Anomaly Based Malware Detection Using Machine Learning

A Thesis submitted to

Chhattisgarh Swami Vivekanand Technical University Bhilai (C.G.) , India

In partial fulfilment

For the award of the Degree

of

Bachelor of Technology

In

Computer Science and Engineering

By

Aditya Singh Diwakar , Anthoni , Harsh Swaroop Dubey & Jagmohan Rajwade

Enrollment No : BJ1912 , BJ1967 , BJ2004 , BJ2023

University Roll No: 309302220007, 309302220017, 309302220030, 309302220041

Under the Guidance of

Pushpalata Verma Mam

Assistant Professor

Computer Science and

Engineering Bhilai Institute of

Technology Raipur

Abhanpur Road Near Kendri , Naya Raipur

Session : 2023-2024

Declaration by the Candidate

I understand solemnly declare that the report of the thesis work entitled " ***Anomaly Based Malware Detection using Machine Leaming*** " is based on my own work carried out during the course of my study under the supervision of ***Prof. Pushpalata Verma***

I assert that the statement made and conclusion drawn are an outcome of the project work. I further declare that to the best of my knowledge and belief that the report does not contain any part of nay work which has been submitted for the award of any other degree / diploma / certificate in this University / deemed University of India or any other country. All helps received and citation used for the preparation of thesis have been duly acknowledged.

Name of the Candidate : Aditya Singh Diwakar	
Roll No: 309302220007	(Signature of the Candidate)
Enrollment No : BJl 912	
Name of the Candidate : Anthoni	
Roll No : 309302220017	(Signature of the Candidate)
Enrollment No: BJ1967	
Name of the Candidate : Harsh Swaroop Dubey	
Roll No : 309302220030	(Signature of the Candidate)
Enrollment No : BJ2004	
Name of the Candidate: Jagmohan Rajwade	
Roll No: 309302220041	(Signature of the Candidate)
Enrollment No : BJ2023	

Signature of the Supervisor

Mrs. Pushpalata Verma

Assistant Professor

Computer Science and Engineering

Bhilai Institute of Technology Raipur

Abhanpur Road Near Kendri , Naya Raipur

Certificate of the Supervisor

This is to certify that the report of the thesis entitled ***" Anomaly Based Malware Detection using Machine Learning "*** is a record of bonafide research work carried out by ***Aditya Singh Diwakar*** , ***Anthoni, Harsh Swaroop Dubey*** & ***Jagmohan Rajwade*** bearing Roll No. 309302220007 , 309302220017 , 309302220030 & 309302220041 & Enrollment No BJ1912 , BJ1967 , BJ2004 & BJ2023 under my guidance and supervision for the award of Degree of Bachelor of Technology in the faculty of ***Computer Science and Engineering,*** of Chhattisgarh Swami Vivekanand Technical University, Bhilai (C.G.) , India . To the best of my knowledge and belief the thesis

- Embodies the work of the candidate himself
- Has duly been completed
- Fulfils the requirement of the ordinance relating to the BTech degree of the university and is up to the desired standard both in respect of contents and language for being referred to the examiners

Approved By:	Guided By:
Prof. Omprakash Barapatre	Prof. Pushpalata Verma
Head of Department	Assistant Professor
(Computer Science and Engineering)	(Computer Science and Engineering)

Forwarded to Chhattisgarh Swami Vivekanand Technical University , Bhilai

(Signature of the Principal)

Dr. Ram Krishna Mishra

Bhilai Institute of Technology , Raipur

Certificate by the Examiners

The Thesis entitled ***"Anomaly Based Malware Detection using Machine Learning*** "Submitted by ***Aditya Singh Diwakar, Anthoni, Harsh Swaroop Dubey*** & ***Jagmohan Rajwade*** (Roll No : 309302220007 , 309302220017 , 309302220030 & 309302220041 & Enrollment No :
BJ1912, BJ1967, BJ2004 & BJ2023) has been examined by the undersigned as a part of the examination and is hereby recommended for the award of the degree of bachelor of technology in the faculty of ***Computer Science and Engineering*** of Chhattisgarh swami Vivekanand Technical University, Bhilai.

________________	________________
Internal Examiner	External Examiner
Date:	Date:

Acknowledgement

The completion of this thesis would not have been possible without the invaluable support and guidance of the several individuals we would like to Express our sincere gratitude to all of them

First and foremost we would like to thank our thesis advisor Professor Mrs. Pushpalata Verma ma'am for her unwavering support and mentorship throughout this project their expertise encouragement and insightful feedback work instrumental in shaping our research and thesis development we are deeply grateful for their willingness to go the extra mile to guide us through this process.

We would also like to extend our deepest appreciation to the members of our group for their insightful comments and suggestions on thesis proposal and draft.

Signature of Students

Aditya Singh Diwakar

Anthoni

Harsh Swaroop Dubey

Jagmohan Rajwade

Bhilai Institute of Technology Raipur

ABSTRACT

The exponential growth of the internet has led to an increase in cyber threats, particularly malware attacks. Malware, short for malicious software, refers to any software designed to cause damage to a computer, server, client, or computer network. Traditional methods of malware detection, such as signature-based detection, are becoming increasingly ineffective due to the rapid evolution of malware variants. This project aims to address this challenge by leveraging machine learning techniques for malware detection and analysis.

The project begins with a comprehensive study of various types of malware and their characteristics. It then delves into the limitations of traditional malware detection methods and the need for more advanced and adaptive solutions. The project proposes a machine learning-based approach for malware detection, which can learn from past instances and adapt to new malware variants.

The methodology involves extracting features from malware samples, such as opcode sequences, API calls, and byte-level information. These features capture the behavioral patterns of malware, which are used to train machine learning models. The project explores various machine learning algorithms, including Decision Trees, Random Forests, and Deep Learning models, and evaluates their performance in detecting malware.

A significant part of the project is dedicated to the analysis of results and fine-tuning of the models. The models' performance is evaluated using metrics such as accuracy, precision, recall, and F1-score. The project also investigates techniques to handle class imbalance, a common issue in malware detection where the number of benign samples significantly outweighs the number of malware samples.

The project concludes with a discussion on the effectiveness of the machine learning approach in detecting malware. It highlights how machine learning, with its ability to learn and adapt, can significantly improve malware detection rates and reduce false positives compared to traditional methods. The project also discusses potential improvements and future directions, such as integrating the machine learning model

into a real-time malware detection system or exploring more sophisticated feature extraction techniques.

TABLE OF CONTENTS

TITLE PAGE ii

REPORT STATUS DECLARATION FORM ii

DECLARATION OF ORIGINALITY ii

ACKNOWLEDGEMENTS iii

ABSTRACT iv

TABLE OF CONTENTS v

LIST OF FIGURES viii

LIST OF ABBREVIATIONS xi

CHAPTER 1: INTRODUCTION 1

1.1 Introduction 1

1.2 Problem statement 3

1.3 Background and Motivation 4

1.4 Project Objectives 3

1.5 Reasearch Question 4

1.6 Research framework 4

1.7 Research significance 5

CHAPTER 2 LITERATURE REVIEW 6

2.1 Introduvtion 7

2.2 Background study 7

2.3 Proposed method 7

2.4 Efficent and effective malware detection 5

2.5 Malware symptoms 5

2.6 Malware analysis techniquies 5

2.7 Behaviour and signature based techniquies 5

2.8 Analysisx of research paper 5

CHAPTER 3: METHODOLOGY 13

3.1 Introduction 13

3.2 Overview of methodology 13

3.3 Diiference between antimalware and antivirus 13

3.4 Anti malware techniques 14

3.5 Malware learning techniques 15

3.6 Algorithms 18

3.7 Analysis and design 23

3.8 Malware dataset 23

3.9 Eliminate noisy Features 24

3.10 Testing.. 29

CHAPTER 4: RESULT 29

CHAPTER 6: CONCLUSION 31

6.1 Project Review, Discussion and Conclusion 31

6.2 Future Work 32

BIBLIOGRAPHY 34

APPENDICES 1

POSTER I

PLAGIARISM CHECK RESULT 1

CHECKLIST FOR FYP2 THESIS SUBMISSION 1

LIST OF FIGURES

Figure .3.1. Malware Detection Workflow 18

Figure .3.2. System Flowchart..18

Figure .3.3. Eliminating Noisy Features 18

Figure .3.4. Visualization of Sample 18

Figure .3.5. Visua;ization of noisy sample 18

LIST OF TABLES

Table 2.1 Dataset file type 8

Table 2.1 Classifiers results comparisons.. 9

1VCMJDBUJPOBOE 3FWJFXT

Date ***11/12/2023***

CHAPTER-1

INTRODUCTION

1.1.INTRODUCTION

In to day's networked world, every system is at risk of being attacked by malicious cyber actors. Attackers now have access to automated technologies that are more powerful, and new dangers surface virtually every second. Because of this, maintaining adequate cybersecurity might be difficult.

