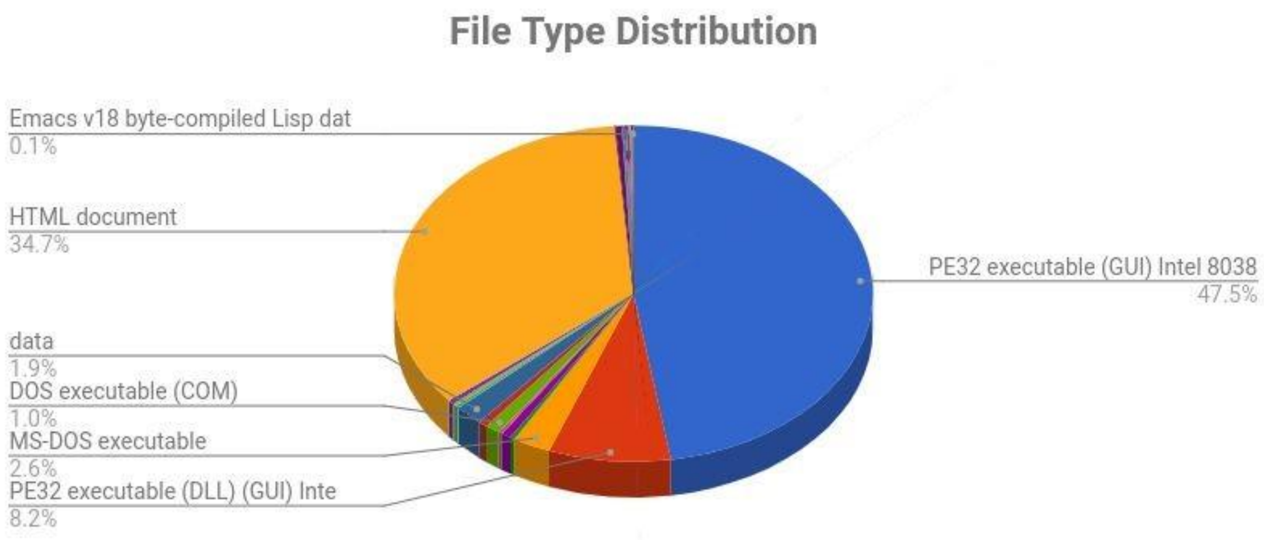


Figure 2: Pie chart representing file types found in the dataset

For analyzing the malware sample, we used an open source automated malware analysis sandbox called Cuckoo. By definition, a sandbox is often used to execute untested or untrusted programs or code, possibly from unverified or untrusted third parties, suppliers, users or websites, without risking harm to the host machine or operating system. It is most frequently used for testing. In this sense, a sandbox is really nothing more than a test or staging server. The goal is to give enough access to allow proper testing while not exposing critical systems to potentially flawed code. In practice, this can take the form of a separate server or mirrored production environment. Cuckoo sandbox can automatically run and analyze files and collect comprehensive analysis results that outline what the malware does while running inside an isolated operating system. It can retrieve the following type of results:

- Traces of calls performed by all processes spawned by the malware.
- Files being created, deleted and downloaded by the malware during its execution.

- Memory dumps of the malware processes.
- Network traffic trace in PCAP format.
- Screenshots taken during the execution of the malware.
- Full memory dumps of the machines.

I had submitted around 8350 samples to cuckoo but due to various hindrances such as system crashes, only 4513 malware analysis reports were generated. The analysis was run on HP laptop .

