

Minor Project Report
on

Armored Antivirus

Submitted in partial fulfillment of the requirements for the award of degree of

Bachelor of Technology
in
Information Technology

by
Harsh Yadav and Keshav Kumar
(19001011025 & 19001011031)

Under supervision of
Dr. Komal Bhatia



Department of Computer Engineering

J. C. BOSE UNIVERSITY OF SCIENCE & TECHNOLOGY, YMCA
FARIDABAD-121006

AUGUST 2021

CANDIDATE'S DECLARATION

I hereby certify that the work which is being carried out in this Minor Project titled **“Armored Antivirus”** in fulfillment of the requirement for the degree of Bachelor of Technology in Information Technology and submitted to **“J. C. Bose University of Science and Technology, YMCA, Faridabad”**, is an authentic record of my own work carried out under the supervision of Dr. Komal Bhatia Sir.

The work contained in this thesis has not been submitted to any other University or Institute for the award of any other degree or diploma by me.

Harsh Yadav and Keshav Kumar
19001011025 & 19001011031

CERTIFICATE

This is to certify that the work carried out in this project titled “**Armored Antivirus**” was submitted by Harsh Yadav and Keshav Kumar to “**J. C. Bose University of Science and Technology, YMCA, Faridabad**” for the award of the degree of Bachelor of Technology in Information Technology is a record of bonafide work carried out by him under my supervision. In my opinion, the submitted report has reached the standards of fulfilling the requirements of the regulations to the degree.

Dr. Komal Bhatia Sir(Supervisor)

Chairperson,

Department of Computer Engg,

J. C. Bose University of Science and Technology, YMCA, Faridabad

Abstract

Viruses and other malicious programs are an ever increasing threat to current computer systems. They can cause damage and consume countless hours and system administration time to combat. The “Armored Antivirus” will scan the system files and detect the virus and alert the user whenever necessary.

The basic ideas, concepts, components and approaches involved in developing an antivirus program from a developer's/software engineer's point of view is discussed here. It will focus on the main elements of an anti-virus engine and will exclude aspects like graphical user interfaces, real-time monitors, file system drivers and plug-ins for certain application software like Microsoft Exchange or Microsoft Office.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION.....	6
<input type="checkbox"/> 1.1 Introduction	
<input type="checkbox"/> 1.2 Basics Terminology	
<input type="checkbox"/> 1.3 Objectives and scope	
CHAPTER 2: DESCRIPTION & METHODOLOGY.....	8
CHAPTER 3: HARDWARE AND SOFTWARE REQUIREMENTS...	12
CHAPTER 4: RESULTS (SCREENSHOTS, GRAPHS Etc.).....	13
CHAPTER 5: CONCLUSION.....	14
<input type="checkbox"/> 5.1 Conclusion	
<input type="checkbox"/> 5.2 Future Scope	
REFERENCES	

CHAPTER 1

1.1 Introduction:-

Antivirus or anti-virus software is used to prevent ,detect and remove malware including but not limited to computer viruses, computer worm, trojan horses, spyware and adware. The software used for the prevention and removal of such threats.

Our software is coded in Python capable of scanning selected files and deleting files that it detects as infected.



1.2 Basics Terminology:-

A computer virus is a potentially dangerous computer program designed with the intent of corrupting data.

Computer viruses are mysteriously hidden beneath seemingly harmless programs, which explain the reason for their effective spread across the internet.

These malicious computer programs are designed to replicate themselves or insert copies of themselves into other programs when executed within the infected program.

Since viruses should be executed to have any effect, files that the computer treats as pure data are safe.

This includes graphics and sound files such as .gif, .jpeg, .mp3, .wav, etc., as well as plain text in .txt files.

Just viewing pictures and playing music won't harm your computer because the computer virus should be in a form such as a .exe program or a word .doc file indicating the means of executing.

When you execute a program that has been infected by a virus, the virus code will start to run on your computer system and thus, will try to infect other computer programs.

1.3 Objectives and scope:-

Simple Antivirus coded in python capable of scanning selected files and deleting files that it detects as infected.

This antivirus uses a large list of MD5, SHA1 and SHA256 malware hashes to determine infections. We would like to implement machine learning detection with the long term goal of becoming a fully functioning antivirus and also for internet protection (Email checker, website checker).

