

**University Institute of Information Technology,**

**PMAS-Arid Agriculture University,**

**Rawalpindi Pakistan**

**Detecting Malware Using Deep Learning**

***By***

**Muhammad Aamir Manzoor 17-ARID-1496**

**Abdul Rehman 16-ARID-839**

***Supervisor*Dr./Mr. Zeeshan Javed**

***Bachelor of Science in Computer Science (2017-2021)***

***OR Information Technology OR Software Engineering***

**The candidate confirms that the work submitted is their own and appropriate  
 credit hasbeen 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.

Muhammad Aamir Manzoor Abdul Rehman

----17-Arid-1496---- ----16-Arid-839----

**CERTIFICATE OF APPROVAL**

It is to certify that the final year project of BS (CS/IT/SE)“Detecting Malware using Deep Learning” was developed by“Muhammad Aamir Manzoor**, (17-Arid-1496) #”**, and “Abdul Rehman**, (16-Arid-839) #”** under the supervision of“Supervisor Name” and that in their opinion; it is fully adequate, in scope and quality for the degree of Bachelors of Science in Computer Science/Information Technology/Software Engineering.

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

**Supervisor**

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

**External Examiner (If any)**

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

**Administrator UIIT**

**Executive Summary**

In public places, there is often a need for monitoring people and different activities going on, which can be referred later for many reasons including security. Appointing humans for this task involves many problems such as increased employee hiring, accuracy problem, trust, no proof for later use, and also the fact that a human can remember things till a certain time limit. Talking about the current security system, they use dumb still cameras with a continuous recording facility ir-respective of the fact that any event may happen or not. Moreover they are usually pointing at a specific user defined locations so more than one cameras are required to cover the entire region.

To prevent all these problems from prevailing, the CSCS is developed. It is a surveillance system, which provides solution to many of these problems. It is a stand-alone application which doesn’t require any computer to operate. It monitors different situations using a camera which is able to rotate intelligently based on sensor messages and captures the scene in the form of video or photos later reference as well.

**C**ustomizable**S**urveillance **C**ontrol **S**ystem**(CSCS)** is a surveillance system that can be assigned a sensor type as in our case a heat sensor is used, it works accordingly, rotates the camera upon event detection and perform user defined actions like capturing video and stores them, for the future use.

It is an embedded system consisting of Linux fox kit with embedded a running server application also a camera, USB storage device and a sensor node base station is attached with fox kit. LAN communication is used by user to download the videos and to operate the system manually.

**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. Kashif Sattar” and ourCo-Supervisor “Dr. Tariq Ali” for personal supervision, advice, valuable guidance and completion of this project. We are deeply indebted to him/her/them forencouragement 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.

Muhammad Aamir Manzoor Abdul Rehman

----17-Arid-1496---- ----16-Arid-839----

**Abbreviations**

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

**Table of Contents**

**Introduction 1**

[1.1 Brief](#_Toc268523777) 2

[1.2 Relevance to Course Modules](#_Toc268523779) 2

[1.3 Project Background](#_Toc268523780) 3

[1.4 Literature Review](#_Toc268523780) 3

[1.5 Methodology and Software Life Cycle](#_Toc268523782) 5

**Problem Definition** 6

[2.1 Purpose](#_Toc268523795) 7

[2.2 Product Functions](#_Toc268523796) 7

[2.3 Proposed Architecture](#_Toc268523804) 7

[2.4 Project Deliverables](#_Toc268523806) 8

[2.5 Operating Environment](#_Toc268523807) 8

[2.6 Assumptions and Dependencies](#_Toc268523808) 8

**Requirement Analysis** 9

[3.1 Functional Requirments](#_Toc268523823) 10

[3.2 Non – Functional Requirments](#_Toc268523825) 11

[3.2.1 Usability](#_Toc268523787) 11

[3.2.2 Reliability](#_Toc268523787) 11

[3.2.3 Performance](#_Toc268523787) 11

[3.2.4 Supportability](#_Toc268523787) 11

[3.2.5 Design Constraints](#_Toc268523787) 11

[3.2.6 Licensing Requirements](#_Toc268523787) 11

[3.3 Use case Model](#_Toc268523823) 12

[3.3.1 Use Case Diagarm](#_Toc268523787) 12

[3.3.2 Actors Discription](#_Toc268523787) 16

[3.3.3 Use Case Discription](#_Toc268523787) 17

**The Design** 21

[4.1 UML Structural Diagrams](#_Toc268523830) 22

[4.1.1 Component Diagram](#_Toc268523787) 22

[4.1.2 System Component Diagram](#_Toc268523787) 24

[4.1.3 Package Diagram](#_Toc268523787) 25

[4.1.4 Deployment Diagram](#_Toc268523787) 26

[4.2 UML Behavioral Diagrams](#_Toc268523830) 27

[4.2.1 Activity Diagrams](#_Toc268523787) 27

[4.2.2 State Machine Diagrams](#_Toc268523787) 29

[4.3 UML Interaction Diagrams](#_Toc268523830) 30

[4.3.1 Sequence Diagrams](#_Toc268523787) 30

[4.4 Node Structure](#_Toc268523830) 31

[4.5 Communication Design Protocol](#_Toc268523830) 32

**Implementation** 33

[5.1 Communication Protocol Implementation](#_Toc268523830) 34

[5.2 PC Application Implementation](#_Toc268523830) 36

[5.3 Embedded Application Implementation](#_Toc268523830) 38

[5.4 Wireless Sensor Application Implementation](#_Toc268523830) 51

**Testing and Evaluation** 52

[6.1 Verification](#_Toc268523830) 53

[6.1.1 Functional Testing](#_Toc268523787) 53

[6.1.2 Static Testing](#_Toc268523787) 57

[6.2 Validation](#_Toc268523830) 57

[6.3 Usability Testing](#_Toc268523830) 57

[6.4 Unit Testing](#_Toc268523830) 57

* 1. [Integration Testing](#_Toc268523830) 57

6.6 [System Testing](#_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 detection is a method for judging the security of computer software, it is a key part of software safety research. Many malware detection technologies, such as malware code structure Therefore, the advancement and completeness of the detection method will determine the effectiveness of the malware detection project and control scheme. Analysis, function analysis, which are all based on the detection and the classiﬁcation. Feature based malware protection schemes such as antivirus software are still the most universal network security products in the current application. Malware is a malicious software or files that are harmful when we are going to execute that type of files or software that can damage the system or may create any effect on the system. Therefore malware affects our daily life and our system need continuous security to defend against that attacks. To provide protection against these attacks many researchers and teams are working on that. Lot of people uses the machine learning to automatic detection and classify the malware to secure their systems.

# Relevance to Course Modules

In our whole degree studied following subjects that have to relate in our project

* Database management system (DBMS)
* Software engineering-1
* Software engineering-2
* Web development.