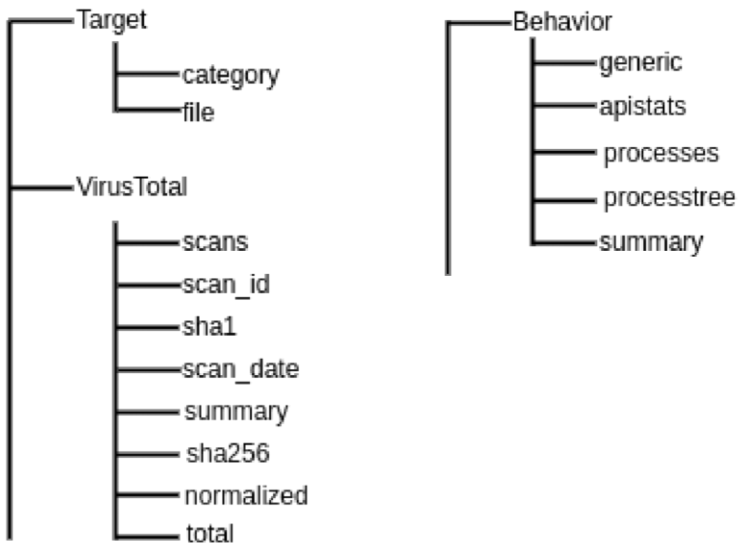


Figure 3: Format of the generated JSON report

5. Algorithms Used

In this section we will discuss the various algorithms that we have used in this in project. We will The whole d-dimensional data-set is taken. If classes exist, they are ignored.

5.1 <u>K-NN</u> :

The KNN algorithm assumes that similar things exist in close proximity. In other words, similar things are near to each other.

The KNN Algorithm :

1. Load the data

2. Initialize K to your chosen number of neighbors

3. For each example in the data

3.1 Calculate the distance between the query example and the current example from the data.

3.2 Add the distance and the index of the example to an ordered collection

4. Sort the ordered collection of distances and indices from smallest to largest (in ascending order) by the distances .

5. Pick the first K entries from the sorted collection

6. Get the labels of the selected K entries

7. If regression, return the mean of the K labels

8. If classification, return the mode of the K labels

Choosing the right value for K

To select the K that's right for your data, we run the KNN algorithm several times with different values of K and choose the K that reduces the number of errors we encounter while maintaining the algorithm's ability to accurately make predictions when it's given data it hasn't seen before.

Here are some things to keep in mind:

1. As we decrease the value of K to 1, our predictions become less stable. Just think for a minute, imagine K=1 and we have a query point surrounded by several reds and one green (I'm thinking about the top left corner of the colored plot above), but the green is the single nearest neighbor. Reasonably, we would think the query point is most likely red, but because K=1, KNN incorrectly predicts that the query point is green.

2. Inversely, as we increase the value of K, our predictions become more stable due to majority voting / averaging, and thus, more likely to make more accurate predictions (up to a certain point). Eventually, we begin to witness an increasing number of errors. It is at this point we know we have pushed the value of K too far.

3. In cases where we are taking a majority vote (e.g. picking the mode in a classification problem) among labels, we usually make K an odd number to have a tiebreaker.

Advantages :

1. The algorithm is simple and easy to implement.
2. There’s no need to build a model, tune several parameters, or make additional assumptions.
3. The algorithm is versatile. It can be used for classification, regression, and search (as we will see in the next section).

Disadvantages :

1. The algorithm gets significantly slower as the number of examples and/or predictors/independent variables increase.

5.2 Random Forest Algorithm :

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of **ensemble learning,** which is a process of *combining multiple classifiers to solve a complex problem and to improve the performance of the model.*

As the name suggests, ***"Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset."*** Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

Working :

Random Forest works in two-phase first is to create the random forest by combining N decision tree, and second is to make predictions for each tree created in the first phase.

The Working process can be explained in the below steps and diagram:

Step-1: Select random K data points from the training set.

Step-2: Build the decision trees associated with the selected data points (Subsets).

Step-3: Choose the number N for decision trees that you want to build.

Step-4: Repeat Step 1 & 2.

Step-5: For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes.

Advantages :

- Random Forest is capable of performing both Classification and Regression tasks.
- It is capable of handling large datasets with high dimensionality.
- It enhances the accuracy of the model and prevents the overfitting issue.

Disadvantages :

- Although random forest can be used for both classification and regression tasks, it is not more suitable for Regression tasks.

5.3 **Obfuscation Techniques** :

Encryption:

- In encryption, malware uses encryption to hide malicious code block in its entire code . Hence, malware becomes invisible in the host.