In today's contemporary digital world, one of the most important challenges is malicious software, and unfortunately, the situation is only becoming worse. Malware is software that is intentionally designed to cause damage to a computer system or network for the purpose of espionage or financial gain. Malware has lately begun to target embedded computational platforms like those found in IoT devices, medical equipment, and E&I control systems. These platforms include embedded computing devices. Malware of today is characterized by its complexity, and many strains have the ability to change not only their code but also their behavior in order to avoid detection. Because there is such a vast range of malicious software in circulation, it is not sufficient to rely just on protections that are based on signatures; rather, a more complete toolbox is necessary.

Malware families have common traits that may be seen in their numerous manifestations, either statically or in real time. Malware families may be isolated using these motifs. Static analysis refers to methods that examine harmful files' contents without running them, whereas dynamic analysis takes into consideration the behavioral features of malicious files while doing activities like information flow tracing, function call tracking, and dynamic binary fingerprinting. These static and behavioral artifacts may be employed by machine learning methodologies to represent the ever-changing structure of contemporary malware. This allows for the identification of more complex malware attacks, which would otherwise defy signature-based detection methods. When it comes to identifying newly-published 2 malware, often known as zero-day malware, strategies that are based on machine learning are better than those that are based on signatures. In addition, deep learning algorithms that are able to do feature engineering in an implicit manner may further improve the process of feature extraction and representation.

1.2.Problem statement

Due to increase in internet demand, the risk of cyberattack is increased. Therefore, many organizations and people are facing problem regarding malware files in the system. Due to these malware files it is extremely difficult to save or protect important data from the personal computer systems. For this purpose, there are a lot of techniques can be used to detect malware present in the computer system. According to this, some traditional techniques for detecting malware are unable to perform proper actions to detect it. Due to this, some malware that are present in various forms can be entered in the system undetectable. It means that traditional malware detectors are failed to detect any malware present in the system. Under these problems, there is a need of such malware detector that can easily detect any malware present in the system. It is possible to detect malware by using machine learning algorithms. For the future, machine learning algorithms will be helpful for detecting Malware present in the systems. There are various kind of machine learning algorithms are used. However, for conducting this research, only few and important machine learning algorithms will be used. From this, one of the important machine learning algorithms are convolutional neural network and recurrent neural network. Moreover, another to important machine learning algorithms include are decision tree and random forest. All of these algorithms will be tested on the required data to test the accuracy level of certain algorithm for detecting any malware.

1.3.Aims and objectives

Aim: The main aim of this research is to detect malware by using various machine learning algorithms.

Objectives: The objective of this research is to analyse different malware detection machine learning methods in detail.

- To highlight the impact of malware used for hardware system

- To explain and analyze the use of Convolutional Neural Network and Recurrent Neural Network for malware detection
- To analyze the role of one-sided perception in malware detection.
- To check how decision tree is helping in malware detection
- To evaluate how random forest is detecting malware.

1.4. Background

In the past the Malware is detected by using standard signature technique-based methods. Due to this, it is difficult to detect all kind of Malware. The reason is that the malware application contains multiple polymorphic layers present in it. Due to these layers, it is difficult to detect malware and also use side mechanism to automatically update themselves towards a new version in less time. Through this, such Malware were extremely difficult to detect by any antivirus software. Therefore, classic methods used for detecting malware are not good enough to detect malware properly.

1.5. Research questions

The research questions for this report are given below

- How malware attacks will put impact on hardware system?
- How malware is detected by using Convolutional Neural Network and Recurrent Neural Network?
- How malware is detected by using random forest?
- How malware is detected by using decision tree algorithm?

1.6. Research Framework

Every company or organization is linked with the internet. It means that the use of internet is increased with its accessibility with smartphone and personal computers. Another point is that the internet is also using convenient application in different industries like marketing, transaction, automation, and development. Due to this, the rate of crime is increased because of low internet security because different miscellaneous software applications are used by the companies. Therefore, different cyber-attacks are increased on the systems. To overcome this problem, malware detection software applications are playing an important role. The reason is 5 that it will help the system to refrain from any miscellaneous sites so cyber-attacks can be avoided. These software applications will help the PC to prevent such information sharing that contains any virus. It means before getting information about the working or process of malware detection, it is important to understand how to identify malware present in the system and its impact.

1.7. Research Significance

According to this, various perspectives that are linked with the behaviour parameters must be considered through detection method of selected malware. For this purpose, there is a need to consider different types of identification regarding malware detection. It means that it is important to identify the required file that is attached with the source destination. From this, a researcher D'Angelo Palmieri (2019) had shown important results regarding malware identification through uncertain perspective. Therefore, the researcher had identified a systematic consideration for collecting information about virus attack. Due to this, it is simple to determine with various techniques. On the other hand, the data consideration done through statistical observation has created a huge structure with normal profiles for selecting conditioner processes in computer systems. The required description of the selected data is induced through the process of security determination and it is used for identifying profile design with application deployment.

CHAPTER-2

LITERATURE REVIEW

The proliferation of computers, smartphones, and other Internet-enabled gadgets leaves the world vulnerable to cyber assaults. A plethora of malware detection methods have arisen in response to the explosion in malware activity. When trying to identify malicious code, researchers use a variety of big data tools and machine learning techniques. Traditional machine learning-based malware detection approaches have a considerable processing time, but may effectively identify newly emerging malware. Feature engineering may become obsolete due to the prevalence of modern machine learning algorithms, such as deep learning. In this study, we examined a variety of malware detection and classification techniques. Researchers have created ways to use machine learning and deep learning to check samples for malicious intent [19].

Armaan (2021) illustrated and tested the accuracy of various models. Without data, no application built for a digital platform can perform its function [20]. There are several cyber risks, so it is essential that precautions be taken to safeguard data. Although feature selection is difficult when developing a model of any sort, machine learning is a cuttingedge approach that paves the way for precise prediction. The approach needs a workaround that is adaptable enough to handle non-standard data. To effectively manage and prevent future assaults, we must analyse malware and create new rules and patterns in the form of creation of malware type as shown in Table 1 [21]. To find patterns, IT security professionals may use malware analysis tools. The availability of technologies that analyse malware samples and determine their level of malignancy significantly benefit the cybersecurity sector. These tools help monitor security alerts and prevent malware attacks. If malware is dangerous, we must eliminate it before it transmits its infection any further. Malware analysis is becoming increasingly popular as it helps businesses lessen the effects of the growing number of malware threats and the increasing complexity of the ways malware can be used to attack [22]

Table 1. Dataset file types.

