Anti-Ghost

The Detection of File-Less Malware

By

Syed Ali Wahaj---Reg# 199868

Muhammad Bin Tariq---Reg# 201971

Supervisor

Asst. Prof. Waleed Bin Shahid

Submitted to the faculty of Department of Computer Software Engineering,
Military College of Signals, National University of Sciences and Technology,
in partial fulfillment for the requirements of B.E Degree in
Computer Software Engineering

July,2020

CERTIFICATE OF CORRECTIONS & APPROVAL

Certified that work contained in this thesis titled “Anti-Ghost” (The Detection of File-less Malware), carried out by Muhammad Bin Tariq and Syed Ali Wahaj under the supervision of Assistant Professor Waleed Bin Shahid for partial fulfillment of Degree of Bachelors of Software Engineering, in Military College of Signals, National University of Sciences and Technology during the academic year 2019-2020 is correct and approved.

Approved By

Supervisor

AP Waleed Bin Shahid

CS Dept, MCS

Dated: July 2020

DECLARATION

No portion of work presented in this thesis has been submitted in support of another award or qualification in either this institution or anywhere else.

Plagiarism Certificate (Turnitin Report)

This thesis has been checked for Plagiarism. Turnitin report endorsed by Supervisor is attached.

Signatures of Students

Syed Ali Wahaj---Reg# 199868

Muhammad Bin Tariq---Reg# 201971

Signature of Supervisor

ACKNOWLEDGEMENTS

All praises for Allah Almighty who gave us the strength to accomplish this mighty task despite numerous difficulties and hardships on our way.

We offer our utmost gratitude to our Supervisor Sir Waleed Bin Shahid for his constant guidance and encouragement. The project would not have been possible without his expert advice, unfailing patience, and invaluable efforts.

We revere the moral support and passionate encouragement of Sir Waleed Bin Shahid (Dept of CS, MCS-NUST) throughout the development of our project.

This research is dedicated to our parents, well-wishers, and most of all, to our supervisor, Sir Waleed Bin Shahid, without whose guidance, cooperation, and support, research on this scale would not be a possibility.

ABSTRACT

The field of information technology has observed considerable development over the last century. Slowly and steadily, computer technology has worked its way into every field imaginable. Now, the use of computer technology can be observed everywhere; from personal use, to use on an enterprise level, rather, all companies rely on a computer in one form or another. In that regard, the computer is considered the most important invention of the 20th century. The integration of computer technology into work, specifically, made life easier. Work became more efficient, as people could easily accomplish things that often took days in a matter of minutes. All in all, the computer proved to be a safe, secure, and efficient tool for work and entertainment alike.

This cyber cloud, however, emerged with a darker lining. Everything went into the cyber world, and thus, people began to think of ways to compromise that safety, and thus, the first malware emerged on the scene in 1986. Over the years, new ways to breach systems are being developed every day, the most recent being the use of file-less attack vectors. In response to the increasing amount of file-less malware all over the cyber world, demand for anti-malware systems has observed a considerable increase. Our project relates to said demand, providing a solution to the detection of file-less malware using behavioral analysis, and Machine Learning. Classification between malicious and benign files is conducted using API data, using SVM, Decision Trees, Naïve Bayes and KNN Algorithms.

Keywords: File-less Malware, Malware Detection, Anti-Ghost, Behavioral Analysis, Machine Learning, KNN, SVM, Naïve Bayes, Decision Tress.

Table of Contents

CERTIFICATE OF CORRECTIONS & APPROVAL ii

DECLARATION iii

Plagiarism Certificate (Turnitin Report) iv

ACKNOWLEDGEMENTS v

ABSTRACT vii

Table of Figures x

List of Tables xi

1. CHAPTER 1: INTRODUCTION 12

1.1. Overview: 12

1.2. Problem Statement: 13

1.3. Objectives: 14

1.4. Approach/Research Methodology 15

1.4.1. Collection of Malware Samples: 15

1.4.2. Collection of Benign Samples: 16

1.4.3. Malware Analysis: 16

1.4.4. Feature Extraction: 17

1.4.5. Machine Learning: 19

1.4.6. Program Development 20

1.5. Limitations: 20

1.6. Document Organization: 22

1.7. Document Conventions 23

2. CHAPTER 2: LITERATURE REVIEW 24

2.1. Project Domain: 24

2.2. Literature Review: 25

2.3. Characteristics of File-less Malware 30

2.4. Previously Available Anti-Malware Solutions 32

3. CHAPTER 3: TECHNOLOGICAL REQUIREMENTS 36

3.1. Hardware Requirements 36

3.2. Software Requirements 39
3.3. Operating System Associations 43
3.4. Working Environment Setup 43
4. CHAPTER 4: ANTI-GHOST DESIGN 45
4.1. Class Diagram: 45
4.2. Activity Diagram: 46
4.3. Object Diagram: 47
4.4. Use Case Diagram: 48
4.5. Sequence Diagram 52
4.6. GUI 53
4.6.1. Home Window 53
4.6.2. Scan Window 54
4.6.3. Settings Window 55
5. CHAPTER 5: DETECTION METHODOLOGY AND RESULTS 56
5.1. Results 59
5.1.1. Naïve Bayes Algorithm 59
5.1.2. K-Nearest Neighbor Algorithm (KNN) 60
5.1.3. Decision Tree Algorithm 61
5.1.4. SVM Algorithm 62
6. CONCLUSION 64
6.1. Future Recommendations 65
7. Bibliography: 66
APPENDIX-A 68
List of Abbreviations 68
Thesis Plagiarism Report: 69

Table of Figures

Figure 1: CSV for Successfully Run Malware 17

Figure 2: CSV for Malware that Failed to Run 18

Figure 3: CSV For Benign Files that Failed to Run..... 18

Figure 4: CSV for Successfully Run Benign Samples..... 19

Figure 5:The Django Based Cuckoo Web-Interface..... 44

Figure 6: Anti-ghost Class Diagram 45

Figure 7: Anti-Ghost Activity Diagram 46

Figure 8: Anti-Ghost Object Diagram..... 47

Figure 9: Anti-Ghost Use Case Diagram 48

Figure 10: Anti-Ghost Sequence Diagram 52

Figure 11: Anti-Ghost Home Window..... 53

Figure 12: Anti-Ghost Scan Window..... 54

Figure 13: Anti-Ghost Settings Window..... 55

Figure 13:Naive Bayes Algorithm with Bernoulli Classifier 59

Figure 14: Naive Bayes Algorithm With Multinomial Classifier..... 60

Figure 15: Naive Bayes Algorithm with Gaussian Classifier..... 60

Figure 16: KNN Algorithm 61

Figure 17: Decision Tree Algorithm..... 62

Figure 18: SVM Algorithm 63

List of Tables

Table 1: Typical Characteristics of File-less Malware 32

Table 2: Complete List of Hardware Used 37

Table 3: Complete List of Software Used for The Anti-ghost Project 43

Table 4: Use Case: Login 49

Table 5: Use Case: Scan 50

Table 6: Use Case: Manual Scan 50

Table 7: Use Case: Stop Scan 51

Table 8: Machine Learning Phases 58

Table 9: Accuracy by Algorithm Used 63

1. CHAPTER 1: INTRODUCTION

The following chapter provides a very comprehensive introduction to the project titled “Anti-Ghost”, enumerating on the project in the form of an overview, explaining the need for the project, the project objectives, an overall approach to the project, the limitations of the project, and an overview into how this document has been organized.

1.1. Overview:

Most over the shelf anti-malware solutions implement the use of static analysis methods that are inefficient against file-less malware. Said class of malware, as their name indicates are a class of malware that do not save a malicious instance or a file on to the system hard drive. Instead, file-less malware loads the malicious program directly into the system’s main memory, and further targets the windows’ legitimate services, integrating their own functionality into said services. Instances of malware that do so have been known to integrate themselves into the windows registry, svchost, and numerous other services to establish themselves on an infected system. In that respect, a number of distinct cyber-attacks have occurred since 2016 inclusive of the PETYA and NOT-PETYA cyber-attacks, or the spread of the ETERNALBLUE malware.

The anti-ghost project fundamentally aims at the detection of malware that employ the use of file-less attack vectors, namely, file-less malware. Towards that end, the project implements the use of behavioral analysis and Machine Learning. Along the scope of the project, i.e. the detection of file-less malware, the use of behavioral analysis is a necessity owing to the file-less nature of malware that make static analysis methods implemented by most over the counter anti-malware software inefficient due to the

unavailability of a malicious file on the system's secondary memory. Secondly, paving the way to an anti-malware solution based on behavioral analysis and Machine Learning involves the use of an adequate dataset based on specific malicious as well as benign samples that will, and has been built from scratch.

1.2. Problem Statement:

The computer is considered the most important invention of the 20th century. The integration of computer technology into work, specifically, made life easier. Work became more efficient, as people could easily accomplish things that often took days in a matter of minutes. All in all, the computer proved to be a safe, secure, and efficient tool for work and entertainment alike. Everything went into the cyber world, and thus, people began to think of ways to compromise that safety, and thus, the first malware emerged on the scene in 1986, i.e. the "Brain" Malware, the first malware to be recognized worldwide, that was the brainchild of a Pakistani programmer.

Over the years, as software platforms were made more and more secure to avert and prevent malicious activity, malware themselves became more and more efficient. New ways to breach systems are being developed every day, the most recent being the use of file-less attack vectors.

In response to the increasing amount of file-less malware all over the cyber world, we require a system capable of the detection of file-less malware, the motivation behind which is the development of a malware detection system indigenous to Pakistan. Furthermore, as the demand for anti-malware systems has observed a considerable increase, the project has considerable feasibility in terms of market. Our project relates to

the aforementioned demand, providing a solution to the detection of file-less malware using behavioral analysis, and Machine Learning.

1.3. Objectives:

The use of Dynamic memory analytics alongside machine learning provides us with a new way to detect malware. While the use of machine learning towards the detection of malicious activity on a computer system is new, it has gained considerable application over the years accounting for the emergence of file-less malware.

The scope of our project is fundamentally the detection of file-less malware. Towards that end, the project makes use of a custom dataset of malware and benign samples. Furthermore, the scope of the project also includes the development of a custom dataset owing to the unavailability of a public instance of said dataset. The project implements the use of behavioral analytics over the characteristics displayed by over 2000 malware samples. These characteristics are extracted in the form of features or API calls made.

Overall, the ultimate goal of our project is the development of a system that is capable of detection of File-less malware, particularly malware that implements the use of PowerShell scripts, as well as Microsoft Office VBA Macros.

The following are a summation of the cardinal objectives of Anti-Ghost:

- Dataset generation.
- Research on how to tackle the emergence of newer classes of malware.
- The development of a system of file-less malware detection.

- Assisting individuals and organizations to keep their systems malware-free.
- Detection of a majority of similar malware based on behavioral heuristics.
- Development of a light-user friendly program.
- Minimizing costs associated with the possibility of ransomware emergence and infections.
- An anti-malware system indigenous to Pakistan.

1.4. Approach/Research Methodology

The following section enumerates on the approach used towards the actualization of the project, i.e. the detection of file-less malware. The section places primary focus on the entire development method, from the initial dataset gathering process, to finalization of the entire project.

1.4.1. Collection of Malware Samples:

The first step along the development of Anti-Ghost involves the generation of a viable dataset based primarily on both malware and benign file samples. Towards that end, the attainment of benign file samples proved a simple task. However, the availability of live malware samples proved challenging. The primary source of malicious samples used is “Virus share”, a repository of live malicious samples. The site has numerous collections of malware that have individually been filtered to those involving the use of file-less methodologies, specifically those using PowerShell, and word Macros. The initial terabyte of malicious samples has been filtered down to 2000 live malware samples.

1.4.2. Collection of Benign Samples:

A sample equal of benign samples equal to the sample set of live malware samples had also been procured. The sample set of benign files includes benign VBA and PowerShell based files.

