MUHAMMAD TAYYAB HASAN
01-134161-045
MUHAMMAD FAIZAN SHAH
01-134161-037

Sandbox Prototype -Installation Auditing System

Bachelor of Science in Computer Science

Supervisor: Dr. Faisal Bashir

Department of Computer Science
Bahria University, Islamabad

November 2019

Certificate

We accept the work contained in the report titled 'Sandbox Prototype -Installation Auditing System", written by Mr. Muhammad Tayyab Hasan and Mr. Muhammad Faizan Shah as a confirmation to the required standard for the partial fulfillment of the degree of Bachelor of Science in Computer Science.

Approved by . . . :

Supervisor: Faisal Bashir (Dr)

_____

Internal Examiner 1: Muhammad Muzammal (Dr.)

_____

External Examiner: Waseem Shahzad (Dr.)

_____

Project Coordinator: Zubaria Inayat (Ms.)

_____

Head of the Department: Muhammad Muzammal (Dr)

_____

November 31^{st}, 2019

Abstract

Today internet users face a very difficult situation, everyday users need to download and install software from internet and it is difficult to assess the integrity of these software. Naïve users are deceived into installation of malicious programs. These Programs mislead the user from their true intent, they behave as benign while they often steal personal information, which leads to real world problems. Currently, there are two ways to conduct Malware Analysis. In Static Malware Analysis a given sample is analyzed without execution in the real environment. The most common technique to perform static analysis are signature-based Antivirus tools. The antivirus tools detect malicious software by scanning the given sample against available known set of signatures. Another technique that is used for malware analysis is Dynamic Malware Analysis. In dynamic analysis a given sample is executed in a controlled environment and the actions performed by the sample are monitored for behavioral analysis. SandMal is an application that is developed, and this project report presents an overview of the system. The intended audience of SandMal application are common internet users. The designed system will help a common internet user to upload a file to the system before they install on their personal computer. This system will install the malicious files on the server and compile the behavioral report of the sample. This behavior report will be sent back to the user that can analyze the report.

Acknowledgments

First and foremost, praises and thanks to the God, The Almighty Allah, for providing us with strength, courage throughout our Final year project. Our profound and humble gratitude towards our Project Supervisor Dr. Faisal Bashir and co-supervisor Ms. Maliha Ismial who had confidence in us and blessed us with the opportunity to do research and providing their invaluable guidance and feedback throughout this research. Lastly, we would like to extend our heartiest falcitations to our parents for all their love, prayers and sacrifices in striving hard to make us better human.

MUHAMMAD TAYYAB HASAN
MUHAMMAD FAIZAN SHAH
Islamabad, Pakistan

November 2019

“ Sometimes it is the people no one can imagine anything of who do the things no one can imagine. ”

Alan Turing

Contents

List of Figures

List of Tables

Acronyms and Abbreviations

API	Application Programming Interface
CSS	Cascading Style Sheets
DLL	Dynamic Link Library
HTML	Hyper Textmarkup Language
OS	Operating System
PgAdmin	Postgresadmin
PL/pgSQL	Procedural Language/Postgresql
RDBMS	Relational Database Management System
SQL	Structured Query Language
UML	Unified Modeling Language

Chapter 1

Introduction

1.1 Overview

Digital economy comprises of more than $20.4 Trillion, almost equivalent to 13.8% of global sales [1] . The ever-growing industry requirements demand an increasing reliance on Computer Systems thus computer security is one of the major challenges of the contemporary world. The Operating System (OS) market share of 87.56% is with Microsoft Windows and it remains the most popular OS [2]. Having the largest market share in industry, Microsoft Windows is also prone to more attacks as compared to others. With having large number of naive users, it is easier to trick them into running malicious computer program such as Trojan-horse. The malware can mislead users from their true intent, often steal their information such as personal identity, passwords and banking information, which leads to real-world problems. That's why, it's far more important to analyze malware behavior and how they infect a host system and how they propagate from one system to another. Software applications access multiple input/output devices as well as the Internet. The software manuals inform users about the resources that will be engaged by the software. Malware analysis system such as Sandbox provide a virtual testing environment to the user to observe how the software will behave after installation. In this project, Sandbox for Windows OS will be developed that will monitor the process's behavior and generate a log file, indicating all the resources engaged by the installing process. The user will identify the target (malware) application and upload the file to the system, System will install the malicious file on the server and monitor the actions of application.

1.2 Objective

To design a software that monitors the installation cycle of any application including the resources that it will utilize independent from the rest of the system. It will help stop

system failures and software vulnerabilities from spreading.

1.3 Problem Description

In the modern digital age, security is becoming a very important part of any system. Users must frequently choose between functionality and security. Users can download many software, but they are afraid to run them because they are never 100% sure of what a software will do in the background. Designed system will provide an analysis to the user, about the installation pattern of the software and can predict about hostile software. System will generate the log file which will identify the disk operations, network connections, registry configuration changes and system calls the software has made.

1.4 Project Scope

The SandMal Application will monitor the installation of any software on Microsoft Windows and will generate a log file indicating the following.

1. **Network Analysis:** Communication done by the program during the installation phase.
2. **Registry Modification:**Registry values altered or created during installation phase.
3. **Folder:** Files and folder accessed, created or deleted during installation phase.
4. **Memory Analysis:** Complete Physical memory dump during the time of installation phase.

1.5 Methodology

Current, security applications widely available to the masses for computer security such as anti-spam filters, intrusion prevention systems, antivirus and other systems, heavily rely on identifying attacks. This can be done either by known signatures or anomaly detection. This kind of protection only works once the software has been installed. Designed system will monitor the overall execution of the software during the installation phase. The developed system will be a sandbox Application. The software to be analyzed is installed in the virtual environment and it cannot harm the actual system. Designed System will only track all the resources (software/hardware) that are engaged by the installing software. System will track the installing process, the sub/child process, interrupts/system calls, communication port etc. A log file will be maintained to inform the user of all the activities that are performed by the installing processes. The tracking in Windows OS is done using process ID of the installing process. The following will be detected disk activity, files

created/accessed, RAM & physical memory acquired, communication channels engaged, and registry entities modified/created.

1.6 Solution Application Areas

This project will be very useful for an organization with enhanced security requirements. Designed system will be useful to study the behavior of viruses and benign software, also it will enable, end users to make their system secure.

1.7 Tools/Technologies

- Cuckoo mon
- PostgreSQL
- Volatility
- TCPDUMP
- Django

1.8 Expertise of the Team Members

The project has its domain in the field of Computer Security and requires deeper understanding due to the complexity of the domain. Both team members have studied Data Encryption and Security and have completed all their necessary course work which has provided sufficient knowledge base on the subject matter. Both the team members are fast learners and believe they are pre-equipped with sufficient knowledge to complete this project. Since the core development of our project will be in Python, both team members have a comprehensive understanding of Python and object-oriented programming paradigm

Chapter 2

Literature Review

In this chapter, the report will discuss the literature review for Sandboxing technique. The previous works related to Malware detection will be discussed. This chapter will also discuss previous existing systems how they detect malware and how SandMal application differentiates from such systems

2.1 Introduction

2.1.1 Malware

Any software that "intentionally accomplishes the harmful intent of an attacker " [3] is referred as malware. Malware is a generic term to describe multiple types of malicious software (e.g., worms, viruses or Trojan horses). Malicious Software are not only a threat to computer security and privacy and data of computer users all around the world, but it also poses serious threat of financial loss.

