Malware Detection of Windows PE files with API Calls using Machine Learning and Deep Learning techniques: Evaluation Study

[Xolani Sibisi] (201606247)

[BSc. Hons Computer Science]

A mini-dissertation was submitted in partial fulfillment of the requirement for the degree of Honors in Computer Science

To

Department of Computer Science,

Faculty of Science and Agriculture

University of Zululand

KwaDlangezwa

RSA

Supervisor: Mr. P. Tarwireyi

November 2021

ABSTRACT

Cyber attacks have become a major threat to computer systems. These threats are designed to endanger low-to-high-level systems, from individuals to large corporations to government agencies. It keeps constantly evolving with regard to their creation technology, and they use advanced tactics for creating multiple existing instances, encrypt malicious payloads, and change the code structure each time it is infected while keeping the same functionality. Due to the polymorphism of malware today, the current solution has not yet adequately addressed this issue. Such solutions are primarily signature-based, and malware changes mean signature changes. In general, the ideal setting should have a detection rate of 100%. However, there is high false positive rate and false negative rate. Frequent updates to the signature-based system require more resources. Consequently, the goal of this research is studying a new approach that can detect new and invisible malware with reduced strong features which could perform the same as the one which used many features without affecting the accuracy and detection rate. In this paper, an effective malware detection model that differentiates goodware and malware executable files by using several strong features extracted from malicious and benign Portable Executable headers is used for the identification and classification of malware. Different techniques such as the CNN-LSTM and numerous machine learning algorithms consisting of KNearest Neighbors, Decision Tree, Gaussian Naive Bayes, Random Forest, Support Vector Machines, Extreme Gradient Boosting, and Logistic Regression has been used for identification and classification. Finally, we have used the model(s) Accuracy, F1 Score, Precision and Recall to compare the results observed. The acquired result can be used further for the improvement of various antivirus systems in detecting malware

Key terms--- Portable Executable (PE), Features, API calls, Malware, Benign

ACKNOWLEDGEMENT

I would like to thank Mr. Tarwireyi for guiding me as my supervisor. His commitment, feedback, and expertise shaped my research to a better level. His advanced technical guidance, helped me to successfully achieve my objectives. I am so grateful for his wisdom and instructions all along this Honours journey. I particularly thank my family for financially supporting my studies. The same level of gratitude goes to my cluster supervisors for also guiding me in cluster meetings. I am very grateful to my grandmother for her encouragement and consistent support during my academic journey, her love, care, and prayers. Finally, I want to extend my gratitude to God for his love, life, guidance, protection, and providence, by which this journey has been possible.

Table of Contents

Figures List

List of Tables

The Statement of Original Authorship

I Xolani Sibisi declare that this mini dissertation contains my work that has not previously been submitted for examination at an institution of higher education. This work was carried out under the supervision of Mr P Tarwireyi, it was submitted with a specialization degree at the Department of Computer Science, Faculty of Science and Agriculture, University of Zululand. The sources used in the essay have been acknowledged.

Signature: …X Sibisi………………………

Date: ……2021/12/05………………………….

Chapter 1 Introduction

1.1 Background of the Study

Malicious software, commonly known as malware represents hostile programs developed with the deliberate intent of compromising the security of computer systems, causing damage, or intentionally violating pertinent information privacy. Malicious software is a general term that comprises all types of hostile programs like rootkits, adware, bots, bugs, spyware, Trojan horses, viruses, and worms[1]. These malware undoubtedly cause an enormous security concern in today`s computing environment, where people`s lives are largely intertwined with digital devices[2]. Various devices such as personal computers, portable laptops, mobile tablets, etc., have undoubtedly gained incredible popularity when used for instantly accessing IT services. However, security in the use of such devices is of great concern because of frequent cyber attacks from malware[3].

According to Kaspersky Security Network, in the second quarter of 2021, Kaspersky solutions blocked 1,686,025,551 attacks from online resources around the world. Web Antivirus recognized 675,832,360 unique URLs as malicious. Malware execution attempts to steal funds from online banking accounts were blocked on the computers of 119,252 unique users. Ransomware attacks were detected on the computers of 97,451 unique users. Their antivirus file detected 68,294,298 malicious and potentially unwanted objects.

During Q2 2021, we detected 14 new ransomware families and 3,905 new modifications of this malware.

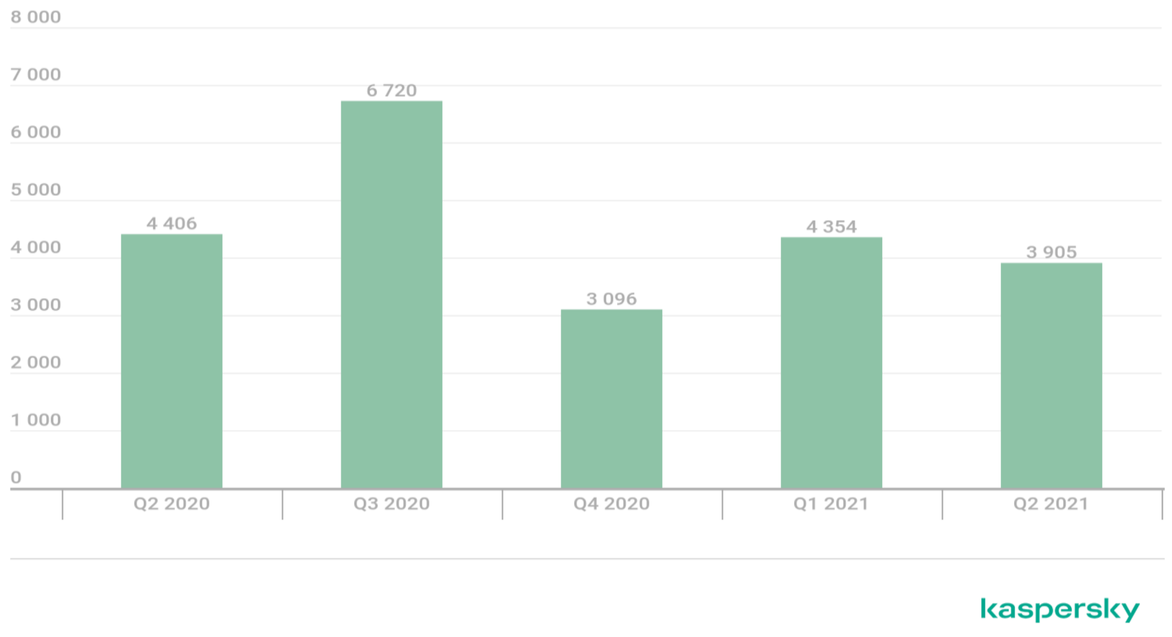


Figure 1: Number of new ransomware modifications, Q2 2020 — Q2 2021

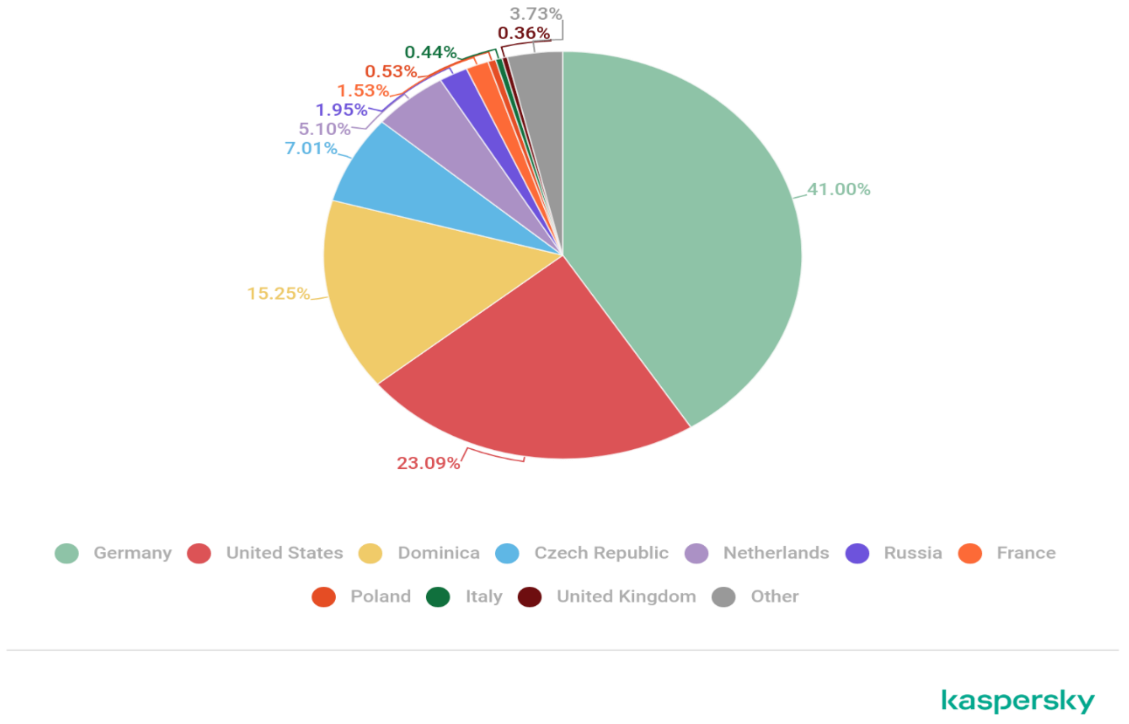


Figure 2: Distribution of web attack sources by country, Q2 2021

The original malware spread was via floppy disks and portable file transfers media. When a user opens contaminated files in a computer that is in good health, the system may be contaminated [9]. Subsequent malware is much more complex, spreading primarily from the Internet, can be embedded in various file types such as portable document files (pdf), portable, etc. Executable (PE) files, web formats, etc. [4], [5], [6], [7], [8]. Currently, there is malware more complicated and more sophisticated in terms of barbaric attacks and evasion detection [9], [10], [11], [12]. They can change their identities with a new attack without changing the main functionality of the malware. They can encrypt and decrypt user data using different keys [13], [14], [15] will result in an infinite number of new variants [16] to bypass and enhance the detection system as well as analytical process [17], [18], [19], [20]. They can also manipulate the key code renaming variables, replacing controls, inserting junk code, etc [21].

Various methods for malware detection had been proposed. The previously proposed detection techniques were based on static analysis. Static analysis checks the binary code, analyzes all possible execution paths, and identifies any malicious code that has not been executed. However, now it is difficult to analyze the binary code. While obfuscation techniques become increasingly complex, static analysis can be avoided by various obfuscation techniques, such as polymorphism, encryption, or packaging. Also, because the static analysis is based on a predefined signature database, new unknown malware cannot be easily detected before the signature is updated. In addition, certain possible execution path can only be explored after execution. To overcome these limitations of static analysis and supplement it, dynamic analysis has been proposed and widely used to achieve more effective malware detection. The techniques based on dynamic analysis executes malware and track its behavior. The two main methods in the dynamic analysis are control flow analysis and API call analysis. Both of these methods detect malware based on similarity analysis between new and known behaviors. However, malware authors try to circumvent these techniques by inserting meaningless code or shuffling program sequences

In our research, We provide technique to recognize the sequence of the malware semantic chain as well as benign sample API calls. By matching the collection of malware API calls and benign programs, we can predict whether any new sequences that emerge are malicious. Inspired by understanding context in Deep Learning, the API call sequence of any application type of factors that act for the calling program. Therefore, series of related API calls can be considered as an important representation of malware behavior. To test our idea, we used invisible test sets that contains new malware and benign samples. The results show the proposed work is superior to the previous work in using the sequence of API calls in the context of, which is important for timely malware detection.

