ACKNOWLEDGEMENT

We would like to profoundly thank Director **Mr. C. M. Faiz** and Management of HKBK College of Engineering for providing such a healthy environment for the successful completion of project work.

We would like to express our thanks to the Principal **Dr. Tabassum Ara**, for her encouragement that motivated us for the successful completion of Project Work.

We wish to express our gratitude to **Dr. Ashok Kumar P. S.**, Professor and Head of Department of Computer Science & Engineering for providing such a healthy environment for the successful completion of Project work.

We are also grateful to **Dr. Deepak N. R.,** Professor, Dept of CSE and **Prof. Simran Pal R** Asst. Prof., Dept of CSE, HKBKCE. I am extremely thankful and indebted to him for sharing expertise, and sincere and valuable guidance and encouragement extended to me.

We express my heartfelt appreciation and gratitude to my Project Guide, **Prof. Tahir Naquash**, Professor of Computer Science and Engineering, HKBK College of Engineering, Bangalore, for his intellectually-motivating support and invaluable encouragement during my Project work. His comprehensive knowledge and understanding of the research topic as well as his uncompromising and sensible attitude towards research and insistence on quality work have profoundly influenced me and will definitely benefit my future work-my heartfelt thanks to his painstaking modification of this report.

We would also like to thank all other teaching and technical staffs of Department of Computer Science and Engineering, who have directly or indirectly helped us in the completion of this Project Work. And lastly, We would hereby acknowledge and thank our parents who have been a source of inspiration and also instrumental in the successful completion of this project.

ABSTRACT

Current antivirus software's are effective against known viruses, if a malware with new signature is Introduced then it will be difficult to detect that it is malicious. Signature-based detection is not that effective during zero-day attacks. Till the signature is created for new (unseen) malware, distributed to the systems and added to the anti-malware database, the systems can be exploited by that malware. Research shows that over the last decade, malware has been growing exponentially, causing substantial financial losses to various organizations. Different anti-malware companies have been proposing solutions to defend attacks from these malwares. The velocity, volume, and the complexity of malware are posing new challenges to the anti-malware community. Current state-of-the-art research shows that recently, researchers and anti-virus organizations started applying machine learning and deep learning methods for malware analysis and detection. Machine learning methods can be used to create more effective anti-malware software which is capable of detecting previously unknown malware, zero-day attack etc. We propose an approach that Various machine learning methods such as Support Vector Machine (SVM), Decision tree, Random Forest and XG Boost will be used.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER -1
INTRODUCTION

CHAPTER 1

INTRODUCTION

Idealistic hackers attacked computers in the early days because they were eager to prove themselves. Cracking machines, however, is an industry in today's world. Despite recent improvements in software and computer hardware security, both in frequency and sophistication, attacks on computer systems have increased. Regrettably, there are major drawbacks to current methods for detecting and analysing unknown code samples. The Internet is a critical part of our everyday lives today. On the internet, there are many services and they are rising daily as well. Numerous reports indicate that malware's effect is worsening at an alarming pace. Although malware diversity is growing, anti- virus scanners are unable to fulfil security needs, resulting in attacks on millions of hosts. Around 65,63,145 different hosts were targeted, according to Kaspersky Labs, and in 2015, 40,00,000 unique malware artefacts were found. Juniper Research (2016), in particular, projected that by 2019 the cost of data breaches will rise to $2.1 trillion globally. Current studies show that script-kiddies are generating more and more attacks or are automated. To date, attacks on commercial and government organizations, such as ransomware and malware, continue to pose a significant threat and challenge. Such attacks can come in various ways and sizes. An enormous challenge is the ability of the global security community to develop and provide expertise in cybersecurity. There is widespread awareness of the global scarcity of cybersecurity and talent. Cybercrimes, such as financial fraud, child exploitation online and payment fraud, are so common that they demand international 24-hour response and collaboration between multinational law enforcement agencies. For single users and organizations, malware defense of computer systems is therefore one of the most critical cybersecurity activities, as even a single attack may result in compromised data and sufficient losses.

Mobile phones have become increasingly important tools in people’s daily life, such as mobile payment, instant messaging, online shopping, etc., but the security problem of mobile phones is becoming more and more serious. Due to the open source nature of the Android platform, it is very easy and profitable to write malware using the vulnerabilities and security defects of the Android system. This is the main reason for the rapid increase in the number of malware

on the Android system. The malicious behaviors of Android malware generally include sending deduction SMS, consuming traffic, stealing user's private information, downloading a large number of malicious applications, remote control, etc., threatening the privacy and property security of mobile phones users.

The number of Android malware is growing rapidly; particularly, more and more malicious software use obfuscation technology. Traditional detection methods of manual analysis and signature matching have exposed some problems, such as slow detection speed and low accuracy. In recent years, many researchers have solved the problems of Android malware detection using machine learning algorithms and had a lot of research results. With the rise of deep learning and the improvement of computer computing power, more and more researchers began to use deep learning models to detect Android malware. This paper proposes an Android malware detection model based on a hybrid deep learning model with deep belief network (DBN) and gate recurrent unit (GRU). The main contributions are as follows:

(i) In order to resist Android malware obfuscation technology, in addition to extracting static features, we also extracted the dynamic features of malware at runtime and constructed a comprehensive feature set to enhance the detection capability of malware.

(ii) A hybrid deep learning model was proposed. According to the characteristics of static features and dynamic features, two different deep learning algorithms of DBN and GRU are used.

(iii) The detection model was verified, and the detection result is better than traditional machine learning algorithms; it also can effectively detect malware samples using obfuscation technology.

