**An approach for malware analysis using hashing**

Submitted in partial fulfillment of the requirement of the degree

**BACHELOR OF ENGINEERING IN COMPUTER ENGINEERING**

By

**Anuj R Tiwari 19CE1088**

**Prathamesh S Vanjape 19CE1010**

**Rohan B Tirmakhe 19CE1036**

Supervisor

**Mrs. Tabassum Maktum**



**Department of Computer Engineering Dr. D. Y. Patil Group’s Ramrao Adik Institute of Technology**

**Dr. D. Y. Patil Vidyanagar, Sector 7, Nerul, Navi Mumbai 400706. University of Mumbai**

**(Ay 2020-21)**

**CERTIFICATE**

This is to certify that the Mini Project entitled **“An approach for malware analysis using hashing”** is a bonafide work of **Anuj R Tiwari (19CE1088), Prathamesh S Vanjape (10CE1010), Rohan B Tirmakhe (19CE1036)** submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of **“Bachelor of Engineering”** in **“Computer Engineering”.**

( **Mrs. Tabassum Maktum** )

Supervisor

**( Dr. Leena Ragha ) ( Dr. Mukesh D. Patil )**

Head of Department Principal

**Mini Project Approval**

This Mini Project entitled “An approach for malware analysis using hashing**”** by **Anuj R Tiwari (19CE1088), Prathamesh S Vanjape (19CE1010)** and **Rohan B Tirmakhe (10CE1036)** is approved for the degree of **Bachelor of Engineering** in **Computer Engineering.**

## Examiners

**1…………………………………………**

(Internal Examiner Name & Sign)

**2…………………………………………**

(External Examiner name & Sign)

Date:

Place:

# Abstract

Malware is a short form of MALICIOUS SOFTWARE, which is a collective phrase for all software developed for disrupting, damaging, or gaining access to data and systems in an unauthorized manner. Malware has remained a consistent threat since its emergence, growing into a plethora of types and in large numbers. Malware Authors have managed to increase their malware’s sophistication to avoid detection against anti-malware technics by implementing new features and specific modifications, such as encryption, polymorphism, and metamorphism to maximize their resilience. Our motive is to Identify the identical files using the context-triggered piecewise hashing (CTPH) technique. “SSDEEP” is capable to find and compares identical files with existing data. “SSDEEP” is a program for computing context-triggered piecewise hashes (CTPH). Also called fuzzy hashes, CTPH can match inputs that have homologies. Such inputs have sequences of identical bytes in the same order, although bytes in between these sequences may be different in both content and length. Fuzzy hash functions hold a certain tolerance for changes and can tell how different two files are by comparing the similarity of their outputs.

## Acknowledgement

We take this privilege to express our sincere thanks to Dr. Mukesh D. Patil, Principal, RAIT for providing the much necessary facilities. We are also thankful to Dr. Leena Ragha, Head of Department of Computer Engineering, Project Co-ordinator Dr. Vanita Mane, Department of Computer Engineering, RAIT and Project Guide Mrs. Tabassum Maktum, Department of Computer Engineering, RAIT for their generous support.

We take this opportunity to express our profound gratitude and deep regards to our guide Mrs. Tabassum Maktum for her exemplary guidance, monitoring, and constant encouragement throughout the completion of this report. We are truly grateful for her efforts to improve my understanding of various concepts and technical skills required in our project. The blessing, help, and guidance given by her from time to time shall carry us a long way in the journey of life on which we are about to embark.

Last but not least we would also like to thank all those who have directly or indirectly helped us in the completion of this thesis.