1.2 Statement of Problem

Most Windows application are using API calls to perform tasks, so malware authors use API calls as a way to perform malicious actions. Our concern is that it is not necessary to detect feature-rich malware when it can be reduced to many powerful features that can do the same thing. To begin with the feature selection step for malware, it is necessary to examine the candidate mechanism or algorithm, as it will help with identifying and classifying new malware by picking up several strong features that can be used to investigate Windows PE files, malicious or benign with API calls without affecting the accuracy and detection rate of the algorithm.

1.3 Aim and Objectives

1.3.1 Research Goal

The goal of this research is to identify and classify Windows portable executable files as malicious or benign using API calls sequence by also taking into consideration the computational complexity involved with regards to identification and classification without affecting the accuracy.

1.3.2 Research Questions, Objectives, How Achieved

Research Question	Objectives	How achieve
1. What is the current state-of-the-art regarding malware detection?	To investigate the current state-of-the-art in malware analysis of PE using machine learning algorithms.	Literature review
2. How can we identify and classify Windows PE files as malicious or benign?	To investigate using deep and machine learning techniques which strong features can be used for identification and classification of Windows PE files.	Using Machine learning techniques and deep learning techniques to train a model
3. How do various machine learning algorithms perform with respect to detection of PE files using API calls?	Compare malware analysis models to see which provides a better detection accuracy.	By using the box and whisker diagram to compare performance on accuracy, ROC curves, False alarm rate
4. What recommendations can I make for a suitable algorithm?	To provide recommendation to suitable algorithm.	After evaluating all the algorithms in my study

Table 1: Research Questions, Objectives, How Achieved

1.4 Significance of the Research

Many security threats occur every day. These threats are primarily malware from the internet. They affect the physical safety of people and their valuable goods, including data and devices. This study aims to identify problems related to polymorphic malware and propose effective solutions for classifying and detecting malware by using several strong features without affecting the accuracy. The solution will help in the proper development of the antivirus sector in developing adequately powerful security applications. This work will continue advanced research in the cybersecurity research community.

1.5 Contributions

Specific contributions are:

1. The research designed a model useful for identifying and classifying Windows portable executables files as malicious or benign using API calls sequence by also taking into consideration the computational complexity involved without affecting the model accuracy. The main advantages of this model are as follows:

a) It provides a unique way of computing feature importances

b) It selects the most discriminating features

c) It reduces overfitting on both training and testing sets

d) It improves modeling accuracy.

e) It minimizes the training time

2. The research proposed a supervised learning model for improved classification performance of polymorphic malware.

3. The research proposed a deep learning model and machine learning models for earlier detection of polymorphic malware.

4. The research provided a thorough discussion on security breach techniques, attack mechanisms and advanced obfuscations techniques.

5. The research provided a comprehensive analysis of polymorphic malware and their evolution, as well as an extensive comparison of gaps, and strengths, and weaknesses of existing techniques.

6. The research provided a comprehensive analysis and discussion on the performance metrics and the bias that exists in some research conclusions.

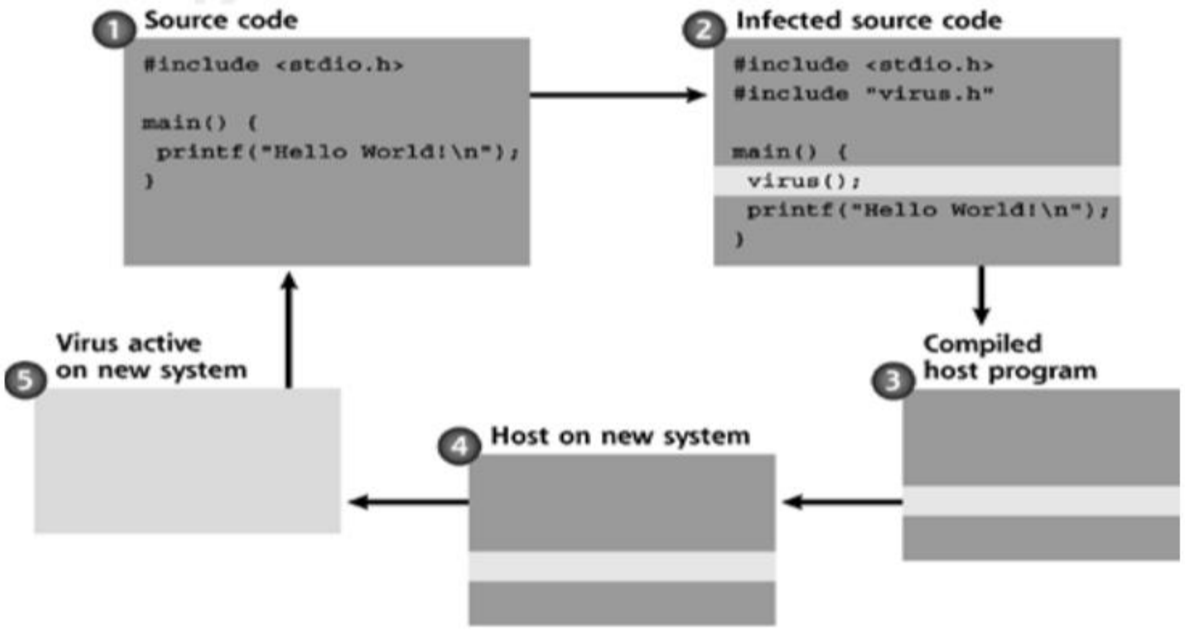
1.6 Organization of document

This thesis has the following structure.

- Chapter 2: This chapter provides a comprehensive overview of the literature on methods for analyzing and detecting polymorphic malware. Learn more about the structure of malware and its polymorphic technologies. We provide an overview of existing techniques that allegedly addressed this issue and identified relevant gaps in the literature.
- Chapter 3: This chapter introduces the methodology for this work. The data set details and possible solutions are shown.
- Chapter 4: This chapter, we analyze and discuss the results which were found
- Chapter 5: Finally, in this chapter, we succinctly summarize the results of this work and discuss our key findings. We present concluding remarks and make recommendations for future research.

2.1 Malware in Computer Systems

Malware is a special type of destructive malicious code. It's like the other normal software. However, there are malicious intents like denial of service and attempts. Possible confidentiality, information theft, or misuse of data integrity[21]. To reasonably achieve its main goals, malware can perform a variety of actions, including file operations (open, read, delete, modify, move), registry operations (open, create, delete, modify, move, query, close), service operations (open, start, create, delete, modify), mutex (Create, Delete), Process activity (Start, Stop), Runtime DLL, Network activity (TCP, UDP, DNS, HTTP), etc. The types of malware can range from simple to more complex polymorphic malware. Malware can self-execute or be controlled locally or remotely over the internet. Malware families include spyware, adware, trapdoors, Trojan horses, sniffers, spam, botnets, logic bombs, worms, viruses, keyloggers, ransomware, backdoors, etc. Adware, spyware, etc. Figure 2.1 below shows an example of a simple virus infection.



```
#include <stdio.h>

main() {
 printf("Hello World!\n");
}
```

```
#include <stdio.h>
#include "virus.h"

main() {
 virus();
 printf("Hello World!\n");
}
```

Figure 3: Infection by a code virus

2.2 Malware Types and Operation

- Virus: This is hidden in a harmless program, It can make multiple copies of itself. It can distribute these copies to files or other programs.
- Ransomware: This is a special type of malware that locks your PC screen with misleading information and frightens victims by pushing vigorously to pay the ransom fees as a condition for regaining access to the system.
- Trojan horse: This malware impersonates a harmless and passive program to persuade the alleged victim to install it. Once installed, it does some destructive action already hidden in the payload.
- Rootkit: This malware can evade detection by staying secretly on the system where it is installed. Hide the service from visibility in a list of processes running on the system.
- Backdoor: This malware can bypass the normal authentication mechanism by endangering network or internet connections. It creates a hidden process that makes it accessible to everyone when needed in the future.
- Adware: This malware places an unwanted advertisement to the user during the software installation process aimed at making money on behalf of the company of their author.
- Exploit: This type of malware exploits a specific vulnerability on the target system.
- Key loggers: This malware records keystrokes and uses them to steal sensitive information anything related to passwords, credit cards, or other login information. Stolen Information is passed to the author.
- Spyware: This malware steals user information without their knowledge and uses it to perform further malicious actions.
- Potentially Undesirable Programs: These are all programs that users can find not desirable. It can jeopardize the security of your system.
- Worms: Worms are built like viruses. It's a subclass of the virus. It can spread by itself from device to device without the help of a specific person.
- Advanced Persistence Threats (APT): This malware uses advanced technology to take advantage of the operating system weaknesses. They can be managed remotely, It can continue to attack specific targets.

- Crypto-Malware: Also known as Crypto-Ransomware or data locker to prevent access to data from target system. Using encryption, the user accesses the data and asks the user to pay for the encryption key.
- Locker Malware: Primarily designed to lock and force prevention access to the device interface. This makes little change to the data.
- Zeus: This malware is designed to steal bank information by collecting keystrokes recording and discontinuing the information entered in the form.
- Shadow brokers: This malware is designed to exploit system vulnerabilities. It can be monitored remotely. They use plugins to record webcams and microphones output, log keystrokes, and access other drives for monitoring purposes.

Malware can spread through many channels, including:

- Drive-by downloads: Unintended malware downloads from the internet.
- Unwanted email: Links or unknown attachments embedded in the email.
- Physical media: Malware that spreads through devices such as USB and other removable media.
- Self-propagation: Malware can spread on or off the network To another computer.

2.3 Malware Analysis Techniques

An analysis is a process that gives analysts a clear picture of the structure and functionality of the malware [22], [23]. Various key features are extracted during the analysis [24] and convey knowledge about how malware works. Unofficial category analysis includes vulnerability analysis [25], source code analysis [26], [25], and behavioral analysis [26], [27]. Similarity analysis [28], [29], [30], [31] are performed to confirm that: Malware is a variant of the existing one. Analytical tasks are very important and it is the first step before developing a reliable detection system. The two analytical methods, namely are static and dynamic analysis [26], [32].

2.3.1 Static analysis

Static analysis is the process of extracting information about malicious programs without executing them [33], [20], [4]. This makes static analysis safer than dynamic analysis as analysis for malware is not running [34]. It can be divided into two categories, such as basic static analysis and advanced static analysis. Basic static analysis is simple, fast. It provides basic information about malicious programs such as their version, files, etc. formats, suspicious imports, etc. Not very effective as it can easily fail important details. The advanced static analysis deals with code / structural analysis where the knowledge of assembly language, compiler code, and operating system concepts is mandatory. Malware functionality is analyzed by internal code Malware [19]. This analysis can provide information about the identity of the malware, passwords, libraries, URLs, programming languages, etc. [19], [35]. Function routine and mutants can be identified [36]. Code with advanced static analysis Disassemble and decompile [20]. However, static analysis cannot handle packing and obfuscation. Indicates the existence of packaging, but the binary file needs to be unzipped [7], [37], [38], [35] for the success of static analysis. In addition to what it was as explained above, the following information can also be revealed by static analysis[22], [26], [39], [40], [41]:

File fingerprinting: Cryptographic hash code (such as md5) computed to know if a binary has not changed.

