

**University Institute of Information Technology,**

**PMAS-Arid Agriculture University,**

**Rawalpindi Pakistan**

**API Based Intelligent Malware Detection System**

***By***

**Anjum Shehzad 18-Arid-2609**

**Haris Amir 18-Arid-2643**

**Zohaib Ali Hassan 18-Arid-2735**

***Supervisor***

**Dr. Zeeshan Javed**

***Bachelor of Science in Computer Science (2018-2022)***

**The candidate confirms that the work submitted is their own and appropriate  
 credit has been given where reference has been made to the work of others**.

**DECLARATION**

We hereby declare that this software, neither whole nor as a part has been copied out from any source. It is further declared that we have developed this software documentation and accompanied report entirely on the basis of our personal efforts. If any part of this project is proved to be copied out from any source or found to be reproduction of some other. We will stand by the consequences. No Portion of the work presented has been submitted of any application for any other degree or qualification of this or any other university or institute of learning.

Anjum Shehzad Haris Amir Zohaib Ali Hassan

--------------------------- --------------------------- ---------------------------

**CERTIFICATE OF APPROVAL**

It is to certify that the final year project of BS (CS) “API Based Intelligent Malware Identification System” was developed by “Anjum Shehzad (18-Arid-2609)”, “Haris Amir (18-Arid-2643)” and “Zohaib Ali Hassan(18-Arid-2735)” under the supervision of “Dr Zeeshan Javed” and that in their opinion; it is fully adequate, in scope and quality for the degree of Bachelors of Science in Computer Science.

---------------------------------------

**Supervisor**

---------------------------------------

**(Mr. Zeeshan Javed)**

**External Examiner**

---------------------------------------

**Administrator UIIT**

**Executive Summary**

We aim to present the functionality and accuracy of different machine learning algorithms to detect whether an executable is infected or clean. The first chapter will present a description of the phenomenon of Malware, software programs or pieces of code that aim to hijack computer systems to steal information or to destroy it. We will dive deeper into this topic in order to have some understanding of these malicious programs. After a brief introduction to this phenomenon, we will present the evolution of malware over time. The following is the presentation of the different protection techniques. The second chapter will introduce the field of Artificial intelligence and its benefits. We will further discuss the importance of artificial intelligence in addressing this situation. The algorithms used by us will be described and their benefits will be presented. Artificial intelligence is widely used in this field by antivirus and antimalware programs as well as by these malicious programs, for example Polymorphic Malware uses artificial intelligence algorithms to encrypt itself in a different pattern each time it infects a new mass, becoming increasingly difficult to detect.

**Acknowledgement**

All praise is to Almighty Allah who bestowed upon us a minute portion of His boundless knowledge by virtue of which we were able to accomplish this challenging task.

We are greatly indebted to our project supervisor “**Dr Zeeshan javed**” and our Co-Supervisor “Dr. Saif ur Rehman” for personal supervision, advice, valuable guidance and completion of this project. We are deeply indebted to him for encouragement and continual help during this work.

And we are also thankful to our parents and family who have been a constant source of encouragement for us and brought us the values of honesty & hard work.

Anjum Shehzad Haris Amir Zohaib Ali Hassan

--------------------------- --------------------------- ---------------------------

**Abbreviations**

|  |  |
| --- | --- |
| **SRS** | Software Requirement Specification |
| **PC** | Personal Computer |
|  |  |
|  |  |
|  |  |

**Table of Contents**

**Introduction 1**

[1.1 Brief](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523777) 2

[1.2 Relevance to Course Modules](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523779) 2

[1.3 Project Background](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523780) 3

[1.4 Literature Review](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523780) 3

[1.5 Methodology and Software Life Cycle](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523782) 5

**Problem Definition** 6

[2.1 Problem Statement](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523795) 7

[2.2 Product Functions](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523796) 7

[2.3 Proposed Architecture](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523804) 7

[2.4 Project Deliverables](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523806) 8

[2.5 Operating Environment](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523807) 8

**Requirement Analysis** 9

[3.1 Functional Requirments](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523823) 10

[3.2 Non – Functional Requirments](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523825) 11

[3.2.1 Usability](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523787) 11

[3.2.2 Reliability](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523787) 11

[3.2.3 Performance](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523787) 11

[3.2.4 Supportability](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523787) 11

[3.2.5 Design Constraints](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523787) 11

[3.2.6 Licensing Requirements](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523787) 11

[3.3 Use case Model](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523823) 12

[3.3.1 Use Case Diagarm](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523787) 12

[3.3.2 Actors Discription](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523787) 16

[3.3.3 Use Case Discription](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523787) 17

**The Design** 21

[4.1 UML Structural Diagrams](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523830) 22

[4.1.1 Component Diagram](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523787) 22

[4.1.2 System Component Diagram](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523787) 24

[4.1.3 Package Diagram](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523787) 25

[4.1.4 Deployment Diagram](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523787) 26

[4.2 UML Behavioral Diagrams](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523830) 27

[4.2.1 Activity Diagrams](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523787) 27