Initially the motivation of the malware creators was to highlight security vulnerabilities and to complete challenging projects to show technical ability of the creator but with the passage of time, motivation of such cyber criminals has changed. It is no longer driven by such simple motivation, today there exist an underground economy with financial benefits where such criminal services exist for a price. [4]

So, to mitigate these attacks and protect computer users, rises a genuine necessity to create such tools that can detect and stop such malicious software. Nowadays the most common tool that is accessible to majority of the users is signature-based anti-virus tools.

2.1.2 Types of Malware

The word Malware is a contraction of two words – Malicious Software. Malware is a program has the ability to cripple or disrupt the operations of the system and allows hackers to steal personal Information. It is important to understand different types of malware in order to stop system vulnerabilities from spreading.

2.1.2.1 Virus

This mechanism of this type of malware is that this type of malware will reproduce it self and distribute copies of itself by any means to spread. Generally, a virus will attach itself to another software and usually requires human interaction to propagate.

2.1.2.2 Worms

Worms are standalone software that are not part or attached to other software. Worms replicate themselves in the target system and destroy the operating system and data files in the drive and keep this behavior until drive is empty.

2.1.2.3 Trojan Horses

A trojan is a type of malware that spreads itself in guise of routine software, they appear useful and persuade user to install it on their personal computer. Trojan are considered to be the most dangerous malware because they steal personal information.

2.1.2.4 Rootkits

The mechanism of this type of malware is they are specifically designed to gather information from the system. Rootkits work in background and acts as a back door for the malware to enter into system and cause loss of data.

2.1.2.5 Ransomware

This type of malware is one of the most devastating for the user. This type of malware will infect the system and will block the access of user files and locks the system. This type of malware will encrypt the user data and demand a ransom payment for decryption.

2.2 Static Malware Analysis

The technique of Analyzing a given sample without executing is called Static Malware Analysis. The most common technique to perform static malware analysis are signature-based Antivirus tools. The Signature-based antivirus tools that exist today, detect malicious

software by scanning the given sample against already available set of signatures. These signatures are carefully generated such that they only match the malicious software. But this technique comes with a fundamental weakness. Using a pre-generated set of signatures prevents the detection of new and unknown malicious software. Even when a new threat is detected and its signature is updated in a centralized database, now this new information must be distributed to every client that is accessing the central database and if a client misses such update it will not detect the malware rendering the system unsecure. Signature based analysis also presents a drawback with the inability to detect completely unknown threats it can also possess a weakness to detect tailored malware. A cyber criminal can target a company and steal sensitive information a long time before such tailored malware sample is submitted for analysis and a signature is produced. Although there exist techniques that identify a software a malicious or not based upon software's behavior, these techniques do allow detection of currently unknown threats, but this technique commonly generates false positives.

2.3 Dynamic Malware Analysis

As opposed to signature-based malware analysis which presents many fundamental flaws, Dynamic Malware Analysis is "Executing a given sample in a controlled environment and monitoring its actions such that it can analyzed for malicious behavior". [5] Since Dynamic malware analysis is performed during the runtime it is easy to analyze the behavior of the malware. Although Dynamic malware analysis is robust as compared to signature-based malware analysis, but it also faces a fundamental weakness of executing a malware on your machine, this also threats the data and privacy of the user so the analysis environment must be properly isolated or restricted.

2.3.1 Function Call Monitor

One of the most Important technique that exist and used by security analysts is Function Call Monitoring. Generally, a Function is a set of instructions that perform a specific task. The use of functions provides code reusability and proper maintenance but functions also play an important role in abstracting the implementation details for simpler semantic representation. Such abstraction helps during the analysis of the malware where we are more concerned with the behavior instead of implementation.

2.3.2 Hooking

Function Call Monitoring technique dynamically analyzes malware behavior by intercepting the calls made by the malware, this process is called hooking. The malware in manipulated in such a way that in addition to that with the intended function an additional

function is invoked. This function is called hook function. The hook function performs the task of analysis.

2.3.3 System Call Interface

Application Programming Interface (API) is a set of functions that provide coherent functionality. Operating Systems (OS) provide many build-in APIs which is used by user-mode programs so that it can request OS to perform some functionality that can only be executed in kernel-mode, where an OS executes. This important separation makes sure that user-mode programs cannot access hardware and memory directly. [6] So, a user-mode application makes a system call to a well-defined OS API sometime referred as system call interface to request the OS to does some functionality on its behalf. So, malware applications that execute in user-mode must invoke some system-call in order to do some functionality like any other genuine application. This technique is achieved by redirecting control flow of the system call. [7] Initially, the instructions of the original function are overwritten with a jump to analysis function. The original function instructions are backed up into another function that is called trampoline function. When the malware makes a system call that is hooked, overwritten instructions change the control flow to the analysis code, which then logs important information and redirects the flow towards trampoline function which then calls the original intended function

Chapter 3

Requirement Specifications

3.1 Existing Systems

There is an application, Process Hacker, that provide dynamic malware analysis to monitor overall System Resources, process's disk activity, network activity, stack traces, but doesn't cover all the proposed functionalities. However, most features provided by Process Hacker can be seen directly using build-in Window debugging systems but one of the fundamental flaws with this approach is running the given sample in an unprotected environment.

3.2 Proposed System

This main purpose of our proposed application is to create a system that emulates complete functionality of a real computer System with all the required components which can install and run our intended Operating System to provide a complete isolated environment to test and analyze Malware samples and prevent any changes to underlying System.

3.3 Functional Requirements

3.3.1 Register

Description: To allow a new user to enroll into the system.
Input: User shall provide first name, last name, email, password and desired username.
Processing: System will validate user input and match if the user already exists or not. it will enroll the user into the system and automatically login.
Output: System will be redirected to upload page.

3.3.2 Login

Description: To allow the user to login to the system if provided valid input.
Input:user name and password.
Processing:: The System will Validate User input and check against the currently enrolled users in the system.
Output: System will redirect the user to upload page.

3.3.3 Logout

Description: To allow the user to safely logout from the system.
Input:user presses logout button.
Processing: System will invalidate the session
Output:system will redirect to login page.

3.3.4 File Upload

Description: To allow the user to upload a file for analysis.
Input: User will select a file.
Processing: The System will verify the file type and upload it to webserver.
Output:: System will be redirected to Reports page.

3.3.5 Search Reports

Description: To allow the user to search from a list of all the reports of all the samples user has submitted for analysis.
Input: User will provide file name.
Processing: The System will access records of all the files that are associated with the user and filter the searched file.
Output: System will display the searched file

3.3.6 View Report

Description: To allow user to view an analysis report in HTML.
Input: User will select the file from list of given files
Processing: System checks if the status of the file, if status is complete it will generate a report in html.
Output: System will redirect to report page.

3.3.7 Download Report

Description: To allow the user to download analysis report in PDF format.
Input: User will click the download pdf button.
Processing: System will check the status of the file, if the status is complete it will generate the report in PDF format.
Output:PDF file.