1-Virus, Malware and Junk file detection

2-System analysis

3-Malware and Junk file removal

4-Ram Booster

5-Real Time Protection

CHAPTER 2

2. Description and Methodology:-

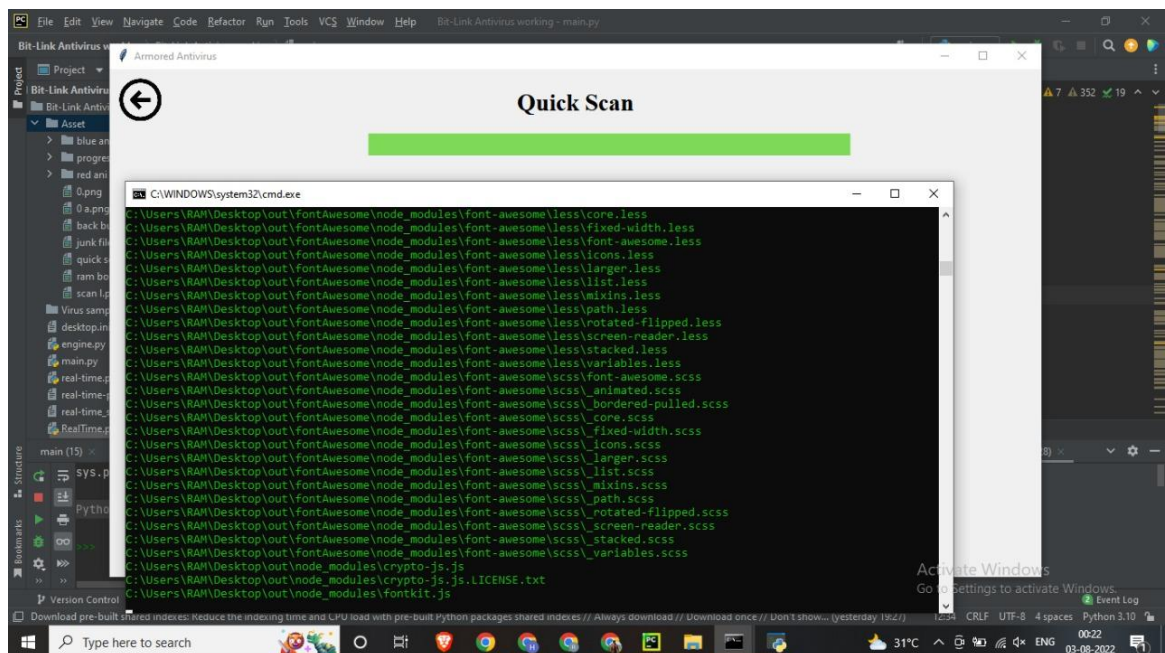
We have added five panels:-

1. Quick Scan
2. Custom Scan
3. Junk File Remover
4. RAM Booster
5. Real Time Scanner

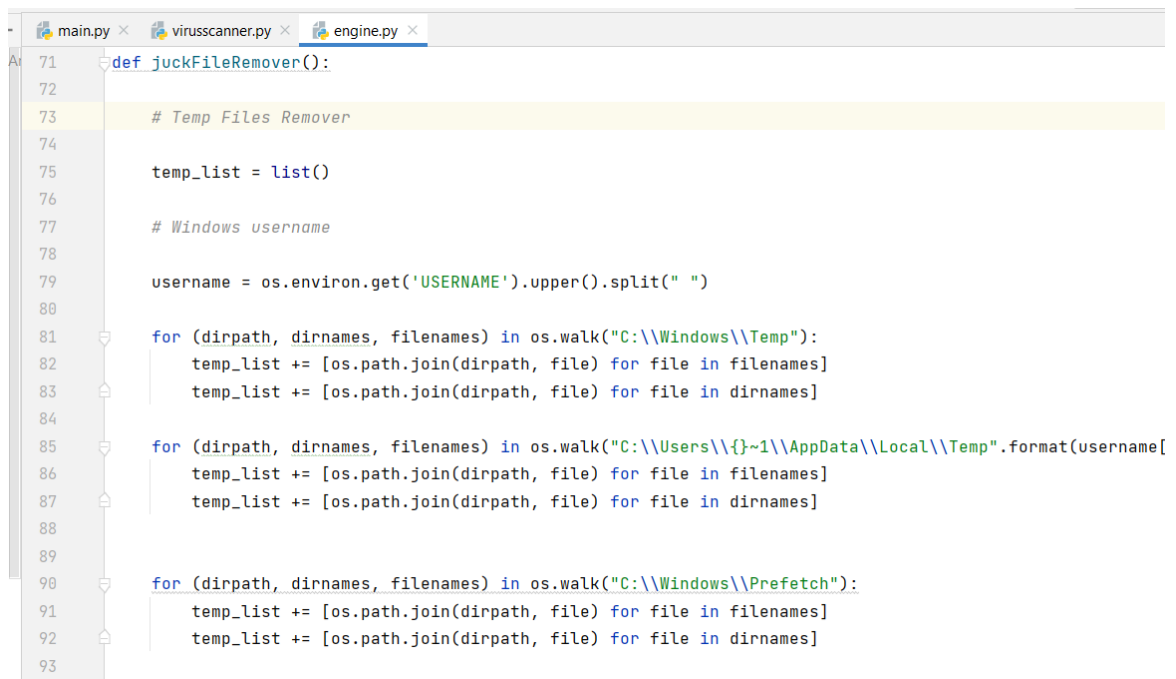


1. In Quick Scan we check the common area for computer viruses. scanned areas can include computer memory in short here we scan the whole C drive.

```
17 |
18 def QUICKSCAN(path):
19
20     # Get the list of all files in directory tree at given path
21     dir_list = list()
22     for (dirpath, dirnames, filenames) in os.walk(path):
23         dir_list += [os.path.join(dirpath, file) for file in filenames]
24         print([os.path.join(dirpath, file) for file in filenames])
25
26     for i in dir_list:
27         print(i)
28         if malware_checker(i) != 0:
29             try:
30                 with open("switch_virusscanner.bb", "a") as bb:
31
32                     bb.write(i+"\n")
33                     bb.close()
34             except: pass
35
36         print(i)
37         virusName.append(malware_checker(i)+" :: File :: "+i)
38         virusPath.append(i)
```



2. In custom Scan we target a particular parent folder to scan where we have the probability of the infection.
3. In Junk File Remover we remove cache-file, temp file, Prefetch files. These files contain information about the environment and applications you run.



```
71 def juckFileRemover():
72
73     # Temp Files Remover
74
75     temp_list = list()
76
77     # Windows username
78
79     username = os.environ.get('USERNAME').upper().split(" ")
80
81     for (dirpath, dirnames, filenames) in os.walk("C:\\Windows\\Temp"):