[4.2.2 State Machine Diagrams](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523787) 29

[4.3 UML Interaction Diagrams](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523830) 30

[4.3.1 Sequence Diagrams](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523787) 30

[4.4 Node Structure](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523830) 31

[4.5 Communication Design Protocol](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523830) 32

**Implementation** 33

[5.1 Communication Protocol Implementation](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523830) 34

[5.2 PC Application Implementation](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523830) 36

[5.3 Embedded Application Implementation](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523830) 38

[5.4 Wireless Sensor Application Implementation](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523830) 51

**Testing and Evaluation** 52

[6.1 Verification](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523830) 53

[6.1.1 Functional Testing](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523787) 53

[6.1.2 Static Testing](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523787) 57

[6.2 Validation](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523830) 57

[6.3 Usability Testing](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523830) 57

[6.4 Unit Testing](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523830) 57

* 1. [Integration Testing](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523830) 57

6.6 [System Testing](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report%20(1).docx#_Toc268523830) 57

**GUI** 58

**Future Work** 70

**References** 72

**List of Figures**

Fig 1.1 Block Diagram 8

Fig 2.1 Use Case Diagram 9

# Chapter 1: Introduction

# Brief

"Malware" is an abbreviation for "malicious software", it is used as a single term to refer to Viruses, Trojans, Worms, etc. These programs have a variety of features, such as stealing, encrypting or deleting sensitive data, modifying or hijacking basic computer functions, and monitoring computer activity. show user permission.

**1.1.2**

**Computer virus:**

It is generally a program that is installed outside the user's will and can cause damage to both the operating system and the hardware (physical) elements of a computer.

Effects generated by the virus:

* Destruction of files.
* Changing the file size.
* Delete all information on the disc, including formatting it.
* destruction of the file allocation table, which makes it impossible to read the information on the disk.

**Worms:**

Computer worms are programs with destructive effects that use communication between computers to spread. Worms have common features with viruses, i.e., Worms are able to multiply like viruses, but not locally, but on other computers. It uses computer networks to spread to other systems.

Types of computer worms:

* E-Mail worms.
* Instant messaging worms.
* Internet worms.
* IRC worms.
* File-sharing files on the network Trojan horses Trojan horses are "disguised" programs that try create gaps in the operating system to allow a user to access the system. Trojans do not have the facility to self-multiply like computer viruses.

**Ransomware:**

Ransomware is a type of malware that blocks the victim's access to the computer and demands payment of a reward. The reward and the official reason why the victim should pay depends on the type of virus. Some versions of ransomware claim that the payment should be made to avoid punishment by a government authority (usually the FBI or a local agency), others inform that this is the only way to decrypt encrypted data.

Effects generated by ransomware are able to encrypt sensitive user data can delete predetermined documents, multimedia objects and any other files that contain important information. They can also try to delete essential components of the system or important parts of other software.

**What spyware can be used for?**

* To steal sensitive information. Such programs are interested in personal information, such as credentials, passwords, bank details, and other similar information. In addition, they can monitor the user's online activity, track their web browsing habits, and send all this data to a remote server.
* Show unwanted creatives. Spyware can display a large number of annoying pop-up ads. Such activity is more associated with adware parasites.
* Redirecting users to questionable or malicious websites against their will. In addition, some types of spyware threats are able to change web browser settings and change the search engine and home page.
* Create numerous links in the search results of the victim and redirect him / her to the desired places (third party spyware sites, websites and other associated fields).

# Relevance to Course Modules:

Our project is related to various courses we have studied in our degree which are mentioned below:

* Object oriented Programming.
* Artificial Intelligence.
* Software Engineering-I.
* Database.

Above mentioned courses not only helped us in developing the UML and Class diagrams as well as Use Cases of the model.

# Project Background

The first versions of Malware were primitive, they infested various machines through floppy disks. With the evolution of Networking and the maturation of the Internet, malware authors have adapted their malicious codes to take full advantage of this new communication environment. Below is a brief overview of the evolution of malware over time.

**1.3.1** The Years 1971-1999

* 1971-Creeper: An experiment designed to test how a program can move between computers.
* 1974-Wabbit: A program that multiplies itself at an accelerated pace, until the speed of the system slows down, the performance is measured, the system is reduced and eventually collapses.
* 1982-Elk Cloner: Written by a 15-year-old child, Elk Cloner is one of the first viruses, and widespread, to multiply itself and display a short "poem" to the infected person: “It will get on all your disks; It will infiltrate your chips; Yes, it's a Cloner! ”
* 1986-Brain Boot Sector Virus: Considered the first virus to infect MS-DOS computers.
* 1986 — PC-Write Trojan: Malware disguised as one of the oldest Trojans as a popular program called "PC-Writer." Once on a system, it deletes all files of a user.
* 1988 — Morris Worm: Infected a substantial percentage of computers connected to ARPANET, the predecessor of the Internet, which brought the network to its knees in 24 hours. this Worm marked a new beginning for malicious software.
* 1991 — Michelangelo Virus: The virus was designed to erase information from hard drives on March 6, the birthday of the famous Renaissance artist.
* 1999 - Melissa Virus: used Outlook addresses from infected machines and was sent to 50 people at once.

