IJARSCT

ISSN (Online) 2581-9429

Impact Factor: 6.252

International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)

Volume 2, Issue 5, May 2022

Malware Detection Using Machine Learning

Shreya Mohanki[1], Shreya Shinde[2], Preeti Pisal[3], Manasi Manikumar[4], Prof. Sagar Dhanake[5]
Students, Department of Computer Engineering[1,2,3,4]
Assistant Professor, Department of Computer Engineering[5]
Dr. D. Y. Patil Institute of Technology, Pune, Maharashtra, India

Abstract: *Malware detection is the study and prevention of malicious software in the realm of computer security. It isn't the only way to protect a business against a cyber-attack. Companies as well as administrators must assess their risk in order to be effective. In this study, we will look at many different ways of detecting computer malware and malicious software, websites, as well as future instructions in this field of study, and we'll also talk about the rise of computer viruses and worms and how to combat them. Innovative procedures and strategies such as the behavioral-based model and the signature-based model are replacing traditional detection methods. Future instructions will include the development of improved security solutions to combat cyber fraud, which has increased in recent years, particularly in the Asia-Pacific region. With the rise in cyber security fraud and other dangerous activities, traditional approaches are no longer sufficient to protect computers, as they have numerous limitations. To address these challenges, researchers have developed new techniques such as heuristic analysis and static and dynamic analysis, which can detect over 90% of malware samples with no false positives or negatives.*

Keywords: Behaviour-based approach, Dynamic analysis, Static analysis, Heuristic, Malware, Ransomware, Signature-based model, Vulnerability

I. Introduction

With the rise in malware threats, it's more necessary than ever to secure our computers and cellphones with anti-virus software. Machine learning is a valuable technology for detecting malicious software. It has been trained on millions of samples so that it can learn to recognise their properties at scale, even when new varieties of malware are discovered. It is a type of artificial intelligence that may be used to detect malware. It works by extracting information from a file and comparing it to known malware signatures. It also entails scanning the entire system or portions of it, extracting harmful software traits, matching these features to known behaviors, and detecting malware. The battle between malware designers and examiners is fierce. Both the research and hacking communities are working in the same direction; one is constructing a malware detection system, while the other is designing malicious software that will assault computer and network resources. Malware examiners look for known malware and try to detect it so that the user's computer systems are not attacked.

The virus has a variety of impacts, including:

- File system destruction.
- The file size has changed.
- Delete the whole contents of the DVD.
- Damage to the file allocation table, rendering the information on the disc unreadable.
- Various harmless but frightening graphic/sound effects

1.1 Motivation

Malicious code can infect our systems. Malicious code is software that is designed to harm or disrupt computer systems. Malicious code is code that is embedded in a program with the goal to destroy, corrupt, disable, or gain access to sensitive information. Malware is constantly growing and becoming more complex in order to take advantage of our devices' capabilities. Manufacturers, developers, and customers all need to be concerned about device security. This will lower the likelihood of malware being used in the future. The manner of delivery is frequently used to classify malicious code.Dropped malware and Drive-by malware are the two most popular varieties.

The malware that is dropped refers to programs that are transmitted by email, Facebook chat, instant messages, Skype contacts, and other social media platforms.

Drive-by Malware is a term used to describe programs or malware that are distributed when a user visits an infected website or downloads an infected file, believing it to be legitimate. Users who download free software from the internet without checking its reputation run the danger of acquiring malware and viruses, which could compromise and invade their privacy and sensitive information.

Before it can cause any harm, the malicious code must infiltrate and subvert the legitimate system. On a compromised device, a malicious program frequently downloads and executes more dangerous programs. It accomplishes this by taking advantage of flaws in the computer's operating system.

II. Related Work

Malware detection can be done using a number of techniques. Malware authors, on the other hand, have employed a variety of tactics to elude detection, necessitating the development of new and more precise approaches. It's hard to identify malware. New malware will continue to emerge even after upgrades, making this a never-ending battle. Traditional signature-based malware detection isn't a good answer because fresh samples can easily circumvent current methods. A vision-based approach is a recent way that employs deep learning techniques in the AI process to detect malicious software features. However, this method is ineffective when there are no examples or when they are not available for training.

III. Literature Review

Jagsir Singh and Jaswinder Singh (2020) published a journal review on Machine learning-based malware detection in executable files, which addresses the problem of harmful file proliferation. This document discusses trojans, worms, backdoors, spyware, logic bombs, ransomware, and viruses, among other harmful applications. Various strategies, including as behavior-based models, signature-based models, dynamic and static analysis, hybrid malware detection techniques, and hypervisor type I and II, have been developed to combat such threats. The malicious material is extracted from training samples.

The review of several potential signature-based techniques is presented in this part. Karnik et al. (2007) proposed a method for malware detection that employed a set of functions. The op-code group was represented by the element in the series. In addition, function series were represented in the signature of the malicious file in order to detect malware variations. To deal with the obfuscation strategies, the cosine similarity measure was developed. However, advanced obfuscation tactics (equivalent instruction replacement) and packaged malware were incompatible with this methodology.

A graph-based malware classification approach was reported by Bruschi et al. (2007). This approach, according to the author, tackles some simple obfuscation strategies. The control flow graphs were created from binary files and then compared to graphs of previously identified dangerous files. Two methods were used to detect the virus. The first method compares the graphs of the binary file B (under test) with the already known file M (malware), while the second algorithm matches the reduced graphs of the B file with the known file M. The author examined 78 malware files against the aforementioned two methods, with a false-positive rate of 4.5 for the first algorithm and 4.5 for the second algorithm, respectively. This method, however, could not tackle zero-day malware.

Zhang et al. demonstrated an approach for detecting and classifying malware using n-gram byte sequence characteristics. The most important n-gram bytes that potentially represent the malware file were retrieved using a selection method. Then, using a probabilistic neural network approach, a classifier was created. A set of decision-rules for identifying malware was included in each classifier. Three types of malware were chosen for categorization from the online VX Heavens database.