1.1 EVOLUTION OF MALWARE

In order to protect networks and computer systems from attacks, the diversity, sophistication and availability of malicious software present enormous challenges. Malware is continually changing and challenges security researchers and scientists to strengthen their cyber defences to keep pace. Owing to the use of polymorphic and metamorphic methods used to avoid detection and conceal its true intent, the prevalence of malware has increased. To mutate the code while keeping the original functionality intact, polymorphic malware uses a polymorphic

engine. The two most common ways to conceal code are packaging and encryption . Through one or more layers of compression, packers cover a program's real code. Then the unpacking routines restore the original code and execute it in memory at runtime. To make it harder for researchers to analyze the software, crypters encrypt and manipulate malware or part of its code. A crypter includes a stub that is used for malicious code encryption and decryption. Whenever it's propagated, metamorphic malware rewrites the code to an equivalent. Multiple transformation techniques, including but not limited to, register renaming, code permutation, code expansion, code shrinking and insertion of garbage code, can be used by malware authors.

The combination of the above techniques resulted in increasingly increasing quantities of malware, making time-consuming, expensive and more complicated forensic investigations of malware cases. There are some issues with conventional antivirus solutions that rely on signature-based and heuristic/behavioural methods. A signature is a unique feature or collection of features that like a fingerprint, uniquely differentiates an executable. Signature-based approaches are unable to identify unknown types of malware, however. Security researchers suggested behaviour-based detection to overcome these problems, which analyses the features and behaviour of the file to decide whether it is indeed malware, although it may take some time to search and evaluate. Researchers have begun implementing machine learning to supplement their solutions in order to solve the previous drawbacks of conventional antivirus engines and keep pace with new attacks and variants, as machine learning is well suited for processing large quantities of data.

1.2 MALWARE DETECTION

In such a way, hackers present malware aimed at persuading people to install it. As it seems legal, users also do not know what the programme is. Usually, we install it thinking that it is secure, but on the contrary, it's a major threat. That's how the malware gets into your system. When on the screen, it disperses and hides in numerous files, making it very difficult to identify. In order to access and record personal or useful information, it may connect directly to the operating system and start encrypting it.

Detection of malware is defined as the search process for malware files and directories. There are several tools and methods available to detect malware that make it efficient and reliable. Some of the general strategies for malware detection are:

- Signature-based
- Heuristic Analysis
- Anti-malware Software
- Sandbox

  Several classifiers have been implemented, such as linear classifiers (logistic regression, naïve Bayes classifier), support for vector machinery, neural networks, random forests, etc.

  Through both static and dynamic analysis, malware can be identified by:
- Without Executing the code
- Behavioral Analysis

1.3 NEED FOR MACHINE LEARNING IN MALWARE DETECTION

Machine learning has created a drastic change in many industries, including cybersecurity, over the last decade. Among cybersecurity experts, there is a general belief that AI-powered anti-malware tools can help detect modern malware attacks and boost scanning engines. Proof of this belief is the number of studies on malware detection strategies that exploit machine learning reported in the last few years. The number of research papers released in 2018 is 7720, a 95 percent rise over 2015 and a 476 percent increase over 2010, according to Google Scholar,1. This rise in the number of studies is the product of several factors, including but not limited to the increase in publicly labelled malware feeds, the increase in computing capacity at the same time as its price decrease, and the evolution of the field of machine learning, which has achieved ground-breaking success in a wide range of tasks such as computer vision and speech recognition. Depending on the type of analysis, conventional machine learning methods can be categorized into two main categories, static and dynamic approaches. The primary difference between them is that static methods extract features from the static malware analysis, while dynamic methods extract features from the dynamic analysis. A third category may be considered, known as hybrid approaches. Hybrid methods incorporate elements of both static

and dynamic analysis. In addition, learning features from raw inputs in diverse fields have outshone neural networks. The performance of neural networks in the malware domain is mirrored by recent developments in machine learning for cybersecurity.

1.4 SUMMARY

This section describes what is actually considered as malware, the need for malware detection and its evolution. The use of Machine Learning Algorithms to easily categorize malware using the static methods, dynamic methods, hybrid methods. Stages for malware detection: Signature based. The need to implement machine learning algorithms.

CHAPTER -2
LITERATURE SURVEY

CHAPTER 2

LITERATURE SURVEY

2.1 Literature Papers Surveyed

[1] Zero-Day malware is of major concern to the analysts and reverse engineers for the evolving threat and unrestrained expansion mainly due to emerging cloud-based systems [47–50]. In this paper, we introduced ZeVigilante, a framework to detect Zero-Day malware with adoption of ML and sandboxing techniques. The whole deploying and validating process is conducted within the Cuckoo sandbox (CS) environment for a safer progression. Afterward, ZeVigilante is to be considered an integrated system that analyzes executables files considering both static and dynamic analyses to generate reports for users with decisions and results. The main findings and contributions of this study are as follows:

1. Providing a detailed literature review that presents several concepts including malware families, features used to classify malware files, datasets that hold benign and malware files, malware analysis techniques, the algorithms, validation and preprocessing, and sandboxing techniques.
2. Integrating different ML algorithms (including RF, NN, DT, K-NN, NB, and SVM) with CS by using Python code.
3. Extracting the required features including both PE imports and API call sequence from the JSON report and converting the extracted features into a csv file.

(4) implementing the proposed framework with interfaces that allow users to conduct experiments and get the validation results of the executable file (malware or a benign). Finally, we depicted the performance of the ML algorithms along with a detailed comparison, where ZeVigilante demonstrated a high accuracy by outperforming most of state-of-the-art approaches. As a future work, for ZeVigilante, dependency on CS imposes limitations on dynamic analysis integration.