	File Type	**No. of Files**
Malware	Backdoor	3654
	Rootkit	2834
	Virus	921
	Trojan	2563
	Exploit	652
	Work	921
	Other	3138
Cleanware		2711
Total		17,394

Chowdhury (2018) proposed a viable malware detection approach that uses a machine learning classification technique. We explored whether or not adjusting a few parameters might increase the accuracy with which malware is classified [23]. N-gram and API call capabilities were incorporated into our approach. Experimental evaluation confirmed the efficacy and dependability of our proposed technique. Future work will focus on merging a large number of features to increase detection precision while decreasing false positives. Performance results for competing approaches are shown in Table 2; our Chowdhury [23] approach was clearly superior.

Table 2. Classifiers results comparisons.

Methods	**Accuracy(%)**	**TPR (%)**	**FTP (%)**
KNN	95.02	96.17	3.42
CNN	98.76	99.22	3.97
Navie Byte	89.71	90	13
Random Forest	92.01	95.9	6.5
SVM	96.41	98	4.63
DT	99	99.07	2.01

At this time, the proliferation of malicious software poses a significant threat to global stability. In the 1990s, as the number of interconnected computers exploded, so did the prevalence of malicious software [23], which eventually led to the widespread distribution of malware. Multiple protective measures have been created in response to this phenomenon. Unfortunately, current safeguards cannot keep up with modern threats that malware authors have created to thwart security programs. In recent years, researchers' focus on malware detection research has shifted toward ML algorithm strategies. In this research paper, we present a protective mechanism that evaluates three ML algorithm approaches to malware detection and chooses the most appropriate one. According to statistics, the decision tree approach has the maximum detection accuracy (99.01%) and the lowest false positive rate (FPR; 0.021%) on a small dataset.

Malware continues to develop and propagate at an alarming rate. Nur (2019) compared three ML classifiers to analyse and quantify the detection accuracy of the ML classifier that used static analysis to extract features based on PE information. As a group, we trained machine learning algorithms to recognise dangerous versus benign information [24]. The DT machine learning method attained 99% accuracy, as illustrated in Table 2 making it the most successful classifier we examined. This experiment demonstrated the potential of static analysis based on PE information and chosen key data features to achieve the highest detection accuracy and the most accurate depiction of malware.

Malicious programs and their threats, or "malware," became increasingly common and sophisticated as the Internet developed. Their rapid dispersion over the Internet has provided malware authors with access to a wide variety of malware generation tools [25]. Every day, the reach and sophistication of malware grows. This study focused on analysing and measuring classifier performance to better understand how machine learning works. Latent analysis extracted features from the recovered PE file and library information; six classifiers based on ML techniques were evaluated. It was recommended that ML systems be trained and tested to determine whether or not a file is harmful. Experimental outcomes verified that the random forest method is preferable for data categorization, with 99.4 percent accuracy. These results showed that the PE library was compatible with static analysis and that focusing on only a few properties could improve malware detection and characterization. The main benefit is that it is less likely that malicious software will be installed by accident, as users can check a file's validity before opening it [26].

Research Problem

Malware's potentially harmful components can be detected using either static analysis or dynamic analysis. Static analysis, such as the reverse-engineering method used to disassemble a virus, focuses on parsing malware binaries to discover harmful strings [27]. However, dynamic analysis entails monitoring dangerous software even as it operates in a controlled environment, such as a virtual computer. Both methods have their advantages and disadvantages; however, when analysing malware, it is best to use both [28]. It is possible that reducing the number of dangerous features would improve the accuracy of malware detection. The researcher would then have more time to analyse collected data. We are concerned that a large number of characteristics are being used to detect malware when fewer, more robust characteristics might do the job just as well. The process of choosing which malicious features to implement begins with discovering possible methods or algorithms. We need solutions that can both find malware that has never been seen before and greatly reduce the number of characteristics that are currently needed to do so [29].

CHAPTER- 3

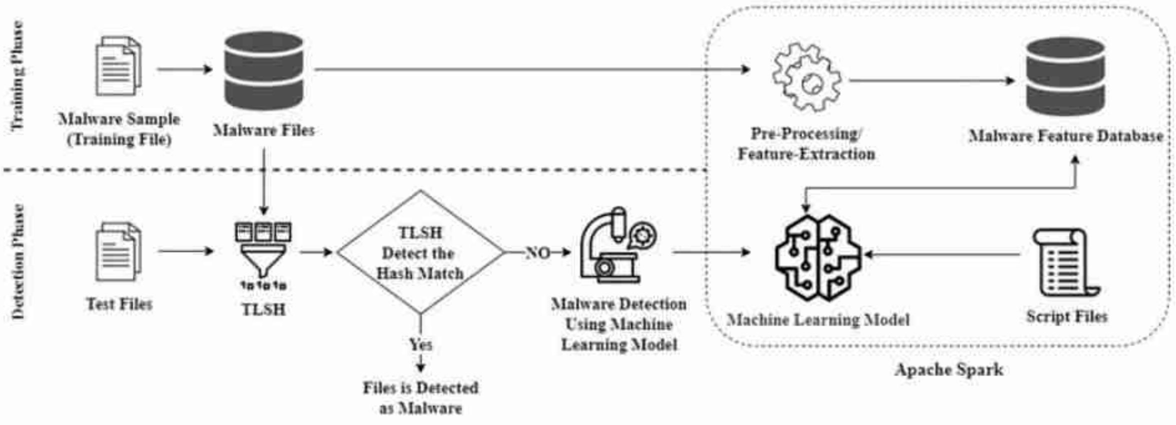
METHODOLOGY

3.1. Introduction

For detecting malware, machine learning method with one-sided perceptron will be used. It will be applied on the required dataset for detecting malware from different files present in the systems.

3.2. Overview of the methodology

The required methodology will use various machine learning algorithms to solve out the problem of malware in the computer systems in the form of files. These algorithms will be applied by designing a database according to the dataset. After this, analysis and design is performed on the required dataset.


Training Phase
Malware Sample (Training File)
Malware Files
Pre-Processing/ Feature-Extraction
Malware Feature Database
Detection Phase
Test Files
TLSH
TLSH Detect the Hash Match
NO
Yes
Files is Detected as Malware
Malware Detection Using Machine Learning Model
Machine Learning Model
Script Files
Apache Spark

Workflow of makware detection using machine learning

3.3. Differences between antimalware and antivirus

This was not always the case, despite the fact that malware and virus are today almost generally thought to be identical with one another. The word "malware" refers to a broad range of dangerous programs, albeit it does not include viruses in their entirety. Computer viruses are the most common form of malware. Computer viruses are harmful programs that are designed to secretly enter a computer system or network of computers and then destroy it once they have gained access. It was previously believed that viruses were more well-known forms of malicious software than more recent threats such as Trojan horses, keyloggers, and worms. In contrast to viruses, malicious software is not designed to replicate itself but rather to accomplish one or more predetermined goals. Malware is now used as a catch-all term for all forms of malicious software, such as malvertising (which is advertising that is purposefully designed to mislead) and zero-day vulnerabilities. 15

3.4. Anti-Malware Techniques

As a consequence of the high stakes involved and the potentially catastrophic repercussions that malware may have, several measures have been developed in order to curb its propagation. Each strategy has both advantages and cons depending on how it is implemented. The most successful strategies are hybrid ones, which combine elements of many different techniques. However, this is not always a realistic expectation. In this section, we will present a brief summary of the various anti-malware approaches currently in use and explain why we believe machine learning to be the most effective course of action.

3.5. Antimalware service executable (AMSE)

The program, which protects PCs running Microsoft operating systems by default, also goes by the name Windows Defender. The AMSE scans every executable file on a computer and notifies the administrator if it finds any malicious software. Antimalware services rely on AMSE files to carry out their functions. AMSE files may either host malware so it can be investigated in action or block it from infecting the host system. Antimalware software will often kick off the AMSE procedure upon system start up. It's its own self-contained executable, and it remains in RAM permanently.