**1.3.2** The Years 2000-2010

* 2000 – ILOVEYOU Worm: the worm infected about 50 million computers. The damage caused major corporations and government agencies, including portions of the Pentagon and the British Parliament, to shut down their e-mail servers. Worms have spread globally and cost more than $ 5.5 billion in damage.
* 2003 – SQL Slammer Worm: One of the fastest spreading worms of all time, SQL Slammer infected nearly 75,000 computers in ten minutes. The worm has had a major effect worldwide, slowing down worldwide Internet traffic by denial of service.
* 2004 – Cabir Virus: Although this virus has caused some damage, it is noteworthy because it is widely recognized as the first cell phone virus.
* 2005 – Koobface Virus: One of the first cases of malware to infect PCs and then spread to social networking sites. If rearranged, the letters in "Koobface" are old, and you get "Facebook". The virus has also attacked other social networks such as MySpace and Twitter.
* 2008 – Conficker Worm: A combination of the words “configure” and “ficker”, this sophisticated worm has caused some of the 11 worst damage observed since Slammer appeared in 2003.

**1.3.3** 2010- Present

* 2010 – Stuxnet Worm: Shortly after its release, security analysts openly speculated that the malware was designed to explicitly attack Iran's nuclear program and include the ability to affects hardware and software. The incredibly sophisticated worm is considered to be the work of a whole team of developers, making it one of the most intensive malware resources created to date.
* 2011 — Zeus Trojan: Often detected for the first time in 2007, the author of the Trojan Zeus released the code to the public in 2011, giving a new life to malware. Sometimes called the Zbot, this Trojan has become one of the most successful pieces of botnet software in the world, impacting millions of machines.
* 2013 – Cryptolocker: had a significant impact globally and helped fuel the ransomware era. • 2014 – Backoff: Malware designed to compromise Point-of-Sale (POS) systems to steal credit card data.
* 2016 – Cerberus: One of the most prolific crypto-malware threats. At one point, Microsoft found more company PCs infected with Cerberus than any other family of ransomware.
* 2017 – WannaCry Ransomware: Exploiting a vulnerability first discovered by the National Security Agent, WannaCry Ransomware has knelt down a number. systems in Russia, China, the United Kingdom and the United States, blocking access to data and demanding redemption or loss of everything. The virus has affected at least 150 countries, including hospitals, banks, telecommunications companies, warehouses and many other industries.

# Literature Review

In this chapter we will go through the previous work which has been done in this field. We will discuss some of the projects which have been made and what was the purpose behind the creation and what benefits did they give to the field.

The list of the projects which we will be discussing is given below:

* Deep learning based Sequential model for malware analysis using Windows exe API Calls.
* Optimal feature configuration for dynamic malware detection.
* Toward Identifying APT Malware through API System Calls.

**1.4.1 Deep learning Based Sequential Model for Malware Analysis using Windows exe API calls:**

This study is focused on metamorphic malware, which is the most advanced member of the malware family. It is quite impossible for anti-virus applications using traditional signature-based methods to detect metamorphic malware, which makes it difficult to classify this type of malware accordingly. Recent research literature about malware detection and classification discusses this issue related to malware behavior. The main goal of this project is to develop a classification method according to malware types by taking into consideration the behavior of malware. This search was started by developing a new dataset containing API calls made on the windows operating system, which represents the behavior of malicious software. The types of malicious malware included in the dataset are Adware, Backdoor, Downloader, Dropper, spyware, Trojan, Virus, and Worm. The classification method used in this study is LSTM (Long Short-Term Memory), which is a widely used classification method in sequential data. The results obtained by the classifier demonstrate accuracy up to 95% with 0.83 $F\_1$-score, which is quite satisfactory. We also run our experiments with binary and multi-class malware datasets to show the classification performance of the LSTM model.

;

**1.4.2 Optimal Feature Configuration for Dynamic Malware Detection:**

Applying machine learning techniques to malware detection is a common approach to try to overcome the limitations of signature-based methods. However, it is difficult to engineer a set of features that characterizes the samples properly, especially when various file types may be a vector of infection. In this work, researchers configured several feature sets for dynamic malware detection extracted from API calls, including an alternative scheme grouping calls in categories, network activity, signatures from the Cuckoo sandbox report, and some interactions with the file system and registry. They tested the combinations of these feature sets to ascertain whether they are good enough to distinguish between benign and malicious samples from a dataset containing several file types, obtained from public sources and applied statistical inference to measure the differences in the performance between the feature sets, and the hyperparameter optimization algorithms applied to construct the models. They also unbalance the datasets to evaluate the model performance on more realistic scenarios in which not many malware samples are available. Although all studied feature configurations provide accuracies greater than 0.98, and several of them a Matthews correlation coefficient greater than 0.95 in the unbalanced datasets, statistically meaningful differences appear, so they analyzed the results to determine which is the optimal set of features. They obtain a model that achieves an accuracy of 0.9937 in the balanced dataset and a Matthews correlation coefficient of 0.964 in the unbalanced dataset with 5% of malware.