[3] Cyber attacks are becoming increasingly complex with obfuscation of network traffic and system level interactions in recent APTs, and polymorphic malware changing the measurable behaviour of the executable. We aimed to use machine data to classify malware using data that is presently more difficult to obfuscate – system level behaviour such as CPU, RAM, process data and network bytes/packet count. We built machine classifiers using state of the art methods

from the extant literature as a baseline, with data that is continuous and represents behaviour on the system without depending on precise system level operations.

We then implemented a two-map SOFM with the aim of enabling 'fuzzy boundaries' to be captured around system behaviour and the hypothesis that this method would improve classification accuracy over the state of the art methods. The ’fuzzy boundaries’ enable us to map new samples onto existing maps and determine that the behaviour may be different but similar enough to previously observed behaviour to label it as malicious - that is, to better generalize between samples over time. Essentially what we have developed are representations of behavioural ‘DNA’ in a malicious or trusted context that provide a model to which potentially malicious executables can be compared for similarity - introducing a new way of thinking about malicious behaviour modeling. We found a 3.45% increase in classification performance using an unseen testing dataset.

We also investigated the impact on classifier performance when using the unseen test dataset versus a k-fold cross validation method, which is frequently used in malware classification research. The rationale for this was to determine whether system behaviour that was not exposed to the classification model during the training phase would evade classification during testing. K-fold cross validation of course leaves a sample out during training, but having a separate dataset of samples allowed us to test beyond the holdout approach. It is effectively a simulation of polymorphic malware or hand-crafted malware behaviour, where the interaction with the system on which it is run leaves a different footprint to previous samples. We found that there was a significant drop in performance when the model was exposed to an unseen dataset with an equal balance between classes.

Finally we combined the SOFM with an ensemble classifier based around a Logistic Regression model and used the Best Matching Unit output from the SOFM – which represents “fuzzy neighbourhoods” of system behaviour – as features, in place of the continuous machine activity data used in the previous experiments. We saw an increase of classifier performance of between 7.24% and 24.68% over the state of the art when using this novel approach to malware classification. The findings suggest that the neural-type model for learning, combined with an ability to provide flexible classification boundaries using the SOFM shows promise as a method for the detection of APTs and polymorphic malware where the activity carried out during the attack may vary between samples.

[5] The ideas of malware, the different types of malware and malware analysis have been discussed in details here. It is inferred from the information collected that dynamic analysis is a better method of malware analysis than static analysis. Although dynamic analysis has the obvious flaw of analyzing only one execution of the malware, static analysis is rather difficult to do properly, for in most cases, the source code is not visible. And even if the source code was available, it can never be ensured that no modifications were done to the binary executables which remained undocumented by the source. Hence, due to many such drawbacks, dynamic analysis is preferred over static malware analysis. On studying malware analysis tools, it can be concluded that the sandbox environment is particularly conducive towards analyzing malware. Recent trends in malware attacks show highly advanced techniques being applied to secure sensitive information. It is with great alarm that experts are noting how malware attacks are being more and more sophisticated in a short period of time, whereas operating systems and other user software are not produced in such brief intervals.

[8] Today the threat of malware has increased many times than before, with the ease of resource availability and technical know how new malwares are emerging every day with intention of not being detected. In order to protect important information security and privacy there a need of a technique that helps in malware analysis on hand and keeps the user safe from further similar attempts by providing necessary security. In this paper we have described the types of malware and after discussing both techniques of malware analysis it can be concluded that upon development of a framework using the above research a successfull technique can be developed that can prevent against malware attacks of many different types. Remote analysis and Local analysis both modules are important in their own respect as they broaden the users ability to mitigate the malware attack themselves. The framework will provide user safety from anti-debugger , packaged /obfuscated malware that tries to conceal its true functionality. Although with the latest development in malware threats such as polymorphic, metamorphic malware etc, this technique does not completely safeguard against those threats. More work is further to be done in this direction to protect the users from such malware. Machine learning is being used to further combat this malware threat by learning the identifying features of malware and deploying them to further identify similar malware before they can cause damage.

[10] In recent years, a few research efforts have been conducted on surveys of data-mining based malware detection methods. The researchers surveyed the feature representation and classification methods for malware detection. Shabtai et al. conducted a comprehensive study on static features using machine learning classifiers for malware detection while Egele surveyed automated dynamic malware analysis techniques and tools. In this article, we not only overview the development of malware and anti-malware industry and present the industrial needs on malware detection, but also provide a comprehensive study on data-mining-based methods for malware detection based on both static and dynamic representations as well as other novel features. Furthermore, we also discuss the additional issues and challenges of malware detection using data mining techniques and finally forecast the trends of malware development.

Due to the exponential growth of malware samples, intelligent methods for efficient and effective malware detection at the cloud (server) side are urgently needed. As a result, much research has been conducted on developing intelligent malware detection systems using data mining and machine learning techniques. In these methods, the process of malware detection is generally divided into two steps: feature extraction and classification/clustering. We provide a comprehensive investigation on both the feature extraction and the classification/clustering steps. We conclude that data-mining-based malware detection framework can be designed to achieve good detection performance with high accuracy while maintaining low false positives. Many developed systems have been successfully integrated into commercial anti-malware products.
The following are some practical insights from our view:

1. Based on different data sets with different feature representation methods, there is no single classifier/clustering algorithm always performing best. In other words, the performance of such malware detection methods critically depend on the extracted features, data distributions, and the categorization methods.
2. Generally, compared with individual classifiers, an ensemble of classifiers can always help improve the detection accuracy. In real applications, a successful malware detection framework should utilize multiple diverse classifiers on various types of feature representations.