Oligomorphic:

In oligomorphic method, a different key is used when encrypting and decrypting malware payload. Thus, it is more difficult to detect malware which uses oligomorphic method than encryption.

Polymorphic:

In polymorphic method, malware uses a different key to encrypt and decrypt likewise the key used in oligomorphic method. However, the encrypted payload portion contains several copies of the decoder and can be encrypted in layered . Thus, it is more difficult to detect polymorphic malware when compared to oligomorphic malware.

Metamorphic:

Metamorphic method does not use encryption. Instead, it uses dynamic code hiding which the opcode changes on each iteration when the malicious process is executed . It is very difficult to detect such malware because each new copy has a completely different signature.

Stealth:

Stealth method also called code protection, implements a number of counter techniques to prevent it from being analyzed correctly .For instance, it can make changes on the system and keep it hidden from detection systems.

6. MALWARE DETECTION TECHNIQUES AND ALGORITHMS :

- In recent years, data mining and ML algorithms have been used extensively for malware detection.
- Malware detection is the process of investigating the content of the program and deciding whether the analyzed program malware or benign.

- **The malware detection process includes 3 stages**:
- Malware analysis
- feature extraction and
- classification.

6.1 SIGNATURE-BASED MALWARE DETECTION :

Signature is a malware feature which encapsulates the program structure and identifies each malware uniquely. Signature- based detection approach is widely used within commercial antivirus. This approach is fast and efficient to detect known malware, but insufficient to detect unknown malware.

In addition, malware belonging to the same family can easily escape the signature-based detection by using obfuscation techniques. General view of signature-based detection schema can be seen in Figure.

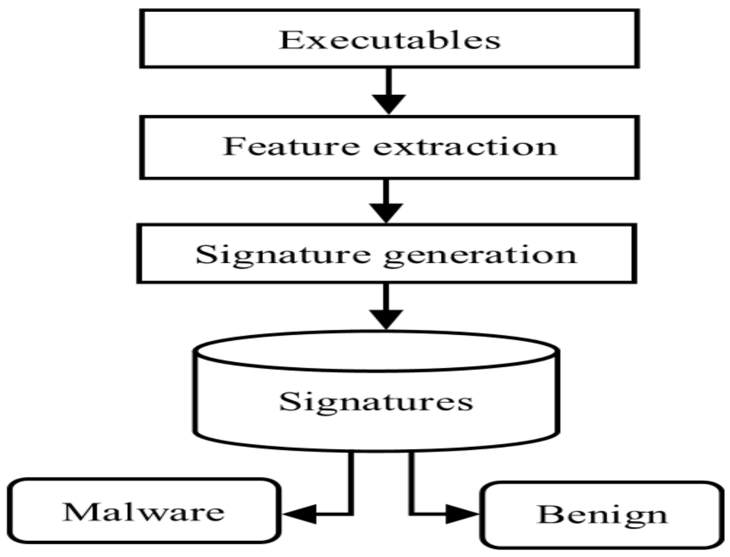


FIGURE 4. Signature-based malware detection schema

During the signature generation, first features are extracted from executables (Figure 1). Then, signature generation engine generates a signatures and stores them into signature database.

When sample program needs to be marked as malware or benign, signature of the related sample is extracted as the same way before and compared with signatures on the database. Based on the comparison, sample program is marked as malware or benign.

. There are many different techniques to create a signature such as : -

string scanning, top-and tail scanning, entry point scanning, and integrity checking.

6.2 BEHAVIOR-BASED MALWARE DETECTION :

- Behavior-based malware detection approach observes the program behaviors with monitoring tools and determines whether the program is malware or benign. Although the program codes are being changed, the behavior of the program will be similar; thus, majority of new malware can be detected with this method .
- On the other hand, some malware binaries do not run properly under protected environment (virtual machine, sandbox environment). Hence , malware samples are may be incorrectly marked as benign.