Hash coded String Extraction: This process helps to extract human-readable strings embedded in the compiled binary file. This information concludes some features of binary files. The imported and exported functions are displayed.

File Format: Examine the file format metadata to find related information such as file type, file format, and compile time.

Antivirus scanning: If a binary file is previously known, it will be recognized by one or more files anti-malware program.

Packer detection: due to the obfuscation structure (encryption, compression, etc.) of that malware, suitable tools such as PEID and Detect it Easy (DIE) can identify packer and compiler information.

Disassembly: This process consists of reverse engineering machine code for assembly language that humans can understand. Tools like IDA Pro can be used. The generated assembly

language code helps analysts further investigate and collect the logic of the code and information about malware intent [42]. The tool OllyBdg is used for this process.
Static analysis has two basic advantages. first, the analysis process because there is no need to run malware. Second, it provides deeper information about the malware execution path. Further static analysis also has some drawbacks, including a lot of experience. It takes a long time and cannot process the packed files. There is also ambiguity in the analysis of malware using self-correction technology.

2.3.2 Dynamic analysis

Dynamic analysis is the process of analysing malicious program by first executing and monitoring its runtime functionalities [33], [7]. Malicious behavior is monitored and logged [22], [26], [27], [43], . During this process, the malware unzips itself and changes made to the system are also observed. Dynamic analysis is performed In a virtual environment to ensure maximum protection of the host machine [20], [44], [4], [19], [45]. Basic dynamic analysis consists of observing basic behavior analysis of malware such as process creation, file activity, and registration activity. On the other hand, in advanced dynamic analysis do a thorough investigation of the internal status of the running malware program. Uses advanced debugging techniques take one step of malicious code and perform a detailed internal inspection to get more complete information about malicious behavior. The code will be analyzed at run time, all code hidden by packing is revealed [37], [41]. Identity of malware is dynamically identified. Function calls, parameter analysis and information flow are all visualized [19]. Dynamic link libraries (dlls), processes, and file activity Not covered [20], [46]. Dynamic analysis helps detect and pack polymorphic variants and obfuscation attributes [7], [38], [47]. Memory analysis [4]. The interactions between malware, file systems, processes, and networks are investigated [4]. However, dynamic analysis is computationally intensive and requires a lot of systems resources[48], [49].

2.4 Malware Detection Techniques

Many studies have been used to analyze the characteristics of malware. Malware analysis methods can be divided into static and dynamic analysis [50]–[54];. In static analysis, malware files are checked by scanning their executable binary files or codes without running malware.

Different from static analysis, dynamic analysis methods control the execution process of malware. Collect, observe and record malware behavior characteristics at runtime. The dynamic analysis method generally runs in a secure virtual environment called a sandbox. Common dynamic analysis sandboxes include Cuckoo Sandbox (Cuckoo Sandbox, 2019) and CW Sandbox (CWSandbox , 2019). The primary goal of the sandbox is to check the malicious behavior of the malware and prevent the malware from attacking the host system. Both static and dynamic analysis methods have their advantages and disadvantages. Compared with dynamic analysis, the main advantage of static analysis is that it does not have the overhead cost of running the program. However, static analysis methods have limitations in the lack of support for packaging and complex obfuscated code[54]; [55] Compared with static analysis, dynamic analysis can effectively analyze packaged and obfuscated malware. The reason is that the malware must unpack itself while running. Therefore, its original and malicious code will be loaded into the main memory. However, the main disadvantage of dynamic analysis is time and resource consumption. Malware samples must be analyzed separately, which leads to the limitations of dynamic analysis in commercial applications. In this research, we will focus on dynamic analysis methods. Specifically, we will focus on analyzing the sequence of API calls generated when the malware sample is executed. We believe that in terms of malware analysis, dynamic analysis methods are considered the most effective and accurate. The reason is that dynamic malware analysis captures feasible characteristics that reflect the actual behavior of the malware. The two methods static and dynamic methods can be used to extract API calls. In static methods[56]; [57] , the API is extracted from the portable executable (PE) header of the executable file. However, in the dynamic approach [56]; [58], the API is collected by looking at the running executable file. In our method, we use a sequence of API calls collected through a dynamic method.

[59] proposed an extensible method that relies on the API name and its input parameter characteristics as an alternative of traditional API calls to achieving high recognition accuracy.

From the technique they have used, these features are extracted to determine the impact on their ability to recognize and distinguish between malware and harmless programs. After feature extraction, a feature selection strategy is then used to save analysis time by minimizing the number of extracted features. Several classifier techniques and tenfold cross-training methods are used. The cross-validation method is useful for small data sets that do not have the data that is enough for training and testing. The disadvantage of the cross-validation method is that it only provides meaningful results when the training and test data are from the same population. Using this proposed technique, the accuracy and false alarm rates of the study reached 98.4% and less than 2%, respectively. Since the number of extracted features is the smallest, the accuracy rate is high, and the false alarm rate is low, this technology is suitable for industrial applications. Although the paper claims that a minimum unit of features can save analysis time, the researchers did not correctly evaluate the analysis time saved in the study. Therefore, it can be said that a false alarm rate of 2% or less does not reach the recommended false alarm rate below zero. The 98.4% accuracy rate is also lower than most behavior-based methods
[60], for the API call sequence using text and data mining, an alternative static analysis method for malware detection was used. Their procedures are as follows. Originally, text mining was performed by using CSMINING records for feature extraction, which contained a sequence of api calls. Subsequently, various data mining techniques, such as multi-layer perceptron (MLP), decision tree (DT), support vector machine (SVM) and data processing group method (GMDH), probabilistic neural network (PNN), and a type of support vector machine (OCSVM) Implementation and use. Multi-layer perceptron is a neural network-based model used to map input data sets to output. It can learn nonlinear models at instantaneous, but it is difficult to scale with large data sets. DT is a tree structure that uses a supervised method for classification; it is easy to implement, but it has scalability issues on larger data sets. Throughout the research process, the ten-fold cross-validation method was used to verify the technology. Therefore, it is found that SVM and OCSVM achieve 100% sensitivity. The study found that under unbalanced and balanced conditions, the sensitivity of feature selection

and SVM based on joint information are both 100%. Additionally, our study shows that oversampling increases the sensitivity and accuracy of OCSVM from 98.5% to 100%. These are the important findings of the research on this data set [61]. Although the study achieved 100% sensitivity, the data set used as part of the paper was too small to legitimize its results. Therefore, more data sets can be used to obtain better results. Additionally, does not pay attention to the time complexity of the proposed technology, which is essential for real-time malware detection.

[62]. He proposed a malware detection system. Their system used the list of file dependency files of the Windows prefetch file for normal and malicious Windows applications to be differentiated

[63]. Introduced a model capable of detecting eight kind of botnets malware using a Hybrid analysis combines static analysis and dynamic analysis. Malware detection model in Windows operating system executables. To detect unknown malware in Windows operating system, the authors proposed an active learning framework. Learning classifier of the SVM machine used in this frame.

[64], has lately present An Android malware detection system, wherein steady chains of the APK file (Android bundle kit) are in comparison to the steady chains of the malicious utility.

If you discover a contrast among steady chains, the consumer can be notified of the inflamed utility. The extracted key phrases and the utility.xml documents of the utility are in comparison to realize approximately the repute of the reality that it's far dangerous or benign. A version known as EnDroid become added relying at the dynamic analysis. The implementation of this version is primarily based totally on special forms of dynamic characteristics. Here, important characteristics are extracted that help to realize the risky behavior of the applications and the behavior of the system. After this stack is applied to implement the malware detection model. For fast and dynamic malware detection, a multi-filter combination framework was searched. This image is designed using additional filters to recognize characteristics of malware execution behavior. The proposed system has developed different types of hybrid algorithms using a combination of filtering methods such as fishing Fscore, vulnerability, and maximum relevance to the SVM envelope. An educational method has been suggested for minimal behavioral data logs for malware detection.

Model for improving cyber resilience based on mobile user behavior. It is designed to automatically scan, detect malware, and warn when malware is present on mobile devices. A new Android malware detection model is developed based on Blockchain technology. The model is designated as a framework for the Blockchain Consortium's effort to detect public chains shared by users and linked chains shared by test members. It targets the problem of detecting malicious code and extracts the corresponding tests on mobile devices.

For automatic malware detection on Android devices, a model has been introduced based on bee and deep learning calls of Android app bundles. Recommended frameworks automatically detect malware. In this image, the authors used the Android application's runtime behavior by collecting the frequency of the application's system calls.

[65] introduces dynamic scanning to detect malware using the executable API call.

The authors used classification methods to classify malware.

[66]. He proposed a method to analyze and then classify an unknown executable. The method is implemented with static and dynamic analysis approaches. Set of functions prepared by extracting binary code from the executable using static analysis.Behavior of executables analyzed during runtime execution is performed in a dynamic analysis. Therefore, machine learning classification methods are applied to classify and detect malware. In the malware analysis survey, it shows that many malware detection techniques have been suggested in the literature. Each technique has its importance and all techniques have advantages and disadvantages for detecting malware and other security breaches. Awan proposed a malware detection method using static and dynamic characteristics extracted from the executable file.

In, the authors presented work based on static malware analysis. According to the methodology, the authors used OPODODE NGR imaging technique, grayscale imaging technique and input functions for the process extraction. We design an efficient method to detect malware in the Windows operating system. Extract different types of functions from the executable. These features are used as input to various machine-based classifiers to classify malware executables. The impetus for the proposed work was the discovery of unknown malware in the executable.

2.5 Summary

This chapter gave us a background about this intended study and identified some of the gaps which were available from literature. The literature also showed the metrics that were used by researchers during their investigations. It also provide a guideline on how we can conduct this experiment going forward.

Chapter 3 METHODOLOGY

3.1 Project Setup

To set up the investigation, It is required to create a large set of executables. The dataset is a segment of our assessment of malware detection and classification using deep learning. It contains 42,797 malicious API call strings and 1,079 benign API call strings. Each API call sequence is made up of the first 100 consecutive non-repeating API calls associated with the parent process, taken from the "call" elements of the Cuckoo Sandbox report. Both sample executables are scanned through the virustotal online tool to facilitate the rapid detection of worms, viruses, Trojans and all types of malware.