# List of Figures

3.1 Architecture Framework 5

3.2 Use Case Diagram 7

3.3 Activity Diagram 8

3.4 Main Page 9

3.5 Add Hash Page-i 10

3.6 Add Hash Page-ii 10

3.7 Compare Files Page-i 11

3.8 Compare Files Page-ii 11

3.9 File Scan Page-i 12

3.10 File Scan Page-ii 12

3.11 Folder Scan Page-i 13

3.12 Folder Scan Page-ii 13

3.13 View Hash Value Page-i 14

3.14 View Hash Value Page-ii 14

3.15 View History Page 15

## Contents

**Abstract i**

**Acknowledgments ii**

**List of Figures iii**

1. **Introduction 1**

Introduction 1

Motivation 2

Problem Statement & Objectives 2

Organization of the Report 2

1. **Literature Survey 3**

Survey of Existing System3

Limitation of existing system or research app3

Mini Project Contribution3

1. **Proposed System 4**

Architecture/Framework4

Algorithm and Process Design5

Details of Hardware & Software8

Experiment and Result9

1. **Conclusion and Future Work 16**

**References 17**

### 1. Introduction

Malware is a short form of MALICIOUS SOFTWARE, which is a collective phrase for all software developed for disrupting, damaging or gaining access to data and systems in an unauthorized manner. Malware has remained a consistent threat since its emergence, growing into a plethora of types and in large numbers. Malware Authors have managed to increase their malware’s sophistication to avoid detection against anti-malware technics by implementing new features and specific modifications, such as encryption, polymorphism and metamorphism to maximize their resilience. However, anti-virus vendors and analysts managed to adapt their identification techniques by relying on automated analysis methods, and tools in order to distinguish malicious from benign code, such as the traditional static analysis using “SSDEEP

HASH” being the most commonly used in Malware Research.

Malware analysis can be performed in two different modes, the static mode, and the dynamic mode. The static mode does not run the suspicious samples and is safer, while the dynamic mode executes the suspicious samples and is more sensitive. There are several techniques for malware analysis out of which we chose the “Hashing” technique as is one of the fastest and efficient technique for such preliminary analysis. In hashing cryptographic hash which is an algorithm that takes data as input and produces a fixed-size output called a hash value is used in malware detection. The common cryptographic hashes to detect identical known malware analysis are md5, sha1, sha254, etc.

In our project we have the ”SSDEEP” (similarity digest) program which is a block-based hash program. “ssdeep” is a block-based hash program which means it divides the data into segments and produces a digest or checksum for each segment using any hash function such as md5, sha1,sha256, etc after which it produces the hash signature (value) of that data by linking the hash functions of the segments. “ssdeep” is effective in finding similarity between text files as it was initially introduced for spam detection, we chose “ssdeep” in our project to find similarity between the text files using hash values.

**1.1. Motivation:**

There has been an exponential growth in the malware attacks in this increasing digitalization. A Malware detection tool significantly helps us to protect our system from such harmful attacks. We aim to develop a Malware Detection tool which will not only help us to detect malicious files in a system but also to keep our files safe. For this tool we decided to use hashing techniques as hashing is one the fastest and reliable technique for such type of systems.

1. **2.Problem Statement and Objective:**

The project aims to create a Malware Analysis Tool. The basic idea of the project is to use the hashing or other encryption algorithm to find an efficient way for evaluating text files. The application must be able to differentiate between the safe and the malware files by comparing the hash value of a file with the existing database.

**Objective:**

The main motive of the topic is to develop a tool with one of the best techniques suitable for such systems to maintain the integrity of our system and files.

* + To decide whether a file is malware or not
  + Get the hash value of a file to check its integrity  Compare similarity between two text files

**1.3. Organization of Report:**

This report gives a brief summary of the project which includes details about every component of the project. Chapter 1 is an introduction to the project and tells motivation for developing this project and objectives about it and also gives an overview of the project. Chapter 2 is a literature survey we made while analyzing all the requirements of the project. In Chapter 3 we have mentioned architecture design/framework, implementation details, the technology used to develop the system, and results. In Chapter 4 conclusion and future work are described. Chapter 5 consists the references that have been used while developing this project.

### 2. Literature Survey

**2.1. Survey of Existing System:**

Similar systems have been found on internet. For developing our system the most important feature was to survey the existing systems present in the market and noting down the main features and the features which were lacking in them. While going through various systems we came across the fact that many systems are still using the md-5 hash algorithm for malware detection, even though practical collisions have been seen in md-5 hashes since 2006.

Some of the examples are listed below:

1. Kaspersky Total Security **[2]**
2. Bromium **[3]**

The above listed antivirus uses the MD-5 hash algorithms as per the “jumpespjump” **[1].**

* 1. **Limitation of Existing System:**

The above systems are using MD-5 hashes even though practical collisions have been seen using this algorithm. Also md5 hash is not able to detect if there will a slight change in a malware file as it produces a completely different hash value of similar files having difference of even a single bit.