1.4.3. Malware Analysis:

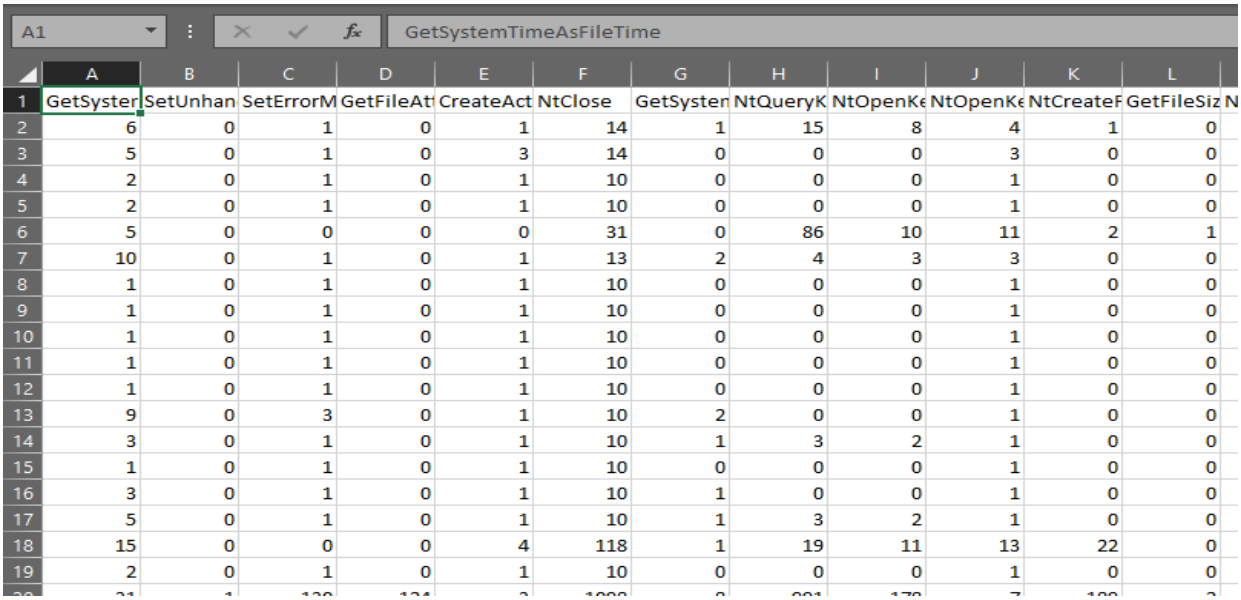
Once the sample sets of malicious and benign files had been gathered, the files were analyzed to assess features of both the live malware and the benign files. This has been accomplished using Cuckoo, the leading open-source, automated malware analysis tool available. Cuckoo allows for the analysis of malware to extract features such as API calls, files run, etc., essentially allowing us to monitor a file's behavior. Cuckoo also cross-validates results with popular anti-malware solutions. Each malware sample was run on a VMWare instance run inside the cuckoo sandbox. Cuckoo allows the malware to run on a sandboxed, windows-based VMWare instance, recording how the malware behaves. The behavior of the malware is recorded, and a report is generated based on the complete set of actions the malware performs. The same is also true for benign file samples.

Cuckoo essentially allows us to monitor how a file behaves when run and generates a report based on:

- System API calls.
- Files used, run, or downloaded and deleted.
- Network activity
- System memory dumps

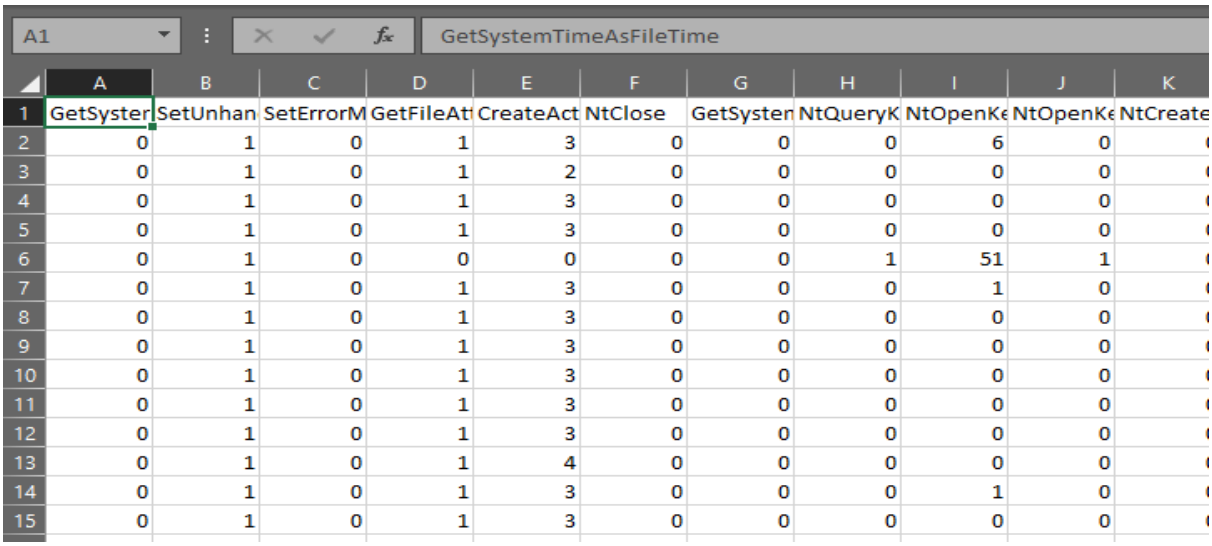
1.4.4. Feature Extraction:

Once all the malware samples and benign files have been analyzed, features are extracted from the generated reports. This includes two CSVs generated each, for both malicious files and benign files. One set of CSVs includes two files, where one file has API data on failed malware, and one has API data on successful entries. The same is also true for benign samples.



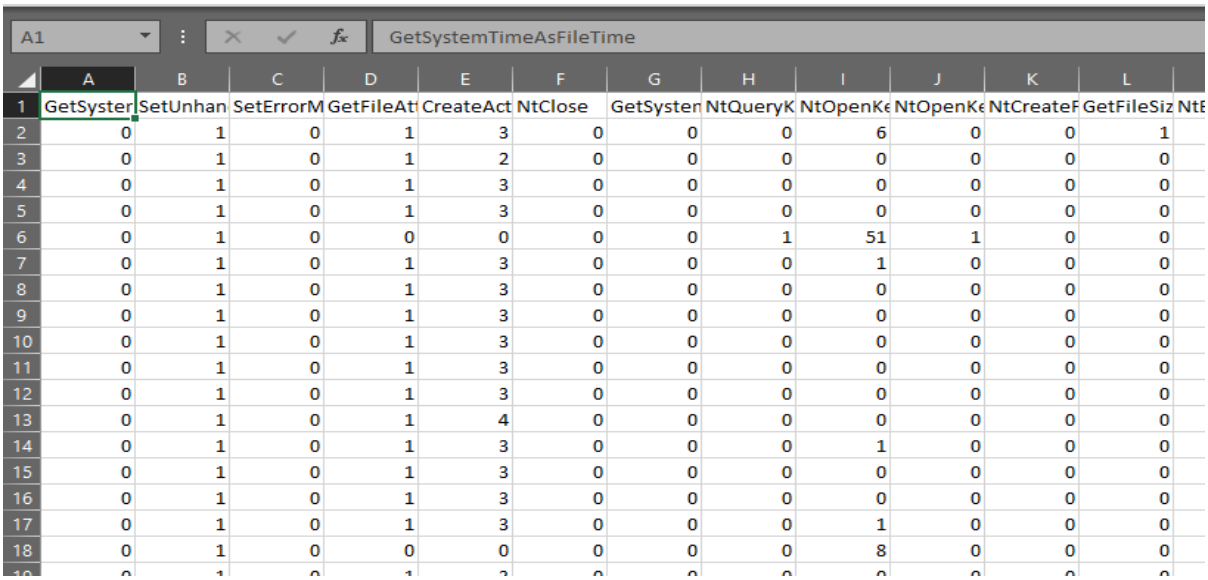
	A	B	C	D	E	F	G	H	I	J	K	L
1	GetSyster	SetUnhan	SetErrorM	GetFileAt	CreateAct	NtClose	GetSysten	NtQueryK	NtOpenKe	NtOpenKe	NtCreateF	GetFileSiz
2	6	0	1	0	1	14	1	15	8	4	1	0
3	5	0	1	0	3	14	0	0	0	3	0	0
4	2	0	1	0	1	10	0	0	0	1	0	0
5	2	0	1	0	1	10	0	0	0	1	0	0
6	5	0	0	0	0	31	0	86	10	11	2	1
7	10	0	1	0	1	13	2	4	3	3	0	0
8	1	0	1	0	1	10	0	0	0	1	0	0
9	1	0	1	0	1	10	0	0	0	1	0	0
10	1	0	1	0	1	10	0	0	0	1	0	0
11	1	0	1	0	1	10	0	0	0	1	0	0
12	1	0	1	0	1	10	0	0	0	1	0	0
13	9	0	3	0	1	10	2	0	0	1	0	0
14	3	0	1	0	1	10	1	3	2	1	0	0
15	1	0	1	0	1	10	0	0	0	1	0	0
16	3	0	1	0	1	10	1	0	0	1	0	0
17	5	0	1	0	1	10	1	3	2	1	0	0
18	15	0	0	0	4	118	1	19	11	13	22	0
19	2	0	1	0	1	10	0	0	0	1	0	0

Figure 1: *CSV for Successfully Run Malware*



	A	B	C	D	E	F	G	H	I	J	K
1	GetSyster	SetUnhan	SetErrorM	GetFileAt	CreateAct	NtClose	GetSysten	NtQueryK	NtOpenKe	NtOpenKe	NtCreate
2	0	1	0	1	3	0	0	0	6	0	
3	0	1	0	1	2	0	0	0	0	0	
4	0	1	0	1	3	0	0	0	0	0	
5	0	1	0	1	3	0	0	0	0	0	
6	0	1	0	0	0	0	0	1	51	1	
7	0	1	0	1	3	0	0	0	1	0	
8	0	1	0	1	3	0	0	0	0	0	
9	0	1	0	1	3	0	0	0	0	0	
10	0	1	0	1	3	0	0	0	0	0	
11	0	1	0	1	3	0	0	0	0	0	
12	0	1	0	1	3	0	0	0	0	0	
13	0	1	0	1	4	0	0	0	0	0	
14	0	1	0	1	3	0	0	0	1	0	
15	0	1	0	1	3	0	0	0	0	0	

Figure 2: *CSV for Malware that Failed to Run*



	A	B	C	D	E	F	G	H	I	J	K	L
1	GetSyster	SetUnhan	SetErrorM	GetFileAt	CreateAct	NtClose	GetSysten	NtQueryK	NtOpenKe	NtOpenKe	NtCreateF	GetFileSiz
2	0	1	0	1	3	0	0	0	6	0	0	1
3	0	1	0	1	2	0	0	0	0	0	0	0
4	0	1	0	1	3	0	0	0	0	0	0	0
5	0	1	0	1	3	0	0	0	0	0	0	0
6	0	1	0	0	0	0	0	1	51	1	0	0
7	0	1	0	1	3	0	0	0	1	0	0	0
8	0	1	0	1	3	0	0	0	0	0	0	0
9	0	1	0	1	3	0	0	0	0	0	0	0
10	0	1	0	1	3	0	0	0	0	0	0	0
11	0	1	0	1	3	0	0	0	0	0	0	0
12	0	1	0	1	3	0	0	0	0	0	0	0
13	0	1	0	1	4	0	0	0	0	0	0	0
14	0	1	0	1	3	0	0	0	1	0	0	0
15	0	1	0	1	3	0	0	0	0	0	0	0
16	0	1	0	1	3	0	0	0	0	0	0	0
17	0	1	0	1	3	0	0	0	1	0	0	0
18	0	1	0	0	0	0	0	0	8	0	0	0

Figure 3: *CSV For Benign Files that Failed to Run*

	A	B	C	D	E	F	G	H	I	J	K	L
1	GetSyster	SetUnhan	SetErrorM	GetFileAtt	CreateAct	NtClose	GetSysten	NtQueryK	NtOpenKe	NtOpenKe	NtCreateF	GetFileSiz
2	6	0	1	0	1	14	1	15	8	4	1	0
3	5	0	1	0	3	14	0	0	0	3	0	0
4	2	0	1	0	1	10	0	0	0	1	0	0
5	2	0	1	0	1	10	0	0	0	1	0	0
6	5	0	0	0	0	31	0	86	10	11	2	1
7	10	0	1	0	1	13	2	4	3	3	0	0
8	1	0	1	0	1	10	0	0	0	1	0	0
9	1	0	1	0	1	10	0	0	0	1	0	0
10	1	0	1	0	1	10	0	0	0	1	0	0
11	1	0	1	0	1	10	0	0	0	1	0	0
12	1	0	1	0	1	10	0	0	0	1	0	0
13	9	0	3	0	1	10	2	0	0	1	0	0
14	3	0	1	0	1	10	1	3	2	1	0	0
15	1	0	1	0	1	10	0	0	0	1	0	0
16	3	0	1	0	1	10	1	0	0	1	0	0
17	5	0	1	0	1	10	1	3	2	1	0	0
18	15	0	0	0	4	118	1	19	11	13	22	0

Figure 4: CSV for Successfully Run Benign Samples

The overall number of APIs extracted include over 250 distinct APIs.