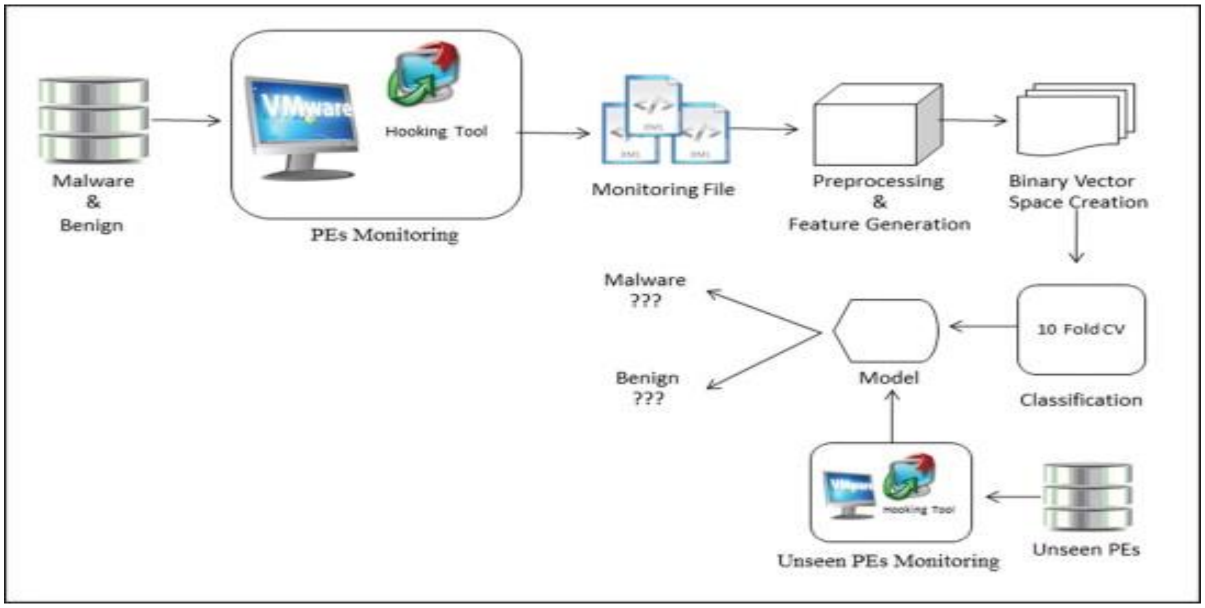


Figure 4: Classification and Detection architecture

3.2 Resource requirements

The main resources needed were malware samples and benign samples. The researcher obtained various samples from online repositories. Sample identified was due virus total. Duplicate samples were removed from the analysis dataset. Malware sample were saved in a compressed .zip folder containing credentials to prevent accidental accidents of any kind infection.

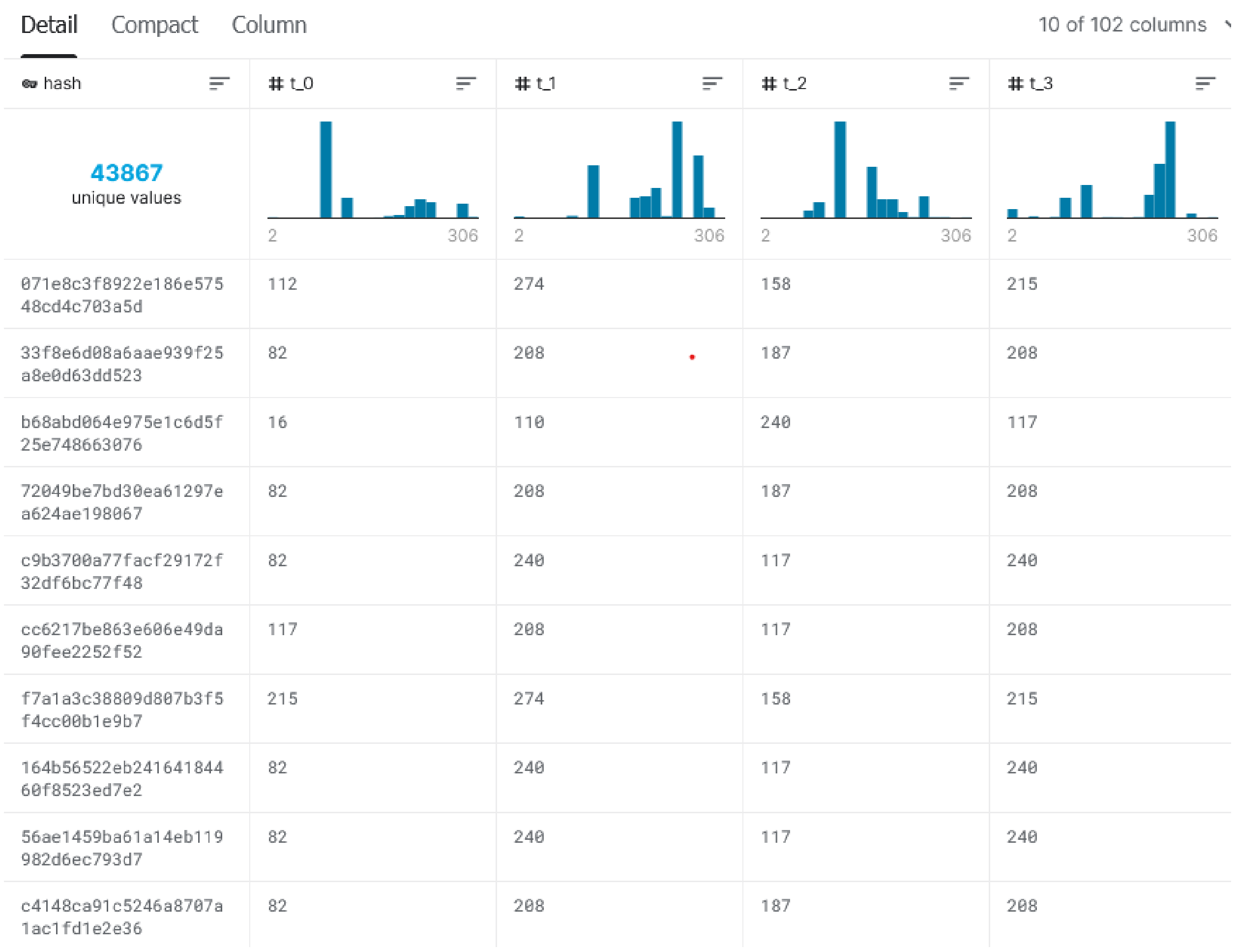
Resource	Description
Datasets	-Malware samples -Benign samples
Analysis computer	i5, 500GB, 16GB RAM
Operating systems	Ubuntu Linux, Windows 7-11
Cuckoo sandbox	Dynamic analyses
Python	For coding and implement all needed tasks
Additional Python modules(Scikitlearn, tensorflow, Pandas, Keras,…	For deep learning, machine learning, and analysis related tasks

Table 2: Resource requirements

3.3 Data Acquisition and preparation

3.3.1 Dataset

Dataset has been provided by Cuckoo Sandbox and was named “Malware Analysis Datasets: API Call Sequences”. This data contains polymorphic malware in the form of API calls. It contains both benign and malware samples.

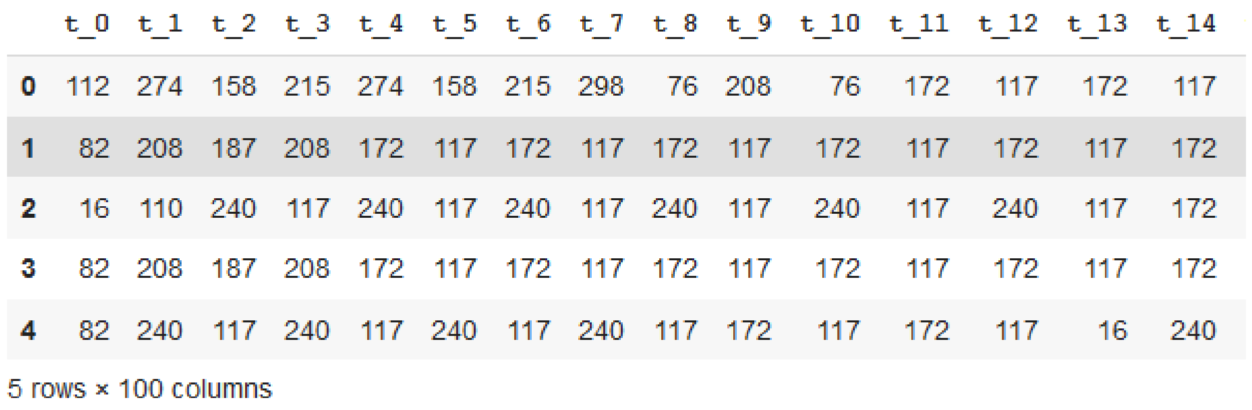


hash	t_0	t_1	t_2	t_3
43867 unique values	2 – 306	2 – 306	2 – 306	2 – 306
071e8c3f8922e186e57548cd4c703a5d	112	274	158	215
33f8e6d08a6aae939f25a8e0d63dd523	82	208	187	208
b68abd064e975e1c6d5f25e748663076	16	110	240	117
72049be7bd30ea61297ea624ae198067	82	208	187	208
c9b3700a77facf29172f32df6bc77f48	82	240	117	240
cc6217be863e606e49da90fee2252f52	117	208	117	208
f7a1a3c38809d807b3f5f4cc00b1e9b7	215	274	158	215
164b56522eb24164184460f8523ed7e2	82	240	117	240
56ae1459ba61a14eb119982d6ec793d7	82	240	117	240
c4148ca91c5246a8707a1ac1fd1e2e36	82	208	187	208

Table 3: Malware Analysis Datasets: API Call Sequences

3.3.2 Feature Selection

The purpose of feature selection is to simplify our model by including only the most relevant and important features. There are many ways to select the most important features. Choosing the most appropriate features will reduce the size of the original data set. Such a reduction, as a return from, would reflect the model variance, overfitting and of course the model reduction. calculation costs. Our dataset have feature selection techniques to remove the MD5 hash column and malware label column and left API calls which are important features for training and testing of our models.



	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_10	t_11	t_12	t_13	t_14
0	112	274	158	215	274	158	215	298	76	208	76	172	117	172	117
1	82	208	187	208	172	117	172	117	172	117	172	117	172	117	172
2	16	110	240	117	240	117	240	117	240	117	240	117	240	117	172
3	82	208	187	208	172	117	172	117	172	117	172	117	172	117	172
4	82	240	117	240	117	240	117	240	117	172	117	172	117	16	240

5 rows × 100 columns

Table 4: New feature selected dataset

3.4 Developing a Malware Detection Model

Signature-based systems use predefined patterns to detect existing malware. So when the malware converts itself, they fail. They have to wait for a new signature to be generated. The other is that the malware can change. indefinitely. It is The number of signatures cannot be unlimited and the database cannot store it with unlimited amount of data.

As the malware stock continues to grow, early detection is crucial for better protection. To meet the need for early detection, researcher have applied standard machine learning and very useful deep learning (DL) model in processing large data sets and provides high accuracy [67]. Based on the characteristics of NFE, the researcher implemented a deep learning model that learn what malware looks like to determine if an unknown program is malicious or harmless with high accuracy as soon as possible. The models are able to learn and detect malware in a very short time, less than a few milliseconds. This may lead to further actions before any damage is done. The researcher used a convolutional neural network and implemented it using a Python library called Keras with a Tensor thread running in the background.

3.5 Learning Models

3.5.1 Supervised Learning Models

KNN (K - Nearest Neighbor): It is one of the easiest algorithms which is for classification and regression. This a non parametric method, which is also known as the lazy learning model with local approximation. It uses data points that are most similar as a base to classify data points. The working of KNN can be briefly explained as:

○Select K as the number of neighbors
○Calculation of Euclidean distance of K number of neighbors
○Select nearest neighbors as the the calculations
○Among these, calculate the unit of data ends in each categories
○Assign newer data in the categories

Decision Tree: This type of algorithm uses supervised machine learning in which the data is split based on a parameter. Decision Nodes and Leaves are the two entities of a tree. Leaves are the last outcomes, whereas the resolution nodes are where the data is divided.