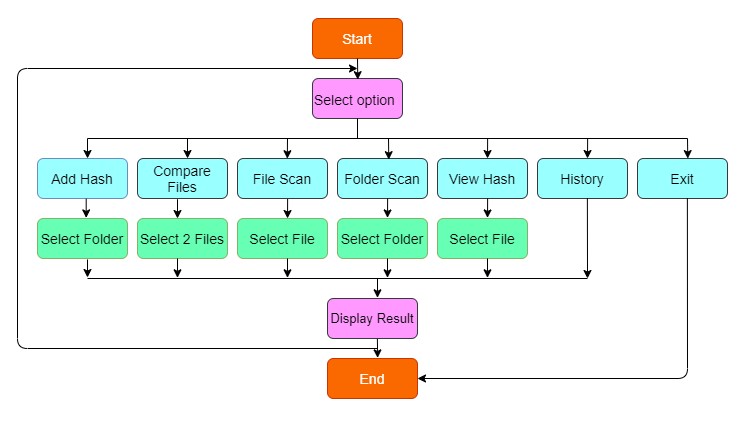
* 1. **Mini project Contribution:**

Our Malware Detection tool is a crucial tool as it helps to main integrity of our system files. This tool not only helps us to detect whether a file is malware or not but also detects a file which has similarity to any malware file. Adding the hash value in the database after every scan when a file is detected as malware. Another important factor is that the system gives the hash values of a file to check its integrity.

### 3. Proposed System

**3.1. Architecture/Framework:**

The proposed model is shown below. The user gets an panel on the left side of the application where he/she can select from the options present as per his/her need. The options on the left panel includes options such as Add Hash, Compare File, File Scan, Folder Scan, View Hash, and History.



#### Figure 3.1 Architecture Framework

**3.2.Algorithm and Process Design:**

**Algorithm:**

Algorithm means a process or a set of rules that needs to be followed. Therefore algorithm refers to a set of rules/instructions that step by step define how a work is to be executed upon in order to get the expected result. Algorithms are language independent that is they are just plain instructions that can be implemented in any language, and yet the output will be the same, as expected. The algorithm that is the basic set of instructions that users can opt for fir desired output is shown below.

Step 1: Start

Step 2: Select option from left Panel

If option == Add Hash

Then user can enter hash values of pre-known malware files into database If option == File Scan

Then user can view whether the selected file is malware or safe

If option == Folder Scan

Then user can view whether the files present in the selected folder are malware or safe If option == View Hash

Then user can view the hash values (md5, ssdeep, sha256, sha512 and sha224) of a selected file

If option == History

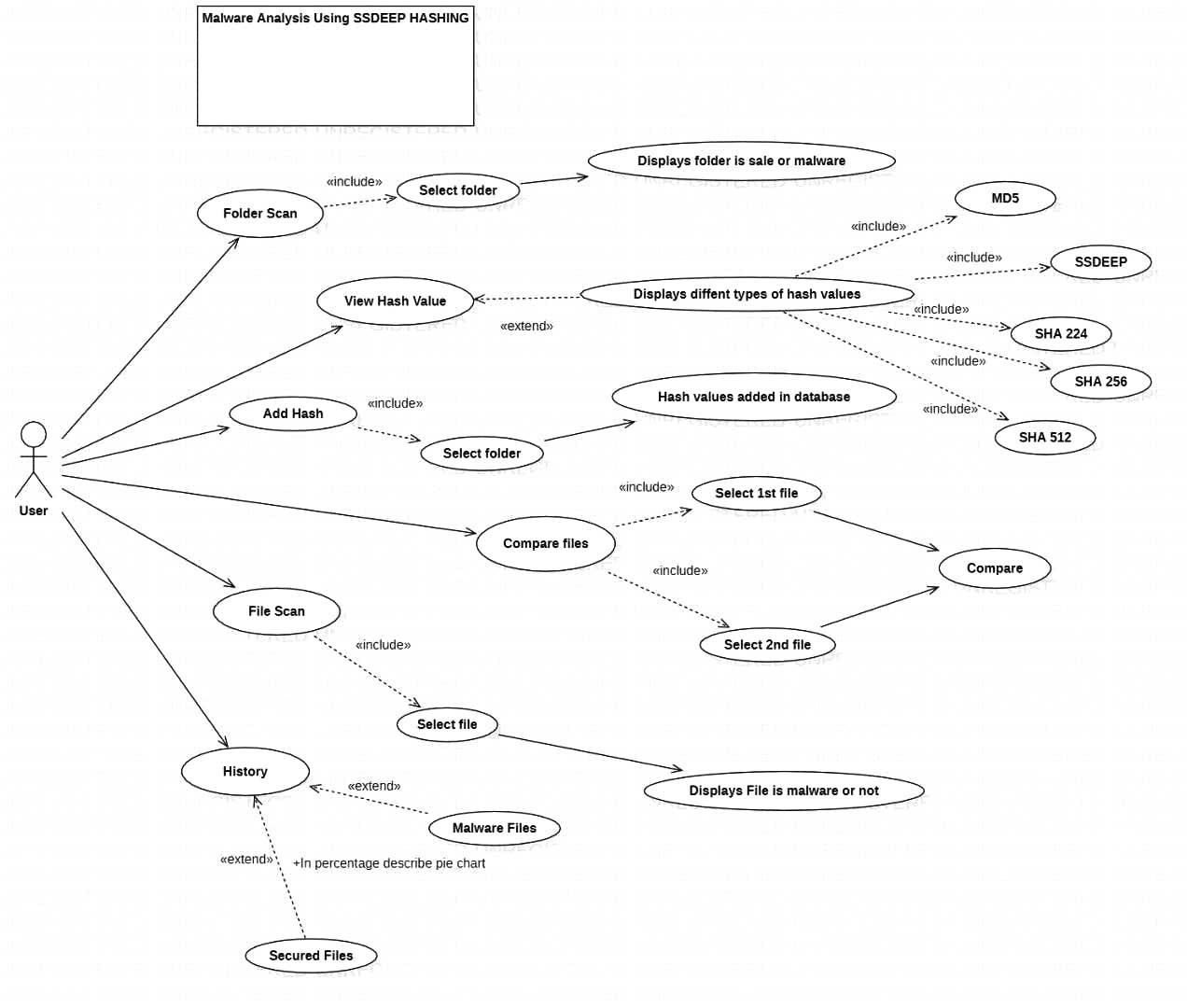
Then user can view the summary of the total number of files scanned in his/her system If option == exit