A database typically requires a comprehensive database software known as database system. A database system serves as an interface between the database and its end users or program allowing users to retrieve, update and manage how the information is organized and optimized.

Software engineering is the application of principles used in the field of engineering, which usually deals with physical systems, to the design, development, testing, deployment and management of software systems.

Web development is the coding or programming that enables website functionality, per the owner's requirements. It mainly deals with the non-design aspect of building websites and Applications which includes coding and writing markup.

# Project Background

In this project we will perform malware detection, we will choose a dataset and perform malware detection on that dataset. This project is being developed for malware detection. This project will identify the malicious files on the system and classify the type of malware. After that we create our web interface and configure with cuckoo sandbox then execute files on cuckoo and associate API calls with parent process, extracted from ‘calls’ elements of cuckoo sandbox reports.

# Literature Review

[1](Gibert, Mateu, & Planes, 2010)The struggle between security analysts and experts and malware developers is a never ending battle with the complications of malware changing as quickly as innovation grows. Current state-of-the-art research focus on the development and application of machine learning techniques for malware detection due to its ability to keep pace with malware evolution. This survey aims at providing an organized and detailed overview of machine learning techniques for malware detection and in particular deep learning techniques. It provides absolute explanation of the methods and features in a traditional machine learning work ﬂow for malware detection and classiﬁcation it explores the challenges and limitations of traditional machine learning and it analyzes recent trends and developments in the ﬁeld with special importance on deep learning approaches.

[2](M. Siddiqui et al, 2009) Used Data Mining for detection of Worms. They used variable length instruction sequence. Their Primary data set consists of 2,775 Windows PE files, in which in which 1,444 were worms and 1,330 were benign. They performed detection of compilers, common packers and crypto before disassembly of files. Sequence reduction was performed and 97% of the sequences were removed. They used Decision Tree, Bagging and Random Forest models using. Random forest performed slightly better than the others.

[3]([Shabtai et al, 2009)](#_bookmark141)Provide a taxonomy for malware detection using machine learning algorithms by reporting some feature types and feature selection techniques used in the literature. They mainly focus on the feature selection techniques(Gain ratio, Fishers core, document frequency, and hierarchical features election) and classiﬁcation algorithms (Artiﬁcial Neural Networks, Bayesian Networks, Natïve Bayes, K-Nearest Neighbor, etc). In addition, they review how ensemble algorithms can be used to combine a set of classiﬁers

[4](J. Z. Kolter et al, 2004.) Used n-gram analysis and data mining approaches to detect malicious executables in the wild. The authors used a hexdump utility to convert each executable to hexadecimal code in an ASCII format and produced n-gram features by combining each four-byte sequence into a single term. Their primary dataset consisted of 1971 clean and 1651 malicious programs They used different classifiers including Instance-based Learner, TFIDF, Naive-Bayes, Support vector machines, Decision tree, boosted Naive-Bayes, SVMs and boosted decision tree. They used information gain to select valued features which are provided as input to all classifiers. The area under an ROC curve (AUC) is a more complete measure compared with the detection accuracy as they reported that the boosted decision trees outperform rest of the classifiers for both classification problems.

# Analysis from Literature Review (in the context of your project)

We have studied the lot of systems and literatures about the malware detection. Some of them are on machine learning and some are on the deep learning techniques. Deep learning technique is pretty good for this system. Machine learning is good but in the machine learning lot of effort is required for the feature engineering. In the deep learning method Auto-Encoders are mostly used for the malware detection system. We are going to use the Deep Learning to build this model. In this model we will develop the system that will detect the malware using deep learning and will classify the malware. It will identify the malware and will also identify it according to their family.

# Methodology and Software Life cycle

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**

**2.1. Purpose**

Purpose of the project is to detect the malware and malicious files from a given dataset. Many of the malicious codes or files attached to your files when you are going to download anything from the internet. Different type of malwares are present on the internet world, every specie has its own purpose and method that how to attack on any computer. So that’s why we are going to build that type of model that can classify any type of malware. Our model is going to classify the family of the malware by which the user administrator of the system can identify that which type of malware is that and against that which action could be performed. How much that file or malware can be harmful for the system? What type of attack is that? What is the purpose of the attacker? The purpose of malware detection is usually to provide the information you need to respond to a network. Your goals will typically be to determine exactly what happened, and to ensure that you’ve located all infected machines and files.

# Problem Statement

Software applications, including malware, naturally evolve over time due to changes resulting from adding features, ﬁxing bugs, porting to new environments and platforms. These changes are expected to be introduced relatively infrequently. Additionally, successive versions of the software are expected to be highly similar to previous versions, with few exceptions such as when the code base under goes signiﬁcant refactoring and there are changes in the compilers or libraries linked to the software. Moreover, the similarity between previous and future versions is expected to degrade slowly over time. In consequence, the prediction quality decays over time as malware evolves and new variants and families appear. Thus, in order to build high quality models for malware detection and classiﬁcation, it is important to identify when the model shows signs of degradation and there by it fails to recognize new malware. Existing solutions aim at periodically retrain the model with the hope that it will automatically adapt to changes in malware overtime. The process of retraining the model can be done from scratch, partially and incrementally, were incremental retraining refers to the process of retraining a given model with new labeled malware samples and all previous training samples without for getting the knowledge obtained from prior datasets.

# Product Functions

The main function of the product is to detect the malware and malicious files from a given dataset. It will detect the malware and classify the malware to their respective family.

# Proposed Architecture

This system is to accurately detect new malware (unknown malware) binaries using a number of data mining techniques. The data set consists of malware files that were collected from two sources in addition to a set of benign files. This proposed architecture is that first to deploy the cuckoo sandbox, which perform dynamic analysis. An malware detection environment is then setup, which consists of all the necessary components to provide a suitable atmosphere to execute the samples and generate the required malware detection reports. The feature identification is then performed, and pre-processing is applied. Finally, the data set is used to train and test the classification accuracy of several deep learning models.

If already Detected

Classify

Feature Extraction

Pre-Processing

Data Set

Detect

# Project Deliverables

We will deliver the project step by step as guided by our supervisor. The most important things will deliver first and then gradually develop the whole System..

# Operating Environment

The malware detection environment involved a machine running Kali Linux as a host operating system and Windows 10 as a guest system, running on one giga byte of RAM. The host system carried out the responsibility of running other necessary programs, such as the Cuckoo Sandbox, VirtualBox and Volatility tool. The Cuckoo Sandbox was first configured and then used for two purposes: Analyzing the behavior of the tested file and getting the memory image by dumping the memory at the end of each file execution.