Random Forest: It an easy-to-use algorithm that gives good results even without hyper-parameter tuning. It is optimal for both regression and classification tasks because of its flexible and diverse nature. Built a "forest" that is set of decision trees formed by a method known as "bagging". It combines learning models that increases the results. The working of Random Forest can be briefly explained as:

○First random sample of the selected data set.
○A decision tree for each sample will be built.
○ The prediction result of each decision tree.
○Voting will be performed for every result that is predicted.
○Finally, we will select the most voted result as the final prediction result.

$$RFfi_i = \frac{\sum_{j \in all\ trees} normfi_{ij}}{T}$$

Gaussian Naïve Bayes: It is a group of supervised algorithms that use Bayes theorem for classification. It is an easy classification technique with high functionality. Gaussian Naïve Bayes uses Gaussian Normal Distribution. It is used for continuous data distribution. After calculating the probability for input values for each class using frequency we calculate mean and standard deviation from the training data.
○We must assume the data given to us is defined by gaussian distribution having no independent

dimensions nor co-variance.

○We can make the model suited by finding the mean and standard deviation of the points within from each marker.

○ At each data point, the distance between the Zscore distance and each class average is calculated i.e, The distance from the class mean divided by the standard deviation.

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

Support Vector Machine - Is a supervised machine learning algorithm that can be used for classification or regression problems. It transforms the data using a technique called kernel tricks, and based on those transformations, finds the best boundaries between possible outputs. Simply put, perform a very complex data transformation and decide how to separate the data based on the label or output you have defined..

Extreme Gradient Boosting - Is an implementation of a gradient-enhanced decision tree designed for speed and performance, which is the mainstream of competitive machine learning..

Logistic regression - Is a statistical model, the basic form of which uses logistic functions to model binary dependent variables, but with many more complex extensions. In regression analysis, logistic regression (or logistic regression) estimates the parameters of a logistic model (a form of binary regression). Mathematically, the binary logistic model has dependent variables with two possible values, such as pass / fail. This is represented by an indicator variable, which distinguishes between the two values. "0" and "1".

$$\ell = \log_b \frac{p}{1-p} = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$

3.5.2 Deep Learning Model

The LSTM model with long-short term memory is an improved version of the RNN model that overcomes backpropagation gradients and gradient explosion variances and is better suited for processing sequence data. Therefore, choose LSTM as your training model. Build a 3-layer LSTM as follows. Each LSTM layer is connected to a dropout layer. Then the LSTM layers join. Finally, a high density layer used as a classifier is attached. The inputs to the LSTM model are digital slices formed during the feature engineering operation. The model will eventually output the digital slice category.

3.6 Evaluation

The researcher used performance metrics to evaluate. Standard metrics have been used such as accuracy, recall, F-score, and confusion matrix. However, the accuracy is is not a good performance measure when the data set is unbalanced. This study focuses on problem of imbalance too.

Confusion matrix

Used to describe the performance of the classifier. The output is displayed in tabular format. The information provided in this matrix is as follows: TruePositive (TP), a unit of malware correctly identified as belonging to a particular category 0. True Negative (TN) is the number of malware correctly identified as Category 0. FalsePositive (FP) results represent the misclassified amount of Category 0 malware. False Negative (FN) is the amount of malware that was misclassified because it was not category 0.

True Positive Rate - It is defined as the number of truly positive results divided by the total number of malicious executables. $TPR = TP/TP + FN$

False Positive Rate - It is defined as the number of false positives divided by the total number of goodware executables. $FPR = FP/FP + TN$

Precision

It assess how frequently the If the malware reported by is correct, the model is correct Category 0. It is defined as follows: Precision = TP/(TP + FP)

Recall

It evaluate how often the model can correctly identify malware as category 0 . Also called sensitivity. It is defined as:

Recall = TP/(TP + FN)

F-score

Also known as the F1 score. This is a medium of harmony between memory and precision that describes the robustness and accuracy of the model. It is defined as:

F_measure = 2x(Precision Recall/Precision + Recall)

Accuracy

Accuracy is the fraction of predictions our model predicted correctly.

Accuracy = (TP + TN)/(TP + TN + FP + FN)

3.7 Summary

This chapter presented the methods which were used to conduct this research. The methods included literature survey, research tools, setup and parameters. This chapter also provide a detailed description about our data.

Chapter 4 RESULTS

4.1 Results presentation and analysis

4.1.1 Model Summary

This model aim to identify and classify polymorphic malware and also address the computational complexity challenges involved in identification and classification of PE files as malicious and benign with API calls. The model is trained using deep learning technique such as cnn-lstm and other various standard machine learning algorithms.

4.1.2 Count plot

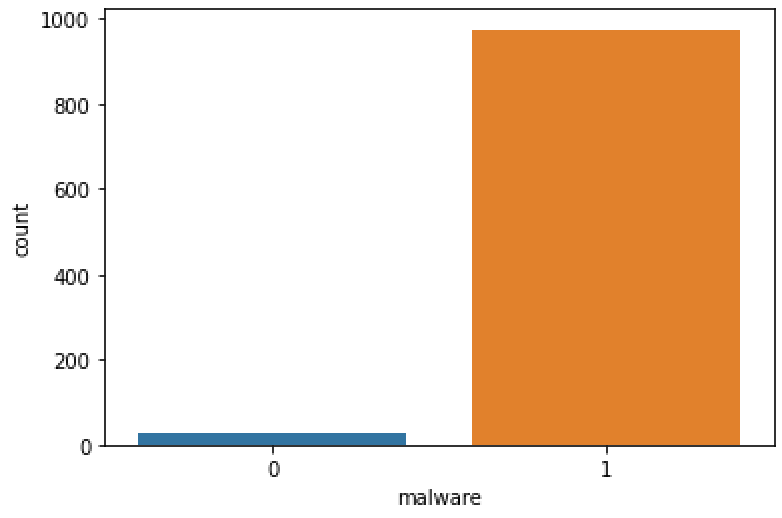
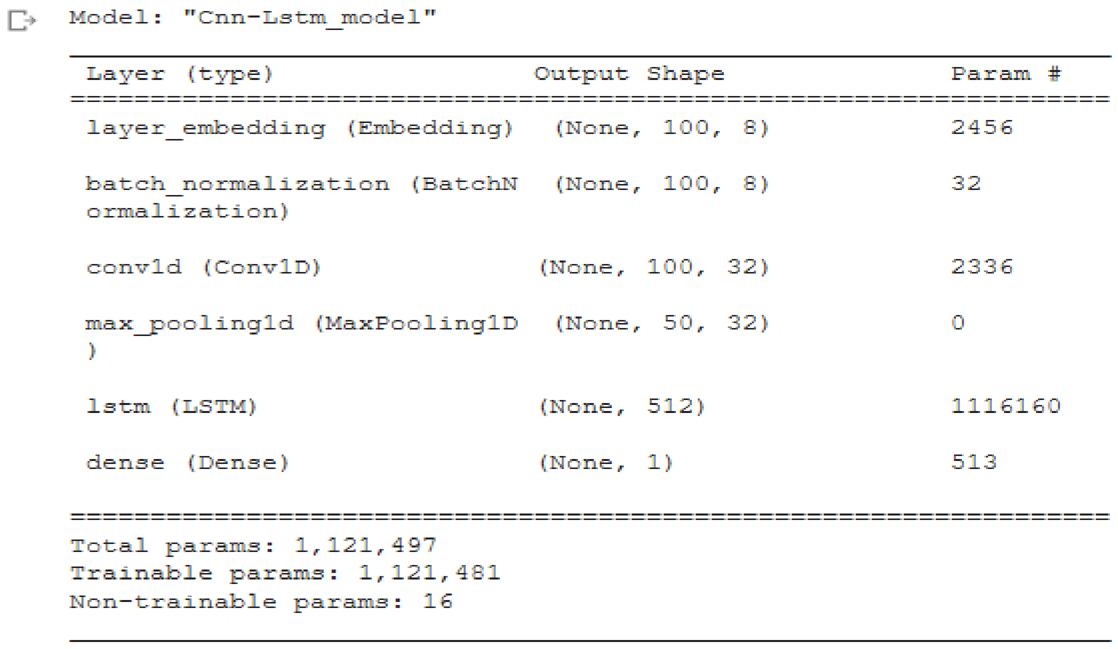


Figure 5: Sample Distribution

4.1.3 LSTM



```
Model: "Cnn-Lstm_model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 layer_embedding (Embedding)  (None, 100, 8)           2456

 batch_normalization (BatchN  (None, 100, 8)           32
 ormalization)

 conv1d (Conv1D)             (None, 100, 32)           2336

 max_pooling1d (MaxPooling1D  (None, 50, 32)           0
 )

 lstm (LSTM)                 (None, 512)               1116160

 dense (Dense)               (None, 1)                 513

=================================================================
Total params: 1,121,497
Trainable params: 1,121,481
Non-trainable params: 16
_________________________________________________________________
```