Then the user can exit from the system

**Process Design:**

* **Use Case Diagram**

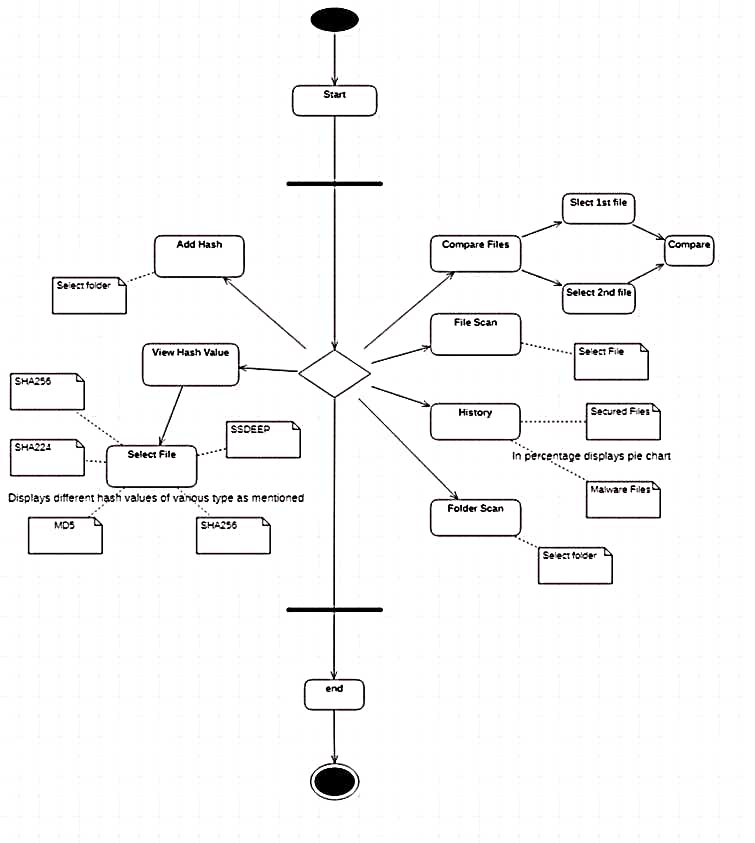
A use case diagram is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. A single use case diagram captures a particular functionality of a system. Here use case diagram of “An approach for malware analysis using hashing” is given



**Figure 3.2 Use Case Diagram**

* **Activity Diagram:**

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. It is a behavioral diagram. The activity can be described as an operation of the system. The control flow is drawn from one operation to another.



**Figure 3.3 Activity Diagram**

**3.3. Details of Hardware & Software:**

The Language we have used in this project is PYTHON.

The libraries used from Python language are

* **tkinter**: Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Our main project part is programmed using Tkinter library. We chose this library as it gave us an option to create multiple frames which was a key need in our project.
* **ssdeep**: ssdeep is a python library for computing context triggered piecewise hashes (CTCH). Also called fuzzy hashes, CTCH can match input that have homologies. Such inputs have sequence of identical bytes in the same order, although bytes in between these sequence may be different in both content and length.
* **matplotlib**: Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in python. We used these library to display the history of scanned files of a particular user in a graphical format.