# Chapter 3: Requirement Analysis

**Use Cases:**

|  |  |
| --- | --- |
| Use case ID | UC01 |
| Use case Name | Report Generation |
| Actor Name | Direct User |
| Description | By selecting the files from the system after processing it will generate report about selected data or files |
| Trigger | To accomplish this task we Click on button (Select File) |
| Precondition | NO info about Dataset |
| Post Condition | Dataset provide complete information on the basis on Malware |
| Normal Flow | 1.Chose the dataset from system  2.select dataset and upload it for action  3.send dataset 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 dataset  3.Uploading Error  In such type of condition user request to restart from initial stage. |
| Includes: | null |

|  |  |
| --- | --- |
| Use case ID: | UC02 |
| Use case Name | Data Collection |
| Actor Name | Direct User |
| Description: | For collection of data first we take report &Api’s,Database when all these steps completed then process start for Data Collection |
| Trigger | For the process this task first get Report and Api’s,Database |
| Precondition | Report in not in data collection |
| Post Condition | Report in Database |
| Normal Flow: | 1.Input report get from sandbox  2.Save report into Data Collection  3.Check Successful Save 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.Report not be selected from Sandbox  2.report not saved |
| Includes: | null |

|  |  |
| --- | --- |
| Use case ID: | UC03 |
| Use case Name | Profiling |
| Actor Name | Direct User |
| Description: | For profiling following steps should take place  1. Start from system Action.  2. Get report from Data Collection.  3. Perform Profiling.  4.Successfully perform or fail. |
| Trigger: | To complete this task take report from Data Collection |
| Precondition | No profile info |
| Post condition | Profiling to their prospective behavior |
| Normal Flow | 1. Get report from Data Collection.  2. Perform Profiling.  3. Successfully perform or fail |
| Alternation | Successfully move next else request for data set  From dataset |
| Exception: | During the performing action some error can be take place  1.Report not be selected from Data Collection  2.report not saved  3.Fail to Profiling |
| Includes: | Null |

|  |  |
| --- | --- |
| Use Case ID: | UC04 |
| Use Case Name | Feature Generation |
| Actor Name: | Direct User |
| Description: | System Action:  1)Get Profiling detail  2)Perform Feature Generation  3)Success |
| Trigger | Get Profile Data() from Api,DLL,Network,File |
| Precondition | The data was only Profiled |
| Post condition | Feature generated from profiled |
| Normal Flow | For performing this process there are following steps  1)Get Profiling detail  2)Perform Feature Generation  3)Success  4)Feature are generated. |
| Alternative Flow | Success full then move to next step else Move again to feature generation |
| Exception: | During the performing action some error can be take place  1.Profiling detail not collected  2.Fail to Perform feature generation |
| Includes | Null |

|  |  |
| --- | --- |
| Use case ID: | UC05 |
| Use case Name | Feature Database |
| Actor Name | Direct User |
| Description: | For process this task there are following steps:  1)Feature generation Report  2)save into Database  3) Successfully save or fail. |
| Trigger | Get Featured \_data From Feature Generation () |
| Precondition | No data in feature data |
| Post condition | Feature data came |
| Normal Flow | 1)System Action  2)Feature generation Report  3)save into Database  4) Successfully save or fail. |
| Alternative | If report get and save success fully then move next else Get report and save |
| Exception | Feature generation Report not selected  Report in found in database  Save to fail |
| Includes | null |

|  |  |
| --- | --- |
| Use case ID: | UC06 |
| Use case Name | Detection |
| Actor Name | Direct User |
| Description | For Detection of Malware  System Action  1)Get Feature generated report database  2)detect the malware from report  3)Success |
| Trigger | Collect feature generation data from database |
| Precondition | Non Detected |
| Post condition, | Detected |
| Normal Flow | System Action  1)Get Feature generated report database  2)detect the malware from report  3)Success |
| Alternative | If successfully detected then move next step  Else again get file from database |
| Exception | Data collection fail from Feature generation report  Detection of malware fail |
| Includes | null |

|  |  |
| --- | --- |
| Use case ID: | UC07 |
| Use case Name | Classification |
| Actor Name | Direct user |
| Description | For Classification of Malware  System Action  1)Get data from detection  2) classify the detected data in families  3)success |
| Trigger | Detected malware file from Detection() |
| Precondition | Non Classified dataset |
| Post Condition | Classified |
| Normal Flow | 1)Get data from detection  2) Classify the detected data in families  3)Classified data set  Compelet detail of malware and their classes |
| Alternative | If success then movie to the final steps else going back to detection |
| Exception | data not collected from detection  fail to Classified data set |
| Includes | null |

# Functional Requirements

1. Give Dataset set to Virtual environment/cuckoo sand box
2. Get Reports from virtual environment report as Data Collecton
3. Malware Profiling of Reports
4. Feature Generation on the basis on malware profiling
5. Collect data in feature database
6. Malware Detection
7. Malware Classification
8. Malware Evaluation

# Non-Functional Requirements

Non-functional Requirements specifies the quality attribute of a software system. They judge the software system based on Responsiveness, Usability, Security, Portability and other non-functional standards that are critical to the success of the software system.

**Performance:**

It will provide performance upto 90% by using deep learning

**System usability scale (SUS):**

By applying the SUS, we were able to obtain a good indication concerning the handling of our KAMAS prototype. The results show a SUS value of 75.83 points out of 100, which can be interpreted as good without significant usability issues according to the SUS. They described the SUS questionnaires result from different perspectives: From the perspective of the acceptability range, a value between 70 and 100 points is labeled as “acceptable and from the adjective rating perspective, the KAMAS SUS result lies in a range between good (73 points) and excellent (85 points). In general, all participants were able to recognize the interaction design and interface changes in relation to the work steps they fulfilled. Based on the SUS description and average evaluation of the system by the participants, the result of the usability assessment was very positive. SUS scores and identified the average score as 68 points. Additionally, he showed that only 36% of the compared tests reached an SUS score higher than 74, and only 10% of the systems reached an SUS score greater than 80, which shows us that our system receives a grade of ‘B” at this implementation state.

During the test, the participants again addressed the tool tip complication and the missing brief introduction of the included visualization techniques.

**Reliability:**

Reliability is based solely on our implementation of a decision tree based classification model that is very naive. The decision tree is built using only the information returned from Virus Total via Cuckoo. This is meant to serve as a proof-of-concept implementation of the framework. The test base for this metric is the same 100 sample test base used for the performance test, but with additional clean and malicious files that were not used for training. The total size of this test base is 147 files.

We are using it along with information pulled from the database to build the model. In order to evaluate this metric, there are three basic steps:

• Build the model

• Test the model

• Check the results

Step one is done by fetching all rows from the .virus total' and the 'malicious answer' database tables, removing irrelevant columns (such as the file's hash), formatting it, writing the formatted results out to a temporary file, and passing the file to the decision tree library to build the model. Step two consists of retrieving the Virus Total results from the database based on the sample's sha256 hash value Finally, step three consists of reviewing the tables and comparing the returned result from step two with the expected value.