1.4.5. Machine Learning:

Once the dataset of both malicious and benign file samples has been developed, the project proceeds to the use of Machine Learning towards the development of a model to determine what files are malicious and what files are benign. The generated dataset is split into two halves. One half of the dataset (now split into two) has been used to train the model, whereas the other half is used to validate the trained dataset. The process determines which files are malicious based on the API calls made.

The dataset has been represented using frequencies of APIs called. This mode of representation has been selected based on a trial and error methodology. Data could be represented in several forms based on the features extracted however API data provided the best mode of classification. The next step along

the process was the use of Machine Learning to train a model to be used towards the detection of malware. Since the developed dataset was supervised, we decided to look into supervised learning algorithms and determined 4 specific algorithms to be used. These include:

- Naïve Bayes (Bernoulli, Gaussian and Multinomial)
- K-Nearest Neighbor
- Decision Trees
- Support Vector Machines

1.4.6. Program Development

Based on the developed Machine Learning model, a program is developed that allows for the use of the model towards the detection of live malware.

1.5. Limitations:

Anti-Ghost as a software-based solution towards the detection of malware allows for the detection of malware that implement the use of file-less attack vectors, i.e. malware that directly run in system memory and achieve persistence through injection into legitimate windows processes.

There are, however, several limitations associated with the development and use of Anti-Ghost. These have been enumerated on in the following section.

- Anti-Ghost has been developed for use with Microsoft Windows only.
- The scope of the project encompasses the detection of file-less malware that invoke PowerShell scripts, as well as VBA macros. These are but a subset of the total types of file-less malware available.
- The scope of the types of malware to be detected has been constrained owing to the lack of available time.
- Detection for malware types apart from those indicated will not be guaranteed.

1.6. Document Organization:

Chapter 1: Provides an overview and introduction to the Anti-Ghost project.

Chapter 2: Deals with the literature review carried out towards the development of Anti-Ghost, as well as its core features and functionality.

Chapter 3: The chapter discusses the software as well as hardware requirements for the Anti-Ghost project. This includes technical specifications, as well as the development of a working environment.

Chapter 4: This chapter is concerned with the design development of the Anti-Ghost project. This includes all UML Diagrams and their explanations.

Chapter 5: The chapter provides an in-depth enumeration of the methodology used by Anti-Ghost towards the development of the project.

Chapter 6: Presents a conclusion to the thesis, as well as recommendations into future work into the project.

1.7. Document Conventions

Headings

Heading are prioritized in a numbered fashion, the highest priority heading having a single digit and subsequent headings having more numbers, per their level.

Sub-Headings

All second level subheadings for every sub section have the same number as their respective main heading.

Figures

All figures in this document have captions and are numbered.

2. CHAPTER 2: LITERATURE REVIEW

This chapter is primarily concerned with the details behind the characteristics of file-less malware, past research into file-less malware, solutions to the problem at hand, i.e. the detection of file-les malware, and lastly, Anti-Ghost, its novelty and limitations.

2.1. Project Domain:

The Anti-Ghost (AG) project is essentially a software-based solution towards the detection of file-less malware, i.e. malware that implements the use of file-less attack patterns. File-less malware, as their name suggests, do not store the malicious file on to the system hard drive, rendering most signature-based detection useless. That is because signature-based detection systems typically detect malware using known file signatures (anti-malware systems typically update their known file signatures in the form of virus definitions every few days). Thus, signature-based detection systems are incapable of detecting file-less threats simply because there is no malicious file to trigger the detection system.

In an attempt to address this new class of malware, Anti-Ghost has been designed to counter these threats using behavioral analysis and Machine Learning. The project is essentially aimed at the detection of a subset of the total sub-types of file-less malware, i.e. malware that implement the use of PowerShell, and VBA macros stored on Microsoft Office files (trojans).

Ultimately, the purpose of Anti-Ghost is research into the detection of File-less malware in response to the ever-increasing threat, and the development of a system of detection indigenous to Pakistan.

2.2. Literature Review:

Owing to the rapid increase of malicious activity over the internet, new forms of malware (i.e. malicious code) are developed roughly every other day. One of the relatively-newer types of malicious threats that exist over the internet are file-less malwares; that are fundamentally your typical malware, however, unlike most malware they do not have files stored on to the system or even require persistence (in some cases). Here we discuss a few of the malware of a "file-less" nature with respect to their functionality, method of spread/propagation, actions as well as some of the steps one can take to prevent said malware from infecting one's systems alongside the detection of file-less malware.

The first malware was developed in 1971, known as the "creeper" virus, that initially infected a small number of systems amounting to a total of 10. Since then, computers systems have gone through numerous stages of evolution, becoming more complex, yet more and more accessible with every stage. However, along with system evolution, the malware themselves have also gone through significant evolution. More and more malicious code is being developed with each passing day, each more "creative" than the last.

One such "evolution" is the advent of file-less malware, the recent development in malware. Where typical malware stores some form of malicious executable that spreads throughout the system, file-less malware directly executes within the memory of the system. This is done with two primary objectives:

- Prevent the code for the malware from being discovered and broken.

- Bypass static checking by anti-viruses.

Since no antivirus classifies code in any language as malicious, especially at runtime, detection of file-less malware becomes impossible with conventional techniques such as checking for signatures due to a lack of a file to check etc. To detect file-less malware, one must first understand how it works, i.e. what methods it uses to propagate, achieve persistence, how it affects the system etc. One proposed method for successful detection involves an in-depth behavioral analysis of system afflicted with such malware to check for malware behavior with respect to the typical behavior of such malware at the propagation stage [4].

Towards that end, several studies have been conducted that have effectively developed methods for detection based on the use of Behavioral analysis and Machine Learning. In a previous study by Chen, Wang, Wen and Lai, the use of a Convolutional Neural Network had been implemented towards the detection of malware where the CNN algorithm allowed for detection ratios up to 90 percent. The study placed emphasis on detection of malicious code embedded into benign code which is essentially one of the most dominate classes of file-less malware [1]. The research proposed conversion of the malicious and benign datasets to a state where each element of the dataset had been converted image format which would then be used with a convolutional neural network algorithm to train a model to be used for the detection of malware. Several similar studies [2,3] indicate the feasibility of detection of malicious behavior by conversion to an image or bitwise signal format which makes for simple processing but have numerous drawbacks, the most notable of which being able to escape detection by simply changing

the malicious code or by obfuscation as well as placement of dummy-API calls within the malicious code as pointed out by Ma. et al, in their study. [4]

Several studies also fall within similar parameters to the Anti-Ghost project that is aimed at the detection of file-less malware using API frequency data [1,5,6]. Studies indicate that the generation of signatures based on malicious file and their use as features of the dataset are a viable method towards the detection of malware, and would essentially improve on the computational performance however, would be prone to an increase in false negatives owing to polymorphic malicious code, and variable malicious implementations [1]. The study suggests the use of algorithms such as CNN and LSTM over SVM, which is regarded as a very accurate deep learning algorithm owing to the nature of the dataset which is prone to very consistent change in implementation methodology [7]. Similar results can be viewed in several notable studies [15]. However, the use of SVM can also be overserved to yield very accurate results as indicated in by Zhang et al., where the use of SVM allows for reduced computational complexity and increased accuracy [9]. Lastly, in a very notable study by Alazab et al. which implemented the use of API frequency analysis, classified given files as malicious or benign based on whether a particular API had been called [10].

We further view previous studies that cover the behavior of well-known malware of the class to determine some of the possible characteristics of file-less malware. A study by Aiden et al. covered one of the most devastating malware attacks of the century which involved the use of the Petya Ransomware in 2016. The malware also proved to inspire numerous reworks the following year. The Petya ransomware attacks had been targeted at numerous multi-national organizations where organizational computer systems had

been encrypted. The study having analyzed the malware determined that the malware operated using file-less principles and thus was able to remain undetected across thousands of computer systems. Owing to the lack of a suitable counter measure at the time, the malware also spread across the internet, where companies like Merck, the National Bank of Ukraine, and several airports had also been infected by the malware. Research indicated that the ransomware implemented the use of a windows exploit that allowed the malware to infect the systems using the Service Message Block (SMBv1) [8]. A year post the Petya malware attacks, Sorebrect emerged, taking the world by storm. The malware affected several industrial facilities worldwide where countries that suffered the most included the US, parts of Japan and China, as well as countries like Lebanon or Kuwait. Unlike the Petya Ransomware, Sorebrect displays characteristics similar to the trojan class, however, is technically still a ransomware. A previous study [11] indicated that the malware fundamentally replicates itself into system memory and executes, where it injects itself into svchost.exe, a legitimate windows process, achieving persistence, post which, the malware proceeds to encrypt the user's files. The analysis of the malware also proved difficult owing to a feature of the malware that forces deletion of system logs once executed, effectively clearing all traces of anything having been executed. The malware forces the use of a legitimate windows process that assumes control of remote execution of system commands (PsEXEC). The malware uses a PowerShell command that runs the malicious payload directly into system memory [11]. Another notable malicious epidemic that implemented the use of a Wrapper (i.e. belonging to the trojan class) was GZipDe, that acted as a front to the startup of a Metasploit-based backdoor into the infected system.

An analysis of the malware in a previous study determined that the malicious process was disguised as a valid Microsoft DOS-header that also housed the Meterpreter payload [12].