**3.4. Experiment and Result:**

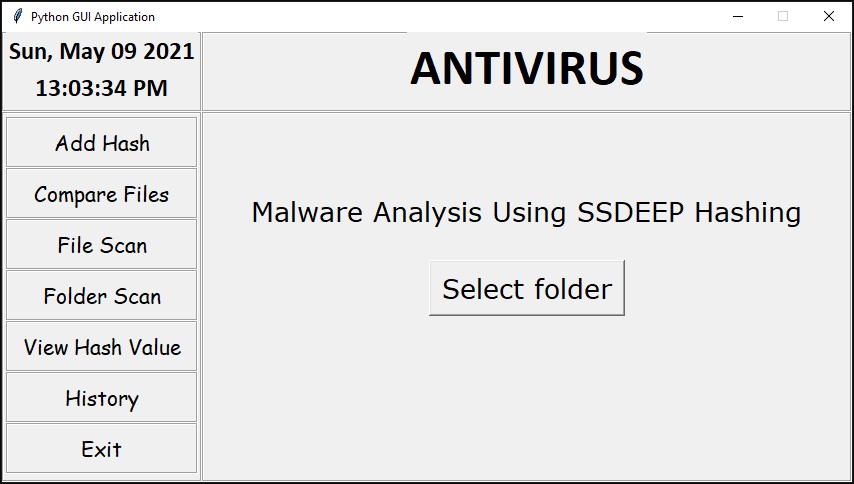
Below are the final pages/frames of our project:

* **Main Page**: It is the home page of our project, which contains the current date and time in the top left side. There is a panel on the left side of the screen with buttons of different features of the project.