3.4 Non-Functional Requirements

3.4.1 Efficiency

The system should be designed in such a way that it should use limited computational resources such as Memory and Diskspace and CPU cycle and should require minimum bandwidth while running an emulated Operating System.

3.4.2 Security

The system should provide Isolation between emulated Operating system and host machine and system should prevent any unauthorized access to the data of the user.

3.4.3 Reliability

The System should behave consistently without failure. System should behave in a user-acceptable manner and should not log unrelated and non-useful information when analyzing a sample.

3.4.4 Usability

The system should be easy enough so that intended user can properly operate, provide inputs and interpret outputs through proper User interface with the system.

3.4.5 Maintainability

The system should be designed in such a way that the system can be modified in future. Design of the system should be able to accommodate changes and improvement in the system and it should be able to handle correction of any defect that may arise.

3.4.6 Portability

The system should be designed in such a way that system should be able to run on multiple platforms.

3.5 Use Cases

The use cases for the system are displayed in this section.

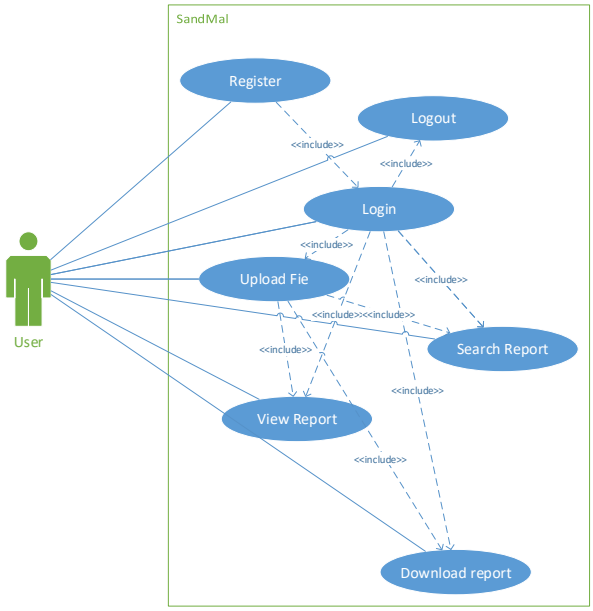

Figure 3.1: Displays the high-level use case diagram of the entire system.

In figure 3.1, a high-level use case diagram of the entire system with one actor(user) is shown. This diagram is a representation of interaction of user with that system. Diagram also show relationship between different use cases and actor.

3.6 Use Case I: Register

Table 3.1: Use Case I: Registeration

Use case id	UC- 01
Title	Register
Primary Actor	User
Description	To allow a new user to enroll into the system.
Pre-Condition	Internet connection
Flow of events	The user enters his credentials. The system checks for existing. The user entry is created in the DB. The user is Logged in.
Post-Condition	User will be enrolled into the system.
Alternate Flow	Entered Information already exist, a toast message will appear alerting, user already exist.

Table 3.1 shows the information of Use case 1. This table shows how user will interact will system to register itself to enroll with the system.

3.7 Use Case II: Login

Table 3.2: Use Case II: login

Use case id	UC- 02
Title	Login
Primary Actor	User
Description	To allow the user to login to the system.
Flow of events	The user enters his credentials. The credentials are verified with DB. The user is Logged in.
Pre-Condition	Enter credentials must exist in the system.
Post-Condition	User will be logged into the system.
Alternate Flow	Entered Information does not exist, a toast message will appear alerting, user of incorrect username or password.

Table 3.2 shows the information of Use case 2. This table shows how user will interact will system to login itself into the system.

3.8 Use Case III: Logout

Table 3.3: Use Case III: logout

Use case id	UC- 03
Title	Logout
Primary Actor	User
Description	To allow the user to safely logout from the system.
Flow of events	The user selects logout button. User is logged out.
Pre-Condition	User must be logged into the system.
Post-Condition	System will redirect to login page.

Table 3.3 shows the information of Use case 3. This table shows how user will interact will system to logout from the system.

3.9 Use Case IV: File Upload

Table 3.4: Use Case IV: file upload

Use case id	UC- 04
Title	File Upload
Primary Actor	User
Description	To allow the user to upload a file for analysis.
Flow of events	User uploads the file File is sent to the SandMal. System Turns on the virtual machine The agent copies the file into host and runs the file The file is run The memory dump is copied to client Tools extract info from dump The info is sent to user through webpage
Pre-Condition	User must be logged in.
Post-Condition	System will generate a task in the background and redirect the system to reports page.

Table 3.4 shows the information of Use case 4. This table shows how user will interact will system to upload a file in the system for analysis.

3.10 Use Case V: Search Reports

Table 3.5: Use Case V: search Reports

Use case id	UC- 05
Title	Search Reports
Primary Actor	User
Description	To allow the user to search from a list of all the reports of all the samples user has submitted for analysis.
Flow of events	The user enters file name. File is searched in db. File report showed.
Pre-Condition	User must be logged in.
Post-Condition	System will search from all the files submitted and filter the requested file.
Alternate Flow	If the file does not exist system will generate message of not found.

Table 3.5 shows the information of Use case 5. This table shows how user will interact will system to search reports from the system.

3.11 Use Case VI: View Report

Table 3.6: Use Case VI: view Report

Use case id	UC- 06
Title	View Reports
Primary Actor	User
Description	To allow user to view an analysis report in HTML.
Flow of events	The user selects file. File is searched in db. File report showed.
Pre-Condition	Selected file status must be complete.
Post-Condition	System will display a HTML report.
Alternate Flow	Wait for analysis to be completed.

Table 3.6 shows the information of Use case 6. This table shows how user will interact will system to the HTML report of the analysis from the system.

3.12 Use Case VII: Download Report

Table 3.7: Use Case VII: Download Report

Use case id	UC- 07
Title	Download Reports
Primary Actor	User
Description	To allow user to download an analysis report in HTML.
Flow of events	The user selects file. File is searched in db. File report showed. File is downloaded.
Pre-Condition	Selected file status must be complete.
Post-Condition	System will generate report in PDF format .

Table 3.7 shows the information of Use case 7. This table shows how user will interact will system to download the PDF report of an analysis from the system.

Chapter 4

Design

This chapter shall discuss the intended architecture and design of dynamic sandbox.

4.1 System Architecture

The system will use client-server Architecture model. This Architecture provides service to many clients that communicate over a computer network to a server that provides one or multiple services to the client.Figure 4.1 represents an architecture of the system.