3. For feature extraction, both static and dynamic analysis approaches have their own advantages and limitations. In real applications, we suggest using static analysis at first, since over
80% of the file sample collection can be well-represented by static features. If the file cannot be well-represented by static extraction, then we can try dynamic analysis.
Novel features, such as file-to-file relation graphs, can also provide invaluable information about the properties of file samples.
4. In order to achieve the best detection performance in real applications, it is often better to have enough training samples with balanced distributions for both classes (malware and benign files).

[13] This paper explored a new direction to extract the API-based dynamic features by analyzing the API calls together with their list of arguments. With the help of machine learning algorithms, we designed two methods to detect Windows malware samples and classify them into their types. The first method deals with the entire list of arguments of each API call as one feature, whereas the second method deals with each argument of each API call separately as one feature. We verified the performance of the proposed methods in malware detection using reasonable datasets of 7105malicious samples belonging to ten distinct types and 7774 benign samples. We showed that our approach outperforms other recent API arguments-based malware detection approaches in terms of accuracy, limitations, and required API information. Our experimental results showed that our malware detection approach gave accuracy of over 99.8992 %, and outperformed the state-of-the-art. Furthermore, we used the same methods to classify the malware samples into their types. The experimental results showed that our malware classification approach gave an accuracy of over 97.9548 % Our approach has promise for wide adoption in API-based malicious behavior detection for Windows platforms. In particular, it can meet the demands of applications that are currently served by malware detectors that rely on extracting statistical information of the API-based dynamic features but desire better robustness against unfavorable sequence interruption attacks.

2.2 SUMMARY

This section concludes the information about the various survey papers and gives a detailed explanation about these papers. These papers and the detailed explanation give us information about the Detection of Malware using many Algorithms and gives us brief glimpse of the process.

CHAPTER -3
METHODOLOGY

CHAPTER 3
METHODOLOGY

To detect the unknown malware using machine learning technique, a flowchart of our approach shown in below figure. It includes pre-processing of dataset, promising feature selection, training of classifier and detection of advanced malware.

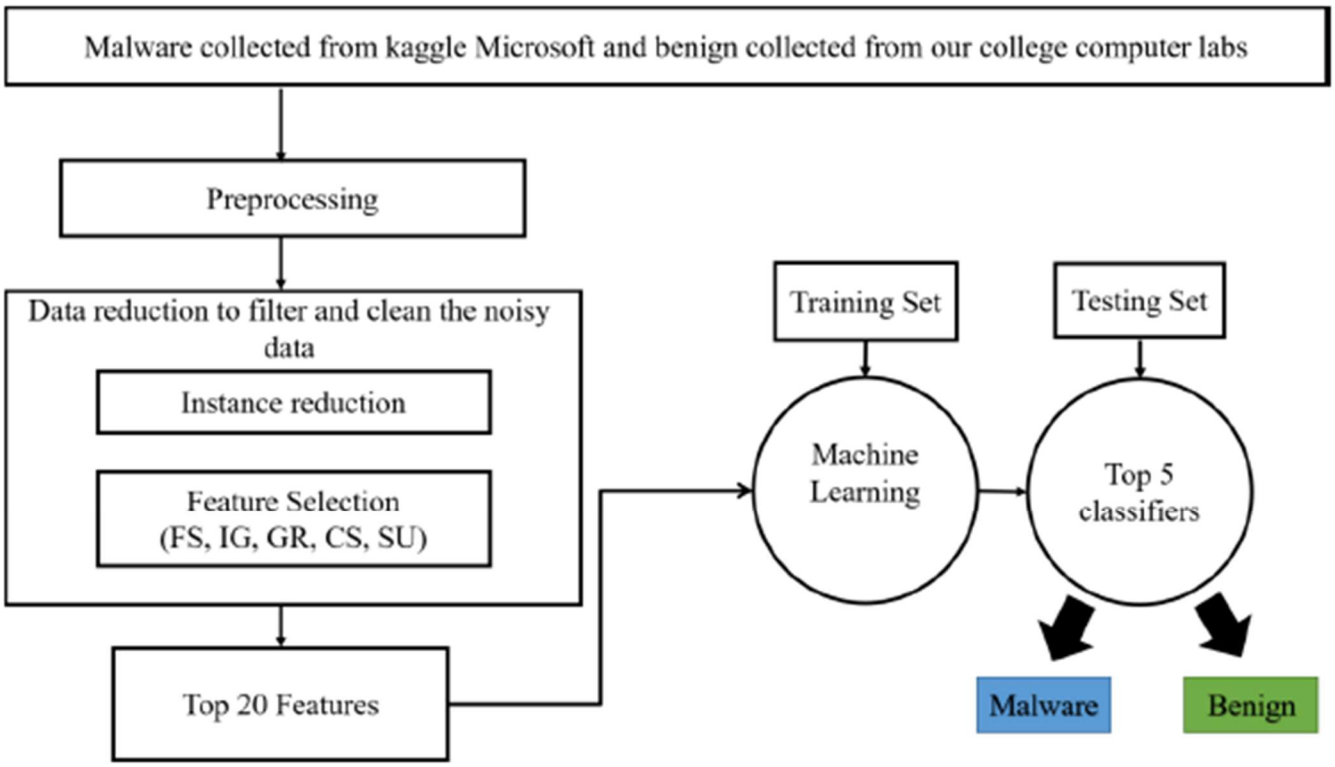

Figure 3 Methodology

3.1 ALGORITMS USED:

3.1.1. RANDOM FOREST

An effective alternative is to use trees with fixed structures and random features . Tree collections are called forests, and classifiers built in so-called random forests. The random water formation algorithm requires three arguments: the data, a desired depth of the decision trees, and a number K of the total decision trees to be built, i. The algorithm generates each of the K trees. independent, which makes it very easy to parallelize. For each tree, build a

complete binary tree. The characteristics used for the branches of this tree are selected randomly, usually with replacement, which means that the same characteristic can occur more than 20 times, even in a single branch. a. the leaves of this tree, where predictions are made, are completed based on training data. The last step is the only point at which the training data is used.

