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

D Y PATIL
RAMRAO ADIK INSTITUTE OF TECHNOLOGY
NAVI MUMBAI

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 (19CE1010), 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)
Head of Department

(Dr. Mukesh D. Patil)
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 (19CE1036)** 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 techniques 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 of finding and comparing 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

Contents

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 techniques 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 techniques 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 that 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 techniques for such types 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 the 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 of the references that have been used while developing this project.

2. Literature Survey

2.1. Survey of Existing System:

Many Anti-Virus systems have been found on Google and other sites . For developing our system the most important thing was to survey the existing systems present in the market and noting down their main features and the features which are lacking in them. While surveying we came across many such softwares. Features which were useful or harmful for maintaining the integrity of a system were obtained on viewing the details of these systems. Some Examples of our surveyed systems from jumpespjump.com **[1]** are Listed Below:-

I. **Kaspersky Total Security**: Kaspersky Total Security is a good choice for users as it provides a real time defence against the latest malware and threats and also has parental control. It includes many interesting features such as password manager, firewall, etc. However it uses the MD5 algorithm to scan the vulnerability in a system and also this tool does not even provide us with the hash values of the file and it also does not provide us with the option of comparing and finding the similarity between two system files. **[2]**

II. **Bromium**: Bromium secure files uses hardware enforced micro virtualization to isolate malicious threats hidden within inbound files and documents, including email attachments, web downloads, and USB files. The process is transparent to the user, with the files completely contained and isolated from other files and processes. However it also uses the MD5 algorithm to check the vulnerability of files in a system. Due to the limitations and inefficiency of the MD5 algorithm, the idea of detecting malware is harmed. **[3]**

"MD5" was the widely used hash function in malware analysis. It used to produce a 128-bit hash value from the data. In early 2000's it was realized that MD5 hash function is easily vulnerable to produce incorrect results. Although MD5 can be used to detect the integrity of a file, it is unable to detect any intentional corruption done to a file. Moreover MD5 was able to detect the known malware whereas malware with a slight change was considered as a new file in MD5 malware and as its data was not present in the existing system it was considered as a safe file. For Example, "Hello **w**orld" — MD5 hash: 3E25960A79DBC69B674CD4EC67A72C62

"Hello **W**orld" — MD5 hash: B10A8DB164E0754105B7A99BE72E3FE5

Just capitalizing a single letter in the second version, results in a different hash value!

Another major disadvantage of using the MD5 algorithm was that until a sample of malware is seen in the world, it can't be properly defended against by traditional antivirus; this means a malware can be missed for a very long time, like in the case of the initial deployment of **Stuxnet**. Stuxnet is a malicious software which was initially uncovered in 2010 but has been in development since 2005.

While going through various systems on the jumpespjump.com **[1]** 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.

2.2. Limitation of Existing System:

The above systems are using MD5 hashes even though practical collisions have been seen using this algorithm. Also md5 hash is not able to detect if there will be a slight change in a malware file as it produces a completely different hash value of similar files having differences of even a single bit as this was one of the most important features of the MD5 hash algorithm. Now it is being exploited by the Malware authors, as they are making a slight change in the Malware File which creates a completely different MD5 hash value due to which it remains undetected from the Anti-Malware Tools.

The existing malware analysis tool does not even provide us with the hash values of the file and it also does not provide us with the option of comparing and finding the similarity between two system files. Using the MD5 Hash function it was considered an impossible task to generate two data that can produce the same hash function. However, there have been some methods that can produce the desired hash functions of the MD5 algorithm intentionally with a particular set of texts. Due to the inefficiency of the MD5 algorithm, the idea of detecting malware using MD5 is harmed. With the help of this particular inefficiency of the MD5 algorithm, the Malware authors can create a malware file with the hash value considered as a safe hash value by many of the Anti-Malware organizations. Due to this inability of the MD5 hashing algorithm the anti-malware tools are finding it difficult to identify the difference between the safe and malware files creating a disorder in the system.

As MD5 is not an encryption technique, you can’t "decrypt" an MD5 hash to get the original data. The goal of any message digest function is to produce digests that appear to be random. To be considered cryptographically secure, the hash function should meet two requirements: first, it is impossible for an attacker to generate a message matching a specific hash value and second, it is impossible for an attacker to create two messages that produce the same hash value. MD5 hashes are no longer considered cryptographically secure. In 2011, the IETF published RFC 6151, "Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms," **[4]** which cited a number of recent attacks against MD5 hashes, especially one that generated hash collisions in a minute or less on a standard notebook and another that could generate a collision in as little as 10 seconds on a 2.66 GHz Pentium 4 system. As a result, the IETF suggested that new protocol designs should not use MD5 at all.

2.3. Mini project Contribution:

Our Malware Detection tool is a crucial tool as it helps to maintain the 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 a 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 include options such as Add Hash, Compare File, File Scan, Folder Scan, View Hash, and History. The add hash option allows adding the pre-known malware in the database. The compare file options give us the similarity percentage between two files using ssdeep hash values of those two files. The file scan and folder scan options compare the ssdeep hash values of the selected files with the hash values present in the database. The view hash options give the different hash values of a particular file to check its integrity. The history section displays the graph of malware and the secured files scanned by the system.