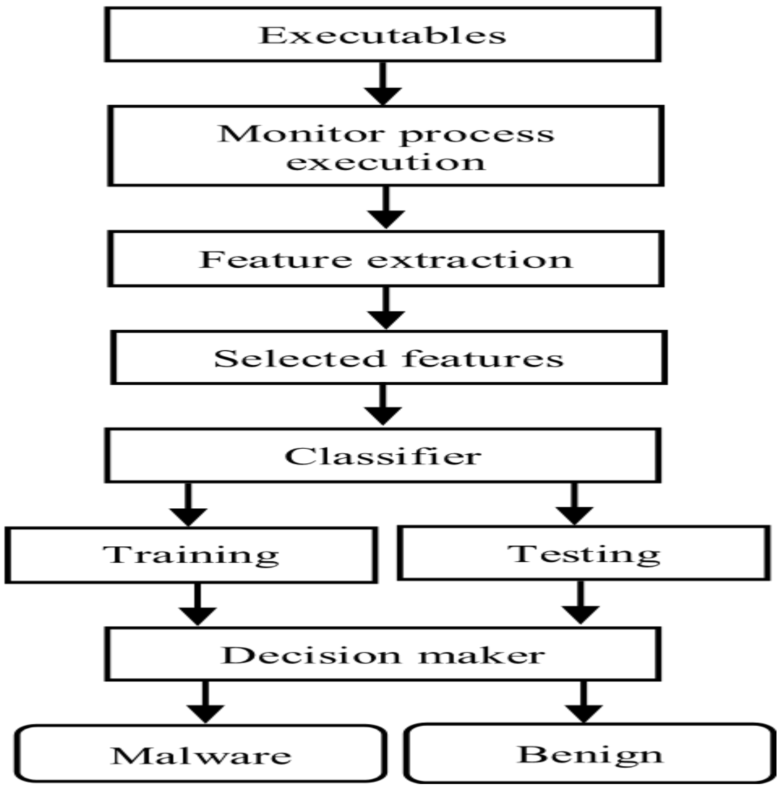


FIGURE 5. Behavior-based malware detection schema

When establishing a behavior-based detection system, behaviors are obtained by using one the following procedure:

Automatic analysis by using sandbox ;

Monitoring of system calls ;

Monitoring of le changes ;

Comparison of registry snapshots ;

Monitoring network activities ;

Process monitoring .

In behavior-based detection, first, behaviors are determined by using one of the technique used above and the dataset is created by subtracting the features using datamining.

Then, specific features from the dataset are obtained and classification done by using ML algorithms. General view of behavior-based schema can be seen in Figure 2.

- Detection schema based on behaviors consists of 3 steps:

Determine behaviors (datamining can be used),

Extract features from behaviors (datamining is used),

Apply classification (machine learning is used).

- Data mining techniques such as *n*-gram, *n*-tuple, bag, graph model, etc. have been used to determine the features from behaviours; Hellinger distance, cosine coefficient, chi-square, etc. (probability and statistical method) distance algorithms are used to specify similarities among features.
- The difficulties in defining a behavior, the large number of extracted

features (when using *n*-grams, etc.), and the difficulties in identifying the similarities and differences among the extracted properties have prevented the creation of an effective detection system.

- Besides, some malware does not run properly within the virtual machines/sandboxes, and advanced code obfuscating techniques prevent malware from being analyzed correctly.

6.3 HEURISTIC-BASED MALWARE DETECTION :

- In recent years, heuristic based detection approach has been used frequently . It is a complex detection method which uses experiences and different techniques such as rules and ML techniques . Although it has a high accuracy rate to detect zero-day malware to a certain degree, it cannot detect complicated malware. Heuristic-based detection schema can be seen in Figure 3.
- Heuristic-based schema can use both strings and some behaviors to generate rules, and based on that rules it generates signature. It uses API calls, CFG, *n*-grams, Opcode, and hybrid features when generates a signature . Although the heuristic-based detection can detect various forms of known and unknown malware, it is insufficient to detect all new generation of malware. In addition, heuristic-based approaches are prone to high *FPR*.