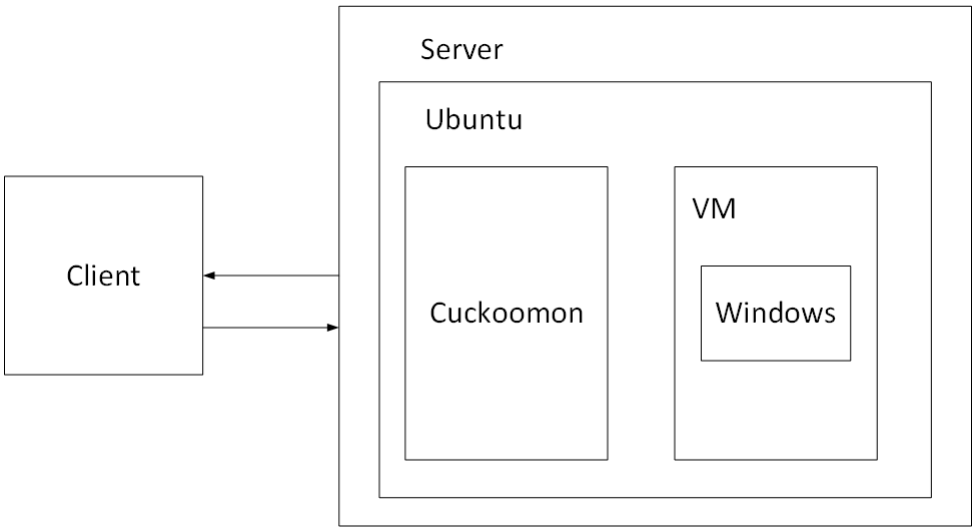

Figure 4.1: System Architecture

4.1.1 Client Layer

In our client-server Architecture, client Layer will provide a Graphical User Interface to our authorized users. This Layer will provide user the ability to login, register, upload a file and view analysis reports.

4.1.2 Server Layer

In our client-server architecture, server layer contains business logic of the system. This layer will manage all the computational task such as validation of data and starting and running virtual machines and generating and compiling analysis reports.

4.1.3 Programming Languages

The project uses Python as main development language. HTML, CSS and bootstrap 4 are used for the client-side development of the system. And Django web framework is used for the development of the server-side of the system. And PL/pgSQL is used with pgAdmin to create the database in PostgreSQL.

4.1.4 Design Standards

System will support all the important UML standards of software development. To support the design and development standards, documentation is carried out at every step.

4.2 Design Methodology

The design this system Iterative Development model was used. We First initially completed the highest priority requirements and later more and more functionality were added to the system as per their priority.

4.3 High Level Design

4.3.1 Level 0 Data Flow Diagram

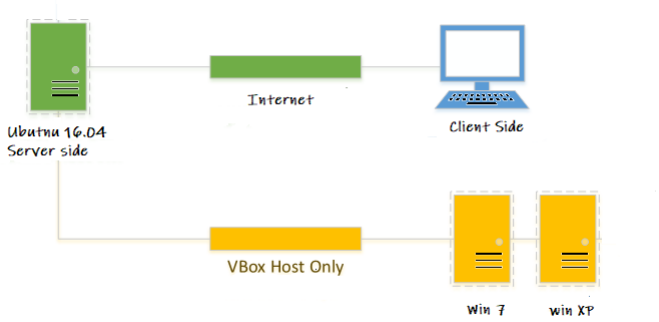

Figure 4.2: Data Flow Diagram

In Figure 4.2 represents flow of data of this system. A user will upload the sample file from client layer that will be processed in server, that will install the given sample in a virtual environment.

4.3.2 High Level Use Case Diagram

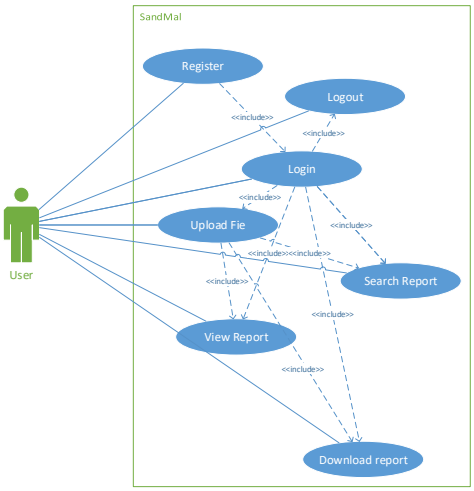

Figure 4.3: Use Case Diagram

In figure 4.3, a high-level use case diagram of the entire system with one actor(user) is shown. This diagram is a representation of interaction of user with that system. Diagram also show relationship between different use cases and actor

4.3.3 Activity Diagrams

The Activity Diagram is given below.

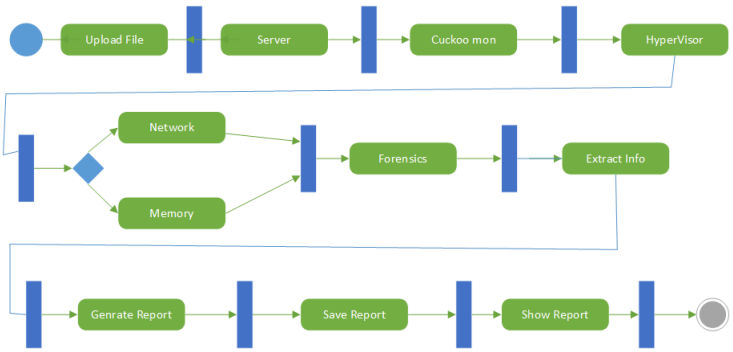

Figure 4.4: Activity Diagram : Upload a File

Figure 4.4, represents workflows and stepwise actions and activities and choices for the use case upload a file