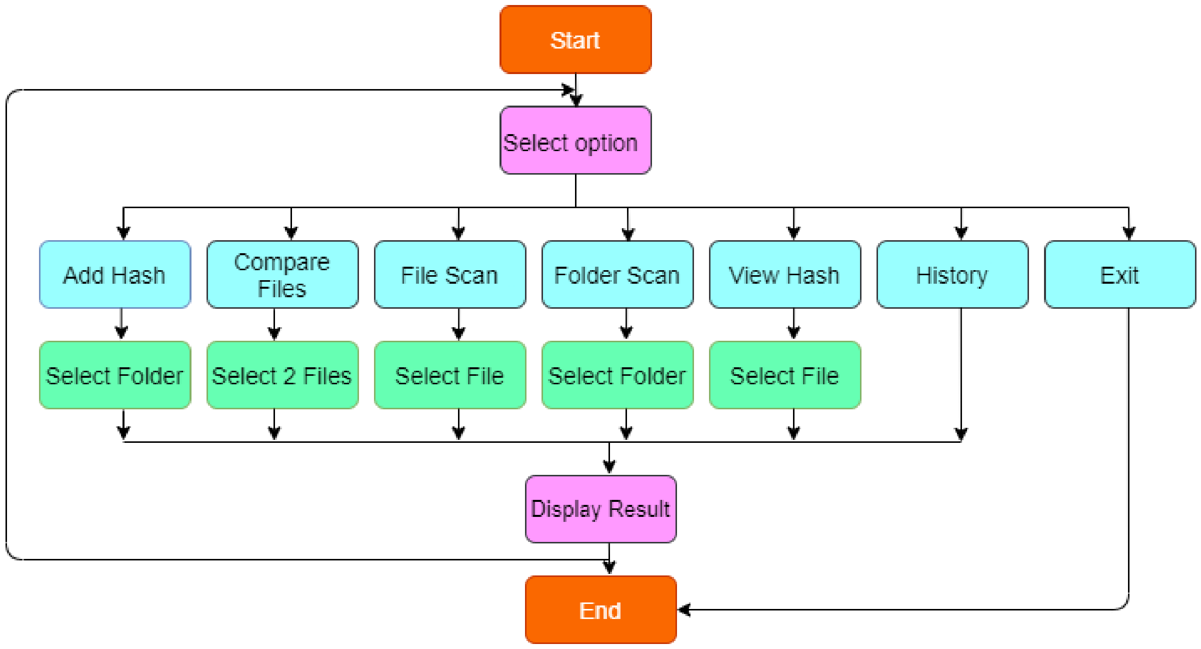


Figure 3.1 Architecture Framework

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.
```

Cryptographic hashes have been the traditional tool for both malware analysis and forensic investigations. The most used Cryptographic Hashes for malware analysis to detect identical known malware signatures are the MD5, SHA-1, and SHA-256. The main difference between these Cryptographic Hashes is their hash value length, despite they share the same concept "to perform known file filtering" by identifying identical matches. As mentioned earlier, changing a single bit in a file/document means that its hash value will also change, which makes it impossible- arguably- to find any associations or similarities between two files or to check if they are virtually similar. Cryptographic hashes are used in day-day life like in digital signatures, message authentication codes, manipulation detection, fingerprints, checksums (message integrity check), hash tables, password storage and much more. They are also used in sending messages over the network for security or storing messages in databases.