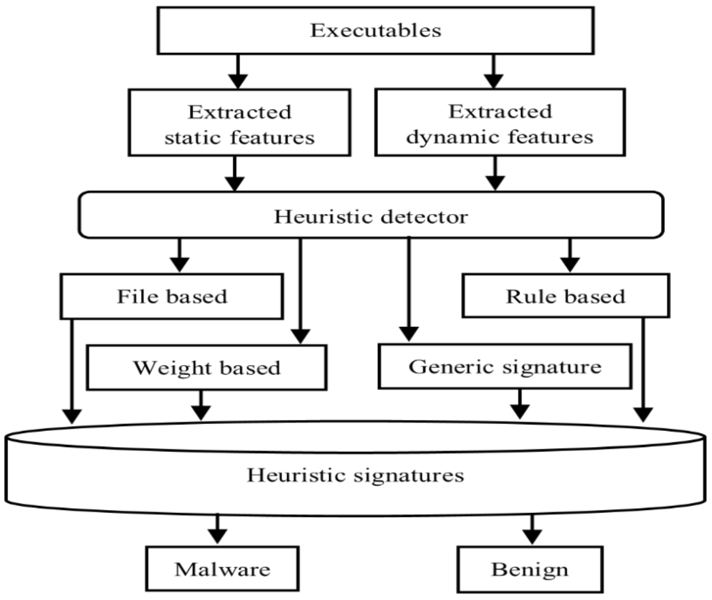


FIGURE 6. Heuristic-based malware detection schema

7. Methodology :

For static analysis , we first preprocessed our dataset , and checked for :

- **Getting the dataset**
- **Importing libraries**
- **Importing datasets**
- **Finding Missing Data**
- **Encoding Categorical Data**
- **Splitting dataset into training and test set**
- **Feature scaling**

Than we trained our dataset with 3 different machine learning algorithms i.e. random forest , k-nn , and logistic regression and checked the accuracy using F-score and confusion matrix .

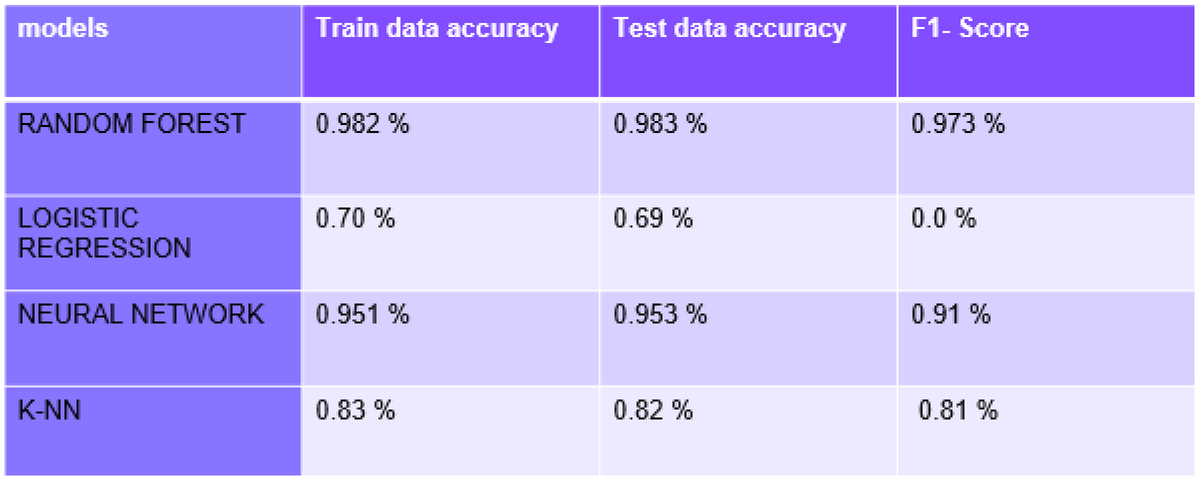
According to the maximum efficiency , we came to a conclusion that , random forest algorithm works better for malware detection .

Further in the upcoming session , I will create (since no standard dataset is available for behavioural analysis) my own data set using cuckoo sandbox and will compare and analyse the behavioural based malware detection technique with static and heuristic approach .

So that a high efficient malware detection system can be designed using automated machine learning approach .