Moskovitch et al suggested a machine learning strategy based on opcodes for detecting unknown harmful files. The operation codes were classified as 1-byte, 2-byte, 3-byte, 4-byte, 5-byte, and 6-byt The heuristic malware detection method was proposed by Griffin et al. The goal of this approach was to develop a 48-byte sequence that might be used to detect virus variations. Despite utilising a single component signature, the author employed numerous component signatures to train the classifier here. During the runtime, however, the impact of various component signatures was unclear.

Gavrilut et al. (2009) employed a one-sided machine learning algorithm cascade to distinguish between benign and malicious code. The goal of this innovation was to use this algorithm to decrease the number of false positives. They started with a basic one-sided perception cascade. Then they included a cascade kernelized one-sided perceptions approach, which improved the detection technique's accuracy (88.78) by chevaliering the execution flow of malware files. On the basis of the execution flow, malware control flow graphs were created utilising function calls. After that, the graph was compared to the malware file's contained templates of malware CFG. Malware templates had previously been saved. If the computed CFG was part of a template, the malware detector was classed as a harmful file. The difficulty with these arcane tactics, such as code reordering, is that they can elude malware detection since they modify the harmful files' real execution pathways.

Searles et al. (2017) developed a malware detection approach based on a control flow graph. The Shortest Path Graph Kernel (SPGK) is a method that was developed to find commonalities between the CFG retrieved from binary files. The SVM approach is then used to train the classifier more efficiently and accurately using a comparable matrix. Parallelization was also employed to cut costs and speed up the classifier, which resulted in higher accuracy than the 2 or 3gram model.

III. Methodology

This method compares the outcomes of five different categorization algorithms for prediction. Based on a previously trained model, an ML model is used to predict the class for a certain file. The Ada-boost, decision tree, gradient boosting, and gaussian machine learning models were among the models tested. Algorithms must be trained to analyse data patterns. Android was originally introduced in 2008, and ML is already infiltrating the system. As the popularity of Android applications expanded, security vulnerabilities became more prominent. Because numerous academics are continually finding and proposing new ML-based solutions, there has been an increasing emphasis on applying machine learning for software security in the previous five years. Based on the study's findings, we devised a number of research topics. The next stage was to come up with a search strategy for finding completed studies that may answer our research objectives. The database's purpose, as well as the criteria for inclusion and exclusion, were set at this time. The study selection criteria were developed in order to find publications that addressed the stated research goals.

3.1 Architecture

A. Naïve bayes

It's the most powerful and accurate probabilistic machine learning algorithm available. The naive bayes algorithm has the benefit of being quick and economical to train, and it can generalise well from modest quantities of training data. The Naive Bayes algorithm is a probabilistic classification technique that is part of the expected-a-posteriori algorithm family. Each event is treated as a random variable in the probabilistic method, and its probability is calculated by dividing the number of occurrences by the total number of occurrences. All features are assumed to be independent in the Naive Bayes classifier.

B. Ada-boost

- Ada-boost is a top-performing algorithm on the market that has been developed for over a decade. It's a patented machine learning approach that enhances results for machine learning challenges by being fast and precise. It has been used to solve challenges ranging from data mining to natural language processing to computer vision.
- The Ada-boost algorithm is a simple machine learning system design algorithm. It works by gradually increasing the effectiveness of a classifier on a training dataset as more data becomes available. It should be emphasised that the approach is designed to fit the classifier's parameters to the training data sequentially. Ada-boost needs sequential training and cannot be used with data that is not in sequence. Bernard Widrow and Ted Smith created the algorithm at the beginning.

C. Decision tree

- A decision tree is a tree structure in which each leaf node signifies the outcome and each interior node reflects a property (or attribute). The uppermost node of the tree is the root node. Recursive partitioning is a technique for recursively dividing a tree. This decision-making aid has a flowchart-like format. It's a flowchart visual representation of human thought. Decision trees are therefore straightforward to understand and interpret.

- Advantages, a new algorithm, adds a lot of new functionality. Advantages is a decision-tree-based method that may be used in a variety of scenarios and disciplines. The objective is to identify the optimal set of options to pursue.
- The goal is to determine the optimal set of options that are worthwhile to pursue for some objective function, but with time, money, or other resources restrictions. One of the algorithm's main advantages is that it may be used to find good near-optimal solutions to a wide range of difficult situations.
- Advantages is a decision tree algorithm that suggests the optimal solution in each given circumstance. Machine learning is used to investigate all possibilities and discover the most efficient solution. Power users may also design their own decision trees for the algorithm to investigate, or download decision trees made by others.

D. Gini Index/Gini Impurity

- It detects impurity in the tree's node. Its value ranges from 0 to 1. As a result, a Gini index of 0 indicates that the sample is fully comparable and that all items are identical, whereas a Gini index of 1 indicates that there is maximal disparity among the elements.

IV. System Design And Flow

4.1 Architecture of Hypervisor

Hypervisor It is an important piece of software that enables virtualization. The machines and operating systems they run on are theoretically guests of the effective hardware. There are two hypervisors in this architecture. There are two types of hypervisors: type I and type II. Type-I hypervisors are referred to as bare metal or native metal, whilst Type-II hypervisors are referred to as hosted hypervisors. Hypervisor is employed.

To separate the operating system and application from their basic hardware in this project. It produces a virtualization that separates the hardware and software.

The virtual machine's CPU, followed by its RAM and other hardware resources.

1. **A bare metal hypervisor (Type-I)** can access hardware resources directly. It is made up of several VMM (Virtual Type-I Hypervisor: Machine Monitors) depending on the hardware and requirements. This hypervisor is likewise thought to be safe. VMware, ESXi, XEN, and AHV are examples of Type-I hypervisors.
2. **Type-II Hypervisor:** A Type-II hypervisor runs inside an operating system and adapts to the hardware requirements. VMM (Virtual Machine Monitors) are used in Type-II hypervisors, which include Virtual Box, VMware Player, and QEMU. The fundamental distinction between Type-I and Type-II hypervisors is that Type-II hypervisors are usually placed on top of an existing operating system.