Wannacry, which is one of the most devastating malicious attacks to date is often regarded as a prime example of why we need preventive measures against threats to cybersecurity. The malware spread across the globe, and infected thousands of systems. An in-depth analysis of the malware indicated that the malware implements the use of two fundamental methods towards the encryption of data on the target system. The first is a modified version of the original EternalBlue vulnerability that had been implemented by the Petya ransomware [8], and a malicious payload. Furthermore, multiple systems had been tested for the effects of the malware, each of which yielded distinct results, indicating the use of polymorphism within the execution framework. Thus, the malware had been incredibly difficult to detect, and was free to spread all over the world, up-until a patch for the Windows Service Message Block (SMB) was released. The study further enumerates on an adequate method of detection through the analysis of system behavioral logs and comparison with the logs from a secure system which also helps counter the polymorphic nature of the code [13]. From an analysis of the popular banking trojan, Dridex, a slightly different method towards the propagation of malware can be observed. The infection method starts off as an email document which has embedded VBA macros within the file's framework that execute as soon as the document is opened. The VBA script runs a PowerShell script that further injects malicious code into windows software processes. The malicious code is placed in order to monitor the system and remains dormant until the user makes use of his/her financial credentials. Furthermore, the malware extended functionality towards the use of the target/infected systems as a

component to a variable bot-net network used to spread the malware [14]. An in-depth analysis of the DNSMessenger based malware indicates functionality based on a back-door into the system. The malware makes use of Meterpreter payloads that are run directly within the system memory, leaving no traces or logs of the execution on the system, making it very difficult to detect. Using the Meterpreter backdoor, malicious files are run on the target system using a DNS protocol, and persistence is achieved through dll-injection [16]. Lastly, we have the Kovter malware that surfaced on the internet late-2015. An analysis of the Kovter malware, rather, distinct variants of the malware that appear over time indicate that the malware has been distributed using several different methods inclusive of root-kits, macro-embedded documents, PowerShell scripts embedded into web-site URLs. The specific malware is often considered a more generic malware that exhibits behavior typical to numerous forms of file-less malware such as registry-injection, dll-injection, and code injection [17].

2.3. Characteristics of File-less Malware

File-less malware have a number of characteristics that are different as opposed to typical classes of malware. File-less malware, in essence, can be considered malware with the typical functionality expected of malware such as encryption of user files for ransom, deletion of OS\User files, monitoring user systems, stealing information, etc. However, file-less malware accomplishes said functionality without the need of a malicious file stored somewhere on the target system, as opposed to typical malware, that store a malicious file somewhere on the user's system. When a file-less variant of a malware propagates, it runs itself directly in the system's memory, without saving a physical file on to the system. Since most anti-malware systems that use signature-based

detection methods towards the detection of malware, said malware runs without being detected by the system's security. Some of these malwares are relatively less dangerous and will be effectively expelled from the system with a simple restart, however, many are still able to achieve persistence by injecting themselves into legitimate windows services such as svchost. On the other hand, some malware hardly have the need for persistence, where ransomware such as Wannacry are primary examples.

The typical characteristics of file-less malware, and whether samples of malware have been used in the developed dataset have been enumerated on in the following table:

Num	Characteristic	Sample Inclusion in Dataset
1	Absence of physical file on hard drive	Yes
2	Non-persistent	Yes
3	Persistent	Yes
4	Use of PowerShell	Yes
5	Use of Automated VBA Scripts	Yes
6	Use of Malicious URLs	Yes
7	Use of Code Injection	Yes
8	Downloading third-party adware	Yes

Table 1: Typical Characteristics of File-less Malware

2.4. Previously Available Anti-Malware Solutions

The following section focuses over some of currently used malware solutions being used all over the globe. Viewing the total number of anti-malware systems available, one can see thousands of solutions that claim to be able to protect a computer system from over 90 percent of the malware spread all over the internet. The section enumerates on whether or not they prove efficient detecting file-less classes of malware and compares them based on price point.

Windows Defender

The most widely used anti-virus system in the world is windows defender. Owing to the placeholder nature of the system, it is essentially only effective against the most basic malware and does not detect the While it may not prove an efficient system for the prevention of defense against more specialized classes of malware, it is a basic form of protection that one acquires at no additional cost. Windows Defender is often considered an entry-level malware solution. Testing numerous malwares against it, we observed its inadequacy considering the detection of file-less malware.

Drawbacks

- Entry level solution
- Uses signature-based detection
- Does not protect against most file-less threats

Avast

Avast is considered to be one of the more widely used anti-virus systems. The anti-virus provides a free version that has been tested and results indicate that the system is in fact capable of detecting file-less malware, however, up to a certain extent. The anti-malware does detect samples that implement the use of Microsoft Office files using embedded VBA macros, however, is oblivious to malicious activities post execution. The anti-malware does also have a paid version that is able to monitor web traffic, which may solve this problem, however it has not been tested.

Drawbacks

- Expensive annual paid subscription
- Inadequate free version

- Only detects the VBA embedded office file and is oblivious to further malicious activity

Kaspersky

Kaspersky is considered to be one of the more high-end/premium anti-malware solutions. Recent improvements to the software have enabled it to detect most classes and sub-types of file-less malware, however, the software comes with a highly expensive price tag.

Drawbacks

- Expensive

McAfee

McAfee has been researching how to detect file-less malware over the years and is considered one of the better anti-malware systems, however, testing malware samples on the software indicated that the software does in fact allow some of the samples to run without detection. That considered alongside the price tag for a single system license renders it infeasible for a good majority of people.

Drawbacks

- Expensive
- Allowed numerous samples to run without detection

ClamAV

ClamAV is an opensource anti-malware solution designed to be both secure and accessible. The program is both free and is able to detect a small subset of file-less malware to a certain extent. The anti-malware does detect samples that implement the use of Microsoft Office files using embedded VBA macros, however, is oblivious to malicious activities post execution similar to how Avast's free version works.

Drawbacks

- Inadequate
- Only detects the VBA embedded office file and is oblivious to further malicious activity
- Lack of software support owing to open-source nature.

3. CHAPTER 3: TECHNOLOGICAL REQUIREMENTS

The following chapter is aimed at the enumeration of comprehensive details into the entire working environment set up required for the development of Anti-Ghost. This includes hardware requirements, software requirements, software tools, as well as packages and libraries installed.

3.1. Hardware Requirements

The Hardware that had been required and used throughout the development of Anti Ghost have been listed and enumerated on as follows.

The first and foremost requirement was a computer system capable of running multiple instances of Windows 7 running on VMWare to speed up the malware analysis performed using Cuckoo, the malware analysis tool. The system was built around an AMD Ryzen 5 2600x, a 6-core, 12-thread processor (i.e. to make use of the high core count) to allow for multiple simultaneous VMWare instances. Real-time use of the processor allowed for 3 simultaneous instances in actual use, paired with high frequency Ram.

The second hardware requirement to be met was storage. While the program itself will essentially consume a fraction of the space required, a fair amount of storage capacity is required to develop the dataset. The total malware samples acquired towards the purpose of analysis were essentially over a terabyte of data. However, post filtration to samples of malware that invoked PowerShell, or made use of VB Macros, the dataset essentially contained 2000 malware samples. In order to analyze the information, we had it collectively stored on to two 500 GB SSD Drives.

The third hardware requirement to be addressed was the dedicated graphical processing unit. The unit selected for this purpose was the GTX 1050ti, a suitable mid-ranged GPU that would be capable of processing the dataset and using Machine Learning Algorithms.

Lastly, two portable laptop computers have also been used to supplement the development of the malware dataset, particularly supplementing to the speed at which malware samples were analyzed.

Num	Hardware Component	Qty
1	Computer System (Ryzen 5 2600x/16GB)	3
2	SSD Drives	2 x 500 GB
3	Dedicated Graphics Card	1
4	Portable Laptop Systems	2

Table 2: Complete List of Hardware Used

3.2. Software Requirements

The Software that had been required and used throughout the development of Anti Ghost have been listed and enumerated on as follows.

Ubuntu

Anti-Ghost has been developed using Ubuntu 16.04-LTS (64-bit) as an OS platform. While the distribution may be considerably outdated, it has been used because the next iterations (both 18.0x and 19.0x) do not support numerous programs, specifically VirtualBox 5.2.

Python

The project implements the use of Python 2.7. This particular version of python is the final fully supported version of Python in terms of the use of the Cuckoo sandbox.

MongoDB

The web-based interface for the cuckoo requires the use of MongoDB and it must be installed as a pre-requisite using:

```
$ sudo apt-get install mongodb
```

PostgreSQL

Cuckoo recommends the use of PostgreSQL within the framework of the program and must be installed as well.

```
$ sudo apt-get install postgresql libpq-dev
```

VirtualBox

The virtual machine used for the purpose of the real time analysis of malware using the cuckoo platform is VirtualBox 5.2. This is the last supported version for the Cuckoo sandbox and support for the version ended with ubuntu 16.04. The purpose of the virtual machine is to actively run malware samples and collect behavioral data over the malware run. Said data includes URLs accessed, APIs called etc. VirtualBox can be installed as follows:

```
$ echo deb http://download.virtualbox.org/virtualbox/debian xenial contrib | sudo tee -a /etc/apt/sources.list.d/virtualbox.list

$ wget -q https://www.virtualbox.org/download/oracle_vbox_2016.asc -O- | sudo apt-key add -

$ sudo apt-get update

$ sudo apt-get install virtualbox-5.2
```

tcpdump

The class of malware intended to be analyzed typically accessed malicious URLs through the internet. In order to capture and analyze the network activity of any particular malware, the packages sent and received must be captured. Tcpdump provides an open source solution towards this and is supported by Cuckoo by default. Tcpdump can be installed using:

```
$ sudo apt-get install tcpdump apparmor-utils
```

```
$ sudo aa-disable /usr/sbin/tcpdump
```

Tcpdump also requires the use of specific root privileges that can be set using:

```
$ sudo groupadd pcap

$ sudo usermod -a -G pcap cuckoo

$ sudo chgrp pcap /usr/sbin/tcpdump

$ sudo setcap cap_net_raw,cap_net_admin=eip /usr/sbin/tcpdump
```

M2Crypto

M2Crypto is a python based library used with cuckoo. It is an SSL and crypto toolkit and can be installed as follows:

```
$ sudo apt-get install swig

$ sudo pip install m2crypto==0.24.0
```

Cuckoo

Installing and running the Cuckoo sandbox software has numerous pre-requisites that have been met. However, before we can install Cuckoo, numerous software packages have to be installed from their respective apt repositories.

```
$ sudo apt-get install python python-pip python-dev libffi-dev libssl-dev

$ sudo apt-get install python-virtualenv python-setuptools

$ sudo apt-get install libjpeg-dev zlib1g-dev swig
```

After all prerequisites have been met, we can proceed to creation of a new user using:

$ sudo adduser cuckoo

As we will be implementing the use of VirtualBox, we add the newly created user to a "vboxusers" group as follows:

$ sudo usermod -a -G vboxusers cuckoo

The next step in the process is the installation of the Cuckoo sandbox. This can be done using the following:

$ sudo pip install -U pip setuptools

$ sudo pip install -U cuckoo

With this, cuckoo has been installed and is running. Furthermore, a complete table of software used with support requirements has been provided as follows:

Num	**Hardware Component**	**Version**	**Support Constraint**
1	Ubuntu	16.04	-
2	Python	2.7	Cuckoo Final Full-Support
3	MongoDB	-	-
4	PostgreSQL	-	-
5	VirtualBox	5.2	Cuckoo Final Supported Version
6	tcpdump	-	-
7	M2Crypto	-	-
8	Cuckoo	-	-

Table 3: Complete List of Software Used for The Anti-ghost Project

3.3. Operating System Associations

The software being developed is essentially being developed for use with the Windows operating system by Microsoft. The operating system being used for the analysis of malware samples is Windows 7, running over instances of VirtualBox, using Ubuntu as a host system.

3.4. Working Environment Setup

The Cuckoo sandbox allows users to run live malware samples on a virtualized operating system instance using VirtualBox or KVM. This allows Cuckoo to analyze the methods used by a particular malware to compromise a target system and report on the behavior of said malware. For the purpose of our project, we have used VirtualBox.

Before we can start analyzing malware using the Cuckoo sandbox, a working directory, i.e. a working environment must be set up and configured. This can be accomplished using:

```
$ sudo mkdir /opt/cuckoo

$ sudo chown cuckoo:cuckoo /opt/cuckoo

$ cuckoo --cwd /opt/cuckoo
```

Once the working environment has been set up, one can simply run cuckoo and analyze malware using a simple web-based interface.

The web-based interface is mostly an automated, easy to use, drag and drop interface that allows a simple alternative to using Cuckoo on a local instance. The

interface also simplifies data visualization and manual analysis and monitoring of malware and their reports.