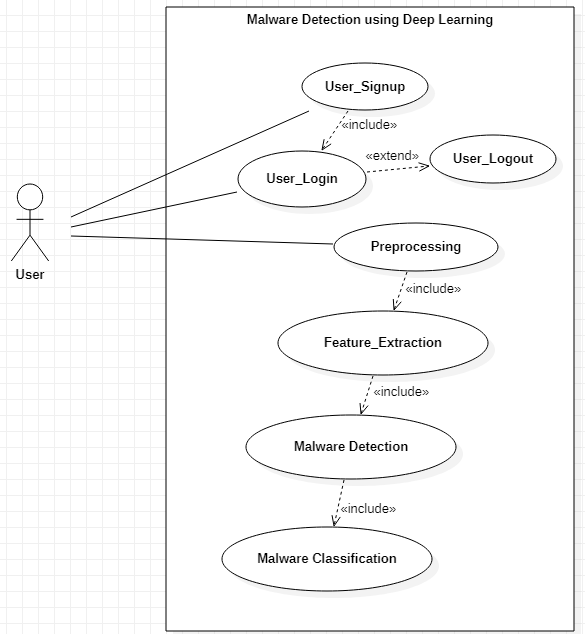
**Performance:**

Malware first of which that it can impact a computer's performance. Furthermore, inexperienced users can be lulled into a false sense of security when using the computer, considering their computers to be invulnerable, and may have problems understanding the prompts and decisions that Malware analysis system presents them with. An incorrect decision may lead to a security breach. If the Malware analysis system employs heuristic detection, it must be fine-tuned to minimize misidentifying harmless software as malicious (false positive).Malware analysis system itself usually runs at the highly trusted kernel level of the operating system to allow it access to all the potential malicious process and files, creating a potential avenue of attack. The US National Security Agency (NSA) and the UK Government Communications Headquarters (GCHQ) intelligence agencies, respectively, have been exploiting anti-virus software to spy on users.Malware analysis system has highly privileged and trusted access to the underlying operating system, which makes it a much more appealing target for remote attacks. Additionally anti-virus software is "years behind security-conscious client-side applications like browsers or document readers.

**System Requirements**

* Operating **System**: Windows 10. Windows 8. Windows 7 SP1. Windows Vista SP2. ...
* Memory: Windows 7 & Windows 8: 32-bit: 1GB RAM **minimum**, 2GB recommended. 64-bit: 2GB RAM **minimum**, 4GB recommended. ...
* Processor: 1.0GHz or higher, as specified by Microsoft for OS install.
* Internet. High-speed Internet .

# Use Case Model

****

**Actor Description:**

For the detection of malware Actor first perform report generate then data collection, Profiling, Feature generate, feature database, Detection, Classification and at the end perform Evaluation and then give output as classified malware file dat**a**

**Use case Description:**

Report generate:

* User (user Click on Button)
* System Action
* Get Dataset from System
* Send dataset to cuckoo
* Generate Report
* success fully Generated or failure

**Data collection:**

* Get Report from Sandbox
* And save that report into Data Collection
* Successful save or not

**Profiling:**

* Get Report from Data Collection
* Perform Profiling
* Successfully perform or fail

**Feature generate:**

* Feature generation Report
* Save into Database
* Successfully save or fail

**Feature database:**

* Get Feature generated report database
* Detect the malware from report
* Success and fail

**Detection:**

* Get data from detection
* Classified the detected data in families
* Success or fail

**Classification:**

* System Action
* Get data from detection
* Classified the detected data in families
* Success or fail

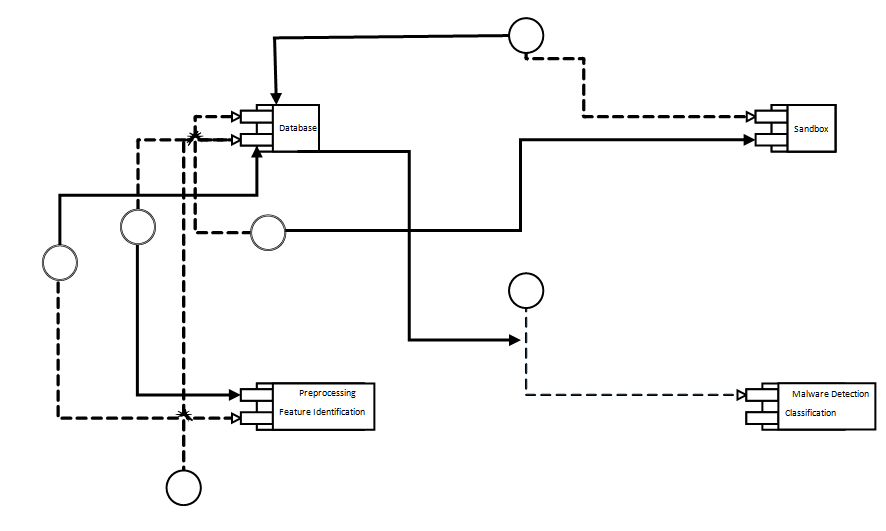
**Evaluation:**

* System Action
* Get the file from classification
* And then evaluate that
* Success send file to the user and fail

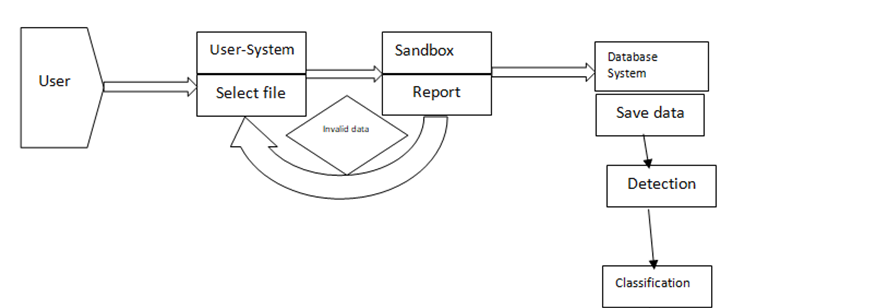
# Chapter 4: Design and Architecture

[**4.1 UML Structural Diagrams**](#_Toc268523830)