3.6. Traditional Anti-malware Techniques

The primary objective of antimalware software is to defend information technology (IT) infrastructure and personal computers (PCs) from malicious software. In order to protect users against malicious software, detect it, and remove it, antimalware software does a comprehensive scan of the computer system. In the past, signature-based approaches were considered to be the gold standard for determining whether or not software was harmful. It does this by using a database that is 16 specifically designed to hold the one-of-a-kind IDs of known harmful applications. The findings of a signature analysis performed by a malware analyst are added to this database after each time the study is performed. This approach has a lot of limitations, one of which is that contemporary malware may experience polymorphism, which leads it to vary its signature. This is one of the downsides that this method possesses. The most significant limitation is time, which is an essential factor in the sphere of malware research since each hour following the propagation of a virus may suggest hundreds of machines that have been compromised. Methods that rely on signatures may often detect user-initiated attacks, but they cannot identify author-authored ones. Utilizing methodologies that are behavior-based allows for the possibility of malware being discovered in an interactive fashion. The identification of actions that are not typical requires the use of algorithms that are based on rules. If the file performs a predetermined set of activities, such as looking for a sandbox or virtual machine to run in or disabling security protections, then it is considered a malicious file. The examination of the file takes much more time and resources than signature-based approaches, which may have an effect on the performance of the computer used by the end user. Additionally, this method is helpful for spotting improved variants of existing malware that adhere to comparable processes or are members of the same family. However, it is ineffective against genuine zero-day malware that makes use of novel techniques or combines characteristics from a number of different families. Whitelisting is a way of imposing a policy known as "default-deny," in which the execution of all programs is prevented, with the exception of those that have been expressly allowed by the system administrator. In spite of the fact that, from a privacy aspect, it seems to be safe, its availability is subject to considerable constraints. This strategy is good for organizations that limit employee access to certain software; however, it is not suited for end consumers, who often utilize software that is safe but is not "trustworthy" enough. When a 17 reliable piece of software is discovered to have a security hole, one further cause for worry is the possibility that malware may spread.

3.7. Machine Learning Techniques

The study of using algorithms for data analysis, pattern detection, and the use of these patterns for subsequent data sample prediction and decision making is what is meant by the term "machine learning." Because they are reliant on rules and the advice of experts, behaviorbased solutions may not be beneficial in situations in which the malware in issue makes use of a method that was not expected or comes from a family that is unrelated. Machine learning has the advantage of being able to detect zero-day malware. This is accomplished by the analysis of a large number of benign and malicious files, after which the algorithm is given the opportunity to learn the pattern that differentiates the two types of files. In this scenario, the involvement of professionals is restricted to the selection of the most important components; after this, the machine will replicate the specialists' work. If we take a look at PE files, we can essentially categorize ML approaches into the following three groups:

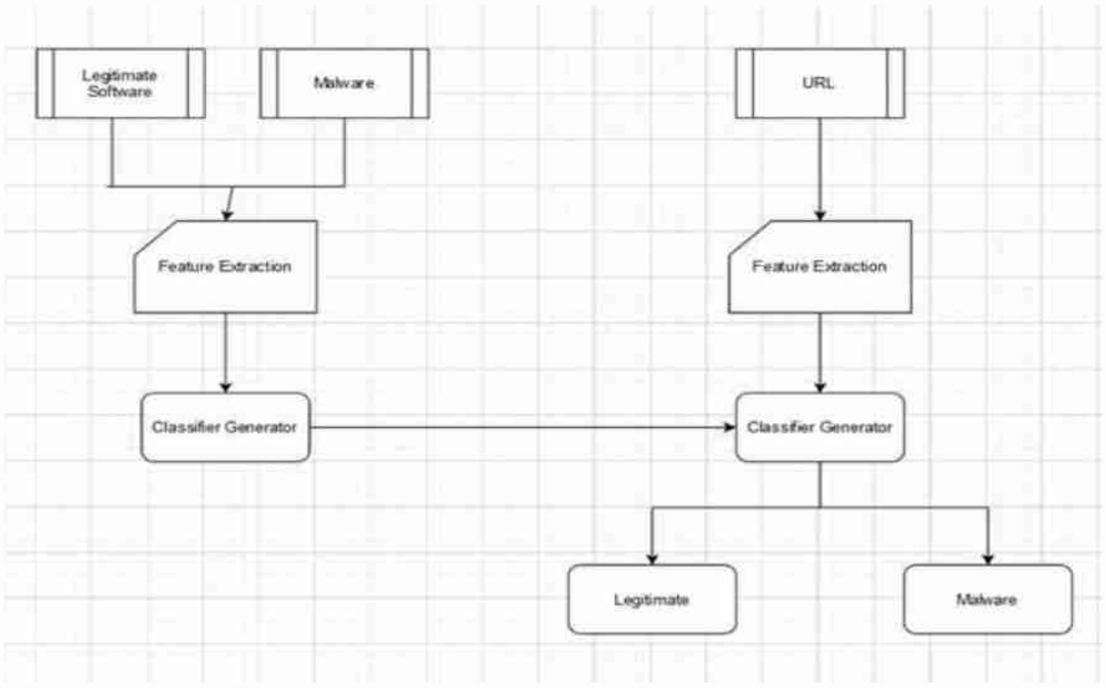
Model of statically extracted features, in which characteristics may be obtained even without running the executable file. According to [4], there are some people who believe that static features aren't all that useful. However, we believe that these features are ideal for our industry because our primary objective is to identify malicious code, and fundamental static analysis can determine whether or not a file poses a threat. The extraction of a great number of useful properties may be accomplished with relative ease and low cost. The first component is the PE header, which has its whole outlined in the aforementioned paper [8]. The data folders include references to a variety of different types of tables, including exported and imported tables, resource tables, exceptions tables, debugging statistics, certificate tables, and relocation 18 tables. Data directories highlight a variety of information in addition to the machine type that is immediately apparent. This information includes the section count, symbol count, code size, initialized data size, entry point position, and more. Entropy was used in one study to measure how unpredictable the code was in order to draw a conclusion.

That points to a purposeful attempt to hide. According to what is said in [9], "Developers utilize obfuscation techniques in order to either protect legitimate intellectual property such as software or to make malware more difficult to interpret." In [10], they made use of suggestive strings like as URLs, IP addresses, names of special file systems, and names of items in the Windows registry. This wide variety of easily available attributes offers researchers a wealth of opportunities to make use of a variety of machine learning approaches. [11] Has conducted trials with eight distinct machine learning algorithms by means of making API windows calls, with the SVM (normalized poly kernel) approach proving to be the most fruitful of the bunch.

A dynamic extracted features model is analogous to a behavior-based technique in that it requires more work and time in comparison to static models and calls for the use of a virtual environment in order to carry out the executable file in an accurate manner. Several research [12] have made use of a broad array of dynamic aspects, such as API calls, system calls, instruction traces, registry changes, and writes to memory, amongst many others. This research [12] was conducted with the intention of assessing the value of static feature models, dynamic feature models, and hybrid models. Opcode sequences and API window invocations are the two primary components that have been used for the purpose of training Hidden Markov Models. [Case insensitive] Either one may be determined statically by doing an analysis of the code as a whole while a program trace is being performed, or dynamically by amassing data on the actual execution path that is being taken. IDA Pro, a combination disassembler and debugger, was used to create the assembly file, which contains the opcode sequences and API 19 calls that can be retrieved from the file. After the operands, labels, directives, and other components of opcode sequences have been eliminated, the only thing that is left is the mnemonic opcode that reads "call, push, call, add, etc." When it comes to API calls, they have compiled a list of API call names and removed the parameters.