Table 5: Research Parameters

4.1.4 Loss and Accuracy graph

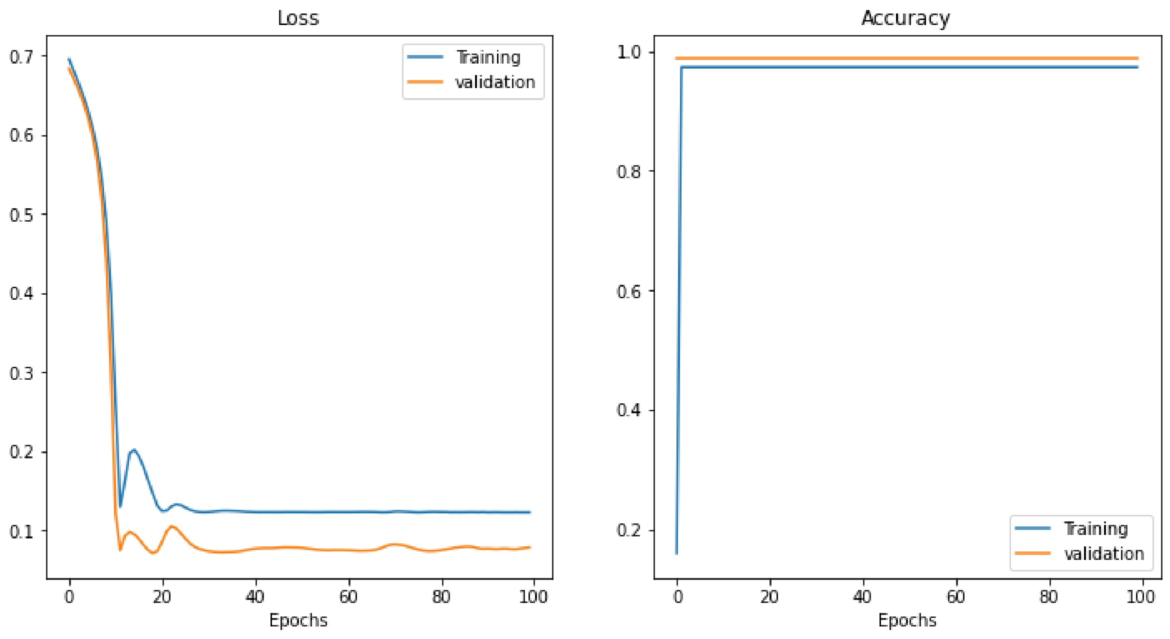


Figure 6: Loss and Accuracy for Training & Validation

4.1.5 ROC Curves

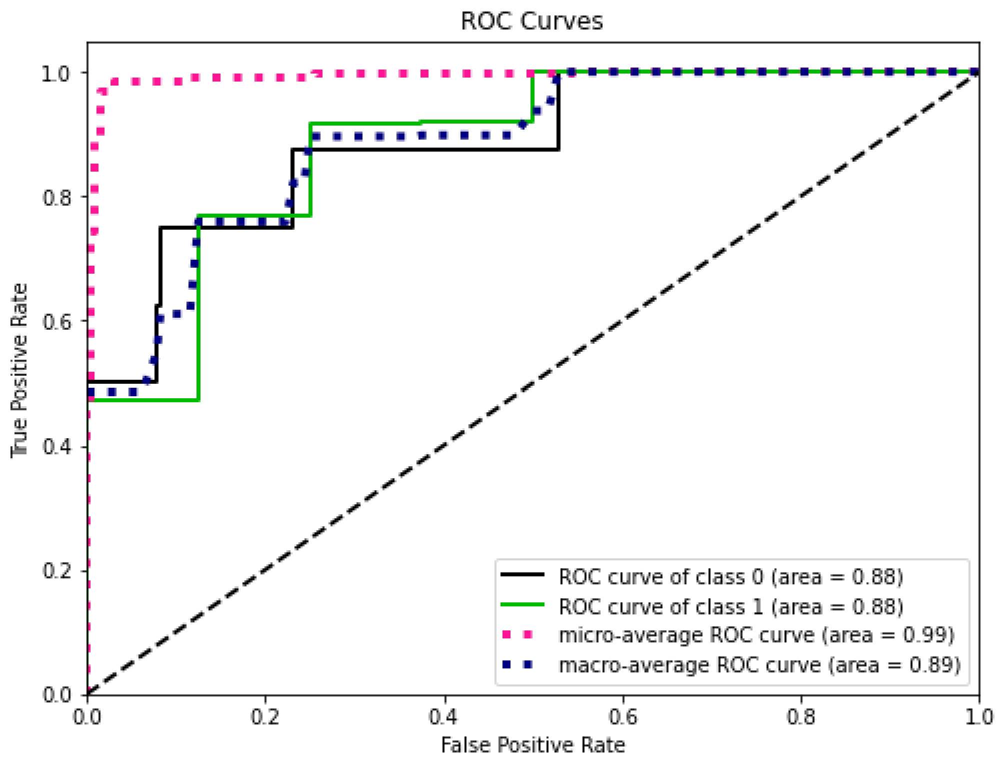


Figure 7: Malware Detection Rate

4.1.6 Confusion Matrix

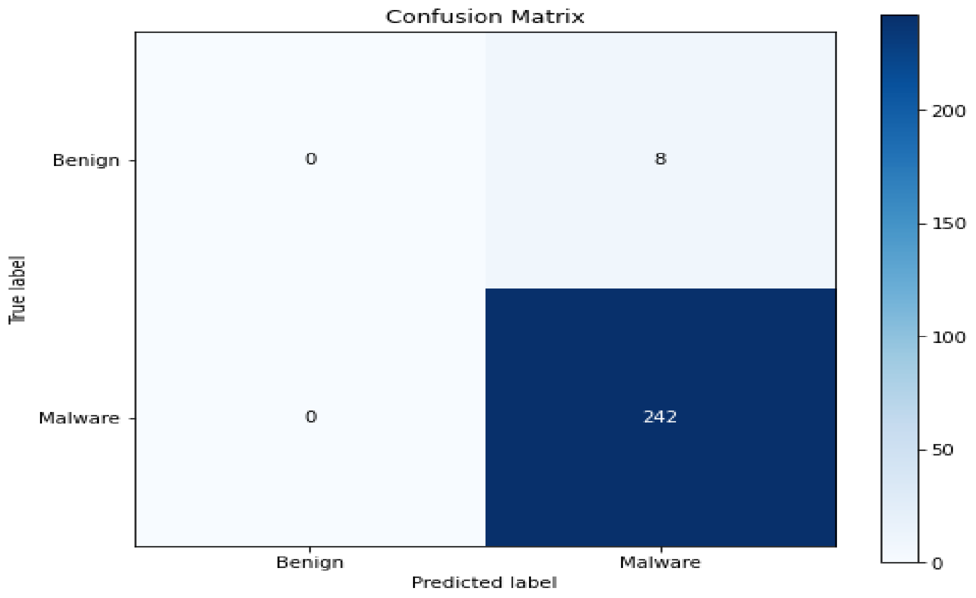


Figure 8: Detections made by the model during testin

4.1.7 Performance Metrics Tables for Benign and Malware Sample

Classifier	**Accuracy**	**Precision**	**Recall**	**f1-score**	**Overall Run time(s)**
Lstm	**97.87%**	**0.97**	**1.00**	**0.98**	**32 706.161**
KNN	**97.73%**	**0.98**	**1.00**	**0.99**	**0.049**
DT	**97.87%**	**0.98**	**1.00**	**0.99**	**0.104**
NB	**92.40%**	**0.98**	**0.94**	**0.96**	**0.152**
RF	**97.87%**	**0.97**	**1.00**	**0.98**	**0.351**
SVM	**98.53%**	**0.98**	**1.00**	**0.99**	**0.407**
XGB	**96.93%**	**0.98**	**1.00**	**0.99**	**0.459**
LR	**97.87%**	**0.98**	**0.99**	**0.99**	**0.590**

Table 6: Performance for Malware sample

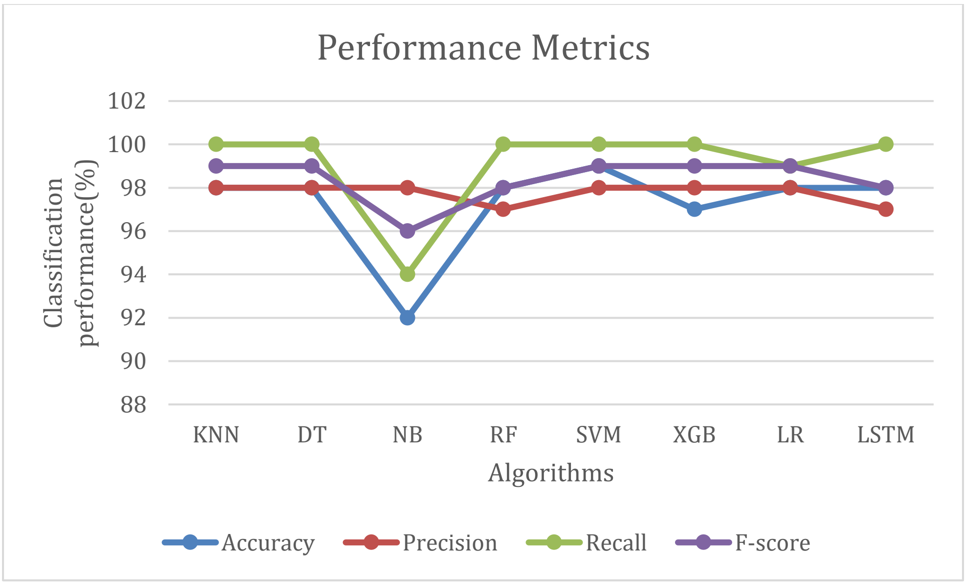
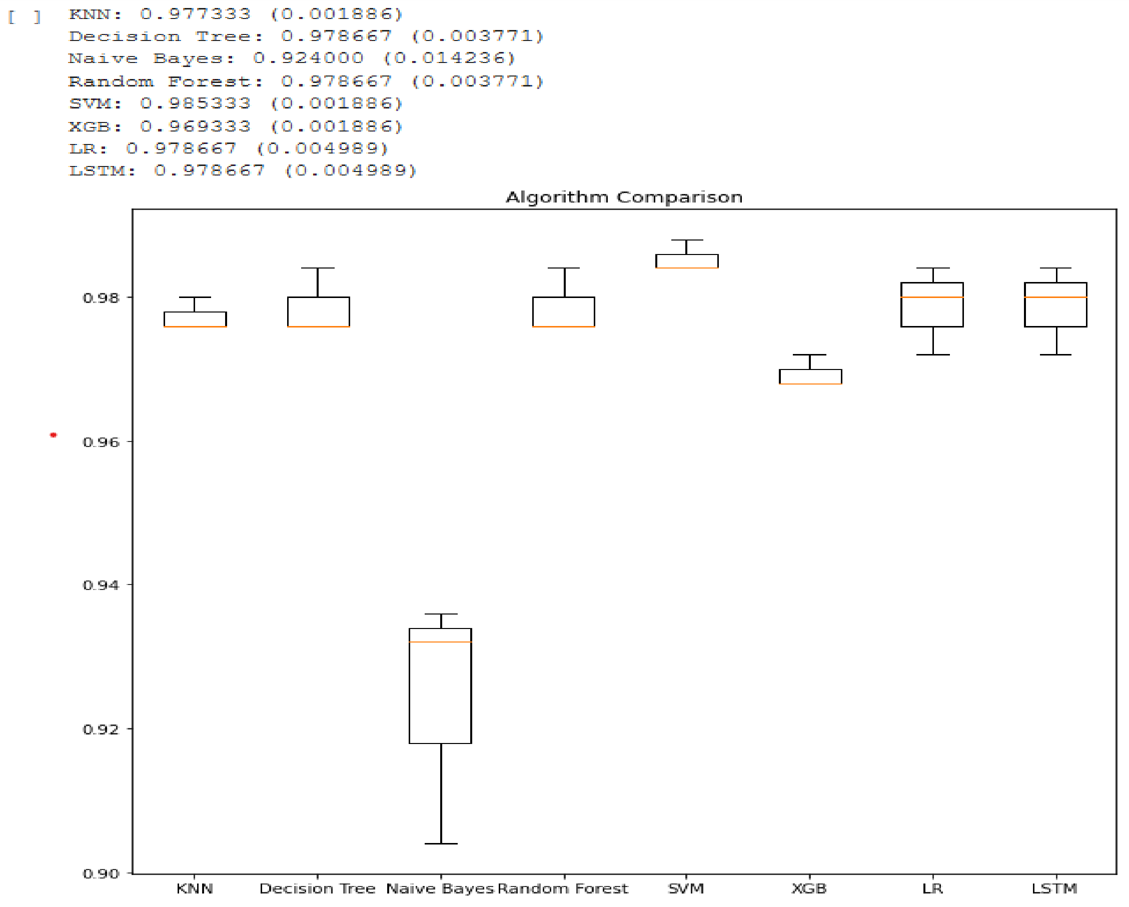


Figure 9: Comparison of performance metrics for algorithms

4.1.8 Box and Whisker Comparison



```
[ ]  KNN: 0.977333 (0.001886)
     Decision Tree: 0.978667 (0.003771)
     Naive Bayes: 0.924000 (0.014236)
     Random Forest: 0.978667 (0.003771)
     SVM: 0.985333 (0.001886)
     XGB: 0.969333 (0.001886)
     LR: 0.978667 (0.004989)
     LSTM: 0.978667 (0.004989)
```

Figure 10: Accuracy comparison

4.2 Comparative Analysis

To prove the effectiveness and efficiency of the proposed model, we tested the model with new examples of malware and goodware. The dataset contains 42,797 malware API call sequences and 1,079 harmless API call sequences. Each API call sequence is associated with the parent process and consists of the first 100 non-repeating, contiguous API calls extracted from the "Call" element of the Cuckoo Sandbox report. A comparison was performed for various merics such as the Accuracy, Precision, Recall, and F1-score. In cross validation almost all models achieved scores near 100% as shown in the performance tables above, when training with a large quantity of balanced data.