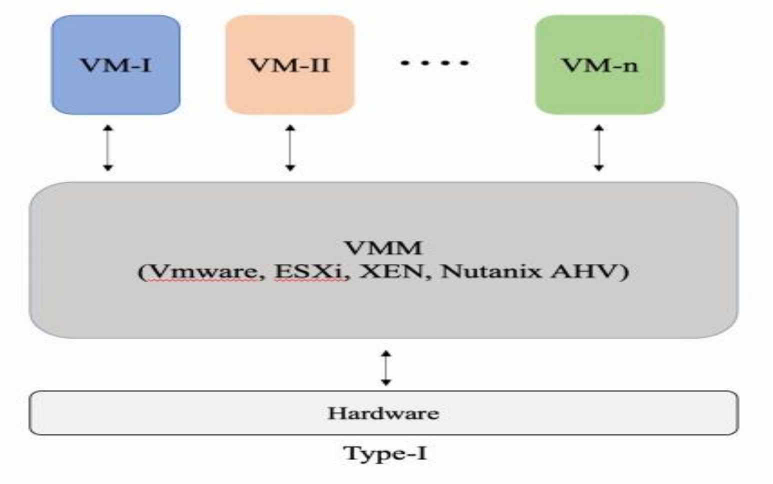


Figure 1: Type I Hypervisor

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Impact Factor: 6.252
Volume 2, Issue 5, May 2022

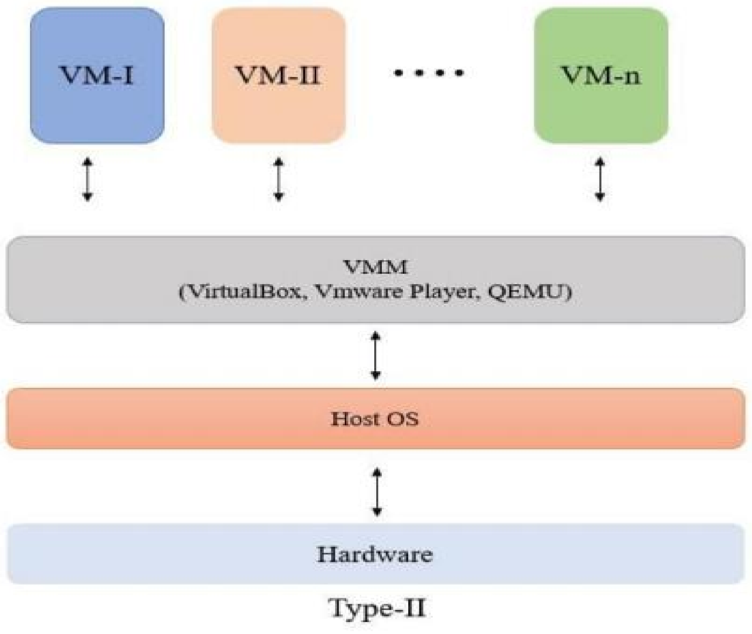


Figure 2: Type II Hypervisor

4.2 Architecture of Signature-based Model

Techniques based on signatures are quick and abstract. Using a hash is one simple technique to create signature-based malware files.

- **Algorithm:** The hash algorithm is an encryption algorithm that is used to ensure data integrity. MD5, SHA-1, SHA-2, NTLM, and LANMAM are some of the most regularly used hash algorithms. The infection is detected using this signature-based approach based on a general pattern of behaviour.
- **Files:** The data samples are first obtained in Malware and Benign samples in this method. Malware samples are examined. Develop defences and resolve the hazard technique.

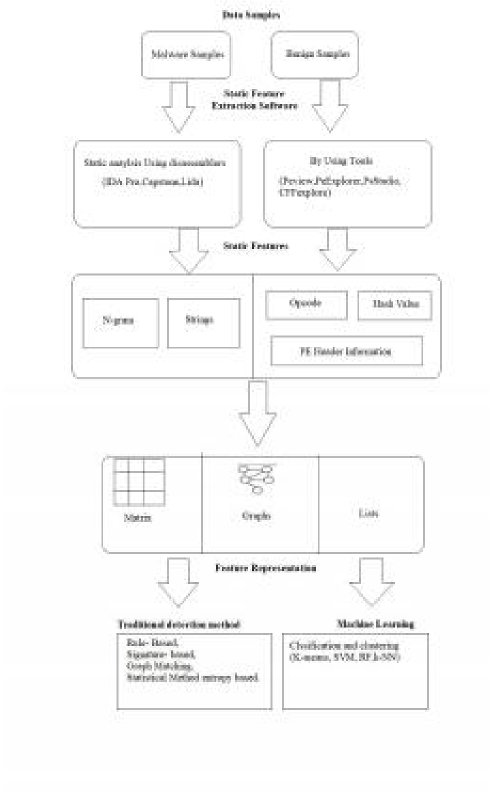
These two categories of samples are static feature extractions of software. Many tools are available in the signature-based model. Malware detection and detection tools have been created. The signature could be made up of strings, bytes, an executable file, or something else different. The use of assemblers such as IDA Pro, Capstone, which further extract n-gram bytes which can detect malicious files, and the four algorithms applied over the n-byte feature sets include Nave Bayes (NB), Ada boost, Decision tree (DT), and Artificial Neutral Networks, is one of the most popular (ANN). Basically, this technology uses a pattern matching approach to identify malicious code. This method detects malicious code by scanning for sequence within it.

However, due of their short execution duration or easy operation, many frequent harmful actions may be missed using this strategy. Despite the fact that this technique can detect malicious files to some extent. This method does not work if the pattern has been changed by a virus, hence it may be difficult to identify new infections using this method alone because it relies on a database of known patterns shared by numerous anti-malware products. Antivirus or anti-malware software frequently use this strategy, relying on a signature database to determine what virus or other harmful code is installed on a computer.