MD5 Algorithm:

MD5 hash function accepts sequence of bytes and returns 128-bit hash value, usually used to check data integrity but has security issues.

Step 1: Start
Step 2: The original data is padded in a way such that its total length will be 64 bits less than the multiple of 512. While adding bits for padding the first bit is 1 and rest bits are 0.
Step 3: After padding 64 bis are inserted in the end to mark the end of the string. At this point the size of the total bits will be a multiple of 512.
Step 4: A four word buffer of 32-bit registers are used to compute all the bits.
Step 5: Message is processed with logical operations on the buffer, resulting in a MD5 value in the 4 buffer where the first buffer contains the lower bit and the last contains the higher bit.
Step 6: MD5 hash value is created by appending values in MD buffer

SHA Algorithm:

SHA-2 (Secure Hash Algorithm 2), of which SHA-256 is a part, is one of the most popular hashing algorithms used in malware analysis. SHA-2 has several variants that all use the same algorithm but use different constants. The 256 and 512 in SHA-256 and SHA-512 refer to their respective digest sizes in bits.

Step 1 - Start
Step 2 - Pre-Processing
Step 3 - Initialize Hash Values
Step 4 - Initialize Round Constants
Step 5 - Chunk Loop
Step 6 - Create Message Schedule
Step 7 - Compression
Step 8 - Modify Final Values
Step 9 - Concatenate

For details of algorithm refer qvault.io **[5]**

SSDEEP:

SSDEEP is the context-triggered piecewise hashing (CTPH) technique to identify the files with previous files that are to find the duplicate content. This SSDEEP can be used effectively to compare the content similarity searches with existing well-known malware databases. Most Antivirus providers use this fuzzy technique to find new malware.

Step 1: Start

Step 2: Rolling Hashing is used to split the document into a 6 bit value segments (blocks of variable length, which are depended of the Rolling Hash Algorithm)

Step 3: Any other hash function, such as MD5 or SHA-1 is used to produce a digest for each segment.

Step 4: All Hash segments are linked together to form the unique hash value.

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