The resulting classifier is just a K-lot vote, and random trees. The most amazing thing about this approach is that it actually works remarkably well. They tend to work best when all the features are at least, well, relevant, because the number of features selected for a particular tree is small. One intuitive reason that it works well is the following. Some trees will query unnecessary features. These trees will essentially make random predictions. But some of the trees will happen to question good characteristics and make good predictions (because the leaves are estimated based on training data).

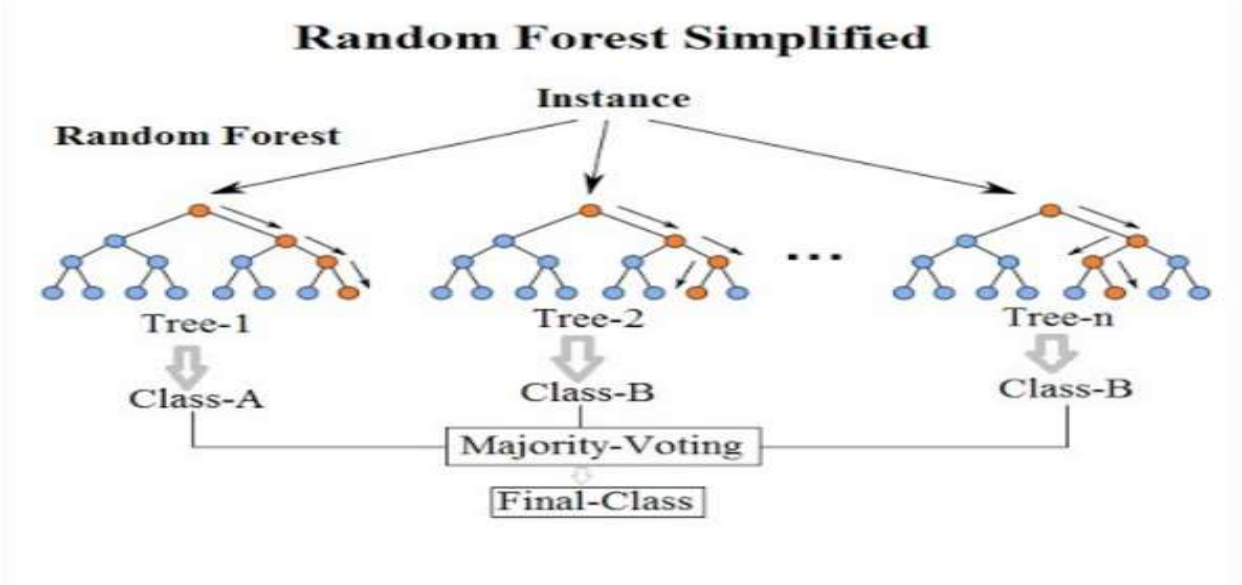

Figure 3.1.1 Random forest view

3.1.2. GRADIENT BOOSTING

The idea behind "gradient boosting" is to take a weak hypothesis or weak learning algorithm and make a series of tweaks to it that will improve the strength of the hypothesis/learner. This

type of Hypothesis Boosting is based on the idea of Probability Approximately Correct Learning (PAC).This PAC learning method investigates machine learning problems to interpret how complex they are, and a similar method is applied to Hypothesis Boosting.

In hypothesis boosting, you look at all the observations that the machine learning algorithm is trained on, and you leave only the observations that the machine learning method successfully classified behind, stripping out the other observations. A new weak learner is created and tested on the set of data that was poorly classified, and then just the examples that were successfully classified are kept.

The Gradient Boosting Classifier depends on a loss function. A custom loss function can be used, and many standardized loss functions are supported by gradient boosting classifiers, but the loss function has to be differentiable. Classification algorithms frequently use logarithmic loss, while regression algorithms can use squared errors. Gradient boosting systems don't have to derive a new loss function every time the boosting algorithm is added, rather any differentiable loss function can be applied to the system.

Gradient boosting systems have two other necessary parts: a weak learner and an additive component. Gradient boosting systems use decision trees as their weak learners. Regression trees are used for the weak learners, and these regression trees output real values..