# 1.4.3 Toward Identifying APT Malware through API System Call:

Self-developed malware was usually used by advanced persistent threat (APT) attackers to launch APT attacks. therefore, we can enhance the understanding and cognition of APT attacks by comprehending the behavior of APT malware. Unfortunately, the current research cannot effectively explain the relationship between the recognition, detection, and defense of APT. The model of similar studies also lacks an explanation about it. To defend against APT attacks and inquire about the similarity of different APT attacks, this study proposes an APT malware classification method based on a combination of multiple deep learning algorithms and transfer learning by collecting malware used in several famous APT groups in public. By extracting the application programming interface (API) system calls, with the vector representation of features by combining dynamic LSTM and attention algorithm, we can obtain API at different APT family’s classification contributions trained dynamic. thus, we used transfer learning to perform multiple classifications of the APT family. This study aims to reduce the burden of network security staff from reviewing a large number of suspicious files when defending against APT attacks. Additionally, it can effectively intercept them in the initial invasion stage of APT to perform targeted defense against specific APT attacks by combining threat intelligence in public. The experimental result shows that the proposed method can achieve 99.2% in distinguishing common malware from APT malware and assign APT malware to different APT families with an accuracy of 95.5%.

# Methodology and Software Lifecycle for This Project

# There are different types of methodologies are used to building a software or any of the project. We have studied all the types of methodologies that can be used but from all of them we select the method that best fit to our project is “Extreme Programming”.

**We are selecting this model because it is:**

Best Suited For: Projects that require maintaining stringent stages and deadlines or projects that have been done various times over where chances of surprises during the development process are relatively high.

One more reason is that this method is applied where the requirements are not very much clear. So that will happen with our project too so that’s why we are selecting this Method.

In systems design, and particularly software design, a common methodology for the development of a new system is the Systems Development Life Cycle, or SDLC. The SDLC contains the following phases of systems development:

• **Planning**

Determine the purpose of the system.

• **Analysis**

Determine what the system needs to do, the goals for the system and how to determine if those goals have been met.

• **Design**

Determine how the system will work, what the overall architecture is, and determine what steps would need to be taken to construct an actual system.

• **Implementation**

Using the existing design, we will construct a system to meet the requirements of the project.

• **Testing**

Establish that the constructed system actually does meet the requirements detailed in the design.

• **Maintenance**

Fix bugs in the system, which are essentially differences between the design (requirements) and the constructed system (reality). As the design inevitably changes, update the actual system to match these changes.

**Chapter 2: Problem Definition**

# Problem Statement:

Applying machine learning techniques to malware detection is a common approach to try to overcome the limitations of signature-based methods. Signature based detection uses the technique of pattern recognition in search of predefined signatures stored in a database. Signature based detection technique is commonly used up by Anti-viruses as their basic block. This technique does not stand on its hundred percent as it is unable to detect malware whose signature is not present in its database record. These methods are unable to detect unknown malware variants and also requires high amount of manpower, time, and money to extract unique signatures. These are the main disadvantages of these methods.

# Purposed Architecture:

1. Run malware sample in sandbox environment.
2. Extract Features from malware reports.
3. Test and train model
4. Perform Classification
5. After classification perform prediction

# Proposed Architecture:

To solve the problem, we configure several feature sets for dynamic malware detection extracted from API calls, including an alternative scheme grouping calls in categories, network activity, signatures from the Cuckoo sandbox report, and some interactions with the file system and registry. API call sequences is one of the most attractive way that reflects the behavior of a piece of code like malware. We test combinations of these feature sets to ascertain whether they are good enough to distinguish between benign and malicious samples from a dataset containing several file types, obtained from public sources.

# Project Deliverables:

* Documentation
* Sandbox Environment
* Features Extraction
* Desktop Application
* Classification System

# Operating Environment:

* VMware
* Virtual Box
* Windows 10(Depends on user)
* Cuckoo Sandbox

# Chapter 3: Requirement Analysis

Software Requirements Specification (SRS) report should be included in this chapter.

# Use Cases:

Use cases are a widely used and highly regarded format for capturing requirements. Before writing functional requirement use cases can help you to understand the requirements in the way user expect. Following table presents you not only the template to write use case(s) as well as guides you to write each section with example.