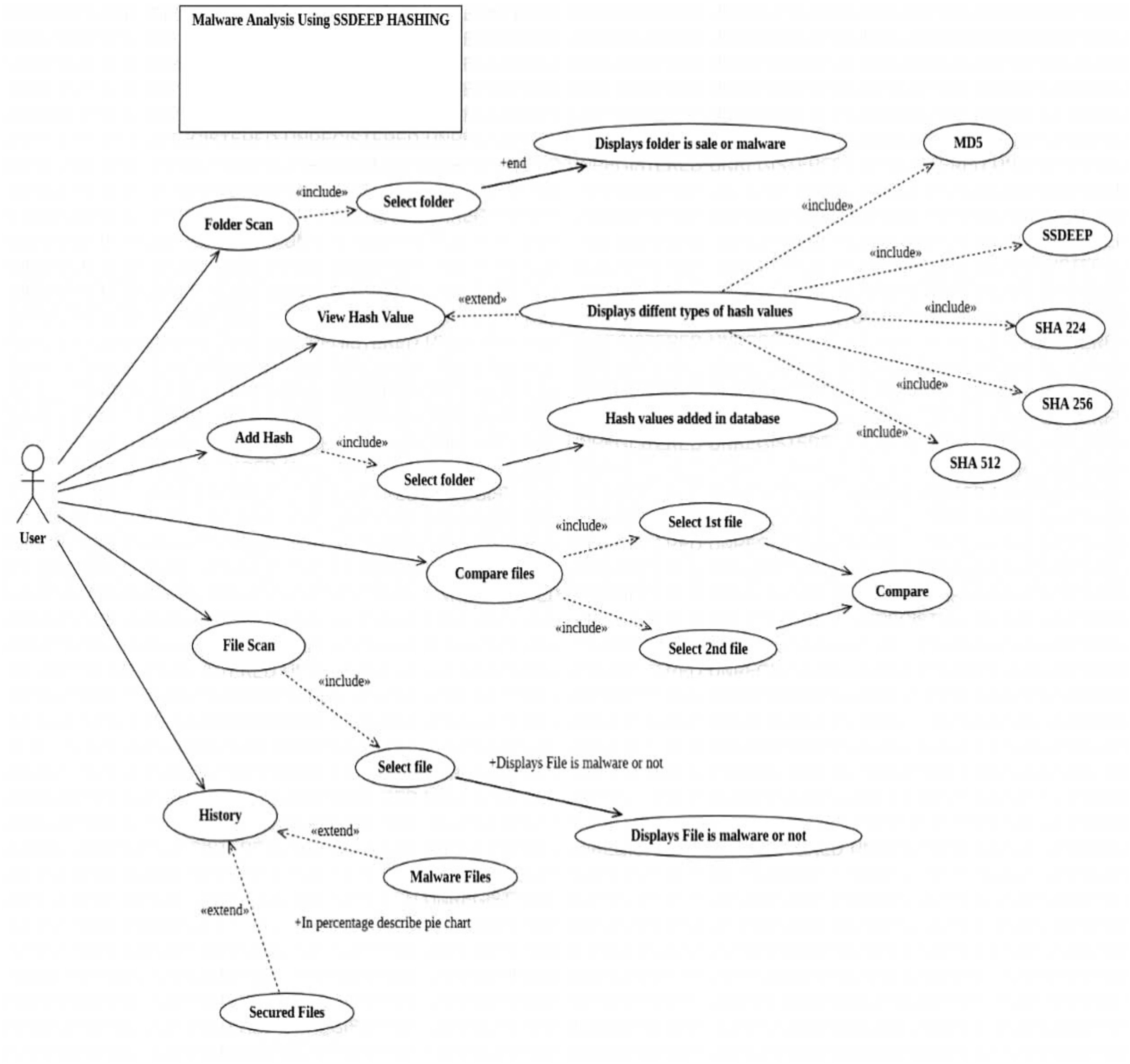


Figure 3.2 Use Case Diagram

- **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.

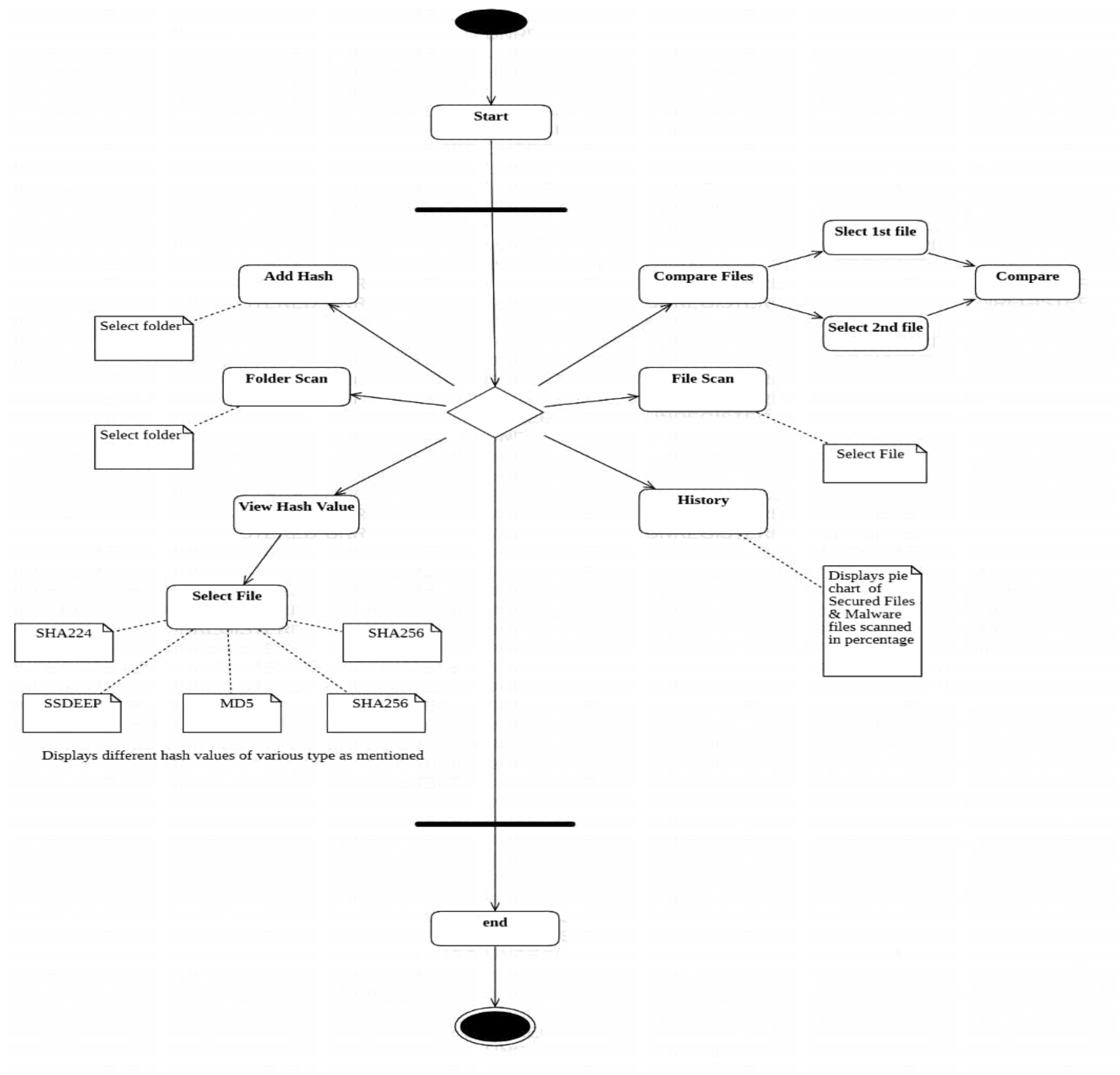


Figure 3.3 Activity Diagram

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 the Tkinter library. We chose this library as it gave us an option to create multiple frames which was a key need in our project. **[6]**
- **ssdeep**: ssdeep is a python library for computing context-triggered piecewise hashes (CTCH). Also called fuzzy hashes, CTCH can match input that has homologies. Such inputs have a sequence of identical bytes in the same order, although bytes in between these sequences may be different in both content and length. **[7]**
- **matplotlib**: Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in python. We used these libraries to display the history of scanned files of a particular user in a graphical format **[8]**

3.4. Experiment and Result:

Here we have developed a malware analysis tool. We are using the ssdeep hash function to find the unique hash value of a file and compare it with the hash values in the existing database. For