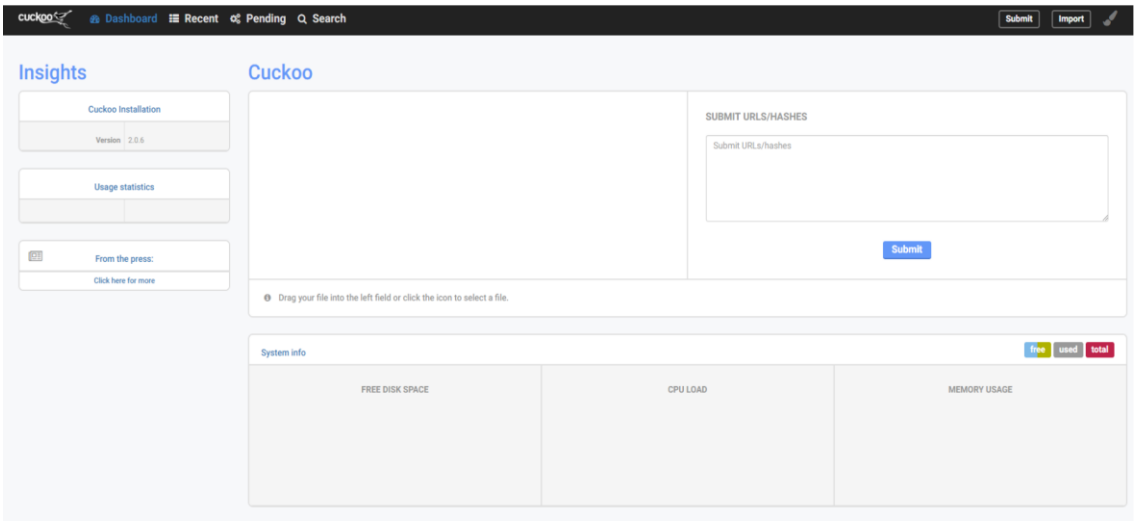


Figure 5:The Django Based Cuckoo Web-Interface

4. CHAPTER 4: ANTI-GHOST DESIGN

This chapter is concerned with the design development of the Anti-Ghost project. This includes all UML Diagrams and their explanations.

4.1. Class Diagram:

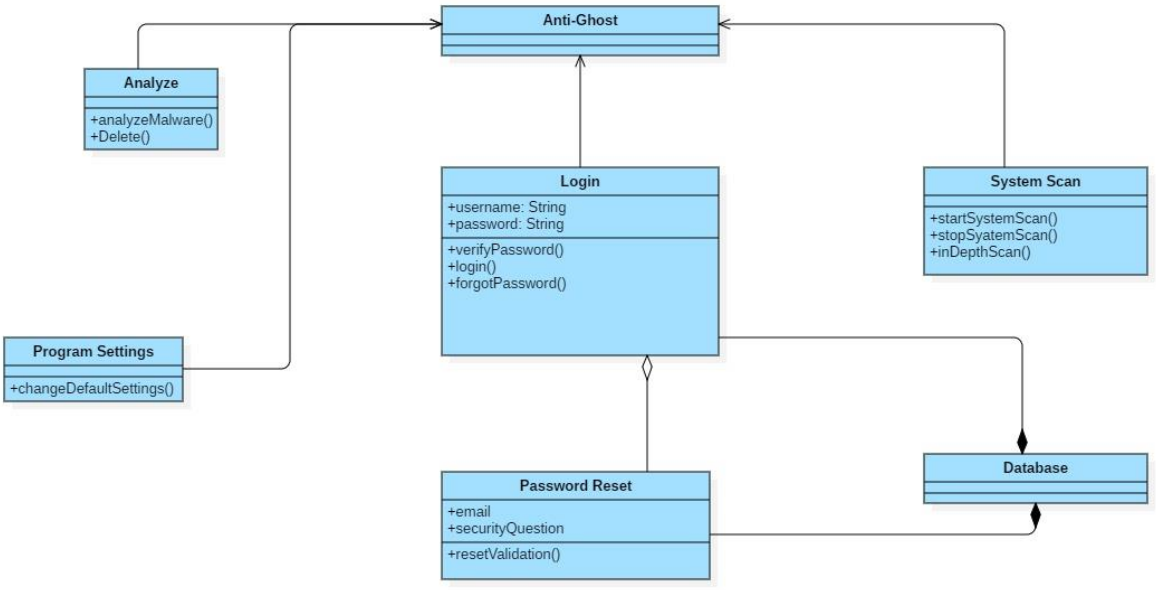


Figure 6: Anti-ghost Class Diagram

Settings: Change Software Settings.

Database: Contains Every data item (user info, password, etc.).

Anti-Ghost: Framework.

Login: Verify and grants access to a registered user.

Password Reset: Resets a user's password.

System Scan: Start System Scan.

Analyze: Analyze Malware after it has been detected.

4.2. Activity Diagram:

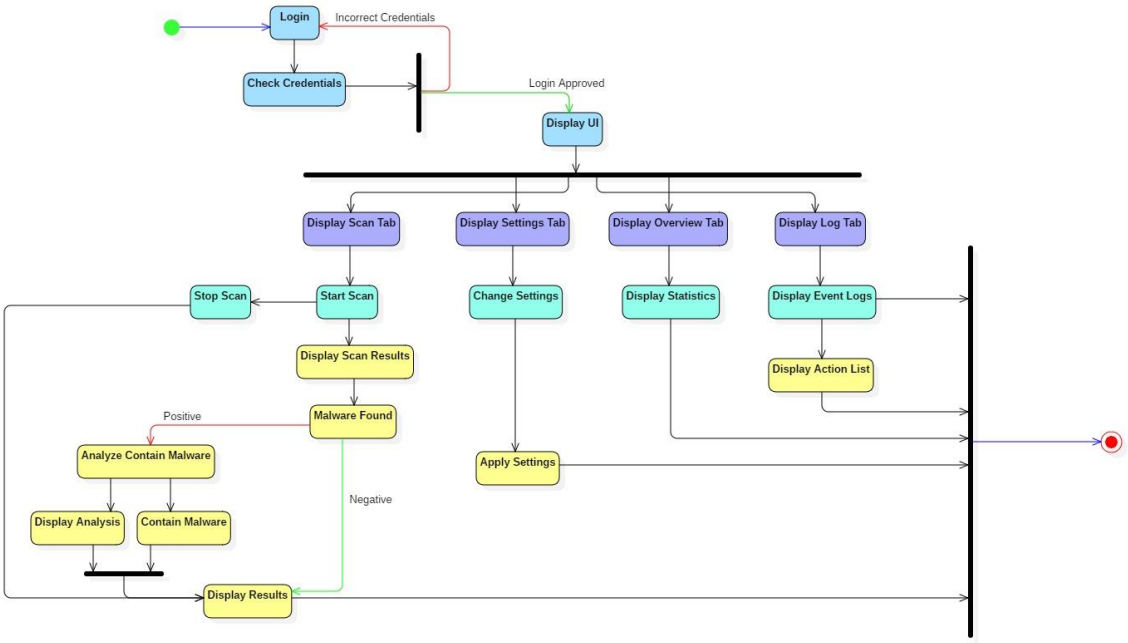


Figure 7: Anti-Ghost Activity Diagram

The activity diagram enumerates on how the Anti-Ghost program was designed to be developed. The program initiates and starts off with a login module where users input their credentials. Once said credentials have been verified, the program displays the main UI. The main UI has several tabs named appropriately (i.e. Scan, Settings, Overview, and Logs). Each tab allows the user to access specific functions such as scanning the system, make changes to settings and display logs.

4.3. Object Diagram:

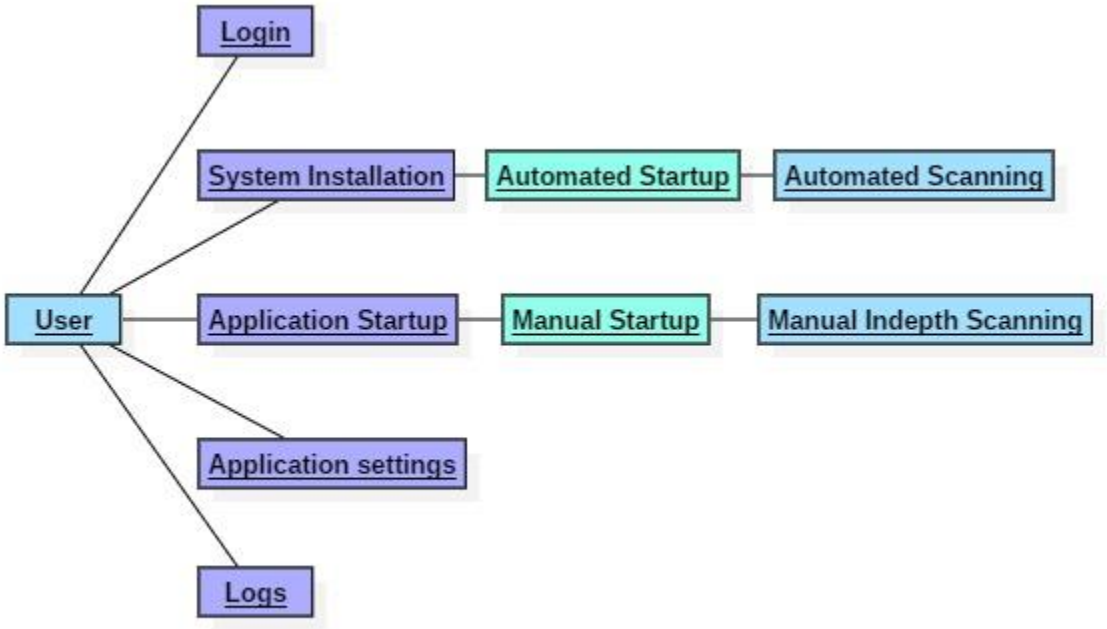


Figure 8: Anti-Ghost Object Diagram

The user will be able to install the application, login to the application. The anti-malware inherently starts up with system startup by default, however, the user will also be able to override this automated startup and manually start up Anti-Ghost. Anti-Ghost analyses system memory in real-time and control any malicious occurrences. Furthermore, the software stores logs of malicious occurrences.

4.4. Use Case Diagram:

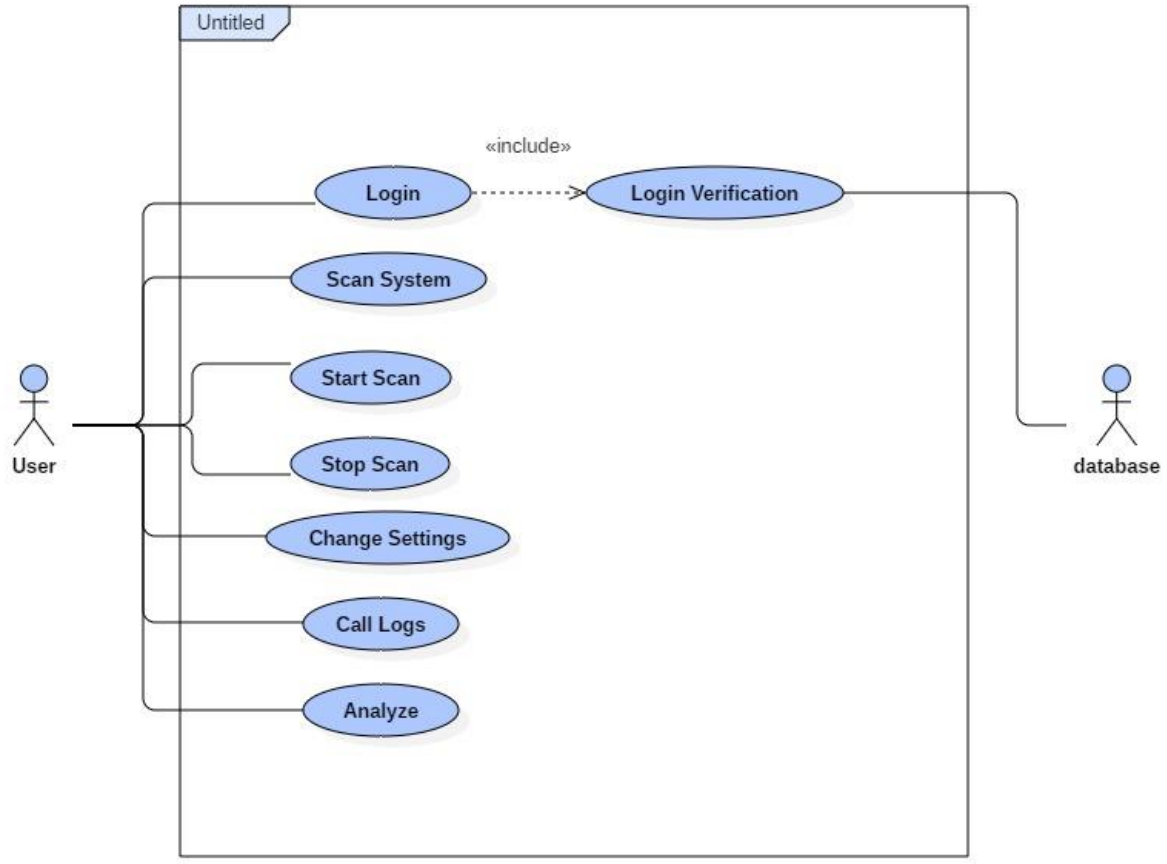


Figure 9: Anti-Ghost Use Case Diagram

Log in: The user enters the credentials to use the software.