|  |  |
| --- | --- |
| **Use case ID** | **U\_id:1** |
| Use case Name | Submit Malware Samples |
| Actor Name | Direct User |
| Description | By selecting the files from the system after processing it will generate report about selected malware sample |
| Trigger | To accomplish this task we Click on button (Select File) |
| Precondition | NO info about samples |
| Post Condition | Report will provide complete information on the basis on Malware |
| Normal Flow | 1.Chose the malware samples from system  2.Submit malware samples to sandbox  4. Then select action report generate. |
| Alternative Flow | If success fully then move to the next process else request user to try again |
| Expectation | During taking action some errors can be occurs  1.No file selected  2.Folder having no malware samples  3.Uploading Error  In such type of condition user request to restart from initial stage. |
| Includes: | Null |

|  |  |
| --- | --- |
| **Use case ID:** | **U\_id:2** |
| Use case Name | Sandbox Environment |
| Actor Name | System |
| Description: | After submitting the malwares in sandbox environment like cuckoo sandbox. Then cuckoo will run those samples in isolated virtual environment like on windows 7 installed on virtual box and connected with cuckoo sandbox |
| Trigger | For this process cuckoo sandbox will run samples on isolated environment using connected network. |
| Precondition | Isolated environment must be configured with cuckoo sandbox and in restore condition. |
| Post Condition | Generate report and save in database. |
| Normal Flow: | 1. Run malware samples in isolated environment.  2. Report generation based on executed malware samples  3.Check Successful or not |
| Alternative Flow: | If save successfully then move to the next step else again request save. |
| Exception: | During the performing action some error can be take place:   1. Cuckoo sandbox is not configured with isolated environment properly. 2. Isolated environment is not in restore mode. |
| Includes: | Report Generation based on executed malware sample. |

|  |  |
| --- | --- |
| **Use case ID:** | **U\_id:3** |
| Use case Name | Feature Extraction |
| Actor Name | User |
| Description: | After the report is generated based on executed malware sample. Next step is to extract features from report in JSON form. |
| Trigger: | To complete this all reports should be collected from cuckoo sandbox. |
| Precondition | No feature is extracted. |
| Post condition | After extracting features save in CSV file. |
| Normal Flow | 1. Get report from Cuckoo Sandbox.  2. Extract features from reports.  3. Prepare dataset from these extracted features. |
| Alternation | Successfully move next else again try to extract features from report. |
| Exception: | During the performing action some error can be take place  1. Index number of desired feature is not defined or correct.  2. Desired feature may not exist.  3.Fail to Extract features |
| Includes: | Dataset |

|  |  |
| --- | --- |
| Use case ID: | U\_id:4 |
| Use case Name | Classification |
| Actor Name | Direct user |
| Description | For Classification of Malware first extract features then prepare dataset based on these extracted features. After dataset perform classification using deep learning model. |
| Trigger | Dataset is prepared based on extracted features. |
| Precondition | Non Classified dataset |
| Post Condition | Classified |
| Normal Flow | 1)Get Dataset  2) Classify the malware in families  3) Then save trained model in H5 file for prediction. |
| Alternative | If success then movie to the final steps else go back for classification |
| Exception | Dataset is not prepared properly. |
| Includes | Null |

**3.2 Functional Requirements:**

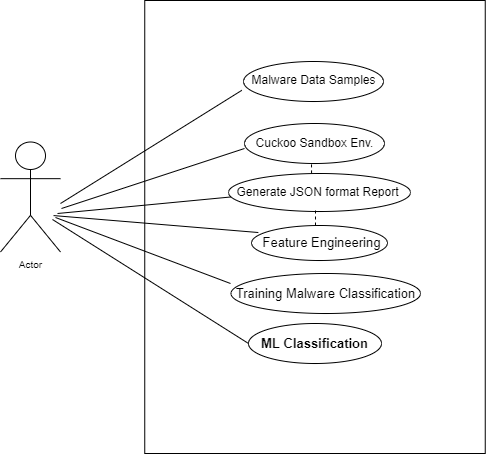
|  |  |
| --- | --- |
| **Functional Requirement No.** | **Functional Requirement Description** |
| Req.1 | User Collect Malware Samples. |
| Req.2 | Submit Malware Samples in sandbox environment. |
| Req.3 | Get generated reports from sandbox. |
| Req.4 | Extract features from these reports and prepare dataset |
| Req.5 | Now test and train dataset. |
| Req.6 | After testing and training prepare model for classification |

**3.3 Non-Functional Requirements**

|  |  |
| --- | --- |
| **Non-Functional Requirement No.** | **Non-Functional Requirement Description** |
| NFR1 | System will take sandbox generated malware reports from user as input and process it. |
| NFR2 | System will extract features from malware reports and prepare CSV File. |
| NFR3 | System will provide interface by using Tkinter. |
| NFR4 | MD5 will be used in for prediction. |

1. **Usability:** System should be easy to extend. The code should be written in a way that it favors implementation of new functions. It will provide the up to date information with good performance to satisfy user needs.
2. **Reliability:** This app should provide appropriate answers to the user. This app should be able to interact efficiently with the user.
3. **Integrity:** This desktop application will requires specific android version to run. It also requires an active internet connection to work and to exchange queries to provide information to the user.