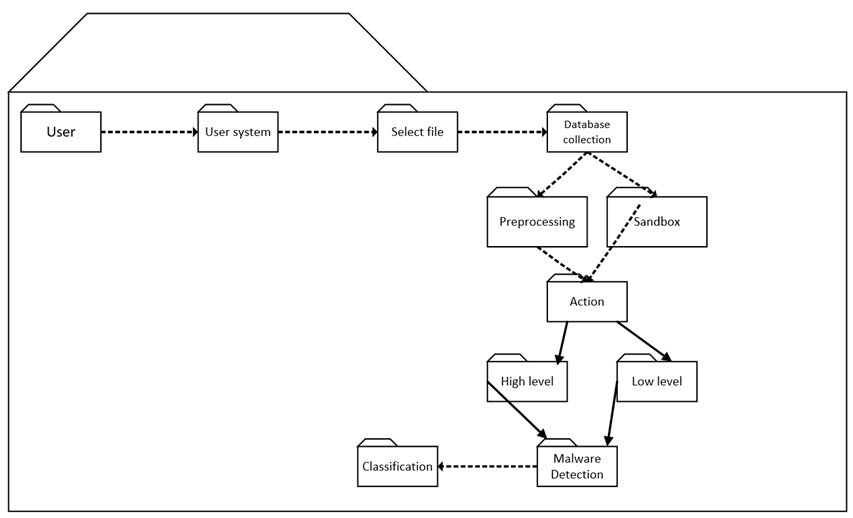
[4.1.1 Component Diagram](#_Toc268523787)



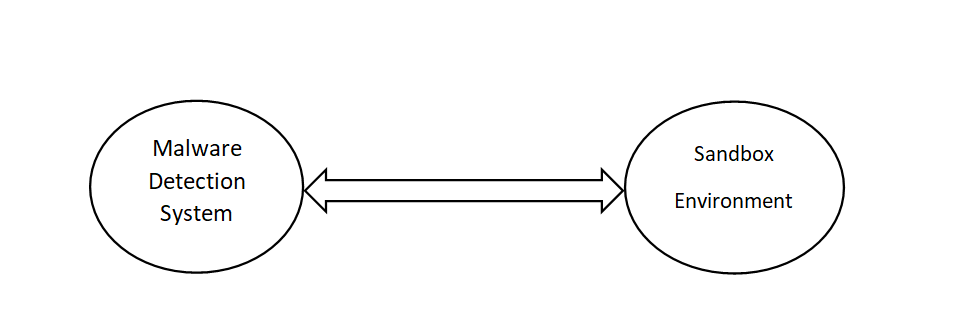
[**4.1.2 System Component Diagram**](#_Toc268523787)



[**4.1.3 Package Diagram**](#_Toc268523787)



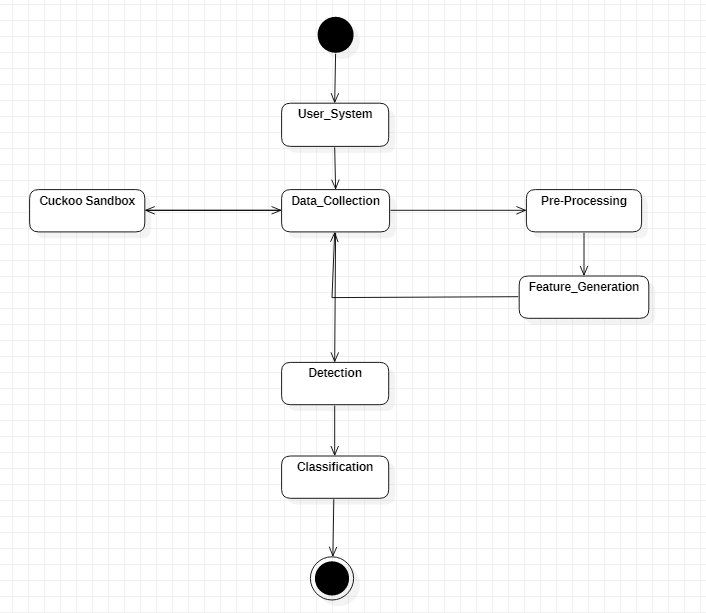
[**4.1.4 Deployment Diagram**](#_Toc268523787)



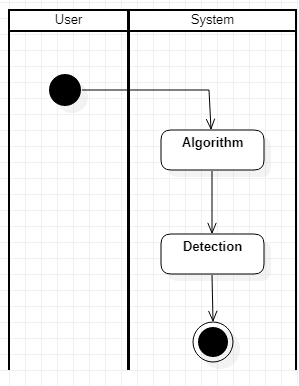
[**4.2 UML Behavioral Diagrams**](#_Toc268523830)

[**4.2.1 Activity Diagrams**](#_Toc268523787)