The results showed SVM model achieving the higher accuracy of 98.53% with 0.407s training time compared to other standard machine learning models which is also high compared to the LSTM model with an accuracy of 97.86% with 32706.161s. Further analysis reveals that the data comes from an API execution sequence with sequence characteristics. Our model is based on LSTMs and is very effective at processing data with sequential relationships. SVMs are classic traditional machine learning models, but their learning capabilities are higher than deep learning models like LSTMs. From the above analysis, we can see that the SVM model is suitable for this type of data.

Chapter 5 DISCUSSION, CONCLUSION AND RECOMMENDATION

4.1 Discussion

This study focuses on the issue of polymorphic malware. The first task was to explore, investigate, understand, and identify gaps in the literature. Result below are answers to the survey questions on page 12. We have found that much of the malware created daily is mostly variants of existing malware, mostly targeting the Windows operating system. We have found that accuracy is a key measure of performance used in many studies. Although this is a powerful evaluation metric, but it may not be enough because of the structure of the dataset. For example, if the dataset is unbalanced and bias to classes with majority representatives. To solve this problem, we have introduced one method to address the problem of imbalance, and have used other methods as well as performance metrics such as recall, precision, and F-score. The study also found that most of the existing methods are based on signatures. As discussed in detail in this study, such techniques detects polymorphic malware. The study also found that even the most popular antivirus products may not detects some basic malicious samples. For example, if you send a sample to VirusTotal, contact major antivirus engines to scan, analyze, and provide reports. Rarely do All engines that come in contact with the specified sample With malicious intent. It is even possible to achieve an identification rate of less than 50%, thereby Some reputable antivirus products have not been able to properly identify the sample. After discovering inconsistent malware classification and detection performance we have found that this is primarily due to the features used in the analysis representing a sample. In order to answer our research questions and accomplish our objectives it was crucial to use the several most relevant features as key to creating successful classification and detection model.

4.2 Limitations

However, we have also faced some limitations. The main one was getting malware dataset for analysis. Finding filtered samples and their variants was not easy. Samples provided by other researchers and online repositories are mixed of malware with their variants and sometimes benign samples not included. Some samples were inconsistent and unlabeled. Some authors of interesting articles failed to deliver those samples for reproducibility. Another limitation of this study is about model provisioning. Due to time limit the original purpose of the study was that the researchers did not deploy the model as part of the software. However, this will be used as a means of application in future work as means to apply large-scale discovery of this work.

4.3 Conclusion

In this study, various techniques in terms of classifying and detecting malware were proposed. A good malware classification and detection system needs to be able to classify even polymorphic variants that can come in any shape. Machine learning and deep learning works well when building these models. However, for any algorithm to work best, a well-balanced, highly preprocessed, well-presented dataset is required. This can be achieved through choosing strong relevant features. The achieved performances in our experiments showed a SVM having a higher accuracy of 98.53% and 0.408s overall runtime compared to other algorithms such KNN, Naïve Bayes, Decision Tree, Random Forest, Extreme Gradient Boosting, Logistic Regression, and LSTM. The lowest being Naïve Bayes with an accuracy of 92.40% and 0.152s run time.

We believe if the techniques proposed in this study are implemented in classification and detection of malware and end-to-end protection applications, especially polymorphic ones can be greatly improved.

4.4 Recommendation

1. For this kind of dataset future research should focus on optimizing the SVM classifer which proved to be the best in terms of classifying and detecting polymorphic malware for better protection using dynamic analysis method in a sandbox.
2. Future research needs to consider what the next generation of antivirus (NGA) will look like. It can be sandboxed and equipped with continuous analysis and learning mechanisms.

BIBLIOGRAPHY

[1] R. Pasha, Y. Prathima, and L. Thirupati, “Malwise System for Packed and Polymorphic Malware,” *Int. J. Adv. Trends Comput. Sci. Eng.*, vol. 3, no. 1, pp. 167–172, 2014.

[2] NATO, “Overcoming inevitable risks of electronic communication.”

[3] M. Kenney, “Cyber-Terrorism in a Post-Stuxnet World,” *Orbis*, vol. 59, no. 1, pp. 111–128, 2015, doi: 10.1016/j.orbis.2014.11.009.

[4] M. Vasilescu, L. Gheorghe, and N. Tapus, “Practical malware analysis based on sandboxing,” *Proc. - RoEduNet IEEE Int. Conf.*, 2014, doi: 10.1109/RoEduNet-RENAM.2014.6955304.

[5] D. Maiorca, I. Corona, and G. Giacinto, “Looking at the bag is not enough to find the bomb: An evasion of structural methods for malicious PDF files detection,” *ASIA CCS 2013 - Proc. 8th ACM SIGSAC Symp. Information, Comput. Commun. Secur.*, pp. 119–129, 2013, doi: 10.1145/2484313.2484327.

[6] Z. Tzermias, G. Sykiotakis, M. Polychronakis, and E. P. Markatos, “Combining static and dynamic analysis for the detection of malicious documents,” *Proc. 4th Work. Eur. Work. Syst. Secur. EUROSEC’11*, 2011, doi: 10.1145/1972551.1972555.

[7] S. Kumar, C. Rama Krishna, N. Aggarwal, R. Sehgal, and S. Chamotra, “Malicious data classification using structural information and behavioral specifications in executables,” *2014 Recent Adv. Eng. Comput. Sci. RAECS 2014*, pp. 6–8, 2014, doi: 10.1109/RAECS.2014.6799525.

[8] D. Liu, H. Wang, and A. Stavrou, “Detecting malicious javascript in PDF through document instrumentation,” *Proc. Int. Conf. Dependable Syst. Networks*, pp. 100–111, 2014, doi: 10.1109/DSN.2014.92.

[9] M. Guri, G. Kedma, T. Sela, B. Carmeli, A. Rosner, and Y. Elovici, “Noninvasive Detection of Anti-Forensic Malware,” pp. 1–10, 2013.

[10] A. Tamersoy, K. Roundy, and D. H. Chau, “Guilt by association: Large scale malware detection by mining file-relation graphs,” *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, pp. 1524–1533, 2014, doi: 10.1145/2623330.2623342.

[11] R. Kaur and M. Singh, “Efficient hybrid technique for detecting zero-day polymorphic worms,” *Souvenir 2014 IEEE Int. Adv. Comput. Conf. IACC 2014*, no. June 2010, pp. 95–

100, 2014, doi: 10.1109/IAdCC.2014.6779301.

[12] M. Norouzi, A. Souri, and M. Samad Zamini, “A Data Mining Classification Approach for Behavioral Malware Detection,” *J. Comput. Networks Commun.*, vol. 2016, pp. 20–22, 2016, doi: 10.1155/2016/8069672.

[13] S. Paul and B. K. Mishra, “Honeypot based signature generation for defense against polymorphic worm attacks in networks,” *Proc. 2013 3rd IEEE Int. Adv. Comput. Conf. IACC 2013*, vol. 6, no. 4, pp. 159–163, 2013, doi: 10.1109/IAdCC.2013.6514213.

[14] L. S. Grini, “Feature Extraction and Static Analysis for Large-Scale Detection of Malware Types and Families,” 2015.

[15] B. J. Kumar, H. Naveen, B. P. Kumar, S. S. Sharma, and J. Villegas, “Logistic regression for polymorphic malware detection using ANOVA F-Test,” *Proc. 2017 Int. Conf. Innov. Information, Embed. Commun. Syst. ICIIECS 2017*, vol. 2018-Janua, pp. 1–5, 2018, doi: 10.1109/ICIIECS.2017.8275880.

[16] V. Naidu and A. Narayanan, “Using different substitution matrices in a string-matching technique for identifying viral polymorphic malware variants,” *2016 IEEE Congr. Evol. Comput. CEC 2016*, no. November, pp. 2903–2910, 2016, doi: 10.1109/CEC.2016.7744156.

[17] V. Naidu and A. Narayanan, “A syntactic approach for detecting viral polymorphic malware variants,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 9650, no. March, pp. 146–165, 2016, doi: 10.1007/978-3-319-31863-9_11.

[18] A. Sharma and S. K. Sahay, “Evolution and Detection of Polymorphic and Metamorphic Malwares: A Survey,” *Int. J. Comput. Appl.*, vol. 90, no. 2, pp. 7–11, 2014, doi: 10.5120/15544-4098.

[19] S. K. Pandey and B. M. Mehtre, “A lifecycle based approach for malware analysis,” *Proc. - 2014 4th Int. Conf. Commun. Syst. Netw. Technol. CSNT 2014*, pp. 767–771, 2014, doi: 10.1109/CSNT.2014.161.

[20] Y. Prayudi and S. Yusirwan, “The recognize of malware characteristics through static and dynamic analysis approach as an effort to prevent cybercrime activities,” *J. Theor. Appl. Inf. Technol.*, vol. 77, no. 3, pp. 438–445, 2015.

[21] I. A.Saeed, A. Selamat, and A. M. A. Abuagoub, “A Survey on Malware and Malware

Detection Systems," *Int. J. Comput. Appl.*, vol. 67, no. 16, pp. 25–31, 2013, doi: 10.5120/11480-7108.

[22] V. Patel and P. H. Bhathawala, "A Literature Review On Anti Virus And Its Analysis," *Think India*, vol. 22, no. 2, pp. 315–328, 2019, doi: 10.26643/think-india.v22i2.8732.

[23] H. Qiu and F. C. C. Osorio, "Static malware detection with Segmented Sandboxing," *Proc. 2013 8th Int. Conf. Malicious Unwanted Softw. "The Am. MALWARE 2013*, pp. 132–141, 2013, doi: 10.1109/MALWARE.2013.6703695.

[24] H. Panchariya and S. Bharkad, "Comparative Analysis of Feature Extraction Methods for Optic Disc Detection," *IOSR J. Comput. Eng.*, vol. 16, no. 3, pp. 49–54, 2014, doi: 10.9790/0661-16334954.

[25] L. Wang, Z. Li, Y. Chen, Z. J. Fu, and X. Li, "Thwarting zero-day polymorphic worms with network-level length-based signature generation," *IEEE/ACM Trans. Netw.*, vol. 18, no. 1, pp. 53–66, 2010, doi: 10.1109/TNET.2009.2020431.

[26] R. Kaur and M. Singh, "A survey on zero-day polymorphic worm detection techniques," *IEEE Commun. Surv. Tutorials*, vol. 16, no. 3, pp. 1520–1549, 2014, doi: 10.1109/SURV.2014.022714.00160.