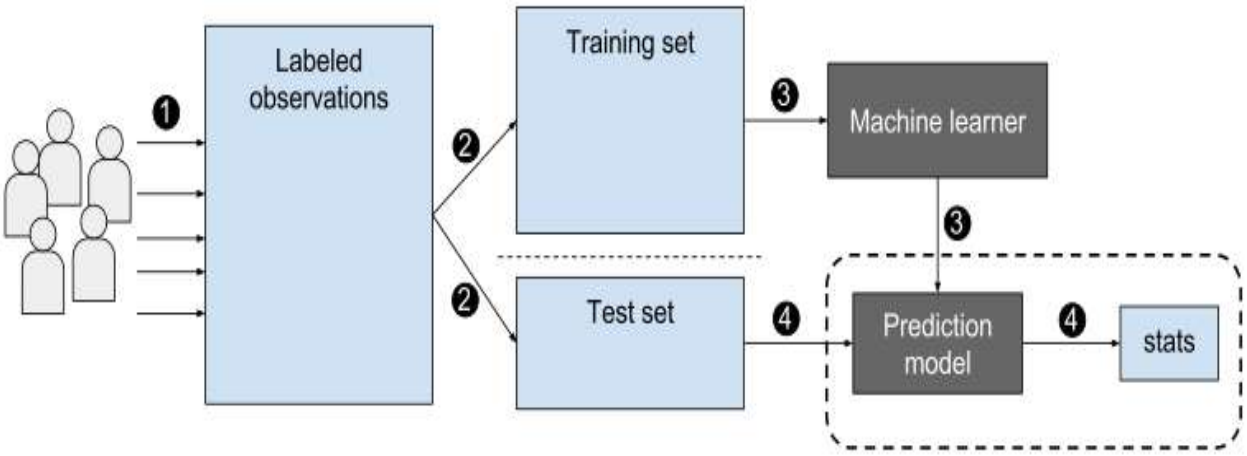

Figure 3.1.2 Gradient boost

3.1.3. DECISION TREE

A decision tree is a flowchart-like tree structure where an internal node represents feature(or attribute), the branch represents a decision rule, and each leaf node represents the outcome. The topmost node in a decision tree is known as the root node. It learns to partition on the basis of the attribute value. It partitions the tree in recursively manner call recursive partitioning. This flowchart-like structure helps you in decision making. It's visualization like a flowchart diagram which easily mimics the human level thinking. That is why decision trees are easy to understand and interpret.

Decision Tree is a white box type of ML algorithm. It shares internal decision-making logic, which is not available in the black box type of algorithms such as Neural Network. Its training time is faster compared to the neural network algorithm. The time complexity of decision trees is a function of the number of records and number of attributes in the given data. The decision tree is a distribution-free or non-parametric method, which does not depend upon probability distribution assumptions. Decision trees can handle high dimensional data with good accuracy.

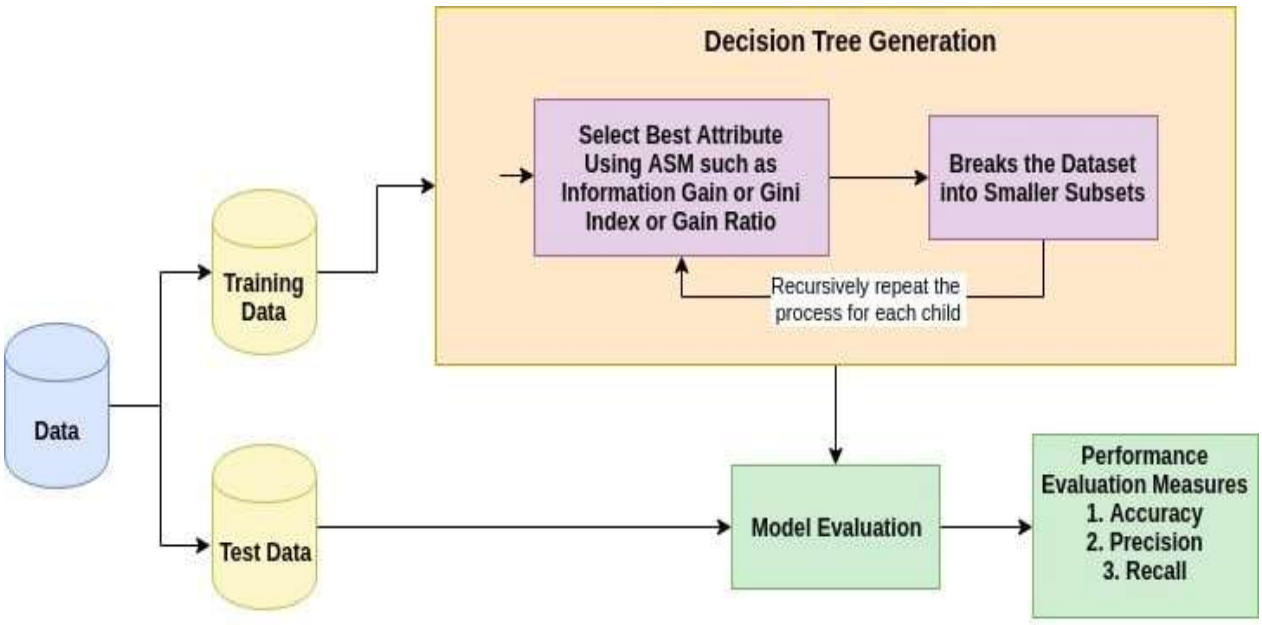

Figure 3.1.3 Decision Tree

3.2 SUMMARY

The methodology described in these touches on the various points and a flowchart of our approach is shown. Here, the brief description of the algorithms used in this process which includes pre-processing of dataset, promising feature selection, training of classifier and detection of advanced malware. In summary, the whole model is explained step by step along with the algorithm to describe the malware detection process.

CHAPTER -4
DESIGN & DEVELOPMENT

CHAPTER 4
DESIGN AND DEVELOPMENT

Here we introduce the various steps and components of a typical machine learning workflow for malware detection and classification, explores the challenges and limitations of such a workflow, and assesses the most recent innovations and trends in the field, with an emphasis on deep learning techniques. The proposed Design of this research study is provided below.

To provide a more complete understanding of the proposed machine learning method for malware detection, **Figure 4.1** and **Figure 4.2** illustrate the workflow process from start to finish.