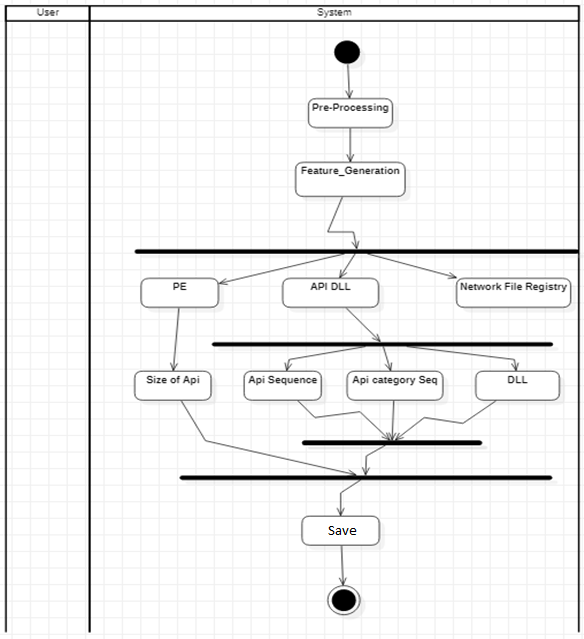
**Data Collection**



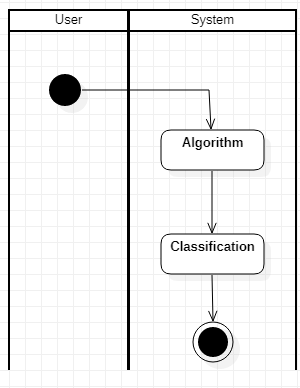
**Detection**



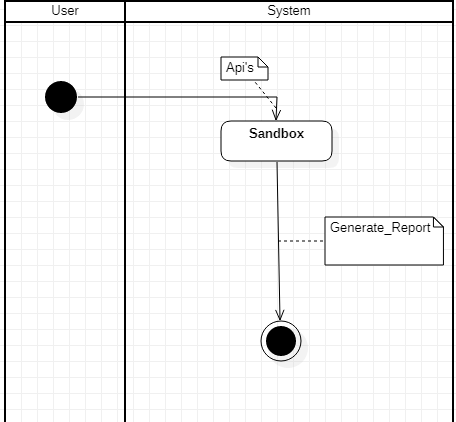
**Feature Generation/Extraction**

****

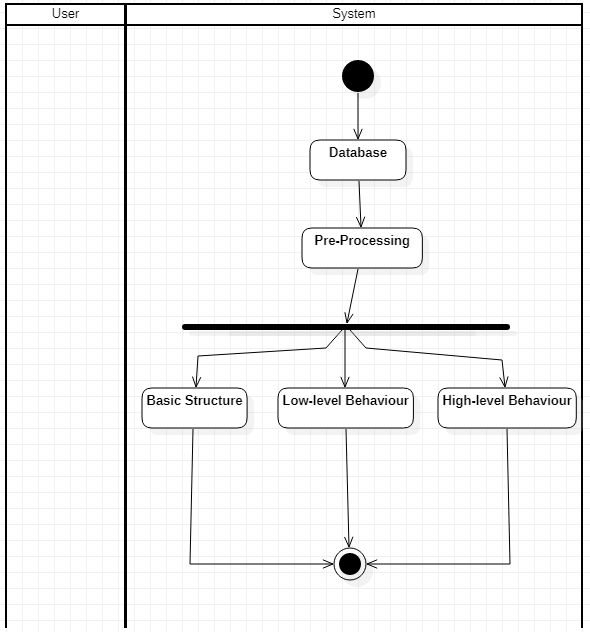
**Classification**

****

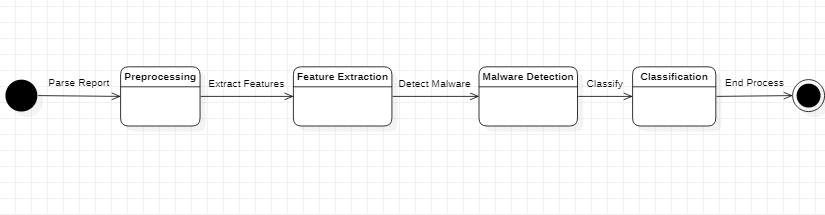
**Generate Report**

****

**Pre-processing**

****

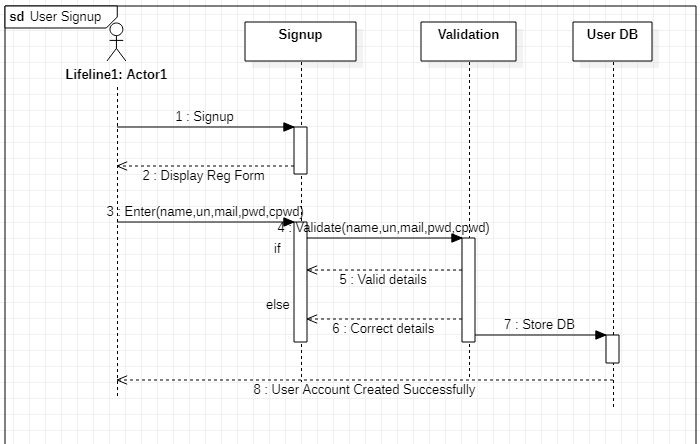
[**4.2.2 State Machine Diagrams**](#_Toc268523787)



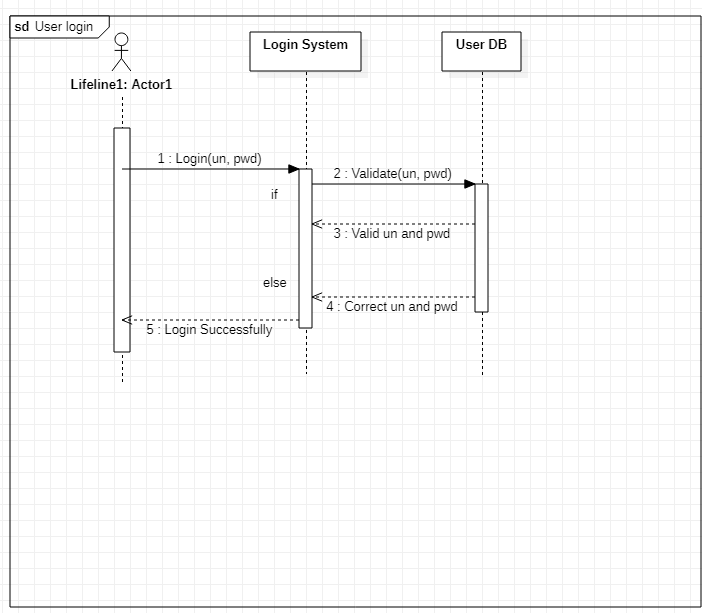
[**4.3 UML Interaction Diagrams**](#_Toc268523830)

[**4.3.1 Sequence Diagrams**](#_Toc268523787)