Verify Log in: The database check for pre-registered user.

Scan System: The user can scan the system.

Start Scan: The user can manually start an in-depth scan.

Stop Scan: The user can stop the scan mid-way.

Analyze: The users can analyze any and all malicious findings.

Call Logs: The user can view logs for each malicious activity prevented.

Use case	Log in
Pre-condition	The user is previously added in the system.
Post-condition	The user is successfully logged into the system.
Basic Path	1. User has to enter username and password. 2. The entered username and password are matched from the entries in the database.
Alternative path	-
Exceptional path	User is not registered with the system and ‘no match found’ error message is generated.

Table 4: Use Case: Login

Use case	Start Scan
Pre-condition	1. The Software is installed onto the system. 2. The software is active and had been given adequate permissions.
Post-condition	The system results of the scans, whether positive or negative.

Basic Path	1. User presses the ‘Scan Now’ button.
Alternative path	-
Exceptional path	-

Table 5: Use Case: Scan

Use case	Start Scan
Pre-condition	The User is already in the scan section.
Post-condition	-
Basic Path	1. User presses the ‘Start Scan’ button.
Alternative path	-
Exceptional path	-

Table 6: Use Case: Manual Scan

Use case	Stop Scan
Pre-condition	The software is scanning the system
Post-condition	The system stops the in-depth scan.
Basic Path	1. User presses the ‘Stop Stream’ button.
Alternative path	-
Exceptional path	-

Table 7: Use Case: Stop Scan

Use case	Analyze
Pre-condition	The software has recorded malicious activity.
Post-condition	-
Basic Path	1. User presses the ‘Analyze’ button.
Alternative path	-

Exceptional path	-

4.5. Sequence Diagram

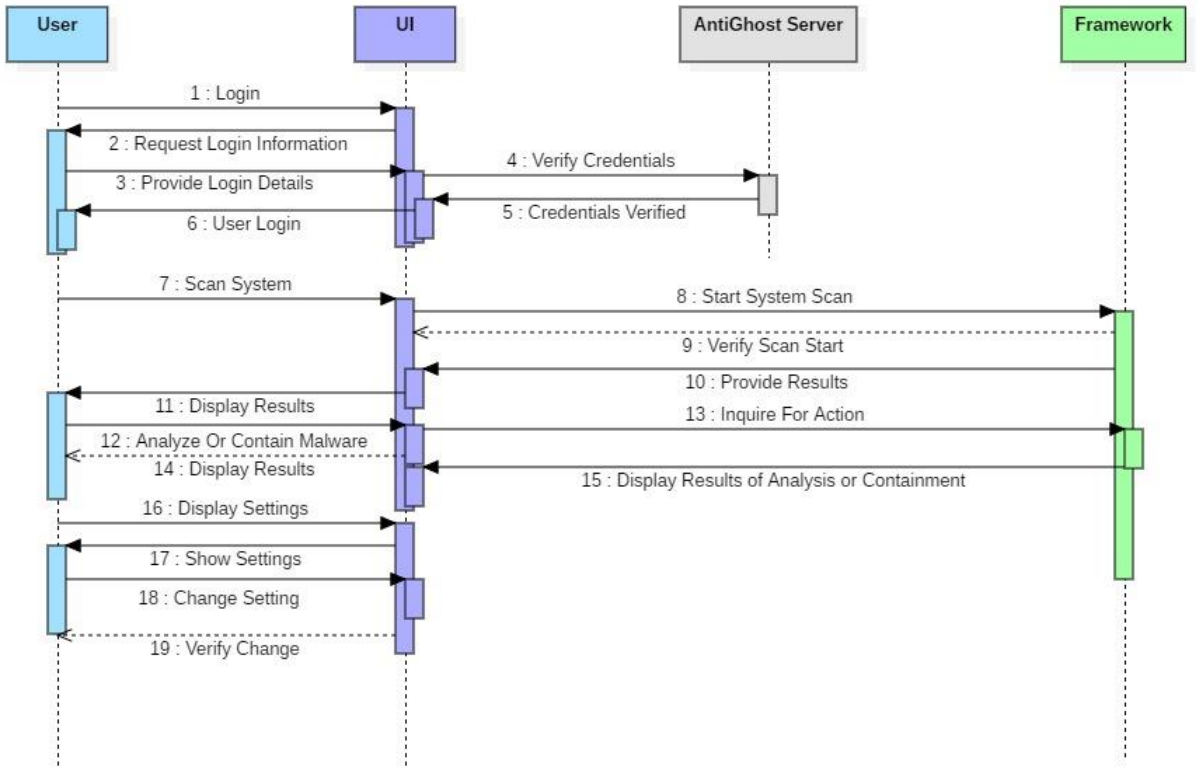


Figure 10: Anti-Ghost Sequence Diagram

The sequence diagram enumerates on the sequence of user activity. The user starts off by logging into the Anti-Ghost system with his/her credentials. Once said credentials have been verified, the program displays the main UI. The main UI has several tabs named appropriately (i.e. Scan, Settings, Overview, and Logs). Each tab allows the user to access specific functions such as scanning the system, make changes to settings, etc.

4.6. GUI

In accordance with the requirements of the system, the GUI for Anti-Ghost Project has been developed and will be enumerated on as follows.

4.6.1. Home Window

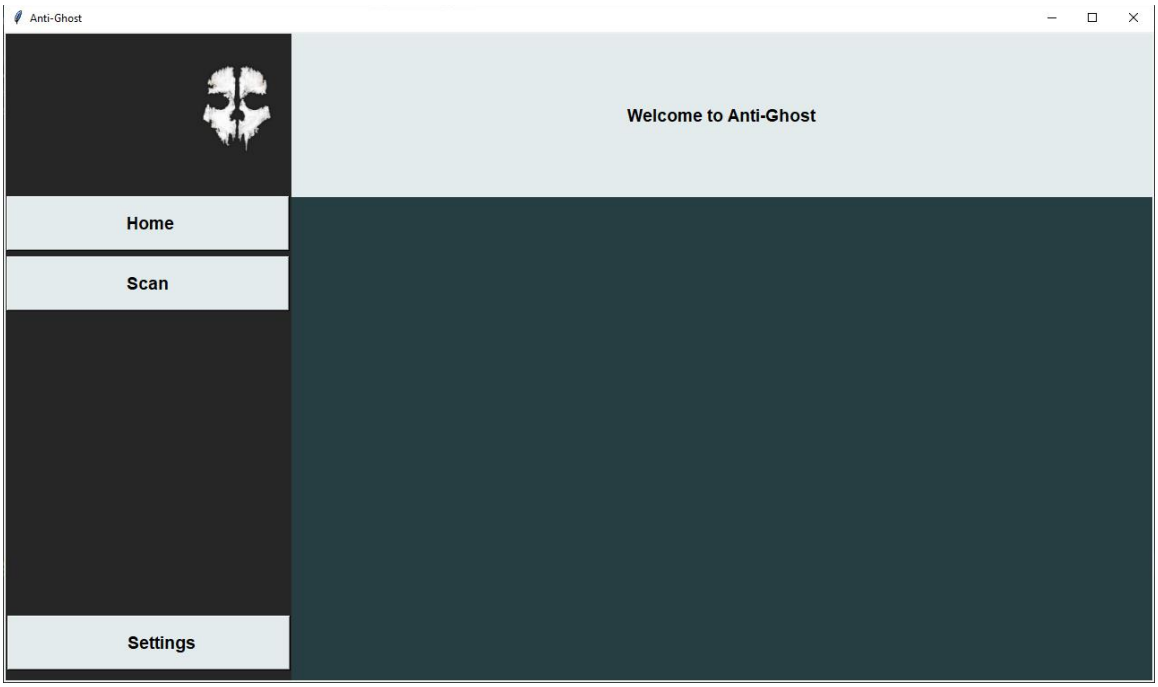


Figure 11: Anti-Ghost Home Window

The image is essentially the home window for the Anti-Ghost system. The home window is the first window visible to users when the software is run. The home window provides users with a simple interface to access system features such as accessing system settings, or system scan.

Users can typically click either of the scan button on to view system scan functionalities, or the settings tab for the settings functionalities (to be added on further update.), alternatively, users can also choose to click the home button to return to the home window.

4.6.2. Scan Window

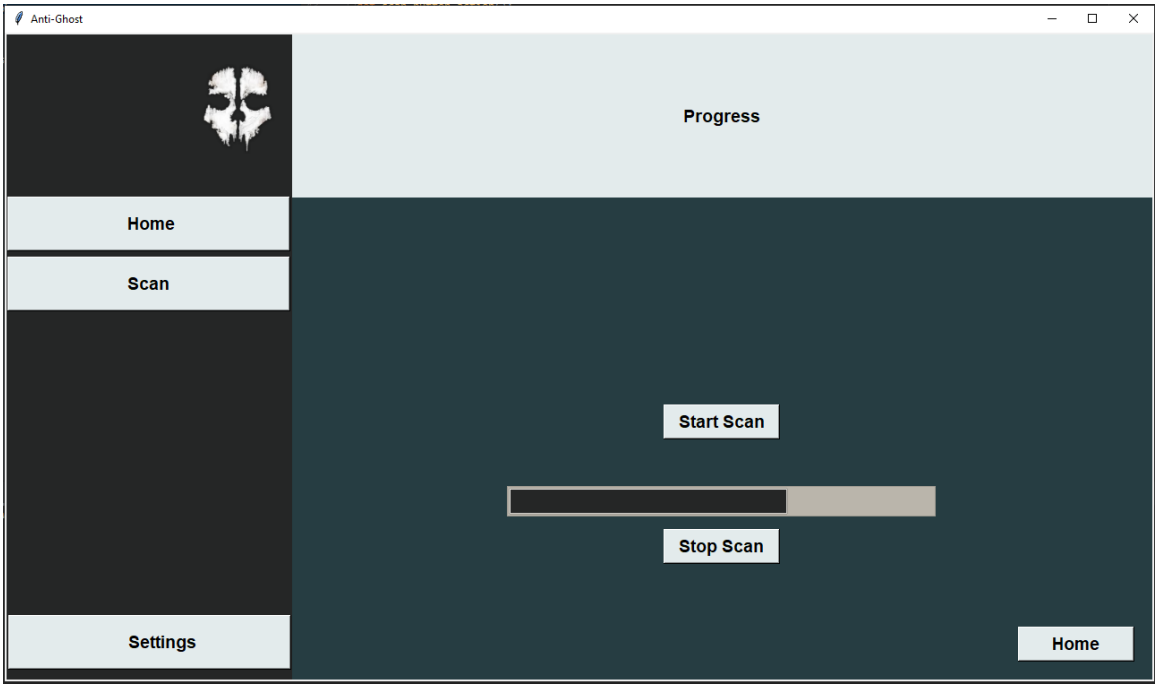


Figure 12: Anti-Ghost Scan Window

The image portrays the scan window for the Anti-Ghost system. While the home window is the first window visible to users when the software is run, the scan window can simply be accessed using the scan button on the left tab. The scan window provides users with a simple interface to access system features such as starting a system.