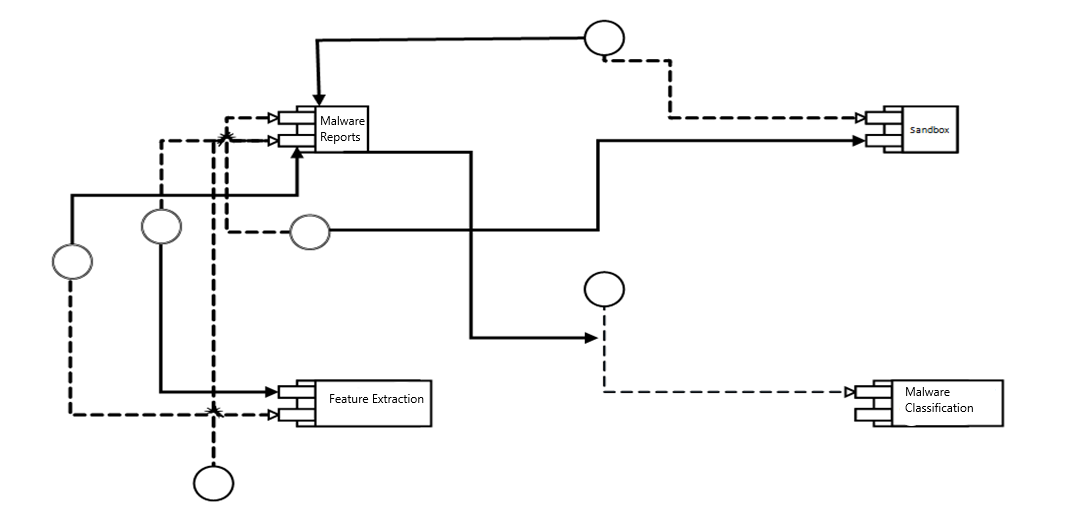
**3.4 Use Case Diagram**

****

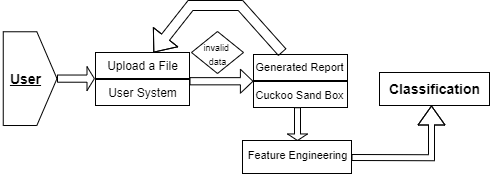
**Chapter 4: Design and Architecture**

**4.1 UML Structural Diagrams**

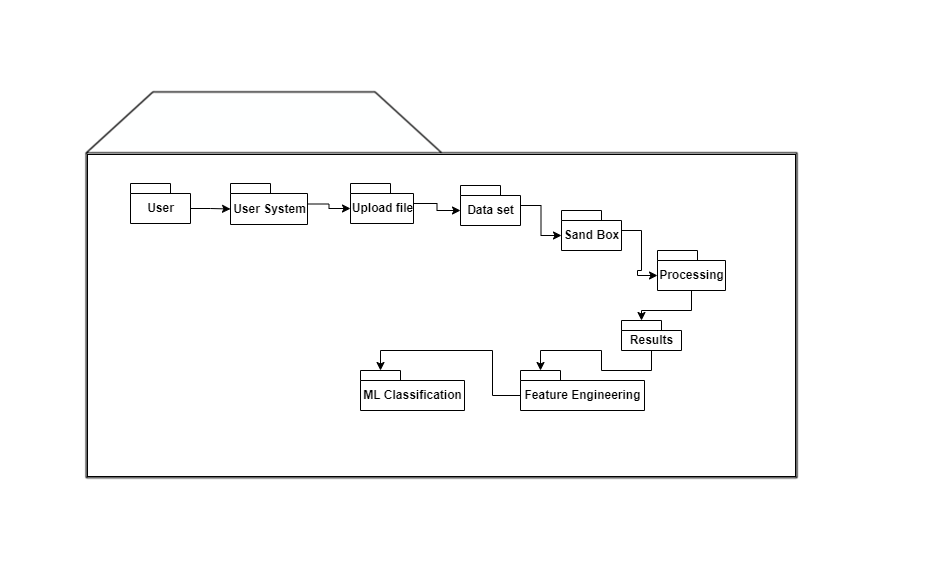
[4.1.1 Component Diagram](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report.docx#_Toc268523787)

****

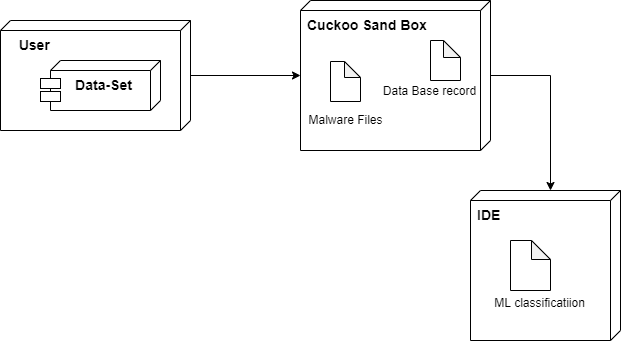
[**4.1.2 System Component Diagram**](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report.docx#_Toc268523787)**:**



[**4.1.3 Package Diagram**](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report.docx#_Toc268523787)