[27] R. Tian, R. Islam, L. Batten, and S. Versteeg, "Differentiating malware from cleanware using behavioural analysis," *Proc. 5th IEEE Int. Conf. Malicious Unwanted Software, Malware 2010*, pp. 23–30, 2010, doi: 10.1109/MALWARE.2010.5665796.

[28] D. Uppal, V. Mehra, and V. Verma, "Basic survey on Malware Analysis, Tools and Techniques," *Int. J. Comput. Sci. Appl.*, vol. 4, no. 1, pp. 103–112, 2014, doi: 10.5121/ijcsa.2014.4110.

[29] U. Bayer, E. Kirda, and C. Kruegel, "Improving the efficiency of dynamic malware analysis," *Proc. ACM Symp. Appl. Comput.*, pp. 1871–1878, 2010, doi: 10.1145/1774088.1774484.

[30] M. Alazab, S. Venkataraman, and P. Watters, "Towards understanding malware behaviour by the extraction of API calls," *Proc. - 2nd Cybercrime Trust. Comput. Work. CTC 2010*, no. July 2009, pp. 52–59, 2010, doi: 10.1109/CTC.2010.8.

[31] M. A. I. Almarshad, M. M. Z. E. Mohammed, and A. S. K. Pathan, "Detecting zero-day polymorphic worms with jaccard similarity algorithm," *Int. J. Commun. Networks Inf. Secur.*, vol. 8, no. 3, pp. 203–214, 2016.

[32] A. K. Sahoo, K. S. Sahoo, and M. Tiwary, “Signature based malware detection for unstructured data in Hadoop,” *2014 Int. Conf. Adv. Electron. Comput. Commun. ICAECC 2014*, 2015, doi: 10.1109/ICAECC.2014.7002394.

[33] M. Ahmadi, A. Sami, H. Rahimi, and B. Yadegari, “Malware detection by behavioural sequential patterns,” *Comput. Fraud Secur.*, vol. 2013, no. 8, pp. 11–19, 2013, doi: 10.1016/S1361-3723(13)70072-1.

[34] U. Baldangombo, N. Jambaljav, and S. Horng, “a S Tatic M Alware D Etection S Ystem U Sing,” vol. 4, no. 4, pp. 113–126, 2013.

[35] A. Javed and M. Akhlaq, “On the approach of static feature extraction in trojans to combat against zero-day threats,” *2014 Int. Conf. IT Converg. Secur. ICITCS 2014,* 2014, doi: 10.1109/ICITCS.2014.7021794.

[36] J. U. Joo, I. Shin, and M. Kim, “Efficient methods to trigger adversarial behaviors from malware during virtual execution in sandbox,” *Int. J. Secur. its Appl.*, vol. 9, no. 1, pp. 369–376, 2015, doi: 10.14257/ijsia.2015.9.1.35.

[37] S. Cesare, Y. Xiang, and W. Zhou, “Malwise-an effective and efficient classification system for packed and polymorphic malware,” *IEEE Trans. Comput.*, vol. 62, no. 6, pp. 1193–1206, 2013, doi: 10.1109/TC.2012.65.

[38] M. A. M. Ali and M. A. Maarof, “Dynamic innate immune system model for malware detection,” *2013 Int. Conf. IT Converg. Secur. ICITCS 2013*, pp. 3–6, 2013, doi: 10.1109/ICITCS.2013.6717828.

[39] R. Islam, R. Tian, L. Batten, and S. Versteeg, “Classification of malware based on string and function feature selection,” *Proc. - 2nd Cybercrime Trust. Comput. Work. CTC 2010*, pp. 9–17, 2010, doi: 10.1109/CTC.2010.11.

[40] Y. Supriya, G. Kumar, D. Sowjanya, D. Yadav, and D. L. Kameshwari, “Malware detection techniques: A survey,” *PDGC 2020 - 2020 6th Int. Conf. Parallel, Distrib. Grid Comput.*, pp. 25–30, 2020, doi: 10.1109/PDGC50313.2020.9315764.

[41] J. Koret and E. Bachaalany, *The Antivirus Hacker’s Handbook*. 2015.

[42] H. Dornhackl, K. Kadletz, R. Luh, and P. Tavolato, “Malicious behavior patterns,” *Proc. - IEEE 8th Int. Symp. Serv. Oriented Syst. Eng. SOSE 2014*, pp. 384–389, 2014, doi: 10.1109/SOSE.2014.52.

[43] Z. Bazrafshan, H. Hashemi, S. Mehdi, H. Fard, and A. Hamzeh, “IKT 2013 - 2013 5th

Conference on Information and Knowledge Technology," *IKT 2013 - 2013 5th Conf. Inf. Knowl. Technol.*, pp. 113–120, 2013.

[44] A. Dhammi, "Behavior A Analysis of Malware Us sing," pp. 14–19, 2015.

[45] M. Egele, T. Scholte, E. Kirda, and C. Kruegel, "A survey on automated dynamic malware-analysis techniques and tools," *ACM Comput. Surv.*, vol. 44, no. 2, 2012, doi: 10.1145/2089125.2089126.

[46] S. YusirwanS, Y. Prayudi, and I. Riadi, "Implementation of Malware Analysis using Static and Dynamic Analysis Method," *Int. J. Comput. Appl.*, vol. 117, no. 6, pp. 11–15, 2015, doi: 10.5120/20557-2943.

[47] G. Liang, J. Pang, and C. Dai, "A Behavior-Based Malware Variant Classification Technique," *Int. J. Inf. Educ. Technol.*, vol. 6, no. 4, pp. 291–295, 2016, doi: 10.7763/ijiet.2016.v6.702.

[48] Q. Jiang, X. Zhao, and K. Huang, "A feature selection method for malware detection," *2011 IEEE Int. Conf. Inf. Autom. ICIA 2011*, no. June, pp. 890–895, 2011, doi: 10.1109/ICINFA.2011.5949122.

[49] J. Landage and M. Wankhade, "Malware and Malware Detection Techniques: A Survey," *Int. J. Eng. Res. Technol.*, vol. 2, no. 12, pp. 61–68, 2013, [Online]. Available: http://www.ijert.org/browse/volume-2-2013/december-2013-edition?download=6744%3Amalware-and-malware-detection-techniques--a-survey&start=10.

[50] S. Cesare, Y. Xiang, and W. Zhou, "Control flow-based malware variant detection," *IEEE Trans. Dependable Secur. Comput.*, vol. 11, no. 4, pp. 307–317, 2014, doi: 10.1109/TDSC.2013.40.

[51] H. S. Galal, Y. B. Mahdy, and M. A. Atiea, "Behavior-based features model for malware detection," *J. Comput. Virol. Hacking Tech.*, vol. 12, no. 2, pp. 59–67, 2016, doi: 10.1007/s11416-015-0244-0.

[52] Z. Salehi, A. Sami, and M. Ghiasi, "MAAR: Robust features to detect malicious activity based on API calls, their arguments and return values," *Eng. Appl. Artif. Intell.*, vol. 59, no. December 2016, pp. 93–102, 2017, doi: 10.1016/j.engappai.2016.12.016.

[53] T. Lee and J. Kwak, "Effective and Reliable Malware Group Classification for a Massive Malware Environment," *Int. J. Distrib. Sens. Networks*, vol. 2016, 2016, doi:

10.1155/2016/4601847.

[54] E. Amer and I. Zelinka, “An ensemble-based malware detection model using minimum feature set,” *Mendel*, vol. 25, no. 2, pp. 1–10, 2019, doi: 10.13164/mendel.2019.2.001.

[55] W. Han, J. Xue, Y. Wang, L. Huang, Z. Kong, and L. Mao, “MalDAE: Detecting and explaining malware based on correlation and fusion of static and dynamic characteristics,” *Comput. Secur.*, vol. 83, pp. 208–233, 2019, doi: 10.1016/j.cose.2019.02.007.

[56] Y. Ki, E. Kim, and H. K. Kim, “A novel approach to detect malware based on API call sequence analysis,” *Int. J. Distrib. Sens. Networks*, vol. 2015, 2015, doi: 10.1155/2015/659101.

[57] B. Ndibanje, K. H. Kim, Y. J. Kang, H. H. Kim, T. Y. Kim, and H. J. Lee, “Cross-method-based analysis and classification of malicious behavior by API calls extraction,” *Appl. Sci.*, vol. 9, no. 2, 2019, doi: 10.3390/app9020239.

[58] Y. Qiao, Y. Yang, L. Ji, and J. He, “Analyzing malware by abstracting the frequent itemsets in API call sequences,” *Proc. - 12th IEEE Int. Conf. Trust. Secur. Priv. Comput. Commun. Trust. 2013*, pp. 265–270, 2013, doi: 10.1109/TrustCom.2013.36.

[59] F. P. Documents, “(12) United States Patent,” vol. 2, no. 12, 2016.

[60] Z. Salehi, M. Ghiasi, and A. Sami, “A miner for malware detection based on API function calls and their arguments,” *AISP 2012 - 16th CSI Int. Symp. Artif. Intell. Signal Process.*, no. Aisp, pp. 563–568, 2012, doi: 10.1109/AISP.2012.6313810.

[61] G. G. Sundarkumar and V. Ravi, “Malware detection by text and data mining,” *2013 IEEE Int. Conf. Comput. Intell. Comput. Res. IEEE ICCIC 2013*, vol. 500057, pp. 1–6, 2013, doi: 10.1109/ICCIC.2013.6724229.

[62] B. Alsulami and S. Mancoridis, “Behavioral Malware Classification using Convolutional Recurrent Neural Networks,” *MALWARE 2018 - Proc. 2018 13th Int. Conf. Malicious Unwanted Softw.*, pp. 103–111, 2019, doi: 10.1109/MALWARE.2018.8659358.

[63] K. Sathya, J. Premalatha, and S. Suwathika, “Reinforcing Cyber World Security with Deep Learning Approaches,” *Proc. 2020 IEEE Int. Conf. Commun. Signal Process. ICCSP 2020*, no. Ml, pp. 766–769, 2020, doi: 10.1109/ICCSP48568.2020.9182067.

[64] Z. U. Rehman *et al.*, “Machine learning-assisted signature and heuristic-based detection of malwares in Android devices,” *Comput. Electr. Eng.*, vol. 69, pp. 828–841, 2018, doi: 10.1016/j.compeleceng.2017.11.028.

[65] S. A. R. Shah and B. Issac, “Performance comparison of intrusion detection systems and application of machine learning to Snort system,” *Futur. Gener. Comput. Syst.*, vol. 80, pp. 157–170, 2018, doi: 10.1016/j.future.2017.10.016.

[66] P. V. Shijo and A. Salim, “Integrated static and dynamic analysis for malware detection,” *Procedia Comput. Sci.*, vol. 46, no. Icict 2014, pp. 804–811, 2015, doi: 10.1016/j.procs.2015.02.149.

[67] X. Yuan, “PhD Forum: Deep Learning-Based Real-Time Malware Detection with Multi-Stage Analysis,” *2017 IEEE Int. Conf. Smart Comput. SMARTCOMP 2017*, pp. 2–3, 2017, doi: 10.1109/SMARTCOMP.2017.7946997.

https://securelist.com/it-threat-evolution-in-q2-2021-pc-statistics/103607/