The system scan can be run using the “Start Scan” button on below the progress bar. Once initiated, the progress bar fills with progress into the scan indicating to users the percentage of the scan completed.

4.6.3. Settings Window

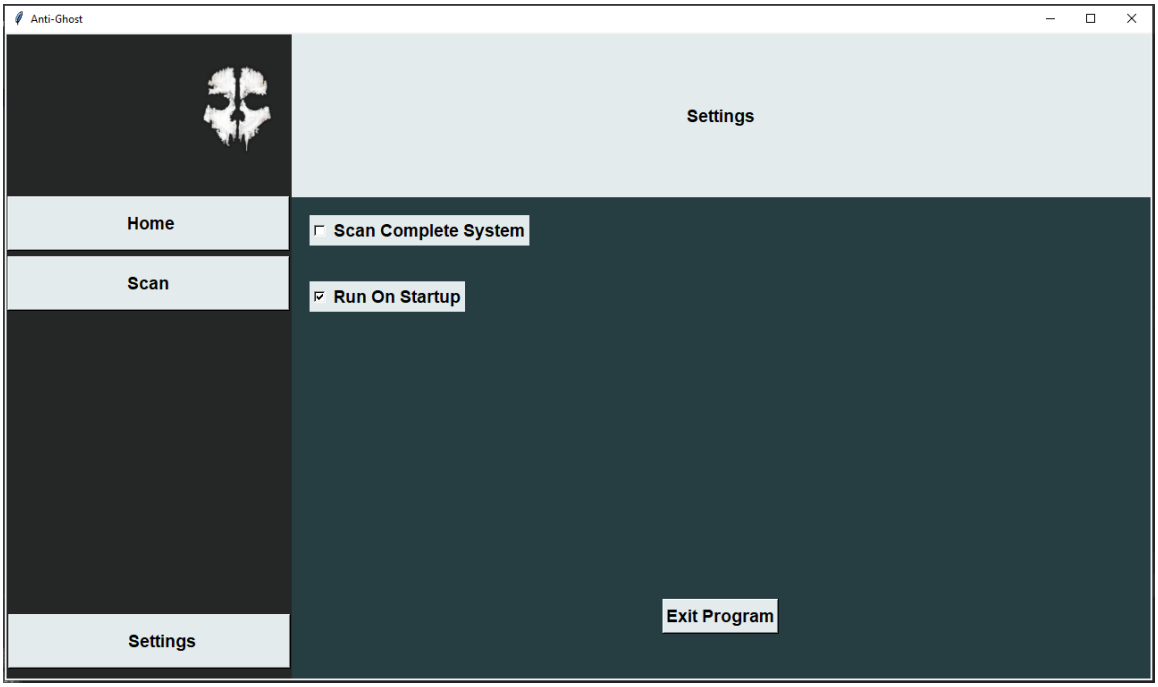


Figure 13: Anti-Ghost Settings Window

The image portrays the settings window for the Anti-Ghost system. While the home window is the first window visible to users when the software is run, the settings window can simply be accessed using the settings button on the left tab. The scan window provides users with a simple interface to access system settings using simple checkboxes.

5. CHAPTER 5: DETECTION METHODOLOGY AND RESULTS

The anti-ghost project fundamentally aims at the detection of malware that employ the use of file-less attack vectors, namely, file-less malware. Towards that end, the project implements the use of behavioral analysis and Machine Learning. Along the scope of the project, i.e. the detection of file-less malware, the use of behavioral analysis is a necessity owing to the file-less nature of malware that make static analysis methods implemented by most over the counter anti-malware software inefficient due to the unavailability of a malicious file on the system's secondary memory. Secondly, paving the way to an anti-malware solution based on behavioral analysis and Machine Learning involves the use of an adequate dataset based on specific malicious as well as benign samples that will, and has been built from scratch.

In an attempt to address this new class of malware, Anti-Ghost has been designed to counter these threats using behavioral analysis and Machine Learning. The project is essentially aimed at the detection of a subset of the total sub-types of file-less malware, i.e. malware that implement the use of PowerShell, and VBA macros stored on Microsoft Office files (trojans). A sample equal of benign samples equal to the sample set of live malware samples had also been procured. The sample set of benign files includes benign VBA and PowerShell based files. Once the sample sets of malicious and benign files had been gathered, the files were analyzed to assess features of both the live malware and the benign files. This has been accomplished using Cuckoo, the leading open-source, automated malware analysis tool available. Cuckoo allows for the analysis of malware to extract features such as API calls, files run, etc., essentially allowing us to monitor a

particular file's behavior. Cuckoo also cross-validates results with popular anti-malware solutions. Each malware sample was run on a VMWare instance run inside the cuckoo sandbox. Cuckoo allows the malware to run on a sandboxed, windows-based VMWare instance, recording how the malware behaves. The behavior of the malware is recorded, and a report is generated based on the complete set of actions the malware performs. The same is also true for benign file samples. Once all the malware samples and benign files have been analyzed, features are extracted from the generated reports. This includes two CSVs generated each, for both malicious files and benign files. One set of CSVs includes two files, where one file has API data on failed malware, and one has API data on successful entries. The same is also true for benign samples.

The dataset has been represented using frequencies of APIs called. This mode of representation has been selected based on a trial and error methodology. Data could be represented in a number of forms based on the features extracted however API data provided the best mode of classification. Another mode of detection considered was conversion of the dataset to image data which could be used to classify benign and malicious files. The mode of detection would drastically improve based on previous studies into the matter, where it method provides an ideal method of detection of malicious code embedded into benign code (the cardinal class of file-less malware based on the available literature). While the mode would be a possible solution based on the nature of file-less malware, the implementation of the ML algorithm would be too dependent on computation resources that were simply unavailable. That is because conversion of files to an image format would increase the resource consumption to an extent unsupported by our systems.

The next step along the process was the use of Machine Learning to train a model to be used towards the detection of malware. The process of Machine Learning can be divided into several steps, where preprocessing and transformation represent up to 90% of the work involved. There are 5 basic stages involved:

Machine Learning Phases	
Intake	The Dataset is loaded into memory from the developed csv.
Transformation	The data is cleaned and normalized for use with the algorithms. The objective of pre-processing is to convert the complete set of data to lie within a single range. Furthermore, we also split the data into two sets, i.e. the training data and the test data.
Training	This phase involves development of the training model using a given algorithm.
Training	The trained model is tested using the test data.
Deployment	The model that indicates more favorable results is selected.

Table 8: Machine Learning Phases

Since the developed dataset was supervised, we decided to investigate supervised learning algorithms and determined 4 specific algorithms to be used.

These include:

- Naïve Bayes (Bernoulli, Gaussian and Multinomial)
- K-Nearest Neighbor
- Decision Trees
- Support Vector Machines

5.1. Results

5.1.1. Naïve Bayes Algorithm

The first algorithm used is the Naïve Bayes algorithm which allows for classification by comparing probabilities based on APIs use. A probability on whether a file is malicious or benign is determined for a given file based on APIs called by the file when run. If a file has a higher probability of malicious character, it is classified as malicious, and vice versa. The algorithm was implemented using three classifiers: the Bernoulli classifier, the Gaussian classifier and the Multinomial classifier. The results of implementation indicated a favorable level of accuracy in terms of detection of malware.

```
BernoulliNB(alpha=1.0, binarize=True, class_prior=None, fit_prior=True)
0.7276386404293381
              precision    recall  f1-score   support

           0       0.73      1.00      0.84      1627
           1       0.00      0.00      0.00       609

    accuracy                           0.73      2236
   macro avg       0.36      0.50      0.42      2236
weighted avg       0.53      0.73      0.61      2236
```

Figure 14:Naive Bayes Algorithm with Bernoulli Classifier

```
MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
0.8492844364937389
              precision    recall  f1-score   support

           0       0.92      0.87      0.89      1627
           1       0.69      0.80      0.74       609

    accuracy                           0.85      2236
   macro avg       0.81      0.83      0.82      2236
weighted avg       0.86      0.85      0.85      2236
```

Figure 15: Naive Bayes Algorithm With Multinomial Classifier

```
GaussianNB(priors=None, var_smoothing=1e-09)
0.6426654740608229
              precision    recall  f1-score   support

           0       0.99      0.51      0.68      1627
           1       0.43      0.99      0.60       609

    accuracy                           0.64      2236
   macro avg       0.71      0.75      0.64      2236
weighted avg       0.84      0.64      0.66      2236
```

Figure 16: Naive Bayes Algorithm with Gaussian Classifier

There are a number of advantages towards the use of Naïve Bayes algorithm. These include, first and foremost a level of simplicity and implementation. Furthermore, the Naïve Bayes algorithm works well with because the probability of an irrelevant aspect/feature has a very low probability of affecting the results.

5.1.2. K-Nearest Neighbor Algorithm (KNN)

The second algorithm used was the K-Nearest Neighbor (KNN) algorithm. The algorithm works through comparison of a test file with all the trained data. The algorithm

further determines the closest examples from the dataset and opts for a label (malicious or benign) based on similarity between API calls made.

```
0.9311270125223614
              precision    recall  f1-score   support

           0       0.96      0.94      0.95      1627
           1       0.85      0.90      0.88       609

    accuracy                           0.93      2236
   macro avg       0.91      0.92      0.91      2236
weighted avg       0.93      0.93      0.93      2236
```

Figure 17: KNN Algorithm

The use of the KNN algorithm bears a number of advantages, the first of which is simplicity in terms of implementation and accuracy in terms of results. The non-parametric nature of the algorithm is also beneficial as it does not make assumptions over the structure of the data, making it suitable for use. In terms of how the algorithm works within the scope of the Anti-Ghost project, the prediction on whether a particular file is malicious or benign is determined through comparison of the file being tested with the nearest training sample.

5.1.3. Decision Tree Algorithm

The third algorithm used towards the detection within the scope of the project is the Decision Tree algorithm. As the name of the algorithm suggests, the algorithm represents the data in the form of a tree. Each of the tree's leaf nodes represents a label, i.e. malicious or benign, where each sub-node represents an attribute, or in this case, APIs. The model compares the file being tested against the tree's nodes and checks for

homogeneity based on all available sub-nodes. The attribute that has the most homogeneous sub-nodes is selected based on the results of the analysis.

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=None, splitter='best')
0.9660107334525939
              precision    recall  f1-score   support

           0       0.97      0.98      0.98      1627
           1       0.95      0.93      0.94       609

    accuracy                           0.97      2236
   macro avg       0.96      0.95      0.96      2236
weighted avg       0.97      0.97      0.97      2236
```

Figure 18: Decision Tree Algorithm

The use of the model bears several advantages where it allows for more rapid decisions and analysis. The algorithm is also very simple in terms of implementation and can be optimized for speed in terms of analysis.

5.1.4. SVM Algorithm

Lastly, the project also makes use of Support Vector Machines (SVM). The SVM algorithm proves an adequate solution towards the classification of files as malicious or benign. As for its working within the scope of our project, it should be observed that the SVM algorithm classifies a given file as malicious or benign based on closeness to malicious or features based on closeness by margin. SVM divides the training data into two fundamental classes (i.e. malicious or benign) and determines a hyperplane between the two in two-dimensional space. Based on where a given file lies on the hyperplane, the developed model can be used to classify whether a file is malicious or benign.