82         temp_list += [os.path.join(dirpath, file) for file in filenames]
83         temp_list += [os.path.join(dirpath, file) for file in dirnames]
84
85     for (dirpath, dirnames, filenames) in os.walk("C:\\Users\\{0}\\AppData\\Local\\Temp".format(username[0])):
86         temp_list += [os.path.join(dirpath, file) for file in filenames]
87         temp_list += [os.path.join(dirpath, file) for file in dirnames]
88
89
90     for (dirpath, dirnames, filenames) in os.walk("C:\\Windows\\Prefetch"):
91         temp_list += [os.path.join(dirpath, file) for file in filenames]
92         temp_list += [os.path.join(dirpath, file) for file in dirnames]
93
```

4. RAM Booster provides memory optimization by closing unnecessary tasks. So for our project we have made a tasklist for demo purposes . Tasklist contain "notepad.exe" , "AnyDesk.exe" , "TeamViewer_Service.exe" , "msedge.exe" , "IDMan.exe" , "chrome.exe"

```

main.py × virusscanner.py × engine.py × RealTime.py ×
def ramBooster():
    taskList = ["notepad.exe", "AnyDesk.exe", "TeamViewer_Service.exe", "msedge.exe", "IDMan.exe", ]

    username = os.environ.get('USERNAME').upper().split(" ")

    for (dirpath, dirnames, filenames) in os.walk("C:\\Windows\\Temp"):
        temp_list += [os.path.join(dirpath, file) for file in filenames]
        temp_list += [os.path.join(dirpath, file) for file in dirnames]

    for (dirpath, dirnames, filenames) in os.walk("C:\\Users\\{}~1\\AppData\\Local\\Temp".format(username)):
        temp_list += [os.path.join(dirpath, file) for file in filenames]
        temp_list += [os.path.join(dirpath, file) for file in dirnames]

    # Task Kill

    for i in taskList:
        os.system("taskkill /f /im {}".format(i))