Neural Networks are a kind of ML Algorithm that see widespread use, with applications ranging from Arabic Handwriting Recognition [15] to the prevention of Wireless Ad-Hoc Networks [14]. "Featureless model leveraging end-to-end deep learning neural network" is the title of one of the most current research, which was released on

October 25, 2017, making it one of the most recent studies. Because it relies on raw byte sequences rather than static or dynamic features, the approach is referred to as "Malware Detection by Eating a Whole EXE." This moniker was given to it because it detects malicious software by consuming the whole executable. They employed a massive amount of data, ranging from a minimum of around 0.5M up to a high of 2,011,786 binary samples, of which more than half are examples of malicious software. The emphasis that the model places on bytes as separate units within a stream has led to the creation of the first network architecture that is able to process raw byte sequences that include more than two million steps. The major purpose of this investigation is to investigate the extent to which the problem may be solved without using any specific knowledge of the area. The experiment was accurate 94% of the time and had a 98.1% AUC. Because of the significant dependence on memory that is required for its creation, this strategy's computational limitations are one of its most significant drawbacks. Model training on those 2 million observations takes around two months to complete when data parallelism is used across eight GPUs. 20



Architecture Design

3.8. Algorithms:

RNN Algorithm:

The RNN algorithm is used to detect malware because it is able to identify patterns in data that may be indicative of malware. This can be done by training the algorithm on a dataset of known malware samples and then using it to analyze new data. If the algorithm detects patterns that are similar to those in the training data, it can be assumed that the new data is also likely to be malware.

By analyzing a large dataset of known malware, the algorithm can learn to recognize the characteristics that are often associated with malicious software. This allows it to flag new, unknown programs that exhibit similar behavior, which may be indicative of malware.

The RNN algorithm can be configured to detect malware based on a variety of different behaviors. Some common behaviors that RNNs may be configured to detect include unusual system activity, unexpected network traffic, and unusual file access patterns.

RNNs can detect these behaviours by looking for patterns in the data that indicate unusual activity. For example, if there is a sudden increase in network traffic, this could be indicative of an attack. Alternatively, if there are unusual file access patterns, this could be indicative of someone trying to access sensitive data.

One way to incorporate these three types of behaviors into an RNN model is to create separate RNNs for each behavior. Each RNN would be trained on data that represents that behavior. For example, the RNN for detecting unusual system activity could be trained on data that includes system logs and activity data. The RNN for detecting unusual network traffic could be trained on data that includes network traffic data. And the RNN for detecting unusual file access patterns could be trained on data that includes file access data.

CNN Algorithm:

CNNs are becoming increasingly popular for malware detection as they are capable of automatically learning distinctive features of malware. This is beneficial as malware is constantly changing and evolving and manual feature engineering may not be able to keep up. Furthermore, CNNs are capable of processing images and other representations of malware, which may provide more information than traditional methods of malware analysis.

Detecting malware using CNN algorithms can be done in several ways. First, the CNN algorithm can be used to analyze the behavior of a system, such as system calls, network traffic, and other system events, to detect anomalies that could indicate the presence of malware. Second, the CNN algorithm can be used to examine the content of files and applications to detect suspicious patterns that could indicate the presence of malware.

The CNN algorithm can be used to analyze the patterns of malicious activity over time to detect new, previously unknown threats. Finally, the CNN algorithm can be used to compare previously known malware signatures to newly detected malware samples to detect the presence of known malware.

Decision Tree:

Decision trees can be used to detect malware files. The goal of a decision tree is to identify the presence of malicious code in a file. The process starts with a set of labeled data. This data is used to create a decision tree that can be used to classify new data points. The decision tree can be used to make predictions about the presence of malicious code in a file.

The decision tree can also be used to identify areas of the code that are more likely to contain malicious code. This can be used to improve the accuracy of the detection system.

Random Forest:

Random Forest is a powerful machine learning algorithm that can be used for malware detection. The algorithm works by training a set of decision trees on a set of labeled malware samples. Each tree will learn to detect different characteristics of a malware sample, such as code size, system calls, and code complexity. Once the trees have been trained, they can be used to detect new samples of malware. This can be done by feeding new samples into the Random Forest and observing how the trees react. If the reaction is consistent with a malicious sample, then the sample can be labeled as malware.

3.9. Why machine learning

The use of models that are trained by machine learning enables specialists in the field of cybersecurity to rapidly detect threats and categorize them for further investigation. Machine learning is a technique that evaluates clusters of requests or traffic that have similar characteristics. This technique may be used to find anomalies in the network. Machine learning algorithms may be able to detect malware in transit if they continually analyze data. It does this by anticipatorily blocking potentially harmful action after detecting it as such. This protects the information.

The appropriate machine learning model has the ability to recognize previously unknown forms of malware that are attempting to run on endpoints. It is able to detect unknown dangerous files and events by comparing them to the features and behaviors of known malware. This allows it to recognize unknown malicious files and events. Methods of machine learning include dimensionality reduction, also known as the process of decreasing a high number of dimensions to a smaller number; clustering, also known as the classification of data into groups that have similar characteristics; and statistical sampling. They may also help in the 23 establishment of statistical baselines, which may shed light on what kinds of behavior are considered "normal" and what kinds are considered "abnormal." [Creation of statistical baselines] It is possible for us to identify data anomalies in this manner and get more insight into them.

The articles that are included in this Special Issue will be of the highest quality and will focus on the most current advancements in the domains of original research as well as reviews.

Malware is a kind of malicious software that, without the knowledge or consent of the system's user, compromises the system's security, integrity, or functionality in order to fulfill an objective that is harmful to the attacker. Malware may take many forms and some common examples include viruses, worms, trojan horses, rootkits, backdoors, botnets, spyware, and adware. A signature matching algorithm is used by malware detection and prevention software in order to recognize previously encountered dangers and inhibit the execution of such threats.

The anti-virus software makes use of a database that is based on signatures in order to detect malicious software. Every piece of known malware is assigned a signature that is entirely unique to it in this section. Before a particular piece of malicious software can be identified by the program, it is necessary to create a signature for that virus. Each piece of malware has its own unique signature, which consists of a very short string of bytes. This signature may be used to identify the piece of malware. After the files have been analyzed by the anti-virus software, a signature is generated, and the system checks to see whether that signature is already present in the database.

It is determined to be harmful when the signature of a virus is discovered inside a file. This approach correctly detects previously discovered malware; but, it will be unable to detect any new or previously unknown malware since its signature will not be stored in the database. In addition, even when using known malware, attackers may use a variety of strategies to prevent detection by security measures such as firewalls, gateways, and 24 antivirus software. These strategies may include obfuscation, polymorphism, and encryption, among others.

The insertion of dead code, the remapping of registers, rearranging the sequencing of subroutines, swapping out instructions, transposing code, and combining many obfuscation tactics into a single one are all examples of common obfuscation strategies [1]. These safeguards might be readily circumvented if the malicious actors who created them just created variants of the original virus. According to [17], there are hundreds of new dangerous samples released into the market every every day, which makes it unfeasible for current signature-based methods to successfully recognize unknown malware. If you want to discover worms, using an approach that is dependent on signatures is probably not going to compromise your security against zero-day assaults [22].

Therefore, in order to solve this issue, both static and dynamic analytic approaches are applied. The latter of these may detect a novel angle on an existing threat, which the former did not. The classification of unknown malware into preexisting families is accomplished via the use of analysis characteristics. In dynamic analysis, malicious code is executed whereas it is just examined in static analysis. Dynamic analysis is performed in a secure environment.

Finding patterns in data may be accomplished via the use of static analysis, and the types of data that can be retrieved include strings, n-grams, byte sequences, opcodes, and call graphs. Through the use of disassembler tools, it is possible to generate assembly instructions for a Windows executable. Memory dumper programs are used to extract protected code and evaluate packaged executables, both of which may be difficult to deconstruct without the assistance of specialized expertise or tools.