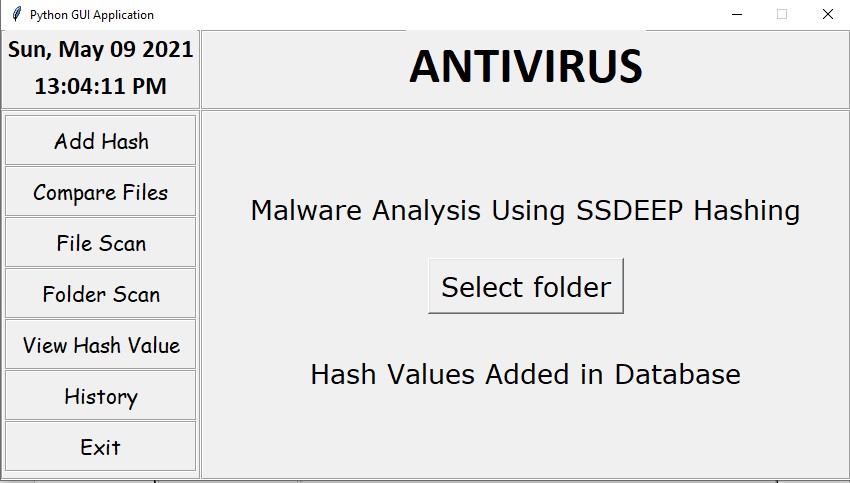


#### Figure 3.4 Main Page

**Add Hash Page:** This page gives access to add update the database of malware files. This page asks the user to enter

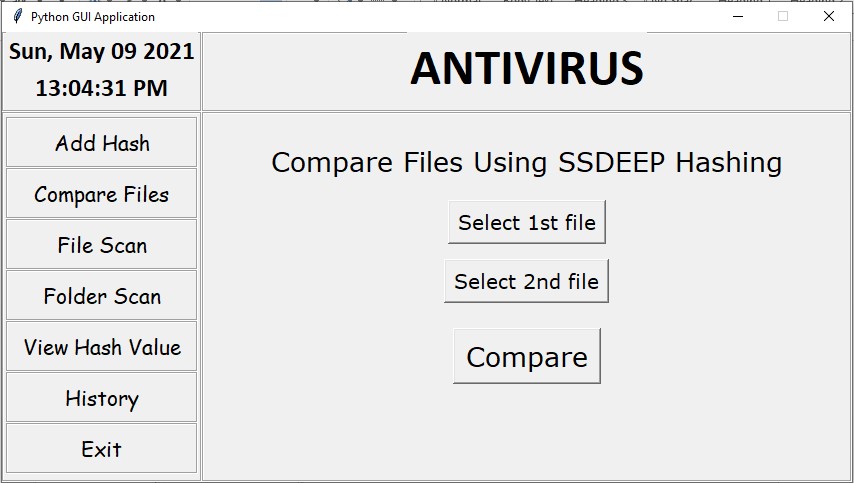


#### Figure 3.5 Add Hash Page-i

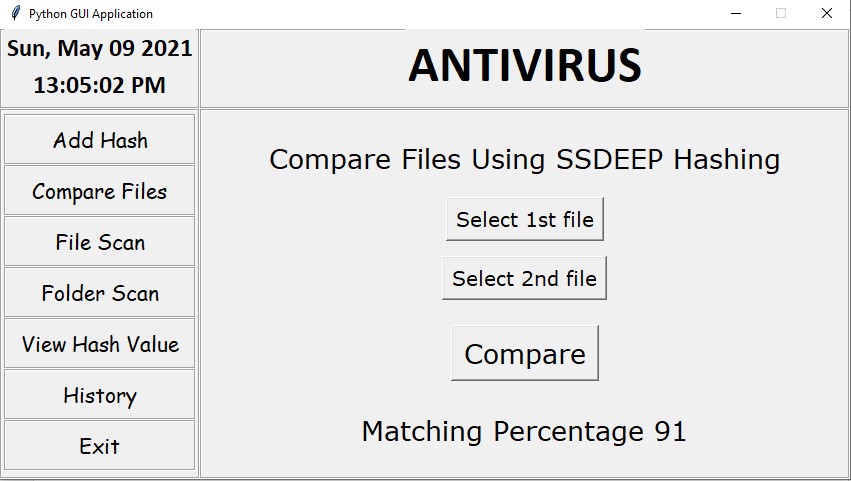


#### Figure 3.6 Add Hash Page-ii

**Compare Files**: In this section the user can select two files from the system after which, on pressing the compared button the matching percentage between those two selected files will be displayed on the screen.

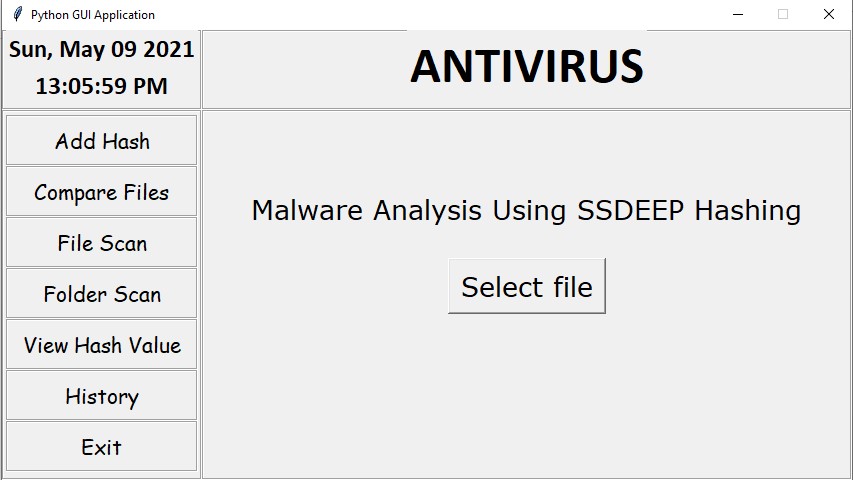


**Figure 3.7 Compare Files Page-i**

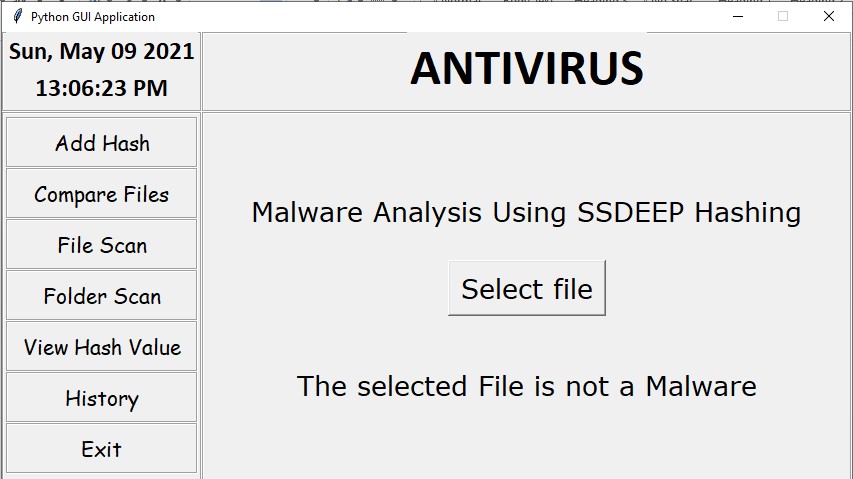


**Figure 3.8 Compare Files Page-ii**

**File Scan:** In this section the user has to select followed by which the result whether the file is a malware or not will be displayed on the screen.