Because of the following factors, this approach has become less effective:

1. The growth of polymorphic code.
2. Malware insertion is increasingly employing encryption and encoding techniques to avoid detection during transmission.
3. The growing number of known and undiscovered variants that attackers can use to sign malicious code with valid keys, preventing it from being identified as such. Clustering is the process of grouping unlabeled examples in

machine learning. Unsupervised machine learning is required. The marked examples, on the other hand, are utilised for clustering and then classification. K-means, SVM, RF, and k-NN are the algorithms employed in this strategy.


Data Samples
Malware Samples
Benign Samples
Static Feature Extraction Software
Static analysis Using disassemblers (IDA Pro,Capstone,Lida)
By Using Tools (Peview,PeExplorer,PeStudio, CFFexplore)
Static Features
N-gram
Strings
Opcode
Hash Value
PE Header Information
Matrix
Graph
Lists
Feature Representation
Traditional detection method
Rule- Based, Signature- based, Graph Matching, Statistical Method entropy based.
Machine Learning
Clssification and clustering (K-means, SVM, RF,k-NN)

4.3 Architecture of Behavior based model

Malware can infect a computer in a variety of ways. Allowing malware to be detected based on behaviour might improve security against new or modified viruses. In the computing industry, behavior-based malware detection is also known as heuristic detection, and it differs from the usual method of scanning executables. For example, there is no need to scan executables that are only performed in memory, such as scripts or web pages, for harmful activities because they cannot be executed without an executable present. It can detect new malware mutations and distinguish between benign and harmful files. Malware is easy to detect on the surface since it manifests as a process or a file on the system. However, behavior-based detection approaches look for more subtle symptoms of infection, such as changes in start up sequence frequency and duration.

Behavior-based approaches are more durable but take a long time to implement. Malware behaviour is influenced by a variety of factors. APIs, browsers, operating systems, and network events all influence behaviour. This method provides a solution to the obfuscated malware. The obfuscation approach is utilised to solve this strategy. It is a technique for making textual and binary data difficult to read, making it difficult for attackers to discover malware files.

Malware samples and benign samples are used to acquire data samples. The behavioural properties of these two types of samples are extracted. Process Explorer, Wireshark, Regshot, and T Dump are among the tools utilised in Malware samples. Static and dynamic analyses are used to examine benign samples. It use sandboxes to identify suspicious activity in virtual machines. Cuckoo, CW Sandbox, Anubis, and Norman are the sandboxes used. File actions, Registry changes, Network artefacts, and System calls are removed and converted. It is the most effective method for distinguishing between malicious and benign files.

Traditional detection methods and machine learning are used to convert the characteristics. The tools utilised in the traditional method are rule-based, graph matching based on API calls, and statistical methods. The methods employed in machine learning algorithms are classification and clustering; the algorithms used in this approach are SVM, DT, KNN, NB, Ensemble, CNN, RNN, K-means, and so on.

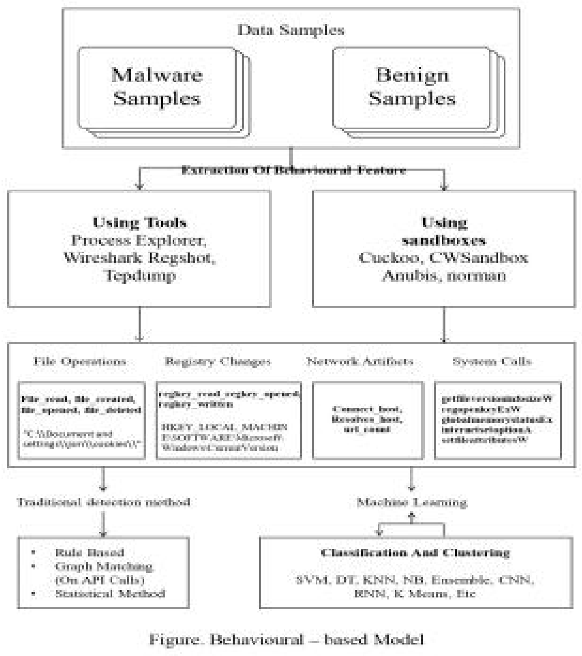

Data Samples
Malware Samples
Benign Samples
Extraction Of Behavioural Feature
Using Tools
Process Explorer, Wireshark Regshot, Tcpdump
Using sandboxes
Cuckoo, CWSandbox Anubis, norman
File Operations
Registry Changes
Network Artifacts
System Calls
File_read, file_created, file_opened, file_deleted
"C:\\Document and settings\\user\\cookies\\"
regkey_read regkey_opened, regkey_written
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion
Connect_host, Resolves_host, url_count
Traditional detection method
Machine Learning
Rule Based
Graph Matching (On API Calls)
Statistical Method
Classification And Clustering
SVM, DT, KNN, NB, Ensemble, CNN, RNN, K Means, Etc
Figure. Behavioural – based Model

Fig: Behaviour-based Model

4.4 Architecture of Hybrid Based Malware Detection

Because signature and behavior-based algorithms have both dos and don'ts, researchers have developed hybrid malware detection to address these difficulties. In this architecture, harmful and benign files are analysed in both static and dynamic methods.

These files are fed into malware classifiers for training. It's a way for categorising malware samples into different malware families. These classifiers are used to solve a variety of key security issues. Finally, testing is carried out in both signature-based and behavior-based analytic techniques. Finally, the malware detector database is updated using these strategies for both files.

Both detection methods are used to identify known and unknown malware files. As a result, utilising this technique reduces malware scanning time and reduces false positives. It also has the best detection rates for polymorphic malware.

Viruses, worms, bots, and Trojans are all examples of polymorphic malware. This hybrid approach to malware threats is more accurate, faster, and more robust. Another way is to use data samples, which are a collection of known malicious files and normal programme files, to uncover patterns that can effectively identify malware.