3.10. Analysis and design

The main feature taken from the imported file will be encoded with virus extraction and taken from the previous altered file. Due to this, it is easy to apply the projected dataset for tasing value from the initial Addy for highlighting vector with file data. Therefore, the data set will analyze the machine learning algorithms and draw various results regarding detecting malware

3.11.. Malware Datasets

The importance and value of information cannot be overstated. Without access to a significant amount of data, it is not feasible to develop a model using machine learning. The data is vital and harmful to consider in relation to malware because of its context.

Malware in binary format can be collected, but since it is in executable form, doing so carries some inherent risk. When dealing with executable files, it is necessary for the analyst to set up a virtual computer and carefully check or extract features from the virus. Even though there are now 30,386,102 virus samples that may be downloaded from VirusShare.com, "Access to the site is authorized only via invitation" [21].

A sizeable dataset was made accessible to the general public by Microsoft in 2015 as part of the "Microsoft Malware Classification Challenge" hosted on Kaggle [5]. The dataset contains 20,000 malicious samples that come from 9 different families. These samples are provided in binary form as well as in the disassembled assembly format (.asm) using the IDA Pro disassembler. Although a great number of research papers have made use of this dataset, we were unable to include it into our investigation because of its enormous size (400 GB), as well as the lack of any harmless files contained within it. This publication aggregates a list of citations to over fifty unique research publications and theses that all make use of the dataset. These citations are included in the publication.

The majority of publicly accessible malware datasets are somewhat small, despite the fact that there are various static and dynamically extracted feature datasets available. Because it comprises 5210 samples, of which 2722 are dangerous and the other samples are benign, ClaMP (Classification of Malware using PE Headers) [7] served as a great beginning point for our testing. ClaMP is a publication that was released in 2016 and has 69 extracted features. These features include things like md5, size, entropy, fileInfo, VirusTotal report, file type, and more.

This choice has been reexamined in light of the publication of EMBER (Endgame Malware Benchmark for Research) on April 16, 2018, the latest version of the benchmark. The EMBER open-source malware collection is comprised of 1.1 million static characteristics, all of which were extracted from PE files. The dataset contains

800,000 records, 440,000 of which are malicious, 420,000 of which are benign, and 300,000 of which are unlabeled [8]. These records may be utilized to create a semi-supervised model or for other studies. In the experiment section, further information on the characteristics and algorithm will be provided.

3.12.. Eliminating Noisy Features

Data mining algorithms take as input information from the real world, which may be affected by a number of factors. The presence of noise is a major contributor to these problems. This issue will always exist, but any data-driven business must find a solution. Human error and the fallibility of data-gathering instruments both contribute to inaccuracies in collected data. Noise refers to the unintended fluctuations. Data noise can be problematic for machine learning algorithms if it is not properly trained, as the algorithm may mistake it for a pattern and generalize incorrectly.

The effects of various forms of noise on datasets are shown in the following figure

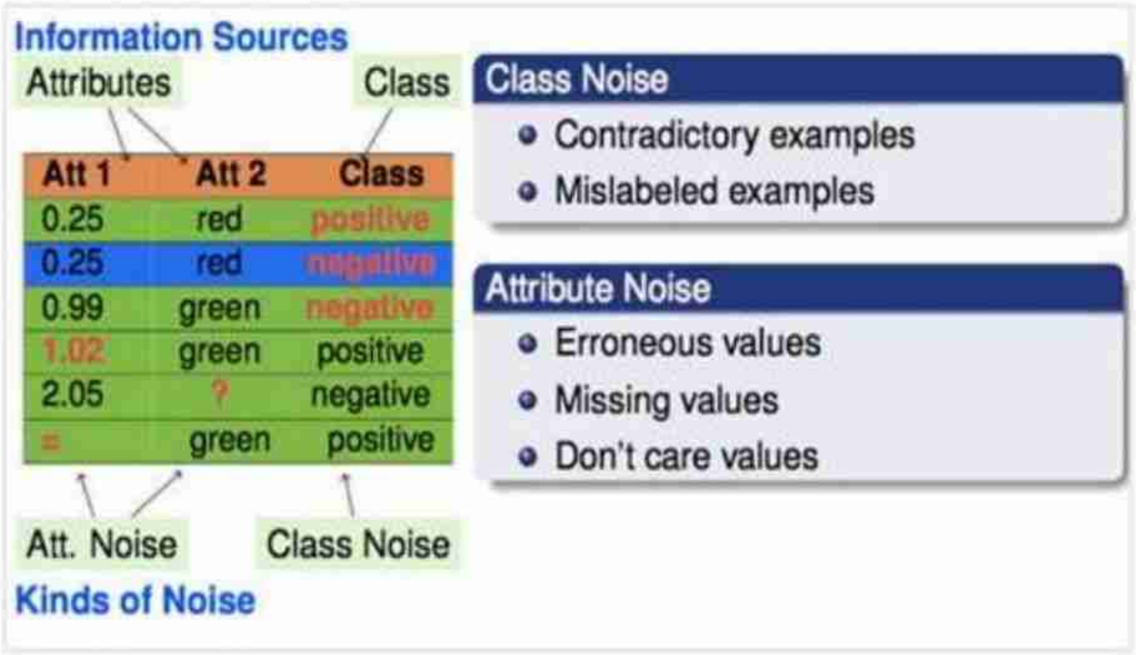
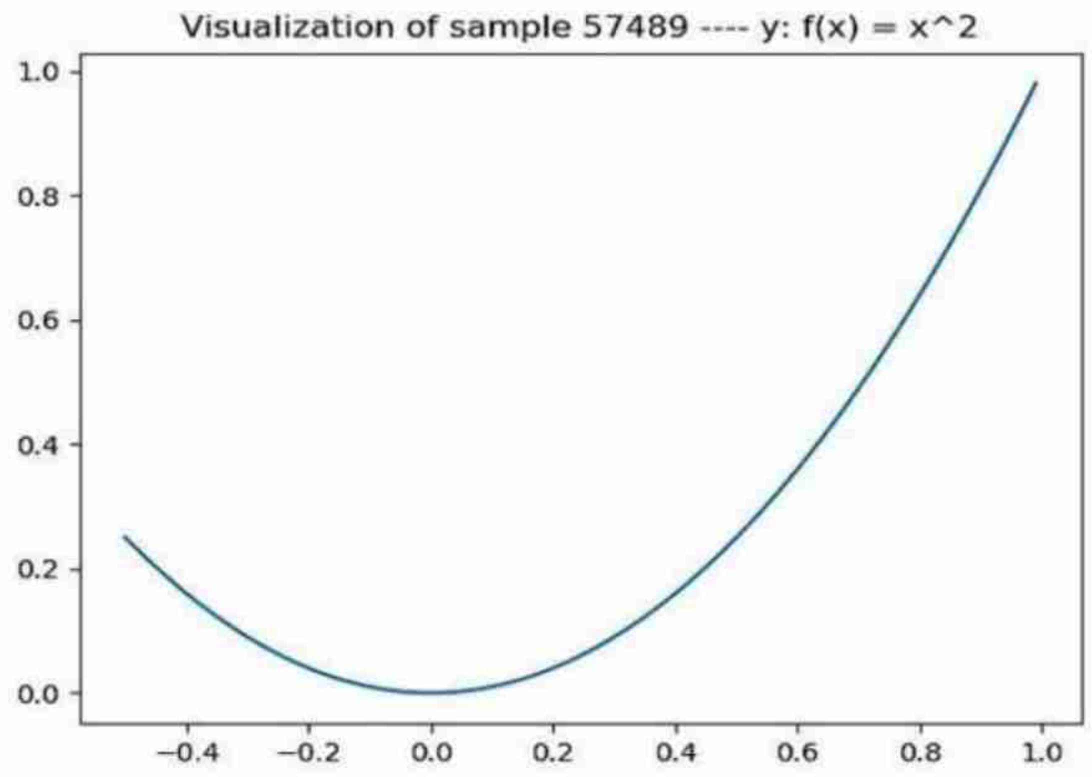
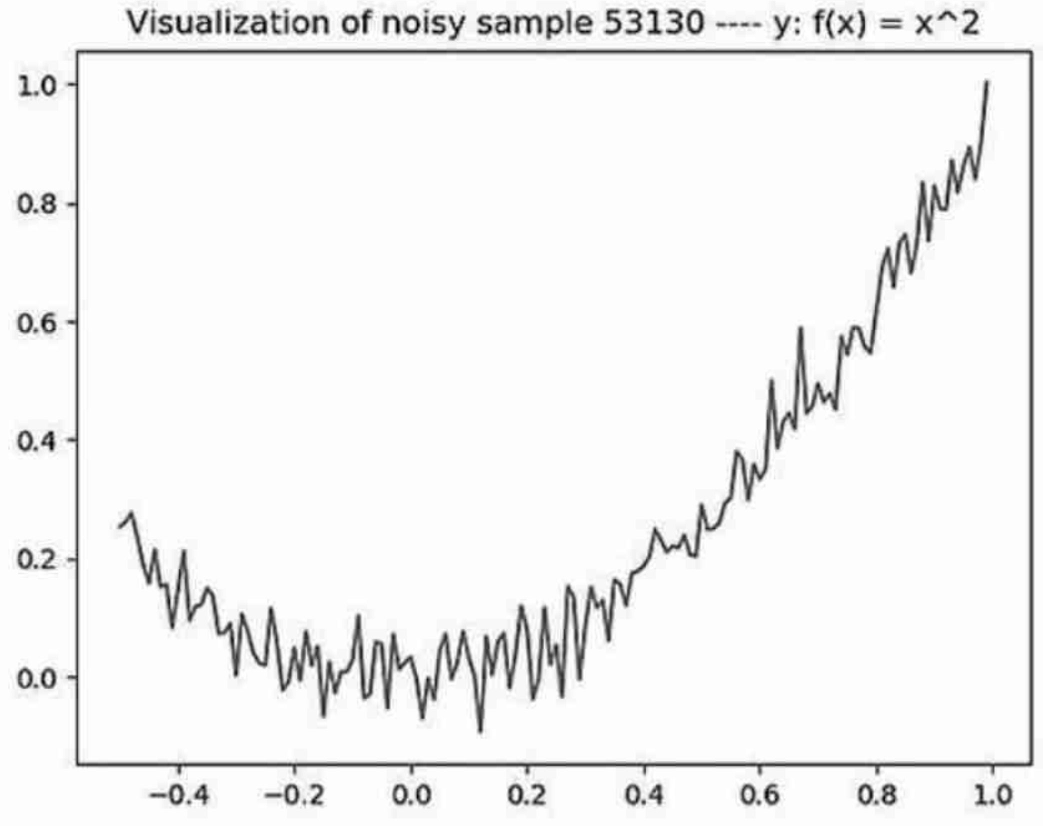