8. Results

<u>RESULTS FOR STATIC ANALYSIS</u> :

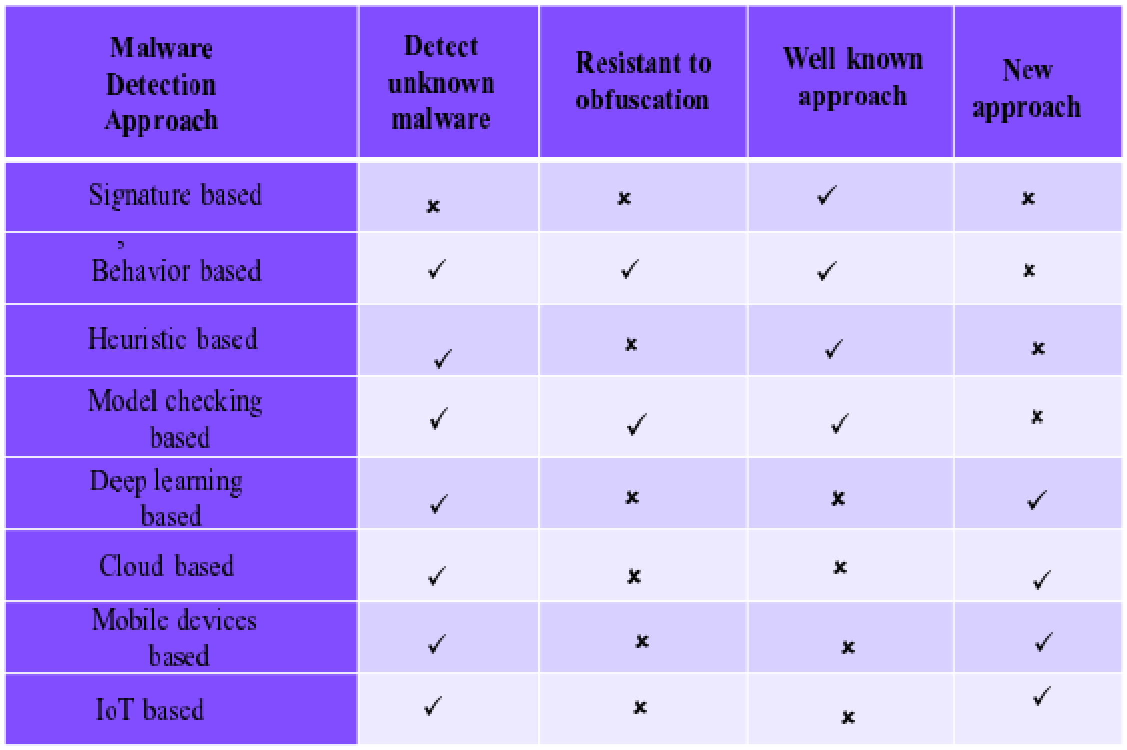


models	Train data accuracy	Test data accuracy	F1- Score
RANDOM FOREST	0.982 %	0.983 %	0.973 %
LOGISTIC REGRESSION	0.70 %	0.69 %	0.0 %
NEURAL NETWORK	0.951 %	0.953 %	0.91 %
K-NN	0.83 %	0.82 %	0.81 %

From the results, it is evident that the AV vendors do not give much importance to the behavioral aspects of malware as both Microsoft and Majority Vote give poor results. The better performance via Majority Vote labels is due to the pooling of labels leading to generalization. However, the results via the clustering labels show that still much more can be achieved if a more behavioral suited approach is taken up.

- For static analysis , all algorithms used in this project gave more than 90% accuracy .
- Out of all algorithms ,Random forest gave the best accuracy .
- Although malware detectors are being improved every day, the following research challenges still remain an open issue in malware detection approaches:
- New generation malware uses some obfuscation and packing techniques to hide itself. By using these techniques malware can prevent itself from being correctly analyzed and avoid detection. Signature-based detection approach is not resistant to malware obfuscation. Even if behavior-, and model checking-based approaches are effective to most of obfuscation techniques, they cannot be resistant to all obfuscation techniques.
- Real-time monitoring and detection are a challenging tasks. Most of the studies have been done so far to detect malware by using dataset and are not appropriate for real-time monitoring.