malware files that have not been detected earlier, we have set a special constraint. If the similarity percentage between a file's hash value and the hash values present in the database is greater than 75%, then we are considering that particular unknown file as a malware file.

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.

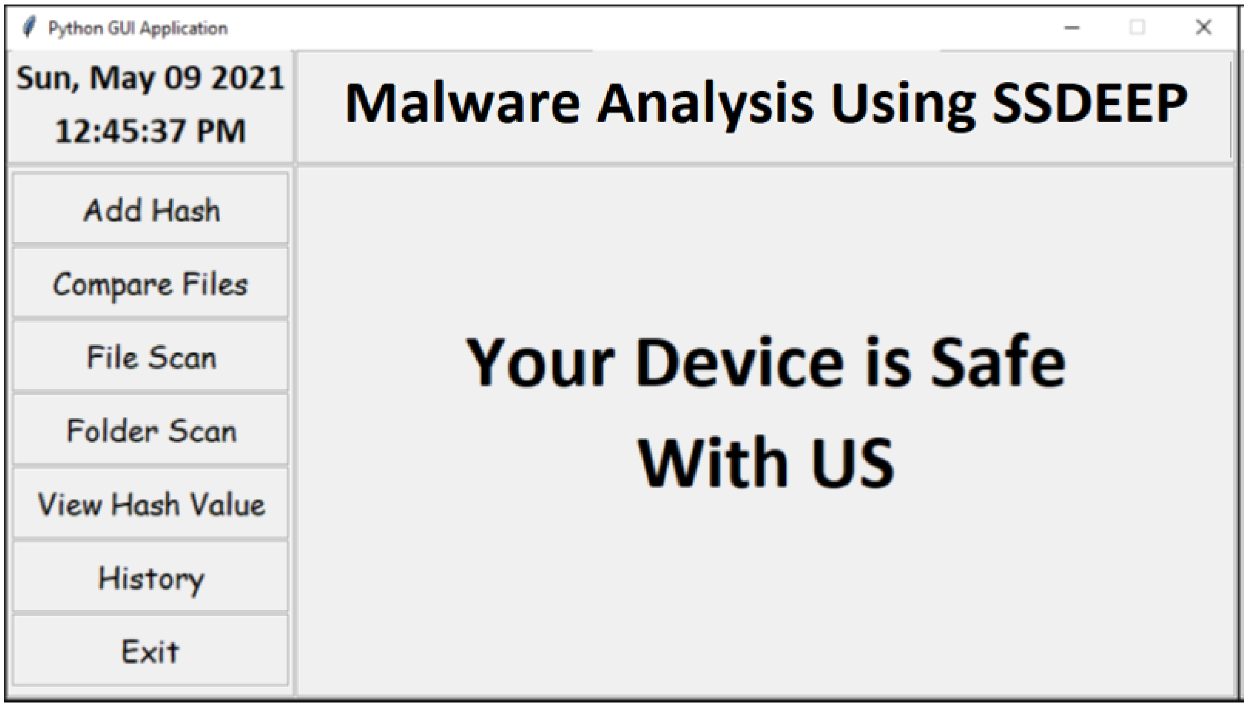


Figure 3.4 Main Page

- **Add Hash Page:** This page gives access to update the database of malware files. This page asks the user to select a folder which contains pre known malware files. The ssdeep hash values of the newly detected malware files are inserted into the database .

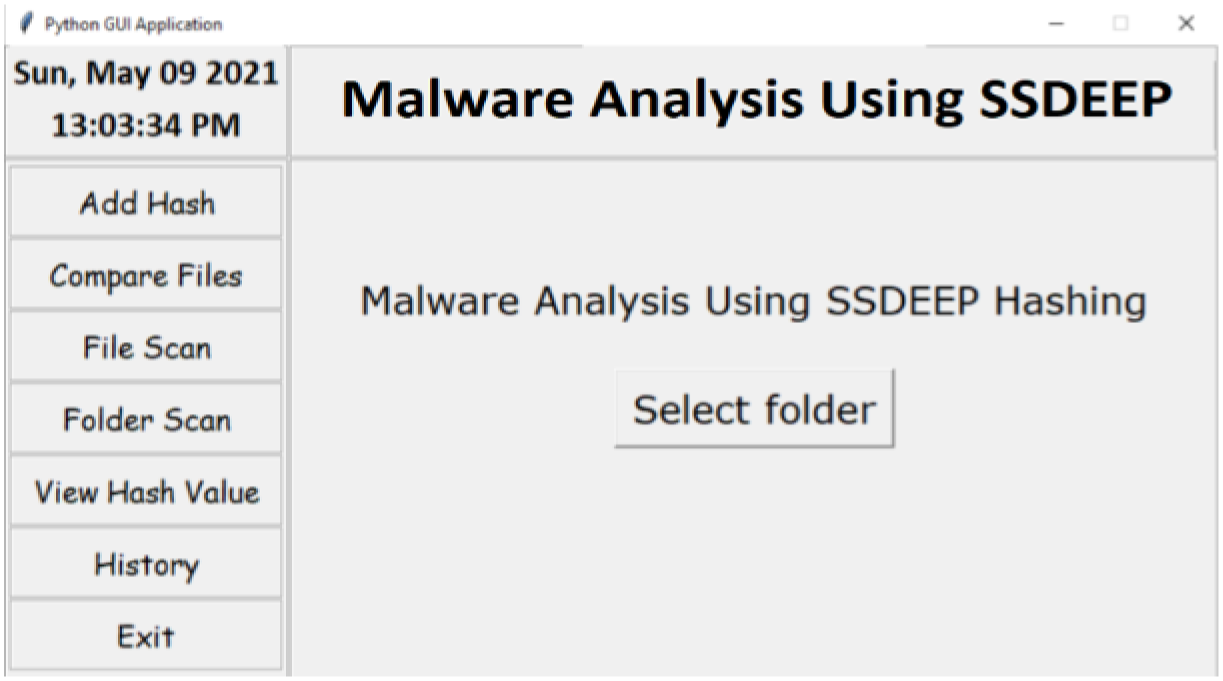
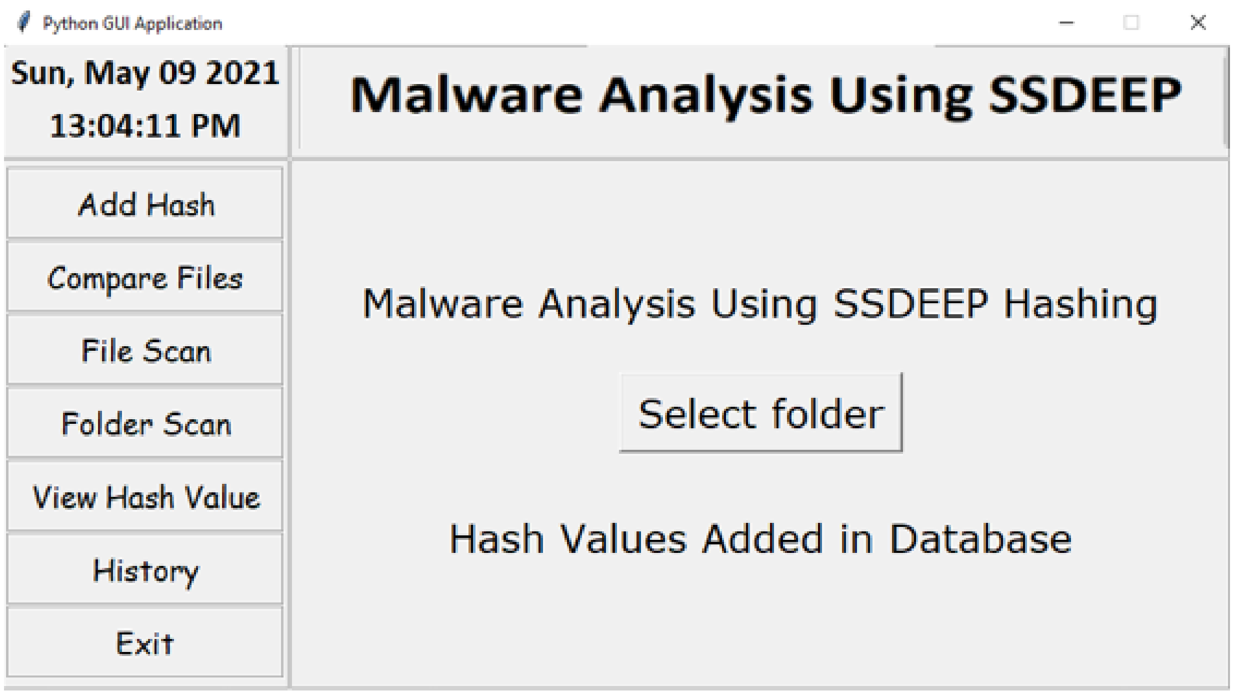


Figure 3.5 Add Hash Page-i



Hash Values Added in Database

Figure 3.6 Add Hash Page-ii

- **Compare Files**: In this section we can select two files from the system after which, on pressing the compare button the matching percentage between those two selected files will be displayed on the screen. The compare button returns the similarity percentage between the ssdeep hash value of the files.