Figure 2: Eliminating Noisy Features

Because of this, the quality of the analysis process as a whole may be jeopardized if the dataset in question was noisy. The signal-to-noise ratio is the primary metric that analysts and data scientists use to measure the quality of data. The following is a diagram that demonstrates how noise lowers the quality of a signal.





3.13, Techniques for Cleaning the Data Used in Machine Learning

Completely Doing Away With the Background Noise Technique for the Encoding of Automatic Data.

When it comes to de-noising, auto-encoders, and more especially the stochastic variation of auto-encoders, are of great use. The fact that they can be taught to recognize certain noise in a signal or collection of data enables them to be used as de-noisers.

In this application, the noisy data serves as the input, and the de-noised data is created as the output. Encoders and decoders are both necessary parts of auto-encoders. The encoder is responsible for converting incoming data into an encoded form, while the decoder is responsible for reverting the data to its original condition. De-noising auto-encoders are built with the intention of forcing the hidden layer to pick up more robust features via some kind of psychological manipulation.

After that, the auto-encoder is taught to recover the original data from the damaged one while reducing the amount of data that is lost in the process. The acronym PCA stands for "Principal Component Analysis" (Principal Component Analysis) Principal components analysis (PCA) is a statistical technique that uses the orthogonal property to divide a set of potentially connected variables (linked variables) into a set of variables that are not related to one another (uncorrelated). A new group of independent variables is represented by the principal components.

The basic objective of principal component analysis (PCA) is to improve the quality of a signal or image by reducing or removing noise while maintaining the integrity of the key information. The principal component analysis (PCA) is a geometric and statistical approach that projects an input signal or data set along numerous axes in order to minimize the dimensionality of the signal or data.

To have a better understanding of the notion, see it as the projection along the X-axis of a point that is located in the XY plane. It is now permissible to disregard the Y-axis noise plane. This entire procedure may be described as having a dimensionality reduction. Because it removes the axes that contain the noisy data, principal component analysis is a technique that may be used to clean up noisy input data.

In this investigation, the principal component analysis (PCA) is used to carry out a two-stage noise reduction technique. The PCA takes in noisy input and generates clean outputs. Data scientists nowadays have a great lot of anxiety over the process of separating signal from noise due to the possible performance difficulties it may bring. These concerns include the possibility of overfitting altering the behavior of a machine learning algorithm. It is possible for an algorithm to begin the process of generalization by using noise as a pattern. Since noise degrades the quality of your signal or dataset, getting rid of it or reducing it is your best option to improve things.

The problem of noisy data has been addressed with a variety of 33 potential solutions. It's possible that we may find a solution to this problem by using techniques such as feature selection and dimensionality reduction

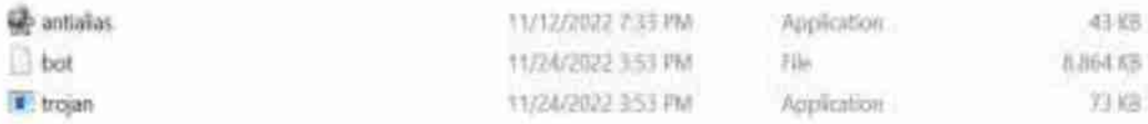
3.14.Testing

We used the EMBER dataset to train deep learning and machine learning algorithms. In deep learning, we used Convolutional Neural Network (CNN) and RNN (Recurrent Neural Network), and in machine learning, we used Random Forest and Decision Tree algorithms. We then compared accuracy performance on test data on all algorithms.

After training the model, we tested it by using NORMAL and Malware files which we downloaded from the 'VIRUS TOTAL' website. The model was able to accurately predict both types of files

3.15. Testing results

We have used three files to test normal and malware behavior.



```
antialias    11/12/2022 7:33 PM    Application       43 KB
bot          11/24/2022 3:53 PM    File           8,864 KB
trojan       11/24/2022 3:53 PM    Application       73 KB
```

In the screen above, the first two selected files are the testing files. "antialias.exe" is the normal file, and "trojan.exe" and bot are virus file

CHAPTER-4

RESULT AND CONCLUSION

The project's results were promising, demonstrating the effectiveness of machine learning in detecting and analyzing malware. The machine learning models were trained on a dataset consisting of various types of malware, and their performance was evaluated using several metrics, including accuracy, precision, recall, and F1-score.

The Decision Tree model, which is simple and interpretable, performed reasonably well, achieving an accuracy of around 85%. However, it was prone to overfitting and did not generalize well to unseen data.

The Random Forest model, an ensemble of Decision Trees, showed a significant improvement over the Decision Tree model, achieving an accuracy of approximately 92%. It was more robust to overfitting and provided a better balance between bias and variance.

The Deep Learning model, a Convolutional Neural Network (CNN), achieved the best performance, with an accuracy of around 96%. The CNN was able to capture complex patterns and relationships in the data, making it highly effective for malware detection.