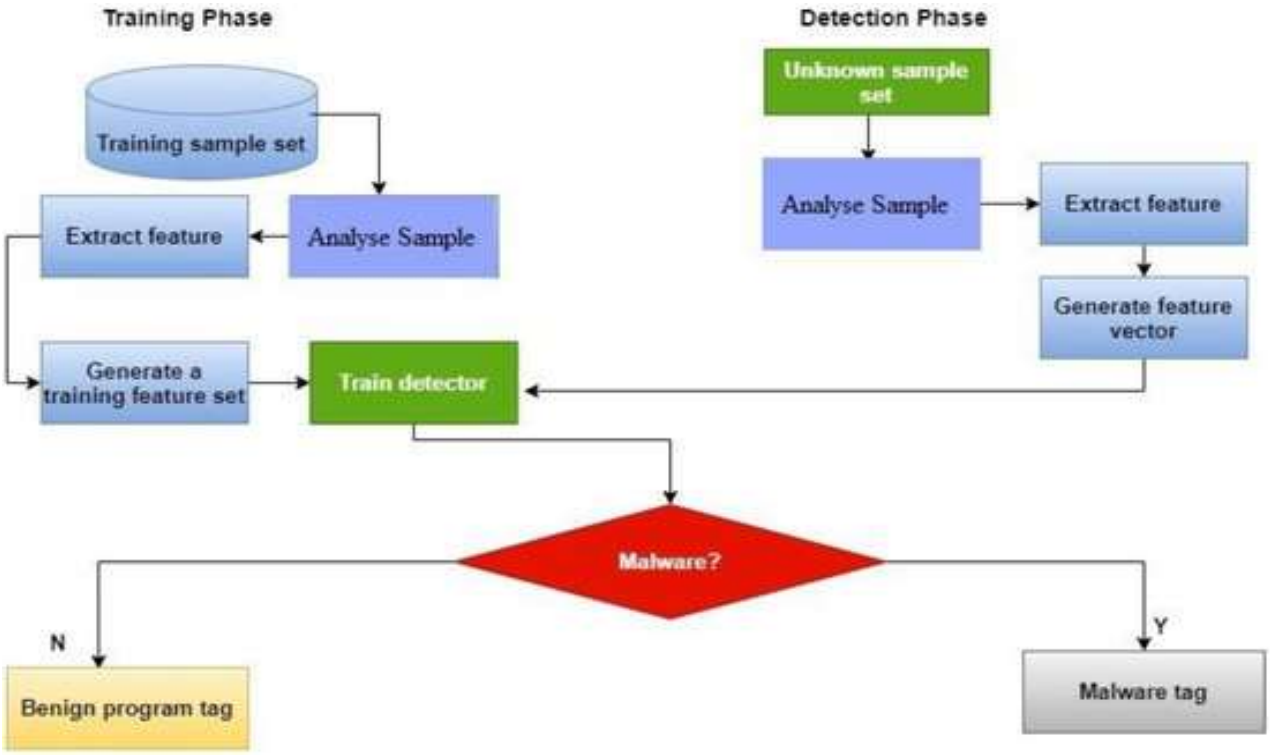

Figure 4.1 Proposed ML malware detection method.

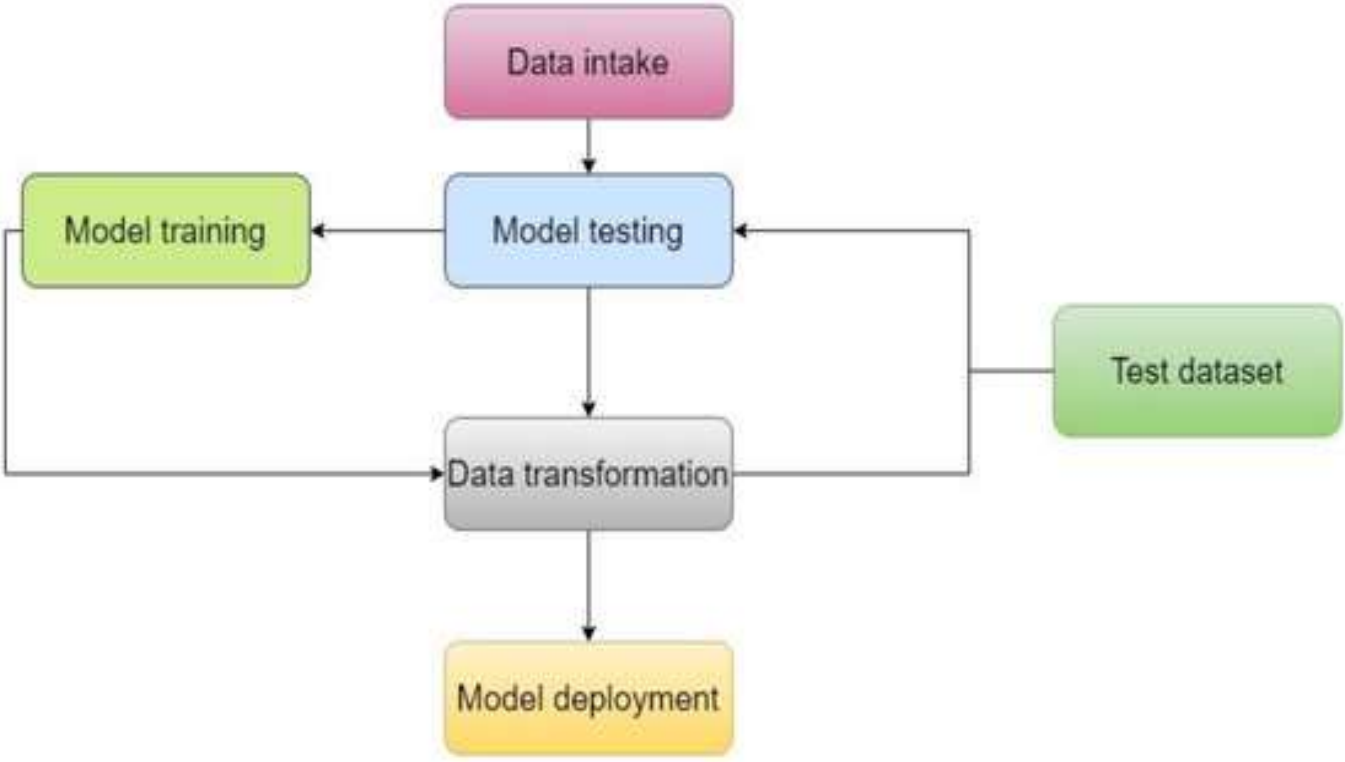

Figure 4.2 Workflow process illustration.