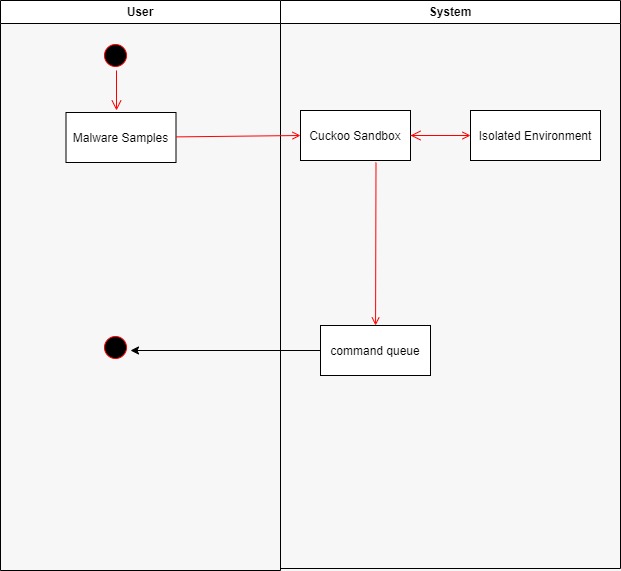
[**4.1.4 Deployment Diagram**](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report.docx#_Toc268523787)