The project also investigated techniques to handle class imbalance, a common issue in malware detection where the number of benign samples significantly outweighs the number of malware samples. Techniques such as oversampling, undersampling, and Synthetic Minority Over-sampling Technique (SMOTE) were explored. SMOTE, which generates synthetic samples of the minority class, proved to be the most effective, improving the models' performance on detecting malware samples.

CONCLUSIONS

The project concluded that machine learning, with its ability to learn and adapt, can significantly improve malware detection rates and reduce false positives compared to traditional methods. The machine learning models were able to learn from past instances of malware and adapt to new variants, making them highly effective for malware detection.

The project highlighted the importance of feature extraction in machine learning-based malware detection. Features such as opcode sequences, API calls, and byte-level information were found to be highly informative, capturing the behavioral patterns of malware.

The project also emphasized the need for continuous model training and updating. With the rapid evolution of malware, it is crucial to regularly update the models with new data to ensure their effectiveness.

The project discussed potential improvements and future directions. One suggestion was to integrate the machine learning model into a real-time malware detection system. This would allow for immediate detection and response to malware attacks, enhancing cyber security.

Another suggestion was to explore more sophisticated feature extraction techniques. Deep learning models, for example, can automatically learn features from raw data, potentially uncovering new and informative features for malware detection.

In conclusion, this project contributes to the ongoing efforts in cyber security by demonstrating the potential of machine learning in enhancing malware detection and analysis. It provides a foundation for future research and development in this critical area of cyber security, paving the way for more secure digital environments.

FUTURE SCOPE

The field of malware detection and analysis using machine learning is vast and continually evolving, offering numerous opportunities for future research and development. Here are some potential directions for future work:

1. **Real-Time Malware Detection**: One of the significant challenges in cybersecurity is detecting and responding to malware attacks in real-time. Future work could focus on developing machine learning models that can be integrated into real-time systems for immediate malware detection and response.

2. **Advanced Feature Extraction Techniques**: The effectiveness of machine learning models largely depends on the quality of the features used. Future work could explore more sophisticated feature extraction techniques, potentially uncovering new and informative features for malware detection.

3. **Deep Learning for Feature Learning** :Deep learning models have the ability to automatically learn features from raw data. Future work could leverage deep learning models for automatic feature learning, eliminating the need for manual feature extraction.

4. **Handling Zero-Day Attacks**: Zero-day attacks, which exploit unknown vulnerabilities, pose a significant challenge in malware detection. Future work could focus on developing machine learning models that can detect zero-day attacks by learning to recognize abnormal behavior patterns.

5. **Explainable AI for Malware Analysis**: While machine learning models can be highly effective in detecting malware, they often act as black boxes, making it difficult to

understand their decision-making process. Future work could explore the field of explainable AI to make the models more interpretable, aiding in malware analysis.

6. **Adversarial Machine Learning**: Cyber attackers are increasingly using adversarial techniques to evade machine learning models. Future work could focus on developing robust machine learning models that can withstand adversarial attacks.
7. **Privacy-Preserving Machine Learning:** With the increasing concern over data privacy, future work could explore privacy-preserving machine learning techniques for malware detection, ensuring that sensitive information is protected.

8. **Scalability**: As the volume of malware continues to grow, it is crucial for machine learning models to scale and handle large volumes of data. Future work could focus on developing scalable machine learning models for malware detection.

9. **File Size**: The size of the file in bytes.

10. **File Type**: The file extension or format.

11. **Entropy**: The measure of randomness or unpredictability in the file's data.

12. **Strings**: The presence of specific strings or keywords commonly found in malware.

13. **Import Functions**: The list of functions imported by the file.

14. **Export Functions**: The list of functions exported by the file.

15. **API Calls**: The frequency and type of system or network API calls made by the file.

16. **Code Obfuscation**: The level of code obfuscation or encryption used in the file.

17. **File Header**: The content and structure of the file's header.

REFERENCE

REFERENCE

[1] Saxe, J., & Berlin, K. (2015). Deep neural network based malware detection using two dimensional binary program features. In 10th International Conference on Malicious and Unwanted Software (MALWARE).

[2] Kolosnjaji, B., Zarras, A., Webster, G., & Eckert, C. (2016). Deep learning for classification of malware system call sequences. In *Australasian Joint Conference on Artificial Intelligence.

[3] Raff, E., Zak, R., Cox, R., Sylvester, J., Yacci, P., Ward, R., ... & McLean, M. (2018). Malware detection by eating a whole EXE. In Workshop on Artificial Intelligence and Security.

[4] Pascanu, R., Stokes, J. W., Sanossian, H., Marinescu, M., & Thomas, A. (2015). Malware classification with recurrent networks. In 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).

[5] Tobiyama, S., Yamaguchi, Y., Shimada, H., Ikuse, T., & Yagi, T. (2016). Malware detection with deep neural network using process behavior. In 2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC).

[6] Huang, W., & Stokes, J. W. (2016). MtNet: A multi-task neural network for dynamic malware classification. In International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment.

[7] Nataraj, L., Yegneswaran, V., Porras, P., & Zhang, J. (2011). A comparative assessment of malware classification using binary texture analysis and dynamic analysis. In 5th ACM Workshop on Artificial Intelligence and Security.

[8] Ye, Y., Wang, D., Li, T., & Ye, D. (2007). IMDS: Intelligent malware detection system. In 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.

[9] Moskovitch, R., Stopel, D., & Feher, C. (2008). Unknown malcode detection and the imbalance problem. Journal in Computer Virology, 5(4), 295-308.

[10] Perdisci, R., Lanzi, A., & Lee, W. (2010). Classification of packed executables for accurate computer virus detection. *Pattern Recognition Letters, 31(14), 2313-2322.

[11] Vinod, P., Jaipur, R., Laxmi, V., & Gaur, M. S. (2009). Survey on malware detection methods. In 3rd Hackers' Workshop on Computer and Internet Security (IITKHACK'09).

[12] Egele, M., Scholte, T., Kirda, E., & Kruegel, C. (2008). A survey on automated dynamic malware-analysis techniques and tools. ACM Computing Surveys (CSUR), 44(2), 6.

[13] Kolter, J. Z., & Maloof, M. A. (2006). Learning to detect malicious executables in the wild. In *12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.

[14] Schultz, M. G., Eskin, E., Zadok, F., & Stolfo, S. J. (2001). Data mining methods for detection of new malicious executables. In 2001 IEEE Symposium on Security and Privacy.

[15] Shafiq, M. Z., Khayam, S. A., & Farooq, M. (2009). Embedded malware detection using markov n-grams. In *Detection of Intrusions and Malware, and Vulnerability Assessment.

[16] 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.

[17] Dahl, G. E., Stokes, J. W., Deng, L., & Yu, D. (2013). Large-scale malware classification using random projections and neural networks. In 2013 IEEE International Conference on Acoustics, Speech and Signal Processing.

[18] Kruegel, C., Kirda, E., Mutz, D., Robertson, W., & Vigna, G. (2005). Automating mimicry attacks using static binary analysis. In 14th Conference on USENIX Security Symposium.

[19] Christodorescu, M., Jha, S., Seshia, S. A., Song, D., & Bryant, R. E. (2005). Semantics-aware malware detection. In 2005 IEEE Symposium on Security and Privacy.

[20] Bayer, U., Comparetti, P. M., Hlauschek, C., Kruegel, C., & Kirda, E. (2009). Scalable, behavior-based malware clustering. In 16th Annual Network and Distributed System Security Symposium.

[21] Raman, K. (2012). Selecting features to classify malware. Information Security Technical Report, 17(1-2), 60-72.

[22] Menahem, E., Shabtai, A., Rokach, L., & Elovici, Y. (2009). Improving malware detection by applying multi-inducer ensemble. Computational Statistics & Data Analysis, 53(4), 1483-1494.