4.3.4 Sequence Diagrams

4.3.4.1 Sequence Diagram I: login

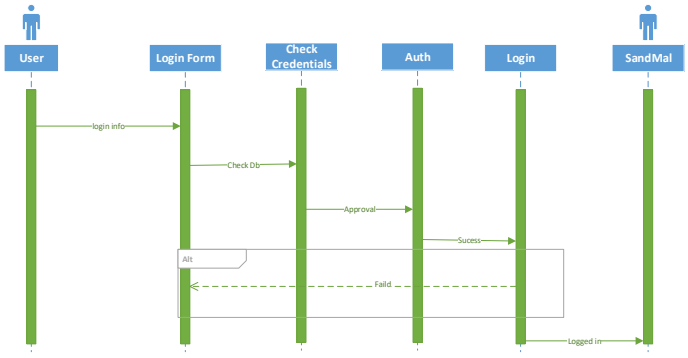

Figure 4.5: Sequence Diagram I: Login

Figure 4.5 shows the sequence of steps necessary to login to the system

4.3.4.2 Sequence Diagram II: Register

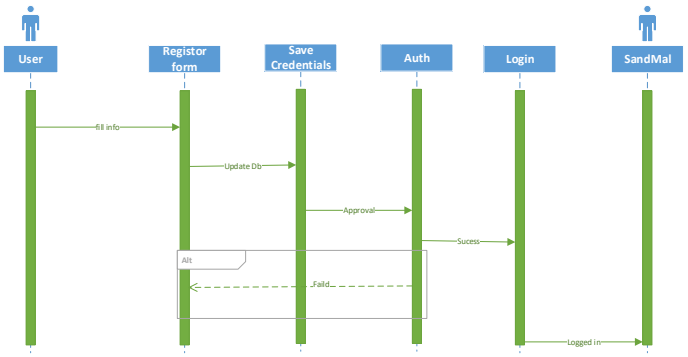

Figure 4.6: Sequence Diagram II: Register

Figure 4.6 shows the sequence of steps necessary to Register to the system

4.3.4.3 Sequence Diagram III: logout

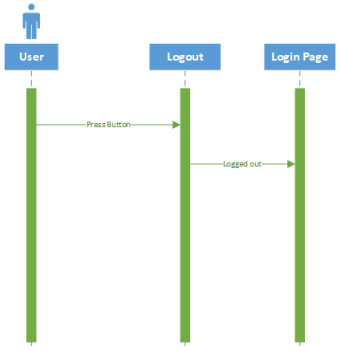

Figure 4.7: Sequence Diagram III: logout

Figure 4.7 shows the sequence of steps necessary to logout from the system

4.3.4.4 Sequence Diagram IV: upload file

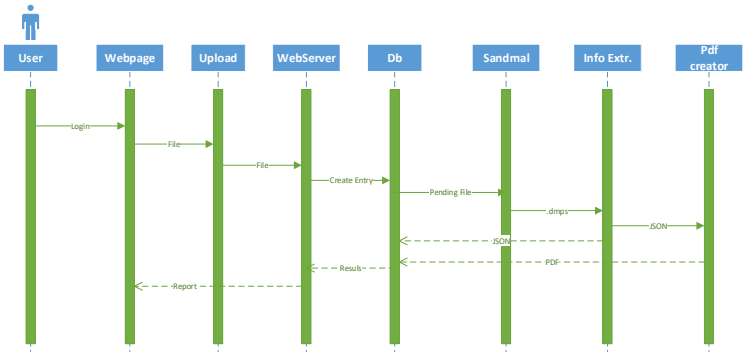

Figure 4.8: Sequence Diagram IV: upload file

Figure 4.8 shows the sequence of steps necessary to upload a file to the system

4.3.4.5 Sequence Diagram V: Search report

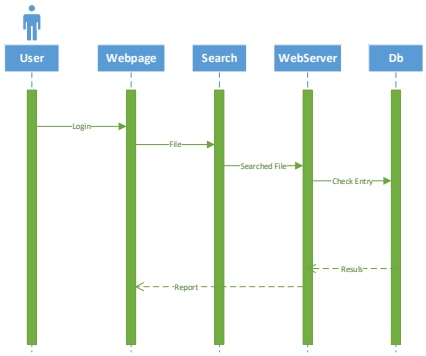

Figure 4.9: Sequence Diagram V: search report

Figure 4.9 shows the sequence of steps necessary to Search a report from the system

4.3.4.6 Sequence Diagram VI: View report

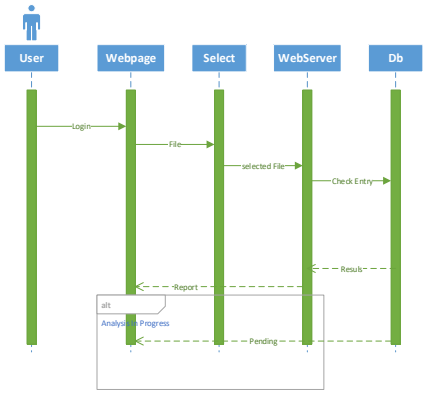

Figure 4.10: Sequence Diagram VI: view report

Figure 4.10 shows the sequence of steps necessary to view a report from the system

4.3.4.7 Sequence Diagram VII: download report

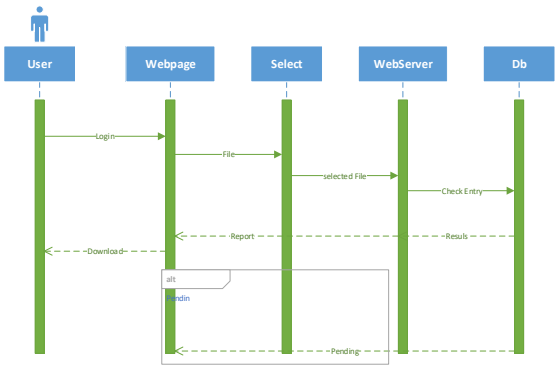

Figure 4.11: Sequence Diagram VII: download report

Figure 4.11 shows the sequence of steps necessary to download a report from the system

4.4 DATABASE DESIGN

Database design shows the organization of data in this system. Database design of this system properly classify data and relationship among the entities. Database design represents theoretical representation of the data.

4.4.1 Entity Relationship diagram

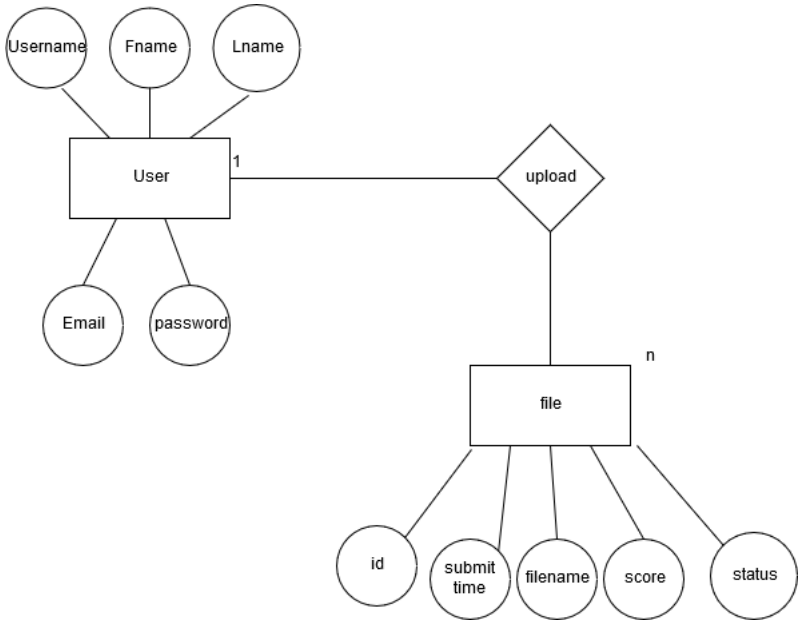

Figure 4.12: Data Flow Diagram

Figure 4.12 represents an entity-relationship model of SandMal system.

4.4.2 Domain tables

4.4.2.1 User

Table 4.1: User

Domain	Data Type	Length	Is Unique	Defualt
First Name	Varchar	64	No	Null
Last Name	Varchar	64	No	Null
User Name	Varchar	128	Yes	Not Null
Email	Varchar	256	Yes	Not Null
Password	Varchar	128	No	Not Null

Table 4.1 represents the user entity and the available fields, their default values, datatypes, length and uniqueness.

4.4.2.2 File

Table 4.2: File

Domain	**Data Type**	**Length**	**Is Unique**	**Defualt**
ID	int		Yes	autogenerated
File Name	Varchar	1024	No	Null
Score	Double		No	0.0
Status	Varchar	64	No	Pending
Submit Time	Timestamp		No	autogenerated

Table 4.2 represents the File entity and the available fields, their default values, datatypes, length and uniqueness.

Chapter 5

System Implementation

This chapter shall discuss the details of system implementation and the tools and the programming languages used to implement this system.

5.1 System Architecture

As mentioned above, our system uses client-server architecture. Now we will explain our architecture in detail.