9. Conclusion :



Malware Detection Approach	Detect unknown malware	Resistant to obfuscation	Well known approach	New approach
Signature based	×	×	✓	×
Behavior based	✓	✓	✓	×
Heuristic based	✓	×	✓	×
Model checking based	✓	✓	✓	×
Deep learning based	✓	×	×	✓
Cloud based	✓	×	×	✓
Mobile devices based	✓	×	×	✓
IoT based	✓	×	×	✓

TABLE 2. Comparison of malware detection approaches.

- For static analysis , all algorithms used in this project gave more than 90% accuracy .
- Out of all algorithms ,Random forest gave the best accuracy .

Above table represents different approaches , and shows the results of different algorithms

- Most of the malware detection approaches are prone to *FPs* and *FNs*. Some features and signatures can be very close in malware and benign samples which raises *FPs* and *FNs.*
- No detection method can affectively detect all unknown malware.
- Generally, learning algorithms are prone to bias, and overfitting. This leads to decreases *DRs* and increases *FPs*.
- There is no well-known and accepted dataset which can be used to evaluate the malware detection approaches performance. This is because each malware detection method uses different malware and dataset.

10. Future Work

The above solution we proposed may work at a basic level but cannot be incorporated in this ever-evolving industry due to numerous factors:

- my future work involves performing behaviour based malware detection and analyzing/comparing it with static and heuristic approaches .
- For performing behavior based malware detection , there is no standard dataset available , so to perform and compare I will create my own dataset , and comparision will be made between all approaches so that we can develop a better malware detection system .
- Comparative study may help to develop an automated machine learning based malware detection system which is more efficient than static approach and which can tackle the problem of obfuscation technique .

1. Researchers till now have incorporated API calls as behaviour based features and not features related to system profile modifications.
2. We also need to formulate an algorithm for efficient selection of behaviour based features.
3. Behaviour based analysis is not widely accepted because researchers believe that some malware detect the presence of a virtual execution environment and hence, do not reveal their behavioural profile and evade detection.
4. Recent research has also shown that aging classifiers i.e infrequently trained classifiers are more prone to zero day attacks.

Keeping this in mind we plan to work on a holistic model that will incorporate the following components:

1. While extracting features to build the behavioural profile corresponding to a sample, we'll include system specific features in addition to features corresponding to API calls.
2. We plan to use autoencoder, an unsupervised deep learning algorithm for feature selection.
3. Use of bare metal environment can prevent malware from evading detection. Bare metal is a computer system or network in which a virtual machine (or Sandbox) is installed directly on hardware rather than within the host operating system (OS). The term "bare metal" refers to a hard disk, the usual medium on which a computer's OS is installed.
4. Develop a model to identify optimal training cycles that a classifier should undergo to prevent aging.

References :