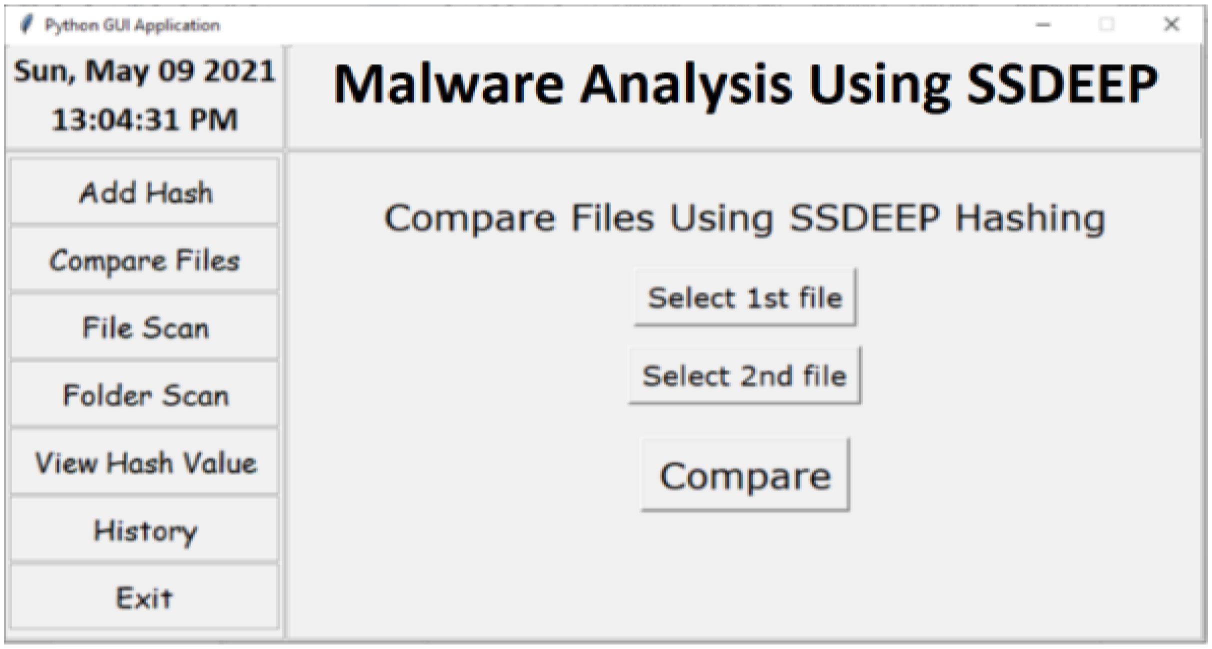
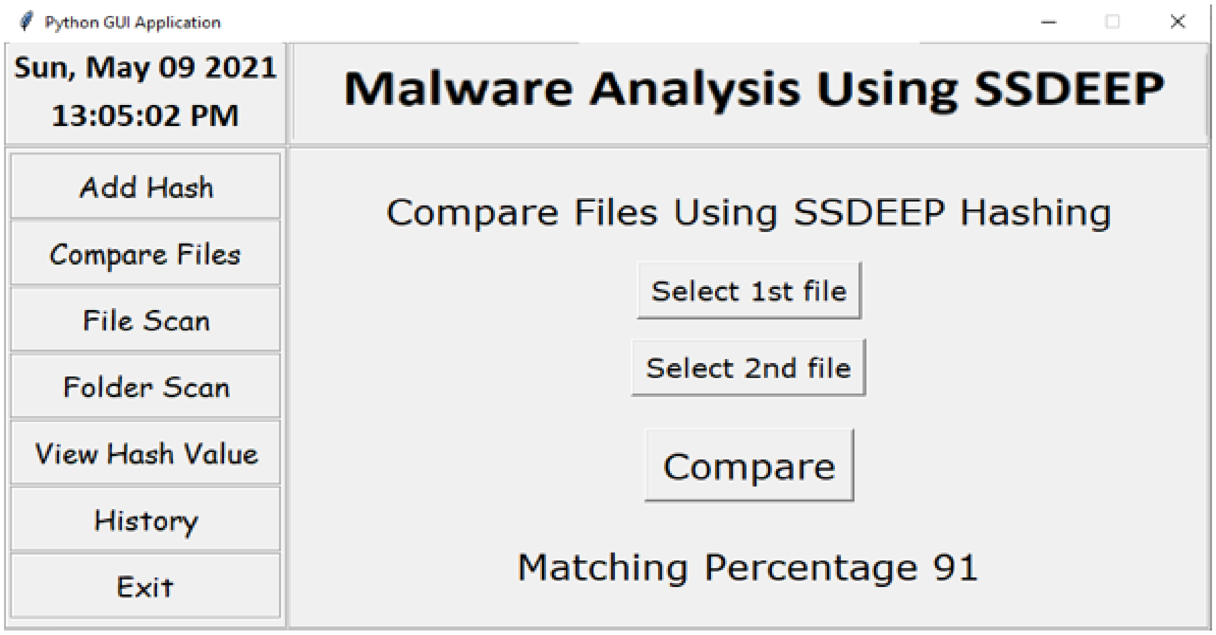


Figure 3.7 Compare Files Page-i



Matching Percentage 91

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. The file is declared as malware file if the similarity of the ssdeep value of the selected file is greater than 75 percent with any of the ssdeep values present in the database.

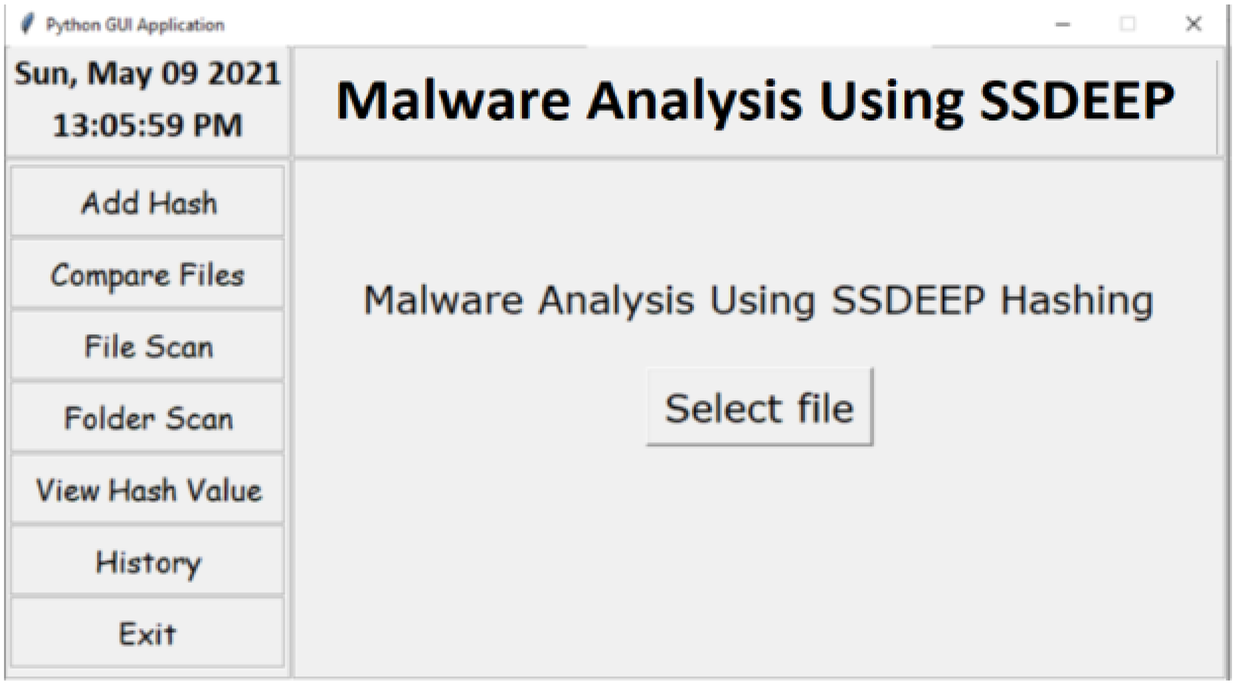
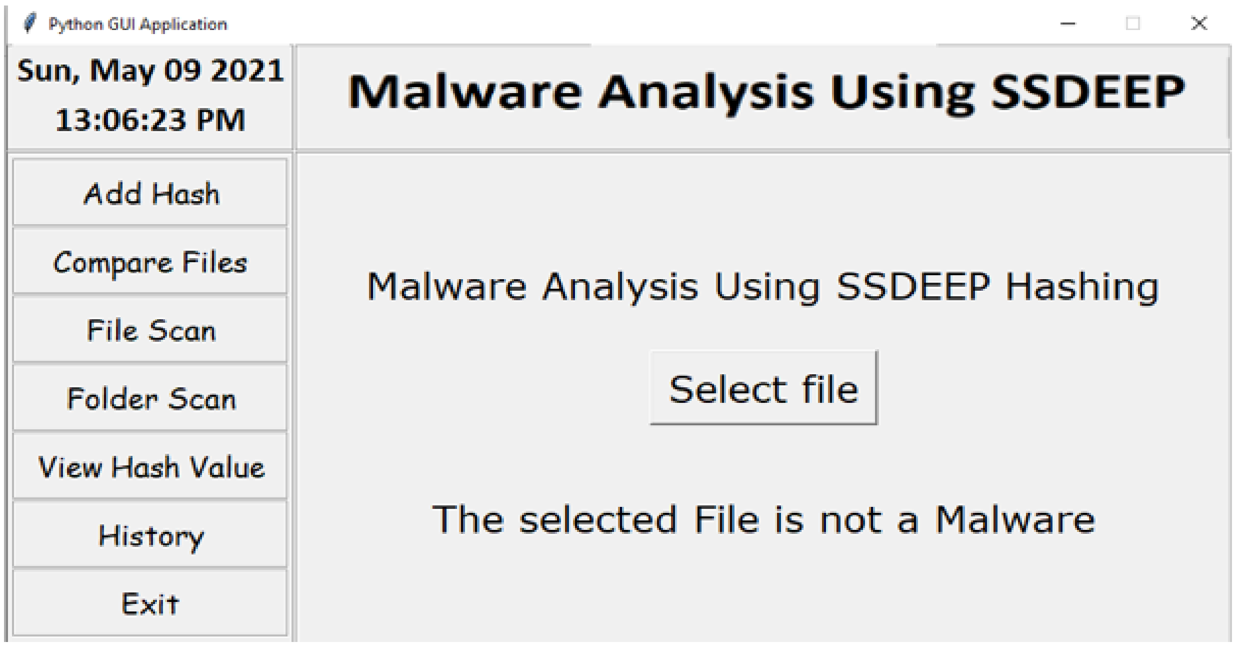


Figure 3.9 File Scan Page-i



The selected File is not a Malware

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.

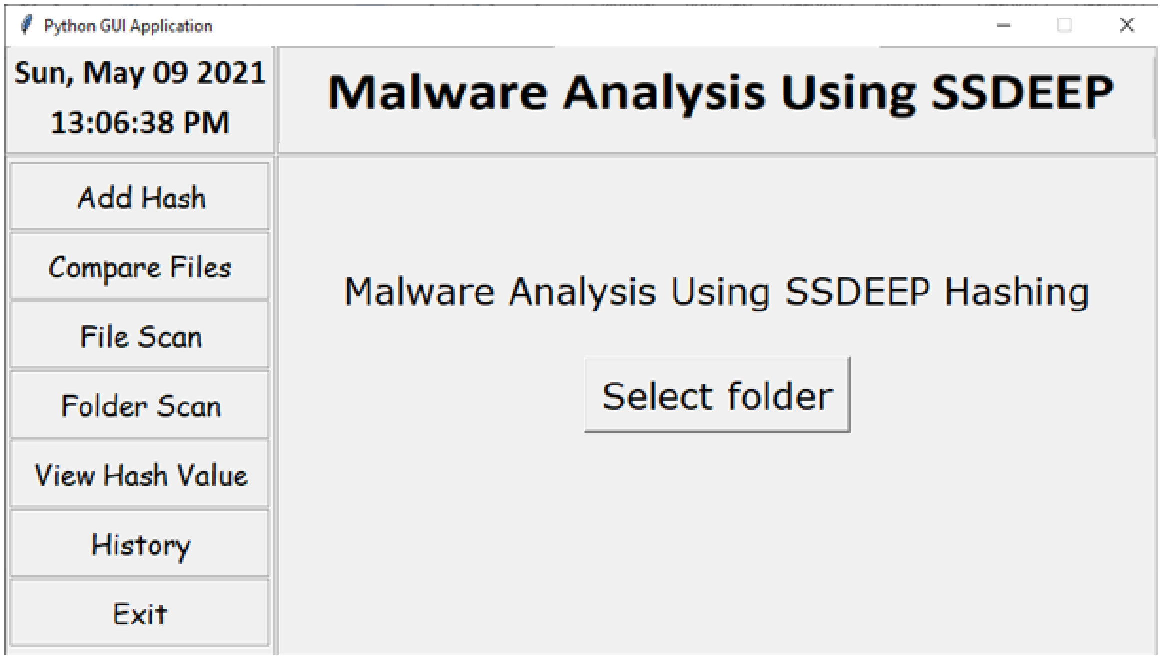
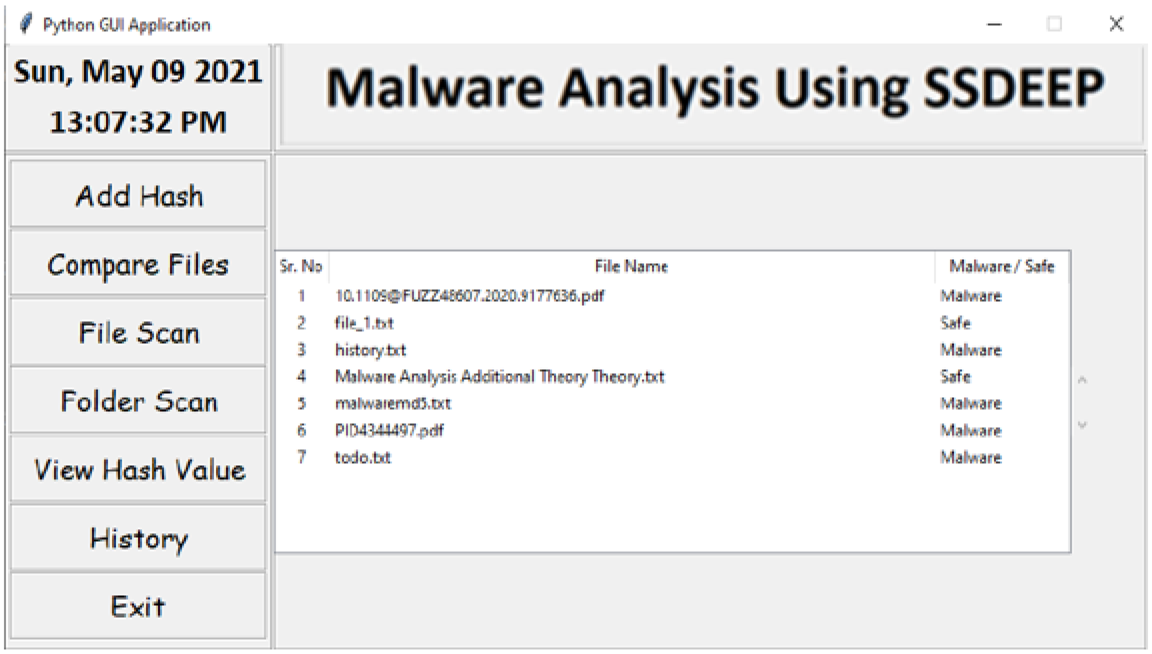


Figure 3.11 Folder Scan Page-i



Sr. No	File Name	Malware / Safe
1	10.1109@FUZZ48607.2020.9177636.pdf	Malware
2	file_1.txt	Safe
3	history.txt	Malware
4	Malware Analysis Additional Theory Theory.txt	Safe
5	malwaremd5.txt	Malware
6	PID4344497.pdf	Malware
7	todo.txt	Malware

Figure 3.12 Folder Scan Page-ii

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

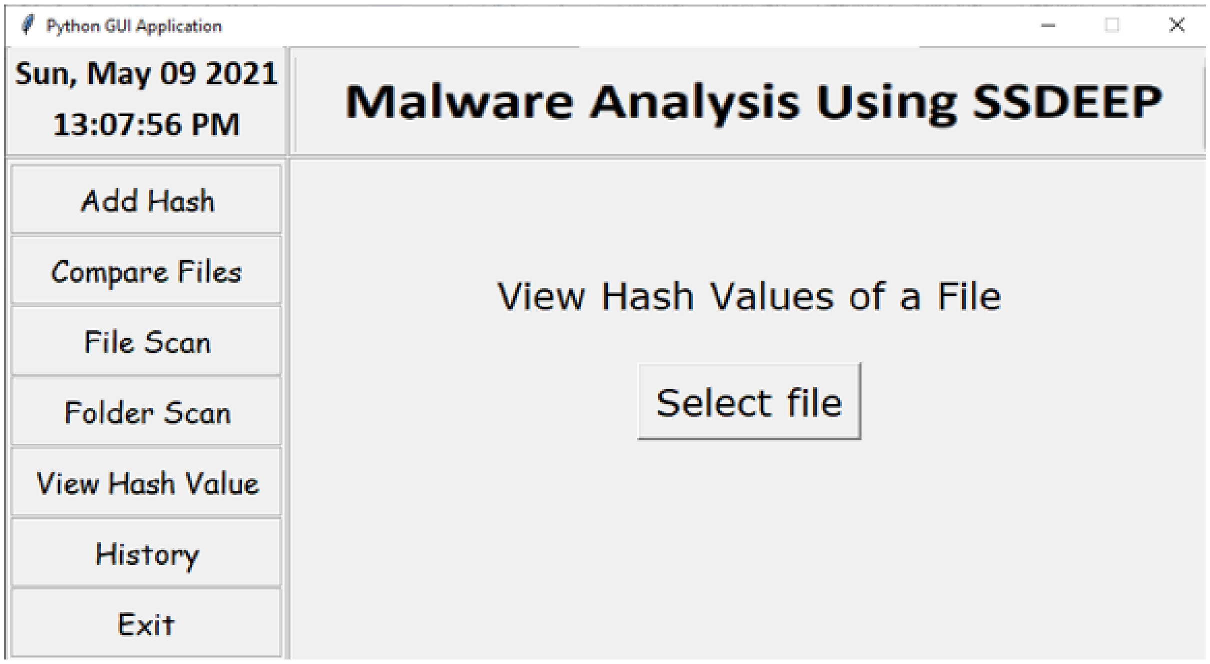
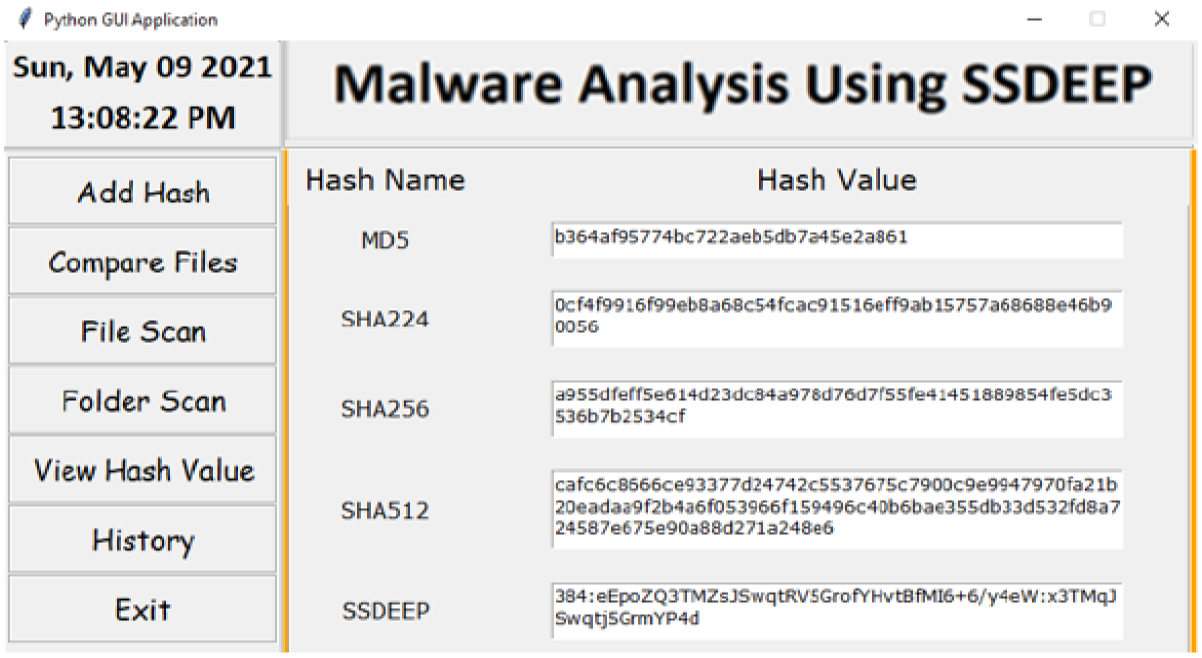