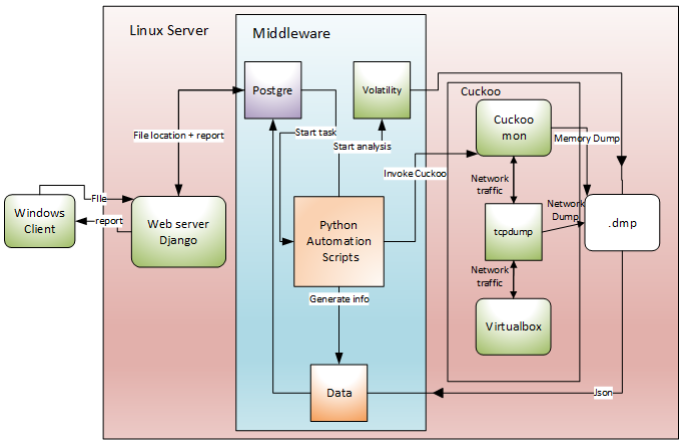

Figure 5.1: Shows System Architecture

Figure 5.1 represents a conceptual model of this system; this diagram represents and defines the structure and behavior and different components of this system how they interact with each other and implement the overall system.

5.1.1 Components and their functionality

5.1.1.1 Scheduling

Since by design this system support multiple user and multiple users can upload files, it is important to schedule the task in our system so system can efficiently manage the resources.

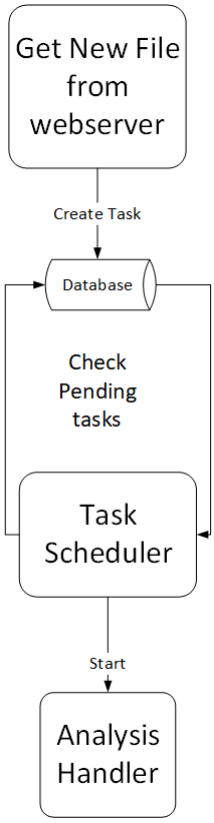

Figure 5.2: Shows Scheduling

Figure 5.2 represents the Scheduling component of the system, it shows steps performed in the system when users upload multiple files in the system. A task entry with status pending is created in database, Task scheduler will create a simple queue of pending task and execute each task when it finds free resources.

5.1.1.2 Automation

This component of the system includes the automation scripts that invoke and manage the cuckoomon in the system. These scripts also invoke Forensics utility. This is an important component of the system since it handles all the automation in the system.

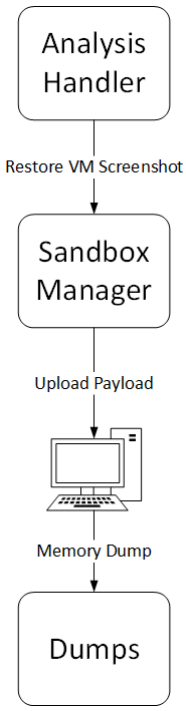

Figure 5.3: Shows Automation

Figure 5.3 represents how system will manage and execute the analysis. When Analysis handler receives a file from the scheduler it will restore a virtual machine and upload the sample in virtual machine which will then execute on the virtual machine, during the installation a network sniffer will log all network traffic and after the installation memory dump of the process will be send back to the system

5.1.1.3 Sandboxing

This component is also an important part of the system since this component manages the sandboxing in our system. In this component Two operating systems i.e. Microsoft windows XP and Microsoft windows 7 are emulated in virtual machine. This component also handles the network traffic of the virtual machines. These virtual machines are configured with host-only adapter and a sniffer collects all the network traffic

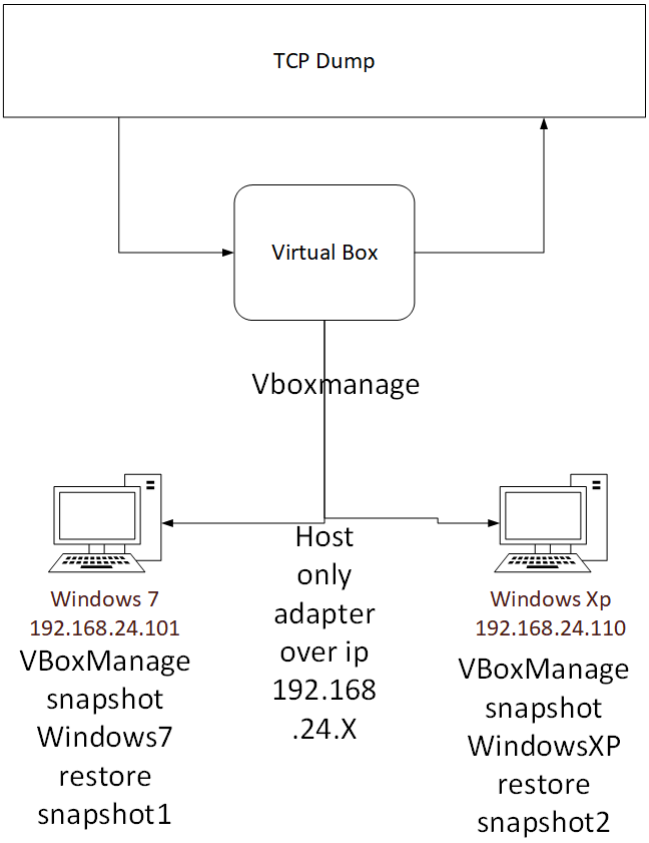

Figure 5.4: Shows Sandboxing

Figure 5.4 shows how multiple virtual machines are connected with the Virtual Box Manager. And a sniffer that will log all the network traffic going in and out of virtual machines.

5.1.1.4 Forensics

This component handles the functionality of memory and network forensics , this component has configured the volatility plugins that run and extract valuable information form memory and network dump.

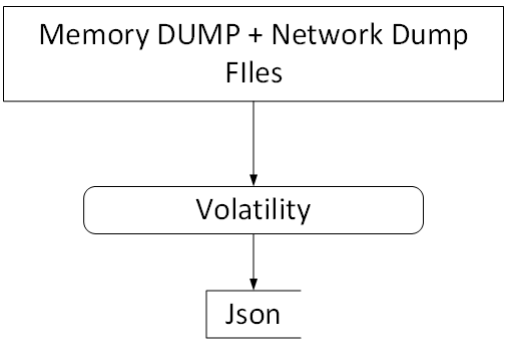

Figure 5.5: shows Memory forensics

Figure 5.5 simple represents, memory and network dump is supplied to Volatility which will extract useful artifacts from the dumps.

5.2 Tools and technology Used

This system is implemented using Python language using Django web framework. This system also uses Cuckoomon for DLL injection, volatility for memory forensics. The client side of the system was designed using BOOTSTRAP 4 framework.

5.2.1 Cuckoomon

This module is used for Hooking Engine of the system. This is an open- source tool that is available for free.

5.2.2 Django Web framework

Django Web framework module is used for the development of server.

5.2.3 Volatility

This module is used for the Memory and Network Analysis of the file. This is also an open-source tool available for free.

5.2.4 PostgreSQL

PostgreSQL RDBMS is used for the database configuration of the system.

5.2.5 Bootstrap 4

Bootstrap 4 is a frontend framework. This tool was used to build the webpages and Graphical User Interface.

5.2.6 VirtualBox

This tool was used to provide sandboxing environment in our system.

5.3 Development Environment and languages

5.3.1 Ubuntu OS

Ubuntu 16.04 Operating system is used as the underlying operating system for our system.

5.3.2 Visual Studio code

This is an Integrated development Environment that was used for the development of the system.