```
0.8278175313059034
              precision    recall  f1-score   support

           0       0.85      0.92      0.89      1627
           1       0.74      0.57      0.65       609

    accuracy                           0.83      2236
   macro avg       0.79      0.75      0.77      2236
weighted avg       0.82      0.83      0.82      2236
```

Figure 19: SVM Algorithm

Several other algorithms had also been considered that included RNN and CNN however, as indicated, we lack the proper resources to run the algorithms with a large dataset.

A summary of the results in terms of accuracy by model can be found in the following table:

S.no	**Algorithm**	**Accuracy**
1	Naïve Bayes (Bernoulli, Multinomial, Gaussian)	(84%, 89%, 64%)
2	K-Nearest Neighbor (KNN)	95%
3	Decision Tree	97%
4	Support Vector Machines	83%

Table 9: Accuracy by Algorithm Used

6. CONCLUSION

Over the years, as software platforms were made more and more secure to avert and prevent malicious activity, malware themselves became more and more efficient. New ways to breach systems are being developed every day, the most recent being the use of file-less attack vectors.

In response to the increasing amount of file-less malware all over the cyber world, we require a system capable of the detection of file-less malware, the motivation behind which is the development of a malware detection system indigenous to Pakistan. Furthermore, as the demand for anti-malware systems has observed a considerable increase, the project has considerable feasibility in terms of market. Our project relates to the aforementioned demand, providing a solution to the detection of file-less malware using behavioral analysis, and Machine Learning. Towards that end, we have implemented the use of cuckoo to analyze the behavior of malware samples and extract features in the form of API calls. Based on the developed dataset, we have further used distinct supervised learning algorithms to inclusive of the Naïve Bayes algorithm, the Decision Tree algorithm, Support Vector Machines and the K-Nearest Neighbors algorithm. The use of each algorithm yielded distinct results ranging from 64% at the worst, and 97% at best. Several other algorithms had also been considered for use inclusive of RNN and CNN, however, we simply lacked the computational resources to implement them.

6.1. Future Recommendations

The Anti-Ghost project was aimed at the development of a system of detection for a subset of file-less malware. Towards that end, we developed a dataset of specific malware which alone is a sizable contribution towards the problem owing to the lack of an available dataset of filtered malware. However, owing to the nature of the project, as well as the limited scope owing to several constraints, the project has room for massive improvements. Future recommendations for possible improvement to the project include:

- Improving the dataset in terms of both quality and quantity.
- Increasing the dataset to support more classes of malware.
- Improving on the GUI component of the system.
- Expansion of the scope of the project to include detection by static analysis.
- Expanding support for the Linux platform
- Expansion of support for the Android platform
- Add System Logs to the software
- Populate Settings Tab with more options provided to the user.

7. Bibliography:

[1] C. Chen, S. Wang, D. Wen, G. Lai and M. Sun, "Applying Convolutional Neural Network for Malware Detection," 2019 IEEE 10th International Conference on Awareness Science and Technology (iCAST), Morioka, Japan, 2019, pp. 1-5, doi: 10.1109/ICAwST.2019.8923568.

[2] Nataraj, Lakshmanan & Karthikeyan, Shanmugavadivel & Jacob, Grégoire & Manjunath, B.. (2011). Malware Images: Visualization and Automatic Classification. 10.1145/2016904.2016908.

[3] M. Christodorescu, S. Jha, S. A. Seshia, D. Song and R. E. Bryant, "Semantics-aware malware detection," 2005 IEEE Symposium on Security and Privacy (S&P'05), Oakland, CA, USA, 2005, pp. 32-46, doi: 10.1109/SP.2005.20.

[4] Ma, Xin & Guo, Shize & Bai, Wei & Chen, Jun & Xia, Shiming & Pan, Zhisong. (2019). An API Semantics-Aware Malware Detection Method Based on Deep Learning. Security and Communication Networks. 2019. 1-9. 10.1155/2019/1315047.

[5] S. Peisert, M. Bishop, S. Karin and K. Marzullo, "Analysis of Computer Intrusions Using Sequences of Function Calls," in IEEE Transactions on Dependable and Secure Computing, vol. 4, no. 2, pp. 137-150, April-June 2007, doi: 10.1109/TDSC.2007.1003.

[6] Sun, H., Lin, Y., & Wu, M. (2006). API Monitoring System for Defeating Worms and Exploits in MS-Windows System. Information Security and Privacy Lecture Notes in Computer Science, 159-170. doi:10.1007/11780656_14

[7] Ki, Youngjoon & Kim, Eunjin & Kim, Huy Kang. (2015). A Novel Approach to Detect Malware Based on API Call Sequence Analysis. International Journal of Distributed Sensor Networks. 2015. 1-9. 10.1155/2015/659101.

[8] J. S. Aidan, H. K. Verma and L. K. Awasthi, "Comprehensive Survey on Petya Ransomware Attack," 2017 International Conference on Next Generation Computing and Information Systems (ICNGCIS), Jammu, 2017, pp. 122-125, doi: 10.1109/ICNGCIS.2017.30.

[9] Zhang, Kai & Li, Chao & Wang, Yong & Zhu, Xiaobin & Wang, Haiping. (2017). Collaborative Support Vector Machine for Malware Detection. Procedia Computer Science. 108. 1682-1691. 10.1016/j.procs.2017.05.063.

[10] Alazab, Mamoun & Venkatraman, Sitalakshmi & Watters, Paul & Alazab, Moutaz. (2011). Zero-day Malware Detection based on Supervised Learning Algorithms of API call Signatures. CRPIT. 121.

[11] Micro, T. (2017). Analyzing the Fileless, Code-injecting SOREBRECT Ransomware [Web log post]. Retrieved 2020, from https://blog.trendmicro.com/trendlabs-security-intelligence/analyzing-fileless-code-injecting-sorebrect-ransomware/

[12] Martin, J. M. (n.d.). GZipDe: An Encrypted Downloader Serving Metasploit. Retrieved July 09, 2020, from https://cybersecurity.att.com/blogs/labs-research/gzipde-an-encrypted-downloader-serving-metasploit

[13] Chen, Qian & Bridges, Robert. (2017). Automated Behavioral Analysis of Malware: A Case Study of WannaCry Ransomware. 454-460. 10.1109/ICMLA.2017.0-119.

[14] Kovacs, E. (2014). XML Files Used to Distribute Dridex Banking Trojan. Retrieved July 09, 2020, from https://www.securityweek.com/xml-files-used-distribute-dridex-banking-trojan

[15] Lu, Xiaofeng & Xiao, Zhou & Jiang, Fangshuo & Shengwei, Yi & Jing, Sha. (2018). ASSCA: API based Sequence and Statistics features Combined malware detection Architecture. Procedia Computer Science. 129. 248-256. 10.1016/j.procs.2018.03.072.

[16] K., Sudhakar & Kumar, Sushil. (2019). An emerging threat Fileless malware: a survey and research challenges. 3. 1. 10.1186/s42400-019-0043-x.

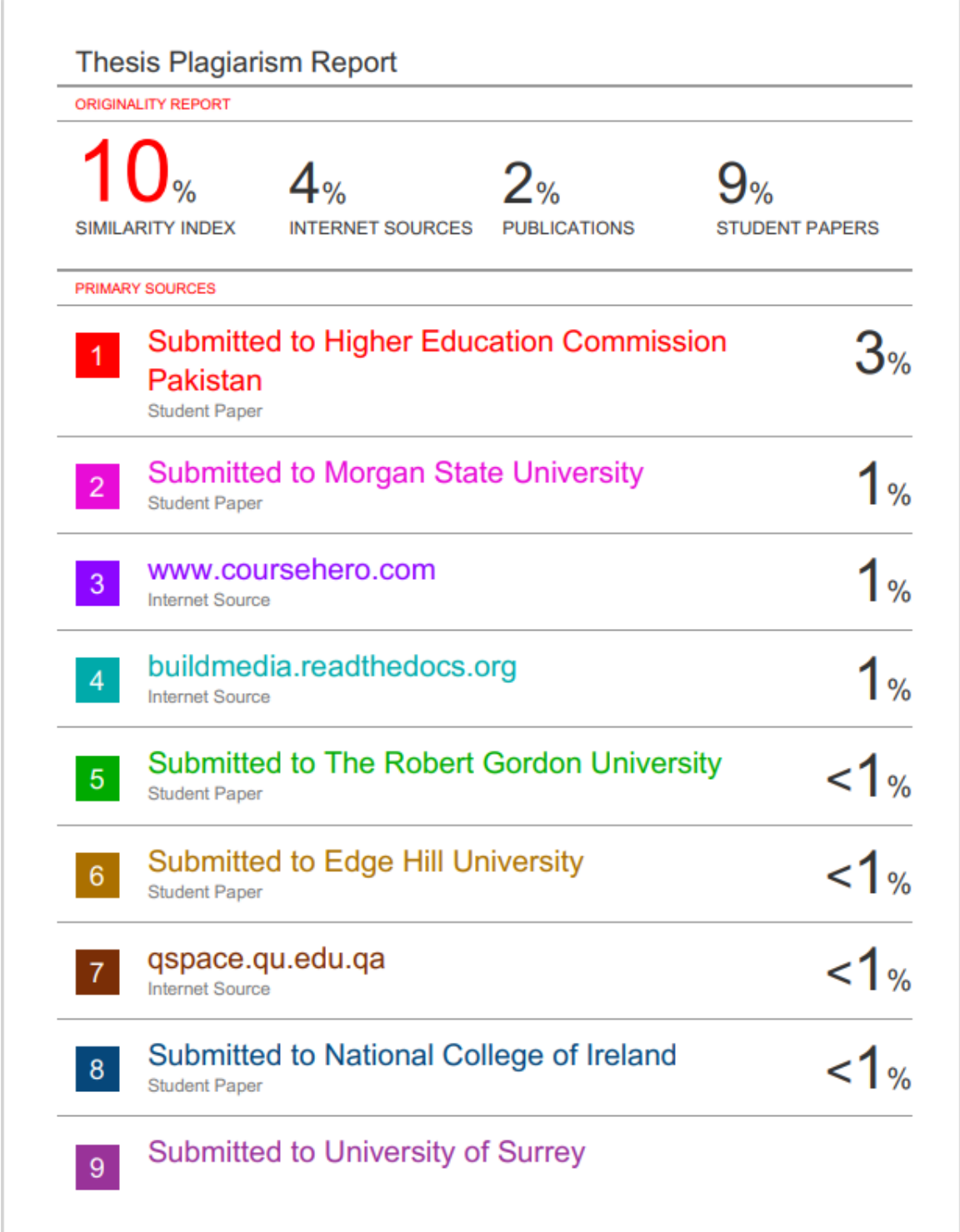
[17] Gorelik, M. (2017). Fileless Malware Attack Trend Exposed. Retrieved July 09, 2020, from https://blog.morphisec.com/fileless-malware-attack-trend-exposed

APPENDIX-A

List of Abbreviations

API	Application Programming Interface
VBA	Visual Basic
CSV	Comma Separated Values
NB	Naïve Bayes
KNN	K-Nearest Neighbors
SVM	Support Vector Machines
UML	Unified Modelling Language
CNN	Convolutional Neural Network
LSTM	Long-Short-Term-Memory
RNN	Recurrent Neural Network
ML	Machine Learning

Thesis Plagiarism Report:



Thesis Plagiarism Report

ORIGINALITY REPORT

10%	4%	2%	9%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Higher Education Commission Pakistan Student Paper	3%
2	Submitted to Morgan State University Student Paper	1%
3	www.coursehero.com Internet Source	1%
4	buildmedia.readthedocs.org Internet Source	1%
5	Submitted to The Robert Gordon University Student Paper	<1%
6	Submitted to Edge Hill University Student Paper	<1%
7	qspace.qu.edu.qa Internet Source	<1%
8	Submitted to National College of Ireland Student Paper	<1%
9	Submitted to University of Surrey	