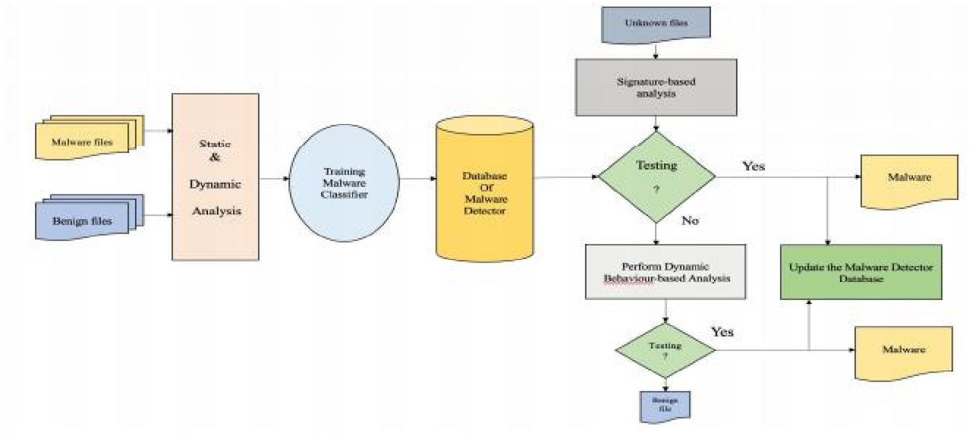
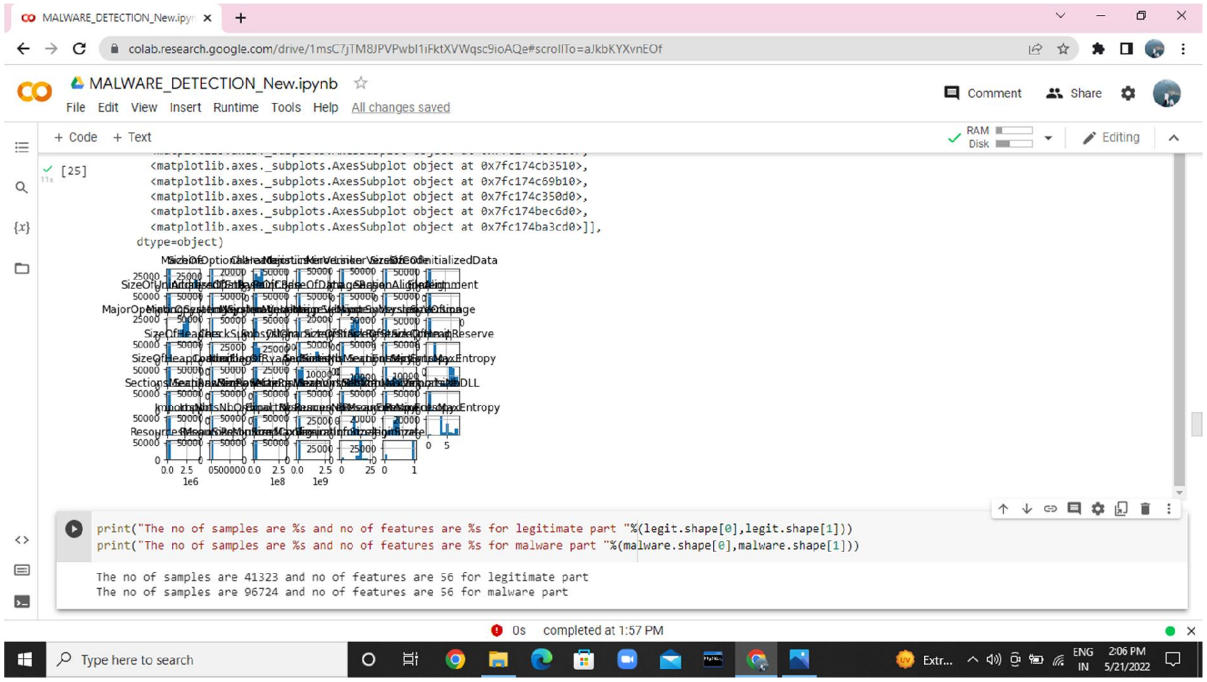

Unknown files
Signature-based analysis
Malware files
Benign files
Static & Dynamic Analysis
Training Malware Classifier
Database Of Malware Detector
Testing ?
Yes
No
Malware
Perform Dynamic Behaviour-based Analysis
Update the Malware Detector Database
Testing ?
Yes
Malware
Benign file

Fig: Hybrid Based Malware Detection

4.5 Application

The performance of this malware detection is to examine and understand how a researcher tackled the problem. The researchers together discover the fastest alternate method of detection and analysis of malware that is by using machine learning. So, by using Machine Learning it is easy to determine the best extraction features, representation and classification methods for the malware detection. Machine Learning consist of various fields that are subdivided into supervised and unsupervised learning, which are further used for malware detection. The techniques that are used are Naïve Bayes and Neural Networks.