4.1. Dataset

This study relied entirely on data provided by the Canadian Institute for Cybersecurity. The collection has many data files that include log data for various types of malware. These recovered log features may be used to train a broad variety of models. Approximate 51 distinct malware families were found in the samples. More than 17,394 data points from different locations were included; the dataset had 279 columns and 17,394 rows.

4.2. Pre-Processing

Data were stored in the file system as binary code, and the files themselves were unprocessed executables. We prepared them in advance of our research. Unpacking the executables required a protected environment, or virtual machine (VM). PEiD software automated unpacking of compressed executables.

4.3. Features Extraction

Twentieth-century datasets frequently contain tens of thousands of features. In recent years, as feature counts have grown, it has become clear that the resultant machine learning model has been overfit. To address this problem, we built a smaller set of features from a

larger set; this technique is commonly used to maintain the same degree of accuracy while using fewer features. The goal of this study was to refine the existing dataset of dynamic and static features by keeping those that were most helpful and eliminating those that were not valuable for data analysis.

4.4. Features Selection

After completing feature extraction, which involved the discovery of more features, feature selection was performed. Feature selection was a crucial process for enhancing accuracy, simplifying the model, and reducing overfitting, as it involved choosing features from a pool of newly recognised qualities. Researchers have used many feature classification strategies in the past in an effort to identify dangerous code in software. As the feature rank technique is very effective at picking the right features for building malware detection models, it was extensively employed in this study.

CHAPTER -5
CONCLUSION

CHAPTER 5

CONCLUSION

We have proposed a malware detection module based on advanced data mining and machine learning. While such a method may not be suitable for home users, being very processor heavy, this can be implemented at enterprise gateway level to act as a central antivirus engine to supplement antiviruses present on end user computers. This will not only easily detect known viruses, but act as a knowledge that will detect newer forms of harmful files. While a costly model requires costly infrastructure, it can help in protecting invaluable enterprise data from security threats, and prevent immense financial damage.

The aim of this paper is to present a machine learning approach to the malware problem. Due to the sudden growth of malware, we need automatic methods to detect infested files. In the first phase of the work, the data set is created using infested and clean executables, in order to extract the data necessary for the creation of the data set, we used a script created in Python. After creating the data set, it must be ready to train machine learning algorithms.

FUTURE ENHANCEMENT

Though the proposed application has covered various aspects, we will add a few more in future. This can be made more accurate with adding more data set and further on adding more algorithms with better performance can add on to the accuracy of the model. It can also be hosted on the web for real time analysis of files on the cloud.

CHAPTER -6
REFERENCES

CHAPTER 6

REFERENCES

[1] Gupta, D., & Rani, R. (2018). Big Data Framework for ZeroDay Malware Detection. Cybernetics and Systems, 49(2), 103-121.

[2] Cho, I. K., Kim, T. G., Shim, Y. J., Ryu, M., & Im, E. G. (2016). Malware Analysis and Classification Using Sequence Alignments. Intelligent Automation & Soft Computing, 22(3), 371-377.

[3] Burnap, P., French, R., Turner, F., & Jones, K. (2018). Malware classification using self organising feature maps and machine activity data. computers & security, 73, 399-410.

[4] Ab Razak, M. F., Anuar, N. B., Salleh, R., & Firdaus, A. (2016). The rise of ―malware‖: Bibliometric analysis of malware study. Journal of Network and Computer Applications, 75, 58-76.

[5] Ray, A., & Nath, A. (2016). Introduction to Malware and Malware Analysis: A brief overview. International Journal

[6] Ucci, D., Aniello, L., & Baldoni, R. (2017). Survey on the usage of machine learning techniques for malware analysis. arXiv preprint arXiv:1710.08189.

[7] AlAhmadi, B. A., & Martinovic, I. (2018, May). MalClassifier: Malware family classification using network flow sequence behaviour. In APWG Symposium on Electronic Crime Research (eCrime), 2018 (pp. 1-13). IEEE.

[8] Khan, M. H., & Khan, I. R. (2017). Malware Detection and Analysis. International Journal of Advanced Research in Computer Science, 8(5).

[9] Ronen, R., Radu, M., Feuerstein, C., Yom-Tov, E., & Ahmadi, M. (2018). Microsoft Malware Classification Challenge. arXiv preprint arXiv:1802.10135.

[10] Ye, Y., Li, T., Adjeroh, D., & Iyengar, S. S. (2017). A survey on malware detection using data mining techniques. ACM Computing Surveys (CSUR), 50(3), 41.

[11] Wang, C., Ding, J., Guo, T., & Cui, B. (2017, November). A Malware Detection Method Based on Sandbox, Binary Instrumentation and Multidimensional Feature Extraction. In International Conference on Broadband and Wireless Computing, Communication and Applications (pp. 427-438). Springer, Cham.

[12] Pai, S., Di Troia, F., Visaggio, C. A., Austin, T. H., & Stamp, M. (2017). Clustering for malware classification. Journal of Computer Virology and Hacking Techniques, 13(2), 95-107.

[13] Gupta, S., Sharma, H., & Kaur, S. (2016, December). Malware Characterization Using Windows API Call Sequences. In International Conference on Security, Privacy, and Applied Cryptography Engineering (pp. 271-280). Springer, Cham.

[14] Liu, L., Wang, B. S., Yu, B., & Zhong, Q. X. (2017). Automatic malware classification and new malware detection using machine learning. Frontiers of Information Technology & Electronic Engineering, 18(9), 1336-1347.