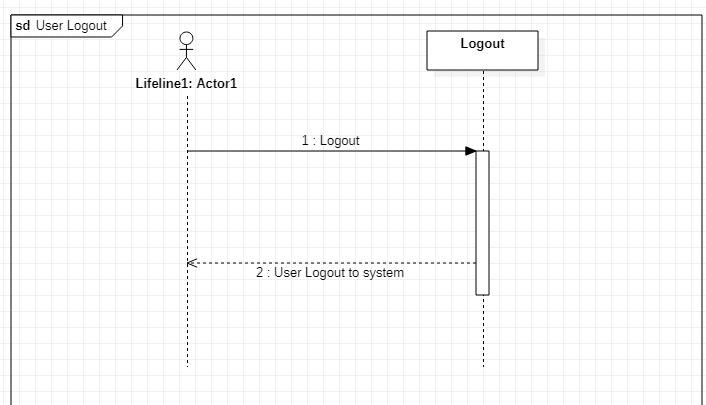
**User Signup(Sequence)**



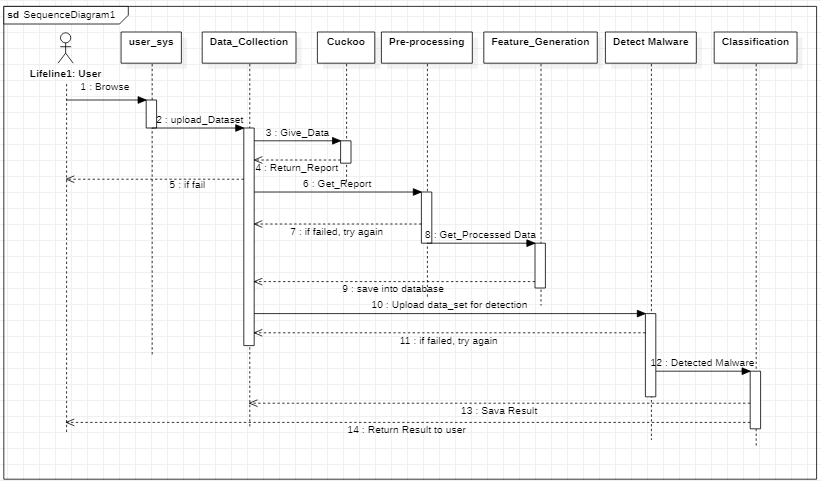
**User Login(Sequence)**



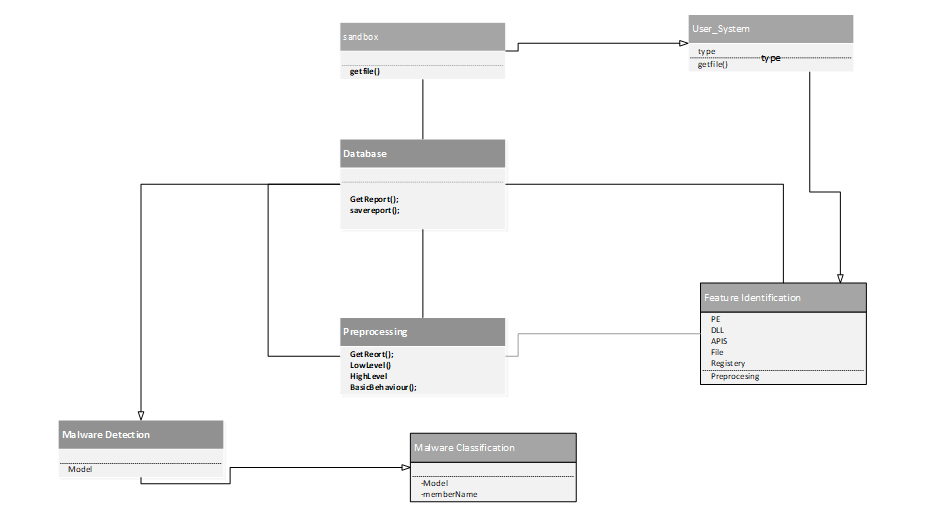
**User Logout(Sequence)**



Sequence Diagrams

****

**Class Diagram:**

****

# Chapter 5: Implementation

5.1 Python Language

Python is a popular programming language. It was created by Guido van Rossum, and released in 1991.

It is used for:

* web development (server-side),
* software development,
* mathematics,
* system scripting.

5.2 Software and Tools

Pycharm

PyCharm is a hybrid-platform developed by JetBrains as an IDE for Python. It is commonly used for Python application development. Some of the unicorn organizations such as Twitter, Facebook, Amazon, and Pinterest use PyCharm as their Python IDE

Operating System

Kali Linux 20.01

Kali Linux is a Debian-derived Linux distribution. Kali Linux is mainly used for advanced Penetration Testing and Security Auditing. Kali contains several hundred tools which are geared towards various information security tasks, such as Penetration Testing, Security research, Computer Forensics and Reverse Engineering.

Databese

PostgreSQL

PostgreSQL is a powerful, open source object-relational database system that uses and extends the SQL language combined with many features that safely store and scale the most complicated data workloads.

Cuckoo sandbox/Cuckoo droid

We have used cuckoo sandbox/cuckoo droid for dynamic analysis. First, we upload file to our interface and then submit to cuckoo perform analysis on that file that file we have been submitted to cuckoo.

5.3 Libraries

Tkinter

[**Tkinter**](https://wiki.python.org/moin/TkInter) commonly comes bundled with Python, using Tk and is Python's standard GUI framework. It is famous for its simplicity and graphical user interface. It is open-source and available under the Python License

Matplotlib

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK+. There is also a procedural "pylab" interface based on a state machine (like OpenGL), designed to closely resemble that of MATLAB, though its use is discouraged. SciPy makes use of Matplotlib.

MS word

* For project documentation

MS Visio

* For UML diagram

PowerPoint

* For presentation

System Requirement

* Laptop
* 8GB RAM
* 256GB ROM
* Octa core processor

5.4 Summary

This chapter is about implementation of our project. We used python for our project because our work is associated with deep learning and we used libraries for deep learning classification and detection as well. Used cuckoo sandbox/cuckoo droid for dynamic analysis that will analysis on uploaded file from system. Tkinter library is used for creating interface from where user submit file from system.

# Chapter 6: Testing and Evaluation

**6.1 Module/Unit Testing**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Project Name: Malware Detection** | | **Test Case: 1** | | | | | | | |
| **Test Case ID:** T\_01 | | | | | | |
| **Test Priority (Low/Medium/High):** High | | | | | | |
| **Module Name:** User Signup | | | | | | |
| **Input:** Data for variable Name, username, password and conform password. | | | | | | |
| **Test Title:** Entering all the Values that must be correct. | | | | | | |
| **Description:** User press the Signup button on splash screen, a new window open with following requirement, Name, username, password and conform password, user enter values against them and press Signup button. | | | | | | |
| **Pre-Condition:** User should enter all the required detail correctly. | | | | | | |
| **Dependencies:** | | | | | | |
| **Steps** | **Test Steps** | **Test Data** | **Expected Result** | **Actual Result** | **Status (Pass/Fail)** | **Notes** |
| 1 | Name | Amir |  |  |  |  |
| 2 | User Name | Amir75 |  |  |  |  |
| 3 | Password | 0022 |  |  |  |  |
| 4 | Conform Password | 0022 |  |  |  |  |
| 5 | Signup Button Clicked |  | All data entered by user will store in database named user dB and user can login through entered username and password. | Show error message Password and conform password should be same | Fail |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Project Name: Malware Detection** | | **Test Case: 2** | | | | | | | |
| **Test Case ID: T\_02** | | | | | | |
| **Test Priority (Low/Medium/High):** High | | | | | | |
| **Module Name:** User Login | | | | | | |
| **Input:** Values against username and password | | | | | | |
| **Test Title:** Entering all the Values that must be correct. | | | | | | |
| **Description:** Test the login page. | | | | | | |
| **Pre-Condition:** User should Have an account and user name and password should correct. | | | | | | |
| **Dependencies:** User must be registered first. | | | | | | |
| **Steps** | **Test Steps** | **Test Data** | **Expected Result** | **Actual Result** | **Status (Pass/Fail)** | **Notes** |
| 1 | User Name | Armaan65 |  |  |  |  |
| 2 | Password | 0011 |  |  |  |  |
| 3 | Login Button Clicked |  | If username and password entered correctly, System granted access to user | Navigate to main screen | Pass |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Project Name: Malware Detection** | | **Test Case: 3** | | | | | | | |
| **Test Case ID:** T\_03 | | | | | | |
| **Test Priority (Low/Medium/High):** High | | | | | | |
| **Module Name:** User Logout | | | | | | |
| **Input:** Press logout button | | | | | | |
| **Test Title:** Logout user from system | | | | | | |
| **Description:** User press logout button. | | | | | | |
| **Pre-Condition:** User login to the system | | | | | | |
| **Dependencies:** User must be login. | | | | | | |
| **Steps** | **Test Steps** | **Test Data** | **Expected Result** | **Actual Result** | **Status (Pass/Fail)** | **Notes** |
| 1 | Logout Button Clicked |  | System logout the user. | System returns the user to login screen | Pass |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Case ID | | | | T\_04 | | |
| Test priority (low, medium, high) | | | | High | | |
| Module Name | | | | Check Cuckoo conf | | |
| Description | | | | Admin Run command of cuckoo to check cuckoo is configure properly | | |
| Test Title | | | | Test configuration | | |
| Precondition | | | | Cuckoo cofig or not | | |
| Despondency | | | | Mongo dB started  VM in restore mode | | |
| s. | Test steps | Expected result | Actual result | | Status fail or pass | Note |
| 1 | * Open terminal * Run commandCuckoo -d | At the end of console  The message will appear  “waiting for analysis” | Waiting for analysis | | Pass | If cuckoo not configure it will give message cuckoo could not find client |