5.3.3 Python

Since many modules such as Cuckoomon and volatility are opensource and written in python language, system uses python as main development Language. This helps save time in integration of different modules.

Chapter 6

System Testing and Evaluation

This chapter will discuss the System testing and evaluation of the system. System testing is a crucial step in order to inject quality into the product. System testing is done to ensure bugs are removed from the system and the product will conform to the requirements. This chapter will discuss the techniques used to evaluate the system.

6.1 Graphical User Interface Testing

Graphical User interface testing is a technique is used to evaluate the Graphical user interface of the system meets the expected user requirement. This technique is important to ensure that User interface of the system is simple and easy to understand for the user. GUI testing ensures that system is easy to navigate and meets expected requirements of the user.

6.2 Usability Testing

Our system is designed in such a way it is very easy to use and require no technical understanding of user interfaces. Our system provides a simple web page where users can login and submit their files for analysis

6.2.1 Unit Testing

In this testing technique the goal is to test the individual parts of the system instead complete system at once. This helps ensures that each unit in the system is giving accurate results according to the requirements of the system.

6.2.2 White box Testing

In White box testing technique, the system was tested for system's internal structure, design and coding. This technique primarily focuses on flow of inputs and their generated outputs. So to figure out if the system is giving accurate results.

6.2.3 Black box Testing

In black box testing technique, the system was tested without looking at the internal structure of the system and implementational details. This technique is used to validate the functional requirements of the system.

6.3 Test Cases

6.3.1 Test Case I: Username Validation

Table 6.1: Test Case I: Username Validation

Test Case ID	**TC-01**
Use case ID	UC-01
Title	Username validation
Description	The user will enter information to register to the system.
Pre-condition	The user is not already registered
Steps	Users enters Name, user name, email and password fields The user presses submit
Expected results	Error- Use Valid username
Status	Pass

6.3.2 Test case II: Email Validation

Table 6.2: Test case II: Email Validation

Test Case ID	**TC-02**
Use case ID	UC-01
Title	email validation
Description	The user will enter information to register to the system.
Pre-condition	The user is not already registered.
Steps	Users enters Name, user name, email and password fields The user presses submit
Expected results	Error- Use Valid email
Status	Pass

6.3.3 Test case III: Account Already Exist

Table 6.3: Test case III: Account Already Exist

Test Case ID	**TC-03**
Use case ID	UC-01
Title	Account Already Exist
Description	The user will enter information to register to the system.
Pre-condition	The user is not already registered
Steps	Users enters Name, user name, email and password fields The user presses submit
Expected results	Error- User already exist
textbfStatus	Pass

6.3.4 Test case IV: Incorrect username

Table 6.4: Test case IV: Incorrect username

Test Case ID	**TC-04**
Use case ID	UC-02
Title	Incorrect Username
Description	The user will enter information login to the system.
Pre-condition	The user is already registered
Steps	Users enters user name and password fields The user presses submit
Expected results	Error- Incorrect User name
Status	Pass

6.3.5 Test case V: incorrect password

Table 6.5: Test case V: Incorrect Password

Test Case ID	**TC-05**
Use case ID	UC-02
Title	Incorrect Password
Description	The user will enter information to login to the system.
Pre-condition	The user is already registered .
Steps	Users enters user name and password fields The user presses submit
Expected results	Error- Incorrect password
Status	Pass

6.3.6 Test case VI: incorrect extension

Table 6.6: Test case VI: Incorrect extension

Test Case ID	**TC-06**
Use case ID	UC-02
Title	Incorrect Extension
Description	The user will select file to upload.
Pre-condition	The user is already logged in.
Steps	Users will select the file The user presses submit
Expected results	Error- Incorrect File extention
Status	Pass

Chapter 7

Conclusions

Today as computers and other mobile devices are becoming increasingly common and essential to business and organizations. Information system security has emerged as one of the most important research field in computer science. This was the main motivation factor for the both team members, to research and learn new techniques that are used to protect information systems. This project has provided us a great learning opportunity. The system developed i.e. Lightweight dynamic sandbox will allow an authorized user to upload a file to the system using a web interface then the server will process this request and generate an analysis report. This approach that is used for the implementation of this system will protect users from running a potential malware file on their system and hide any implementational details of the system. This provides an excellent opportunity for the user to test the file for being potential malware. The system is intentionally designed to have simpler User Interface that will not require any major learning curve for new users.

7.1 Results

Since this system uses multiple virtual machines it is important to understand that the server side of the system will utilize 5GB to run one instance of Windows 7 and 2.5 GB to run one instance of Windows XP separately so it is important to schedule the tasks to properly handle system resources.

7.2 Future improvements

This is the initial version of this system and further Improvements and newer features can be added to the system.

7.2.1 Newer Platforms

This system currently uses Microsoft windows desktop platform, the system can be extended to include support for Google Android mobile operating system. As Android OS is the most common mobile operating system [8]. Adding functionality to add support for Android OS will be an excellent achievement.

7.2.2 improvements in User Interface

User interface can be improved by adding more user-friendly features to the system.

Appendix A

User Manual

The steps to use the system are given below.

A.1 User Interfaces

A.1.1 Register

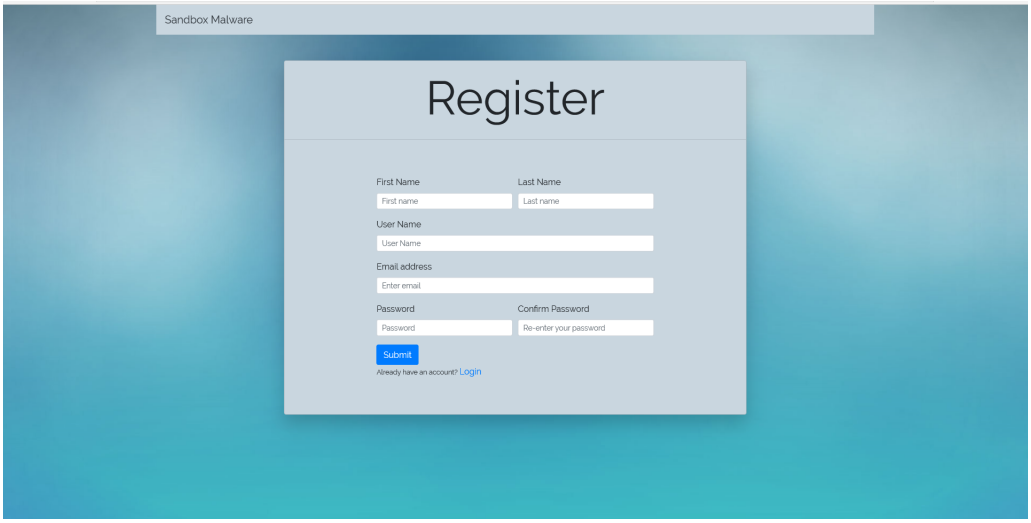

Figure A.1: Register for user access

The figure A.1 shows us the register page. A user must have been register to use the system. A user will enter their info to get the access of the system.