Figure 3.13 View Hash Value Page-i



Hash Name	Hash Value
MD5	b364af95774bc722aeb5db7a45e2a861
SHA224	0cf4f9916f99eb8a68c54fcac91516eff9ab15757a68688e46b9 0056
SHA256	a955dfeff5e614d23dc84a978d76d7f55fe41451889854fe5dc3 536b7b2534cf
SHA512	cafc6c8666ce93377d24742c5537675c7900c9e9947970fa21b 20eadaa9f2b4a6f053966f159496c40b6bae355db33d532fd8a7 24587e675e90a88d271a248e6
SSDEEP	384:eEpoZQ3TMZsJSwqtRV5GrofYHvtBfMI6+6/y4eW:x3TMqJ Swqtj5GrmYP4d

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.

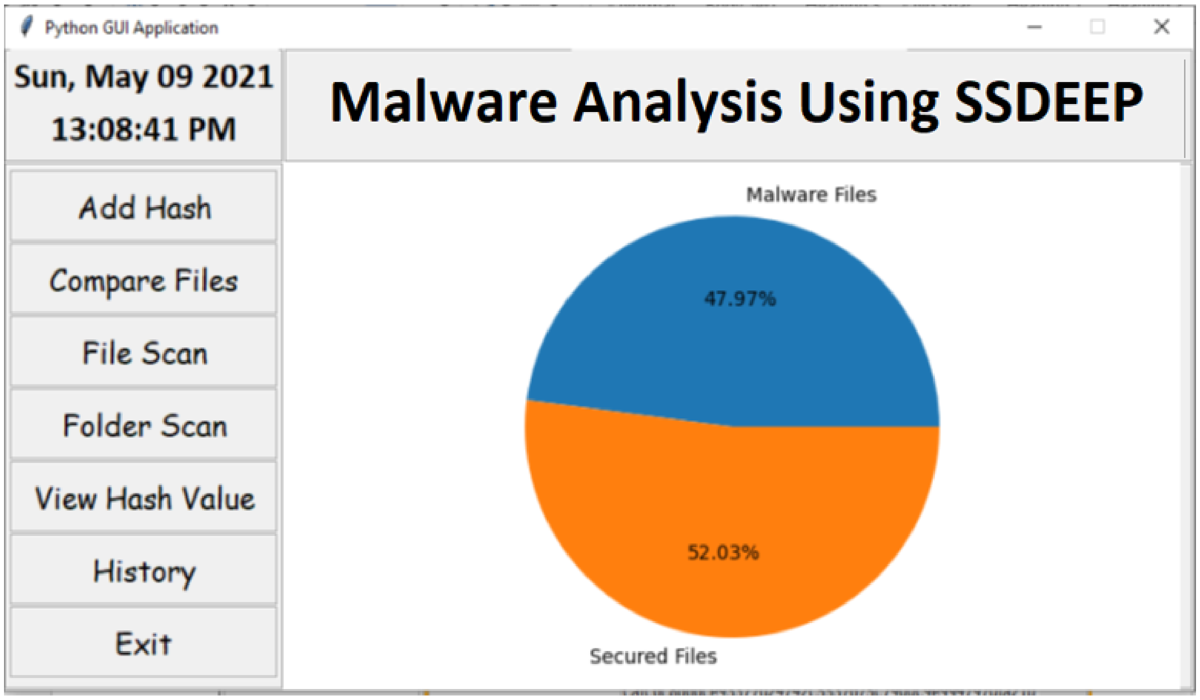


Figure 3.15 View History Page

4. Conclusion and Future Work

We have seen a big spike in the increasing number of attacks done with different malwares and ransomware 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

[3] https://www.bromium.com/tag/anti-virus/

[4] https://searchsecurity.techtarget.com/definition/MD5

[5] https://qvault.io/cryptography/how-sha-2-works-step-by-step-sha-256/

[6] Tkinter Library: **https://docs.python.org/3/library/tkinter.html**

[7] Ssdeep Library: **https://pypi.org/project/ssdeep/**

[8] Matplotlib Library: **https://matplotlib.org/**