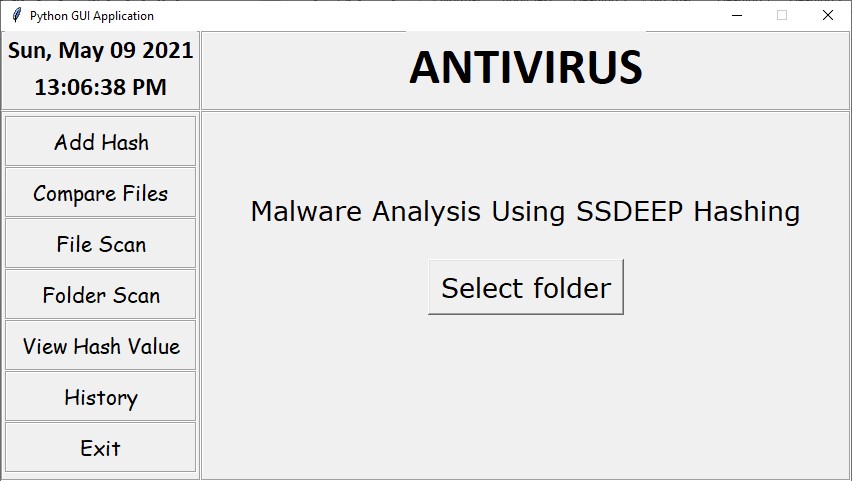


#### Figure 3.9 File Scan Page-i

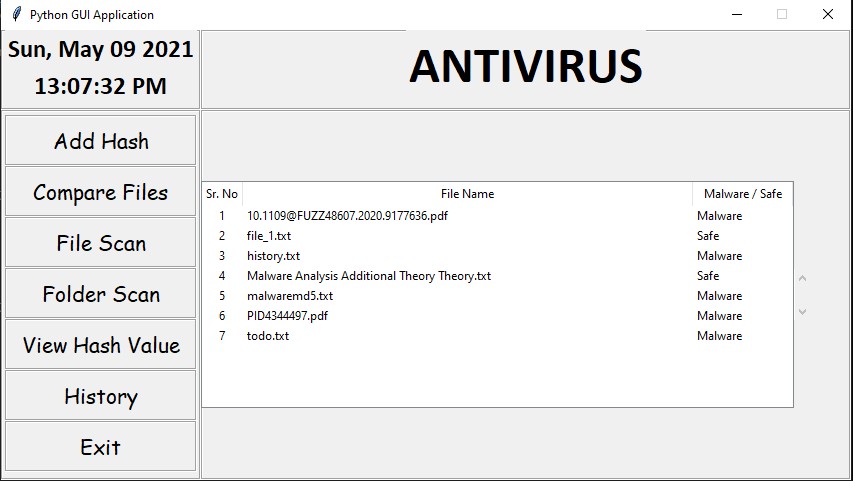


**Figure 3.10 File Scan Page-ii**

**Folder Scan**: In the folder scan session the user is asked to select a particular folder. The files in that folder will be scanned one by one and the result whether the file is a malware file or not will be displayed on the screen.