1. Shanklin, S. D., Bernhard, T. E., & Lathem, G. S. (2002). U.S. Patent No. 6,487,666. Washington, DC: U.S. Patent and Trademark Office.
2. Schmidt, A. D., Bye, R., Schmidt, H. G., Clausen, J., Kiraz, O., Yuksel, K. A., ... & Albayrak, S. (2009, June). Static analysis of executables for collaborative malware detection on android. In Communications, 2009. ICC'09. IEEE International Conference on (pp. 1-5). IEEE.
3. Chua, Z., Shen, S., Saxena, P. (2017). Neural Nets Can Learn Function Type Signatures From Binaries.
4. Moser, A., Kruegel, C., & Kirda, E. (2007, December). Limits of static analysis for malware detection. In Computer security applications conference, 2007. ACSAC 2007. Twenty-third annual (pp. 421-430). IEEE.
5. Rieck, K., Trinius, P., Willems, C., & Holz, T. (2011). Automatic analysis of malware behavior using machine learning. Journal of Computer Security, 19(4), 639-668.
6. Tian, R., Islam, R., Batten, L., & Versteeg, S. (2010, October). Differentiating malware from cleanware using behavioural analysis. In Malicious and Unwanted Software (MALWARE), 2010 5th International Conference on (pp. 23-30). IEEE.
7. Chen, S., Xue, M., Tang, Z., Xu, L., & Zhu, H. (2016, May). Stormdroid: A streaminglized machine learning-based system for detecting android malware. In Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security (pp. 377-388). ACM.
8. Bayer, U., Kirda, E., & Kruegel, C. (2010, March). Improving the efficiency of dynamic malware analysis. In Proceedings of the 2010 ACM Symposium on Applied Computing (pp. 1871-1878). ACM.
9. Wang, J., Xue, Y., Liu, Y., & Tan, T. H. (2015, April). JSDC: A hybrid approach for JavaScript malware detection and classification. In Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security (pp. 109-120). ACM.
10. Graziano, M., Canali, D., Bilge, L., Lanzi, A., & Balzarotti, D. (2015). Needles in a haystack: Mining information from public dynamic analysis sandboxes for malware intelligence. In USENIX Security Symposium (pp. 1057-1072). USENIX Association.
11. Jordaney, R., Sharad, K., Dash, S. K., Wang, Z., Papini, D., Nouretdinov, I., & Cavallaro, L. (2017). Transcend: Detecting Concept Drift in Malware Classification Models.
12. Miramirkhani, N., Appini, M. P., Nikiforakis, N., & Polychronakis, M. (2017, May). Spotless Sandboxes: Evading Malware Analysis Systems using Wear-and-Tear Artifacts.

In Security and Privacy (SP), 2017 IEEE Symposium on (pp. 1009-1024). IEEE.

13. Xu, W., Qi, Y., & Evans, D. (2016). Automatically evading classifiers. In Proceedings of the 2016 Network and Distributed Systems Symposium.
14. Kirat, D., & Vigna, G. (2015, October). MalGene: Automatic extraction of malware analysis evasion signature. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (pp. 769-780). ACM.
15. Smutz, C., & Stavrou, A. (2016). When a Tree Falls: Using Diversity in Ensemble Classifiers to Identify Evasion in Malware Detectors. In NDSS.
16. Xu, M., & Kim, T. (2017). PlatPal: Detecting Malicious Documents with Platform Diversity.
17. Xu, Z., Nappa, A., Baykov, R., Yang, G., Caballero, J., & Gu, G. (2014, November). Autoprobe: Towards automatic active malicious server probing using dynamic binary analysis. In Proceedings of the 2014 ACM SIGSAC Conference (pp. 179-190).
18. Bartos, K., Sofka, M., & Franc, V. (2016). Optimized Invariant Representation of Network Traffic for Detecting Unseen Malware Variants.
19. Fossi, M., Turner, D., Johnson, E., Mack, T., Adams, T., Blackbird, J., ... & Wueest, C. (2009). Symantec global internet security threat report. White Paper, Symantec Enterprise Security, 1.
20. Forecast and Methodology, 2014–2019 White Paper, Cisco, 2015
21. Griffin, K., Schneider, S., Hu, X., & Chiueh, T. C. (2009, September). Automatic Generation of String Signatures for Malware Detection. In RAID (Vol. 5758, pp. 101-120).
22. Kephart, J. O. (1994). Automatic extraction of computer virus signatures. In Proc. 4th Virus Bulletin International Conference, Abingdon, England, 1994 (pp. 178-184).
23. Bayer, U., Comparetti, P. M., Hlauschek, C., Kruegel, C., & Kirda, E. (2009, February). Scalable, Behavior-Based Malware Clustering. In NDSS (Vol. 9, pp. 8-11).
24. Tibshirani, R., Walther, G., & Hastie, T. (2001). Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *63*(2), 411-423.
25. Hartigan, J. A., & Wong, M. A. (1979). Algorithm AS 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, *28*(1), 100-108.
26. Arthur, D., & Vassilvitskii, S. (2007, January). k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms* (pp. 1027-1035). Society for Industrial and Applied Mathematics.