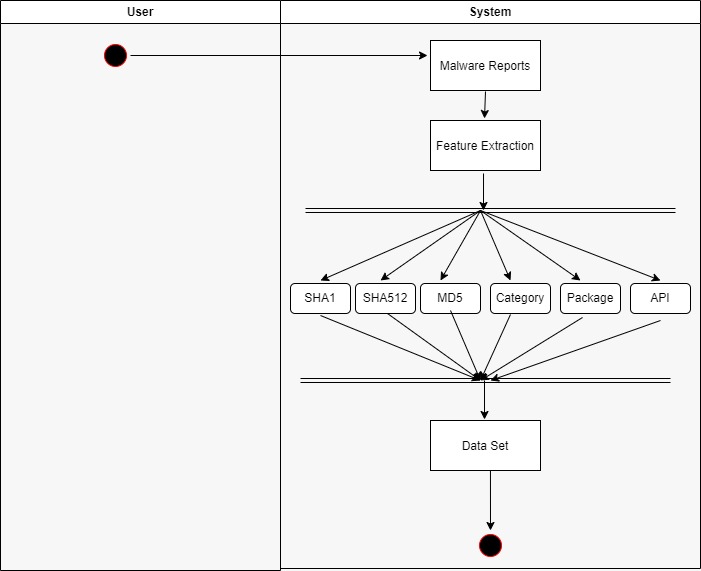
**4.2. UML Behavioral Diagram**

**4.2.1 Activity Diagram**

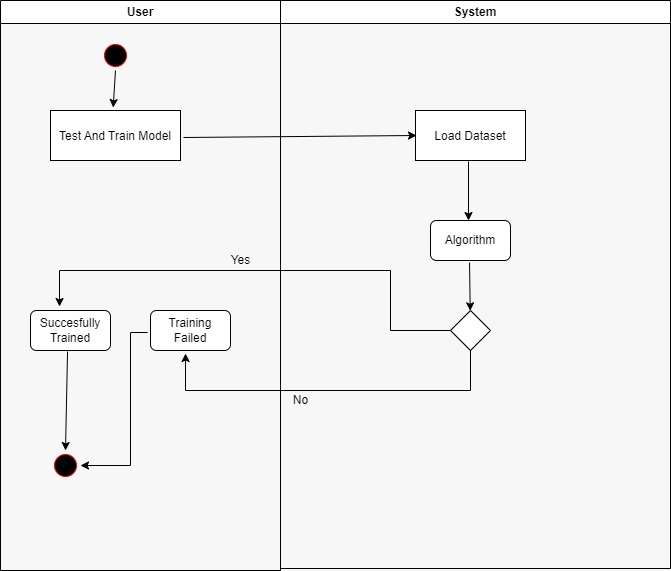
**Run Malware Samples**

****

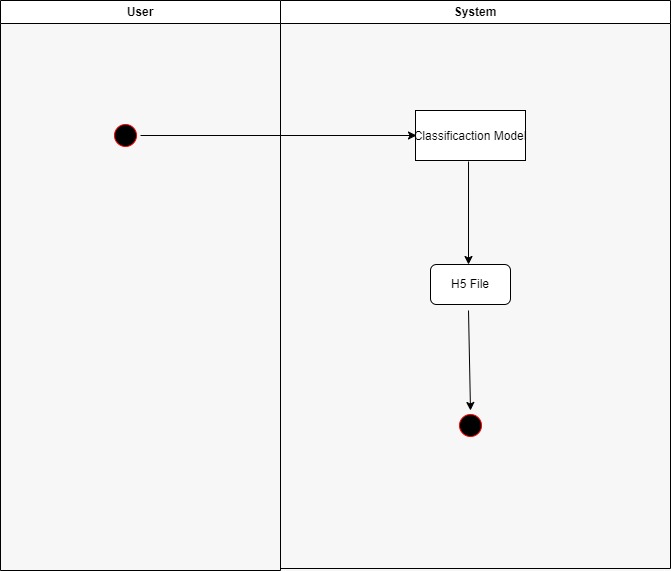
**Feature Extraction**



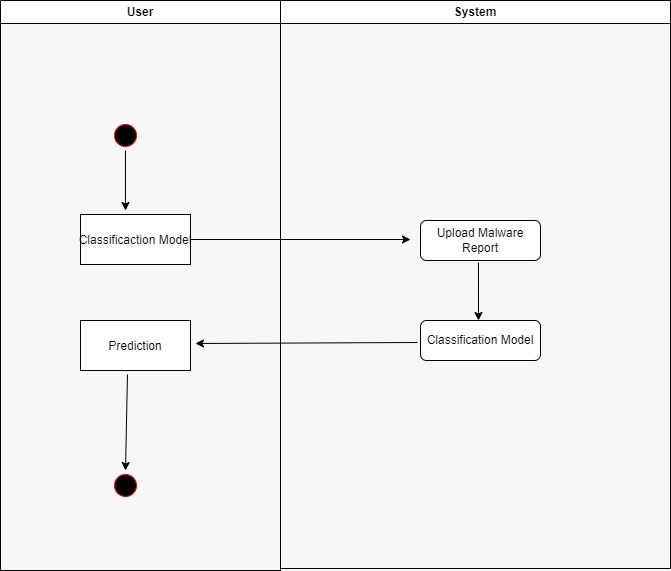
**Test and Train Model**



**Saving Classification Model in H5 File**

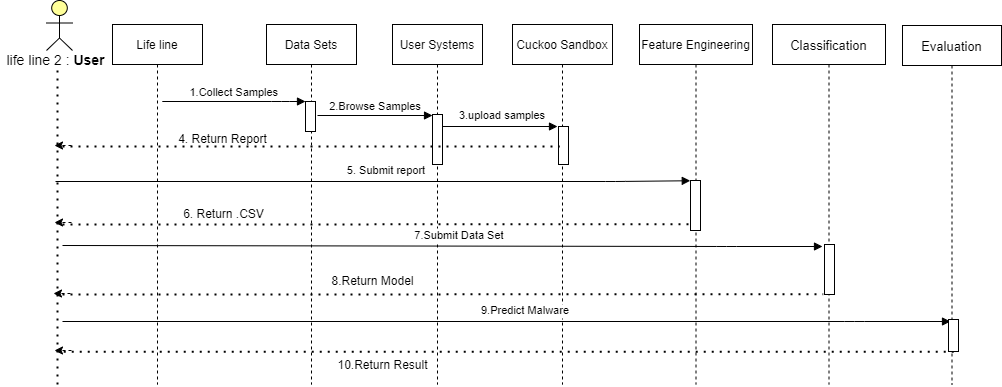


**Predict Malware**



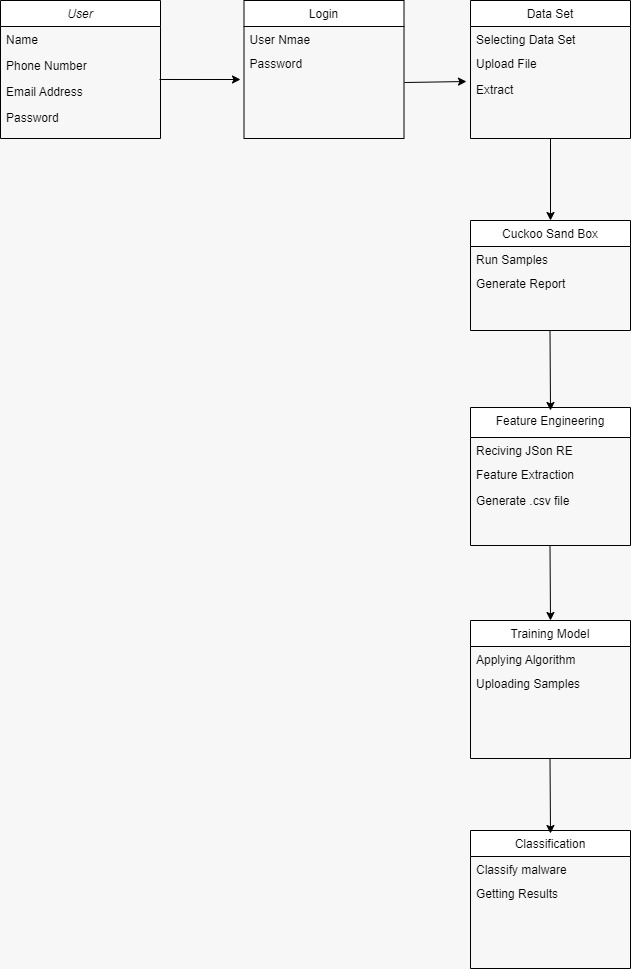
[**4.3 UML Interaction Diagrams**](file:///C:\Users\MudassirRiaz\Downloads\FYP%20Final%20Report.docx#_Toc268523830)

4.3.1 Sequence Diagram



**4.4 Class Diagram**

The class diagram is used to show the different objects in a system, their attriutes, their operations and the relationship among them.

****