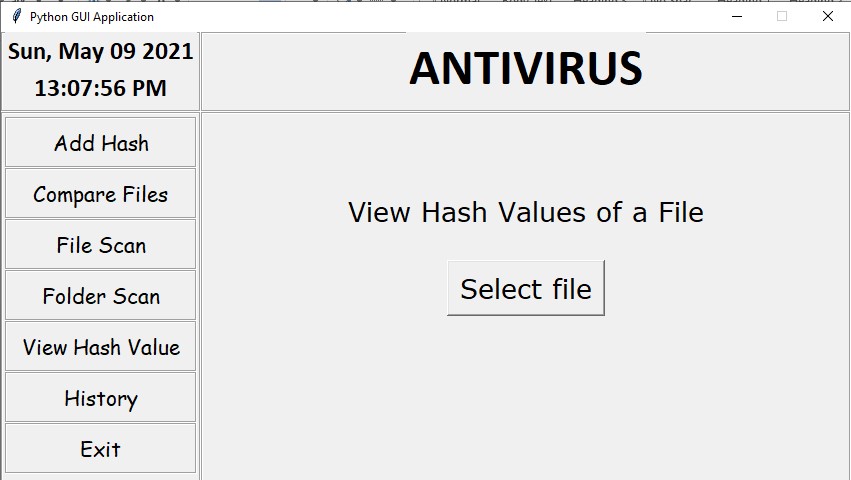


**Figure 3.11 Folder Scan Page-i**

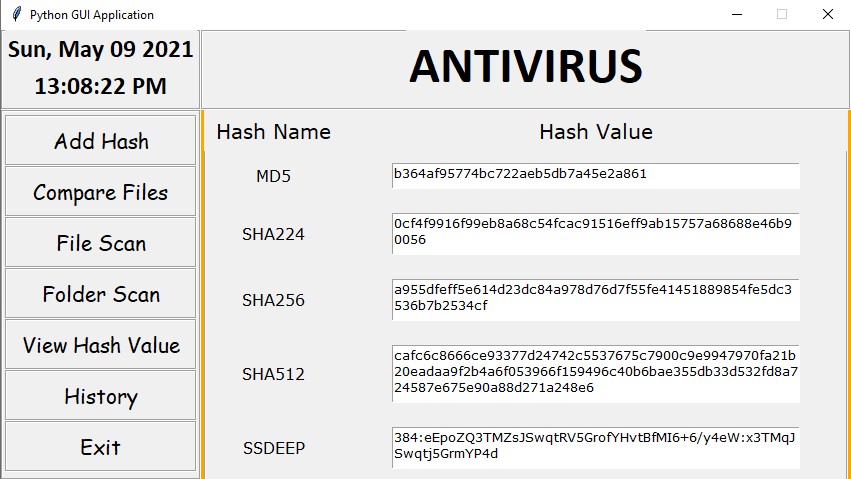


**Figure 3.12 Folder Scan Page-i**

**View Hash Value:** In this section the user is asked to select a file after with a list of hashes of that particular file will be displayed such SMD5, SHA224, SHA256, SHA512 and SSDEEP. User can use this hash values to to verify the integrity of that file.

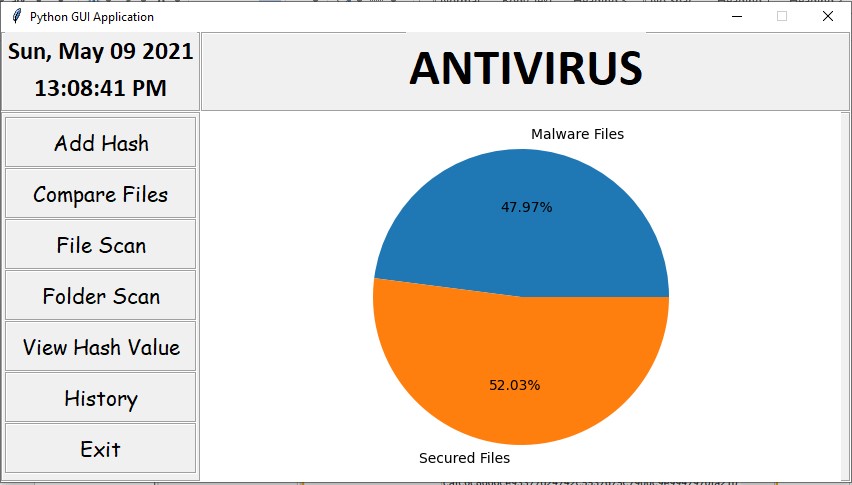


#### Figure 3.13 View Hash Value Page-i



#### Figure 3.14 View Hash Value Page-ii

**History**: In the history section, a graph is displayed. The graph displays the total number of malware files and secured files scanned by a particular user. The data used in this section is updated after every scan done by the system.



**Figure 3.15 View History Page**

### 4. Conclusion and Future Work

We have seen a big spike in the increasing number of attacks done with different malwares and ransom ware with the increasing digitalization. Thus we decided to implement a method for malware analysis to improve the similarity detection, accuracy and performance of the analysis. Currently the project has been implemented using the SSDEEP hash function which can be further upgraded to advanced algorithms in the future.

**Future Scope**

* Using different algorithms such as mvHash, sdHash or a combination of hash algorithms along with the SSDEEP hash to get more precise results.
* Using SSDEEP along with various other algorithms to make the system scan through files in less time.
* Adding a Drive Scan section to scan all the folders present in the drive.

### References

1. **https://jumpespjump.blogspot.com/2014/03/stop-using-md-5-now.html**
2. [**https://www.kaspersky.co.in/total-security**](https://www.kaspersky.co.in/total-security)
3. **https://www.bromium.com/tag/anti-virus/**
4. Tkinter Library: **https://docs.python.org/3/library/tkinter.html**
5. Ssdeep Library: **https://pypi.org/project/ssdeep/**
6. Matplotlib Library: **https://matplotlib.org/**