|  |
| --- |
| **Project Name: Malware Detection** |
| **Test Case: 4** |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Project Name: Malware Detection** | | | | | | |
| **Test Case: 5** | | | | | | |
| Test Case ID | | | T\_05 | | | |
| Test priority(low, medium, high) | | | High | | | |
| Module Name | | | Check Cuckoo Result | | | |
| Description | | | Admin Run cuckoo console and cuckoo web | | | |
| Test Title | | | Test result | | | |
| Precondition | | | Cuckoo produce result or not | | | |
| Despondency | | | Mongo dB started  VM in restore mode  Cuckoo droid | | | |
| s. | Test steps | Expected result | | Actual result | Status fail or pass | Note |
| 1 | * Open terminal * Run command   Cuckoo –d   * Cuckoo web * Open browser * Type localhost:8000 * Submit upload file * Submit file | At the end of cuckoo console show message  “Analysis is completed”  And browser it will show All detail  If score is 0.0 this means cuckoo is not configure properly  It the score between  1-10  This mean cuckoo is producing accurate result | | Score 3.5 | Pass | If score is 0.0 it means we have to review cuckoo configuration |

|  |
| --- |
| **Project Name: Malware Detection** |
| **Test Case: 6** |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Case ID | | | | T\_06 | | |
| Test priority (low, medium, high) | | | | High | | |
| Module Name | | | | Detect Malware | | |
| Description | | | | Admin Run the malware detection | | |
| Test Title | | | | Test Malware  result | | |
| Precondition | | | | Result on desktop interface | | |
| Despondency | | | | Internet | | |
| s. | Test steps | Expected result | Actual result | | Status fail or pass | Note |
| 1 | * Open desktop app * Browse file * Upload file | It should show the Report of file | It shows a short information of file. | | Pass |  |

|  |
| --- |
| **Project Name: Malware Detection** |
| **Test Case: 7** |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Case ID | | | | T\_07 | | |
| Test priority(low,medium,high) | | | | High | | |
| Module Name | | | | Integration | | |
| Description | | | | Upload file submitted to cuckoo | | |
| Test Title | | | | Web interface integration | | |
| Precondition | | | | No Task is running | | |
| Despondency | | | | Running cuckoo  Mongo dB should start  Virtual machine starts or restore mode  Cuckoo api is running | | |
| s. | Test steps | Expected result | Actual result | | Status fail or pass | Note |
| 1 | * Open browser * Open web * In cuckoo module * Browse file * Upload file | It should be submitted to cuckoo. Cuckoo perform complete analysis.  The message will be shown like  “successfully completed” | It shows Message successfully completed analysis | | Pass | If no internet it might not show any result |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Project Name: Malware Detection** | | | | | |
| **Test Case: 8** | | | | | |
| Test Case ID | | | T\_08 | | |
| Test priority (low, medium, high) | | | High | | |
| Module Name | | | Classification | | |
| Description | | | After uploading file and completion of analysis it will show of malware classification. | | |
| Test Title | | | Classification | | |
| Precondition | | | Unclassified | | |
| Despondency | | | Running cuckoo  Mongo dB should start  Virtual machine starts or restore mode  Should complete analysis | | |
| S. | Test steps | Expected result | Actual result | Status fail or pass | Note |
| 1 | * Open browser * Open web * In cuckoo module * Browse file * Upload file | Predict Malware class | It shows message this a trojan horse | Pass |  |

|  |
| --- |
| **Project Name: Malware Detection** |
| **Test Case: 9** |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Case ID | | | | T\_09 | | |
| Test priority(low, medium, high) | | | | High | | |
| Module Name | | | | Display Report | | |
| Description | | | | After uploading file and completion of analysis it will show report. | | |
| Test Title | | | | Report display | | |
| Precondition | | | | No result on Interface | | |
| Despondency | | | | Running cuckoo  Mongo dB should start  Virtual machine starts or restore mode | | |
| s. | Test steps | Expected result | Actual result | | Status fail or pass | Note |
| 1 | * Open browser * Open web * In cuckoo module * Browse file * Upload file | It will show complete file report on Web interface | It shows complete analysis Report on our web interface | | Pass |  |

# 6.2 Integration Testing

**Integration testing** is a level of software testing where individual units or components are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units. Test drivers and test stubs are used to assist in Integration Testing. The tests are conducted to ensure that the components are working properly after interfacing.

Integration testing of the system is performed as follows:

* Integrate login module and test.
* Integrate Malware detection modules and test.

# Chapter 7: Conclusion and Future Work

This chapter concludes the project and highlights future work.

# Conclusion

In the developed model we can analysis the malware in given dataset by using cuckoo droid. We can perform static and dynamic analysis on the system. We can further detect the malware from the report generated and dataset using developed model. When detection process is completed we can classify the detected malware according their family.

# Future Work

The system can be integrated by training the model that can remove the malware from the files or protect files against malware and can be upgraded to provide the security to datasets. In future the system can be integrated further according to our needs.

# References

References to any book, journal paper or website should properly be acknowledged. Please consistently follow the style. The following are few examples of different resources i.e. journal article, book, and website.

1 Lyda M.S. Lau, Jayne Curson, Richard Drew, Peter Dew and Christine Leigh, (1999), Use Of VSP Resource Rooms to Support Group Work in a Learning Environment, ACM 99, pp-2. (Journal paper example)

2 Hideyuki Nakanishi, Chikara Yoshida, Toshikazu Nishmora and TuruIshada, (1996), FreeWalk: Supporting Casual Meetings in a Network, pp 308-314 (paper on web) http://www.acm.org/pubs/articles/proceedings/cscw/240080/p308-nakanishi.pdf

3 Ali Behforooz& Frederick J.Hudson, (1996), Software Engineering Fundamentals, Oxford University Press. Chapter 8, pp255-235. (book reference example)

4 Page Author, Page Title, http://www.bt.com/bttj/archive.htm, Last date accessed. (web site)