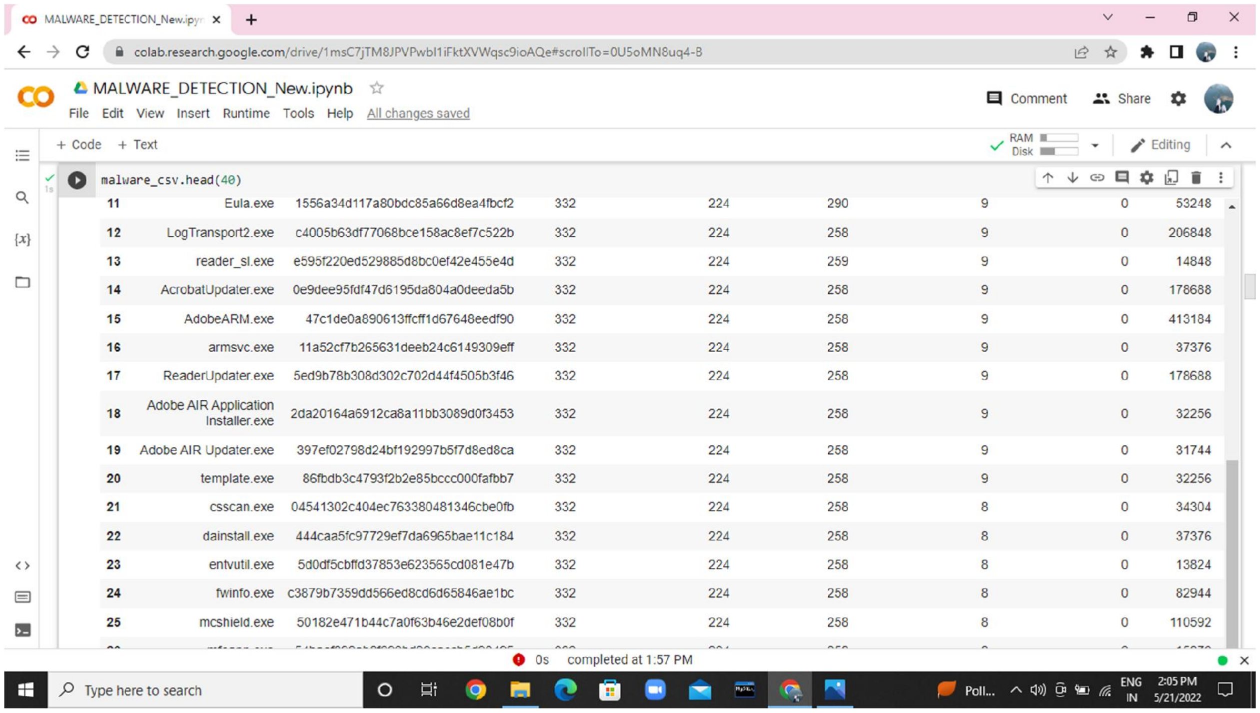
V. Project Implementation


MALWARE_DETECTION_New.ipynb
colab.research.google.com/drive/1msC7jTM8JPVPwbl1iFktXVWqsc9ioAQe#scrollTo=aJkbKYXvnEOf
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
RAM Disk
Editing
[25]
completed at 1:57 PM
Type here to search

```
    <matplotlib.axes._subplots.AxesSubplot object at 0x7fc174cb3510>,
    <matplotlib.axes._subplots.AxesSubplot object at 0x7fc174c69b10>,
    <matplotlib.axes._subplots.AxesSubplot object at 0x7fc174c350d0>,
    <matplotlib.axes._subplots.AxesSubplot object at 0x7fc174bec6d0>,
    <matplotlib.axes._subplots.AxesSubplot object at 0x7fc174ba3cd0>]],
  dtype=object)
```

```
print("The no of samples are %s and no of features are %s for legitimate part "%(legit.shape[0],legit.shape[1]))
print("The no of samples are %s and no of features are %s for malware part "%(malware.shape[0],malware.shape[1]))
```

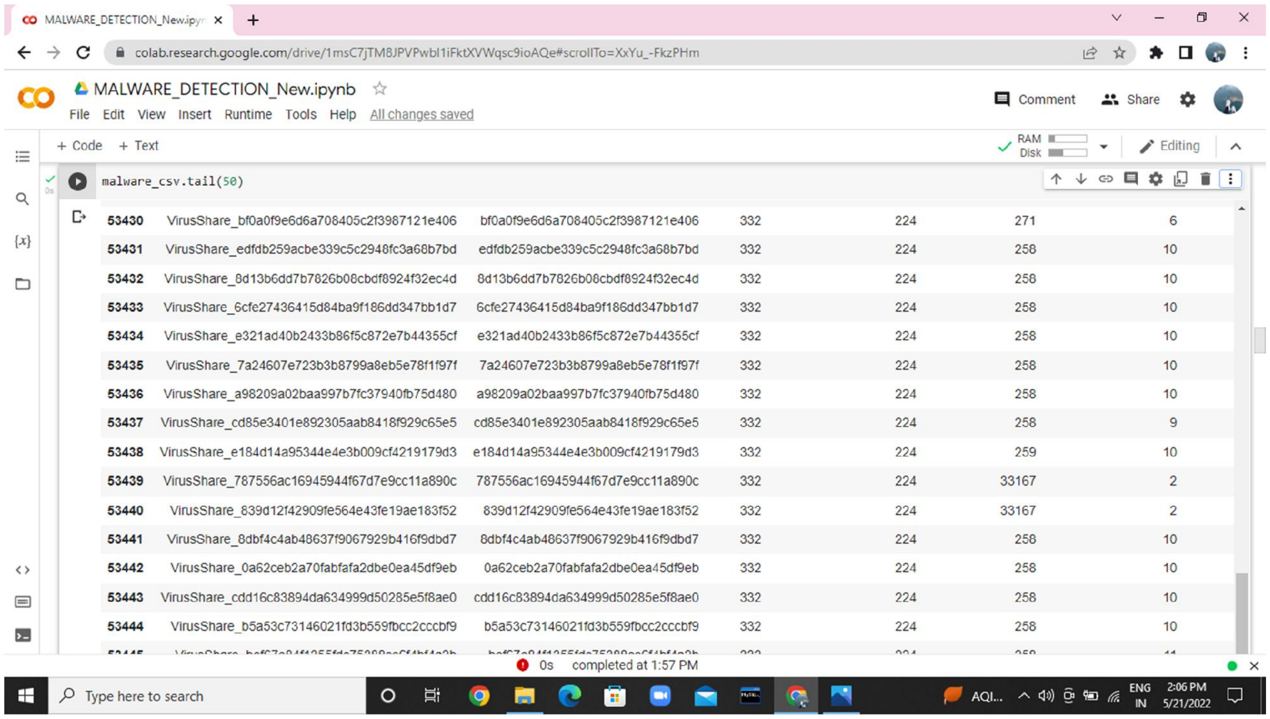
```
The no of samples are 41323 and no of features are 56 for legitimate part
The no of samples are 96724 and no of features are 56 for malware part
```