A.1.2 Login

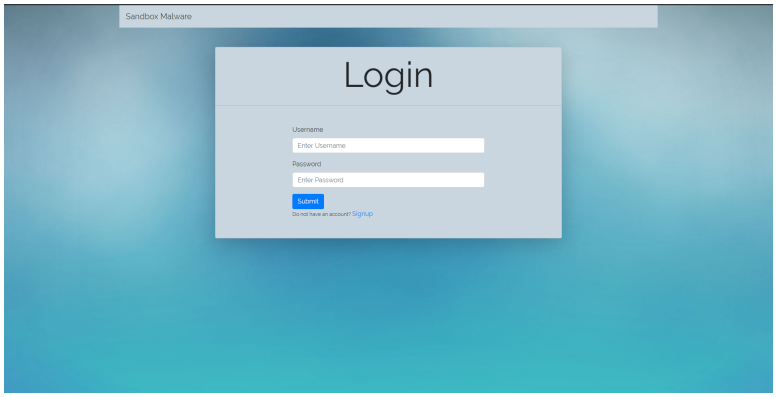

Figure A.2: login to system

The figure A.2 shows us the login page.From here the register user can go to the upload file page from where the system can be used.

A.1.3 Upload File

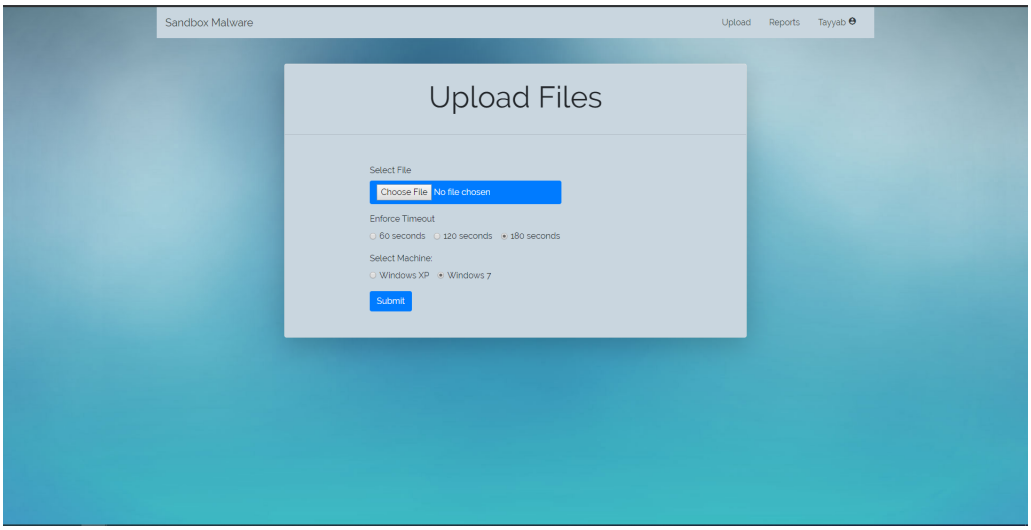

Figure A.3: upload file to the system

The figure A.3 shows us the menue to upload file. The file uploaded here with be analyzed.

A.1.4 Search Report

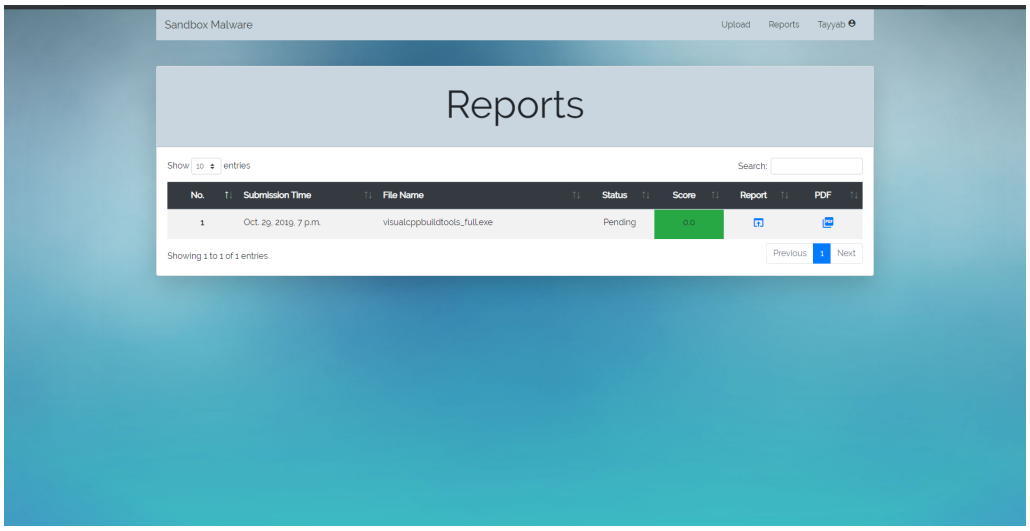

Figure A.4: search report

The figure A.4 shows us the menue to search file.

A.1.5 View Report

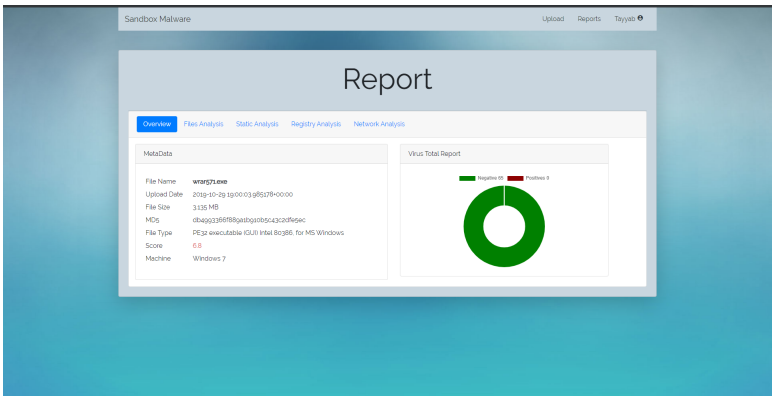

Figure A.5: view report

The figure A.5 shows us the report interface.

Bibliography

[1] Oxford Economies, "The New Digital Economy, How it will transform business" Cited on p. 1.

[2] 2019. [Online]. Available: https://netmarketshare.com/operating-system-market-share.aspx. Cited on p. 1.

[3] C. K. a. E. K. Andreas Moser, "Exploring Multiple Execution Paths for Malware Analysis," Secure Systems Lab , 2007. Cited on p. 5.

[4] Symantec., "Internet Security Threat Report: Volume 24," 2019. [Online]. Available: https://www.symantec.com/security-center/threat-report. Cited on p. 5.

[5] K. B. Savan Gadhiya, "Techniques for Malware Analysis," International Journal of Advanced Research in Computer Science and Software Engineering, vol. 3, no. 4, p. 4, 2013. Cited on p. 7.

[6] J. Atwood, "Understanding User and Kernel Mode," Jan 2008. [Online]. Available: https://blog.codinghorror.com/understanding-user-and-kernel-mode/. Cited on p. 8.

[7] Galen Hunt, Doug Brubacher, "Detours: Binary Interception of Win32 Functions," in WINSYM'99 Proceedings of the 3rd conference on USENIX Windows NT Symposium - Volume 3, Seattle, Washington , 1999. Cited on p. 8.

[8] https://www.digitalseoguide.com/technology/top-mobile-phones-operating-systems-os/ Cited on p. 40.