MALWARE_DETECTION_New.ipynb
colab.research.google.com/drive/1msC7jTM8JPVPwbl1iFktXVWqsc9ioAQe#scrollTo=0U5oMN8uq4-B
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
Comment Share Editing

```
malware_csv.head(40)
```

11	Eula.exe	1556a34d117a80bdc85a66d8ea4fbcf2	332	224	290	9	0	53248
12	LogTransport2.exe	c4005b63df77068bce158ac8ef7c522b	332	224	258	9	0	206848
13	reader_sl.exe	e595f220ed529885d8bc0ef42e455e4d	332	224	259	9	0	14848
14	AcrobatUpdater.exe	0e9dee95fdf47d6195da804a0deeda5b	332	224	258	9	0	178688
15	AdobeARM.exe	47c1de0a890613ffcff1d67648eedf90	332	224	258	9	0	413184
16	armsvc.exe	11a52cf7b265631deeb24c6149309eff	332	224	258	9	0	37376
17	ReaderUpdater.exe	5ed9b78b308d302c702d44f4505b3f46	332	224	258	9	0	178688
18	Adobe AIR Application Installer.exe	2da20164a6912ca8a11bb3089d0f3453	332	224	258	9	0	32256
19	Adobe AIR Updater.exe	397ef02798d24bf192997b5f7d8ed8ca	332	224	258	9	0	31744
20	template.exe	86fbdb3c4793f2b2e85bccc000fafbb7	332	224	258	9	0	32256
21	csscan.exe	04541302c404ec763380481346cbe0fb	332	224	258	8	0	34304
22	dainstall.exe	444caa5fc97729ef7da6965bae11c184	332	224	258	8	0	37376
23	entvutil.exe	5d0df5cbffd37853e623565cd081e47b	332	224	258	8	0	13824
24	fwinfo.exe	c3879b7359dd566ed8cd6d65846ae1bc	332	224	258	8	0	82944
25	mcshield.exe	50182e471b44c7a0f63b46e2def08b0f	332	224	258	8	0	110592

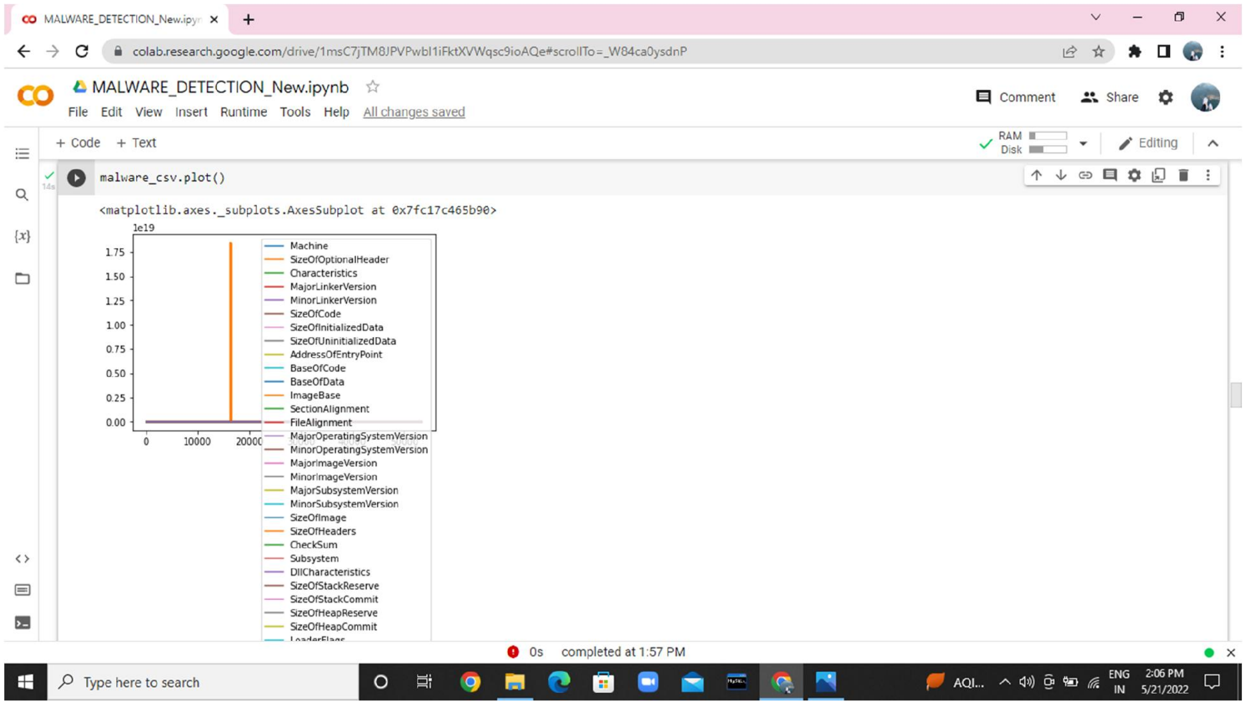
0s completed at 1:57 PM


MALWARE_DETECTION_New.ipynb
colab.research.google.com/drive/1msC7jTM8JPVPwbl1iFktXVWqsc9ioAQe#scrollTo=XxYu_-FkzPHm
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
Comment Share Editing

```
malware_csv.tail(50)
```

53430	VirusShare_bf0a0f9e6d6a708405c2f3987121e406	bf0a0f9e6d6a708405c2f3987121e406	332	224	271	6
53431	VirusShare_edfdb259acbe339c5c2948fc3a68b7bd	edfdb259acbe339c5c2948fc3a68b7bd	332	224	258	10
53432	VirusShare_8d13b6dd7b7826b08cbdf8924f32ec4d	8d13b6dd7b7826b08cbdf8924f32ec4d	332	224	258	10
53433	VirusShare_6cfe27436415d84ba9f186dd347bb1d7	6cfe27436415d84ba9f186dd347bb1d7	332	224	258	10
53434	VirusShare_e321ad40b2433b86f5c872e7b44355cf	e321ad40b2433b86f5c872e7b44355cf	332	224	258	10
53435	VirusShare_7a24607e723b3b8799a8eb5e78f1f97f	7a24607e723b3b8799a8eb5e78f1f97f	332	224	258	10
53436	VirusShare_a98209a02baa997b7fc37940fb75d480	a98209a02baa997b7fc37940fb75d480	332	224	258	10
53437	VirusShare_cd85e3401e892305aab8418f929c65e5	cd85e3401e892305aab8418f929c65e5	332	224	258	9
53438	VirusShare_e184d14a95344e4e3b009cf4219179d3	e184d14a95344e4e3b009cf4219179d3	332	224	259	10
53439	VirusShare_787556ac16945944f67d7e9cc11a890c	787556ac16945944f67d7e9cc11a890c	332	224	33167	2
53440	VirusShare_839d12f42909fe564e43fe19ae183f52	839d12f42909fe564e43fe19ae183f52	332	224	33167	2
53441	VirusShare_8dbf4c4ab48637f9067929b416f9dbd7	8dbf4c4ab48637f9067929b416f9dbd7	332	224	258	10
53442	VirusShare_0a62ceb2a70fabfafa2dbe0ea45df9eb	0a62ceb2a70fabfafa2dbe0ea45df9eb	332	224	258	10
53443	VirusShare_cdd16c83894da634999d50285e5f8ae0	cdd16c83894da634999d50285e5f8ae0	332	224	258	10
53444	VirusShare_b5a53c73146021fd3b559fbcc2cccbf9	b5a53c73146021fd3b559fbcc2cccbf9	332	224	258	10

0s completed at 1:57 PM



```
malware_csv.plot()

<matplotlib.axes._subplots.AxesSubplot at 0x7fc17c465b90>
```

VI. Advantages and Disadvantages

6.1 Advantages

- Detect polymorphic malwares.
- Can analyse patterns and prevent from similar attacks.
- Could distinguish between malicious and legitimate files.
- Functions as an early warning system for computer security.
- Detects unconceived type of malware attacks.

6.2 Disadvantages

- Algorithms need to be taught to analyse data patterns.
- ML displays a risk of running insufficient algorithms and making limited predictions.
- Sales number of threats aggressively rising so it's the amount of data analysed could built Robert defense against attack.

VII. Conclusion and Future Scope

This research intention is to detect malware using malware analysis techniques such as behavioral and dynamic analysis and various machine learning techniques such as navie bayes, random forest, decision tree is used to identify malware. So this project present some of existing machine learning algorithms directly applied implemented on data or datasets of malware. The Navie Bayes classifier is a probabilistic machine learning model. It uses bayes theorem to calculate the posterior probability of each class given a particular piece of feature. This paper also explains how algorithms will play a role in detecting malware with excessive accuracy and predictions.

Other data sets can be added to enhance accuracy, and more algorithms with higher performance can be added to enhance accuracy. It can be specified on the web for real-time reasoning of exe files in the cloud. One could attempt categorizing data into separate malware categories. One can further build a valid set, experiment with different methods. Evolution of a hybrid approach with two heads. Dynamic analysis will also be taken out, to use both automatic and boring methods, as well as a diversity of dynamic instruments. Naïve Bayes is a simple, but strong probability-based classifier that automatically does a few of the heavy-lifting for you. You can also utilize it to calculate some useful parameters for some other machine learning algorithms. Ada-boost is a set of tools to make marketing more predictive with insights from machine learning. It supports with marketing attribution, forecasting marketing spend & ROI, and showing marketing ROI over time

ACKNOWLEDGEMENTS

The completion of our project brings with it a sense of satisfaction, but it is never complete without those people who made it possible and whose constant support has crowned our efforts with success. One cannot even imagine our completion of the project without guidance and neither can we succeed without acknowledging it. It is a great pleasure that we acknowledge the enormous assistance and excellent co-operation to us by the respected personalities.

REFERENCES

[1]. W. Han, J. Xue, Y. Wang, L. Huang, Z. Kong, Mal DAE : Detecting and explaining malware based on correlation and fusion of static and dynamic characteristics, Computer Secure 83 (2019) 208–233, http://dx.doi.org/10.1016/j.cose.2019.02. 007.

[2]. P. Burnap, R. French, F. Turner, K. Jones, Malware classification using self organising feature maps and machine activity data, Computer Secure 73 (2017) 399–410, http://dx.doi.org/10.1016/j.cose.2017.11.016, http://linkinghub.elsevier.com/retrieve/pii/S0167404817302535.

[3]. A. Damodaran, F.D. Troia, C.A. Visaggio, T.H. Austin, M. Stamp, A comparison of static, dynamic, and hybrid analysis for malware detection, J. Comput. Virol. Hacking Tech. 13 (1) (2017) 1–24, http://dx.doi.org/10.1007/s11416- 015-0261-z.

[4]. E.M. Dovom, A. Azmoodeh, A. Dehghantanha, D.E. Newton, R.M. Parizi, H. Karimipour, Fuzzy pattern tree for edge malware detection and categorization in iot, J. Syst. Archit. 97 (March) (2019) 1–7, http://dx.doi.org/10.1016/j.sysarc. 2019.01.017.

[5]. M. Ficco, F. Palmieri, Leaf : An open-source cyber security training platform for realistic edge-iot scenarios, J. Syst. Archit. 97 (September 2018) (2019) 107–129, http://dx.doi.org/10.1016/j.sysarc.2019.04.004.