```

5. Real Time Scanner monitors these actions created, deleted, updated, Renamed from something, rename to something. Whenever any of this action happens our software will monitor this action to console.

```

main.py × virusscanner.py × engine.py × RealTime.py ×
def RealTime():
    usernameUp = os.environ.get('USERNAME').upper().split(" ")
    username = os.environ.get('USERNAME')

    ACTIONS = {
        1: "Created",
        2: "Deleted",
        3: "Updated",
        4: "Renamed from something",
        5: "Renamed to something"
    }

    FILE_LIST_DIRECTORY = 0x0001

    path_to_watch = "C:\\\\"
    hDir = win32file.CreateFile(
        path_to_watch,
        FILE_LIST_DIRECTORY,
        win32con.FILE_SHARE_READ | win32con.FILE_SHARE_WRITE | win32con.FILE_SHARE_DELETE,
        None,
        win32con.OPEN_EXISTING,
        win32con.FILE_FLAG_BACKUP_SEMANTICS,

```

```
main.py × virusscanner.py × engine.py × RealTime.py ×

paths = []

for action, file in results:
    paths.append(os.path.join(path_to_watch, file))

    if paths[0][0:32] == "C:\\Users\\{}\\AppData\\".format(username):paths.clear()
    elif paths[0][0:18] == "C:\\Users\\{}~1\\".format(usernameUp[0]):paths.clear()
    elif paths[0][0:20] == "C:\\Windows\\Prefetch\\":paths.clear()
    elif paths[0][0:15] == "C:\\Windows\\Temp":paths.clear()
    elif paths[0][0:15] == "C:\\$Recycle.Bin":paths.clear()
    elif paths[0][0:14] == "C:\\ProgramData":paths.clear()
    elif paths[0][0:23] == "C:\\Windows\\ServiceState":paths.clear()
    elif paths[0][0:15] == "C:\\Windows\\Logs":paths.clear()
    elif paths[0][0:26] == "C:\\Windows\\ServiceProfiles":paths.clear()
    elif paths[0][0:19] == "C:\\Windows\\System32":paths.clear()
    elif paths[0][0:28] == "C:\\Program Files\\CUAssistant":paths.clear()
    elif paths[0][0:23] == "C:\\Windows\\bootstat.dat":paths.clear()

    # try:print (paths[0], ACTIONS.get (action, "Unknown"))
    # except:pass
    try:engine.virusScanner(paths[0])
    except:pass
    paths.clear()
```

CHAPTER 3

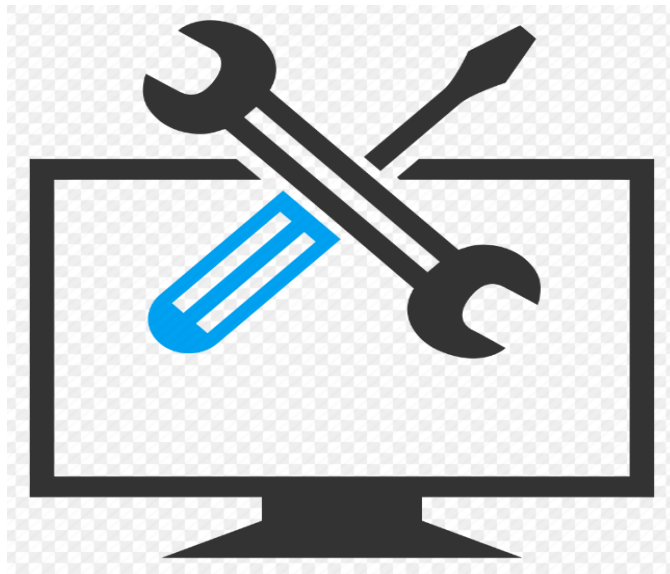
3. HARDWARE AND SOFTWARE REQUIREMENTS:-

☐ **Software:-**

Pycharm, VScode.

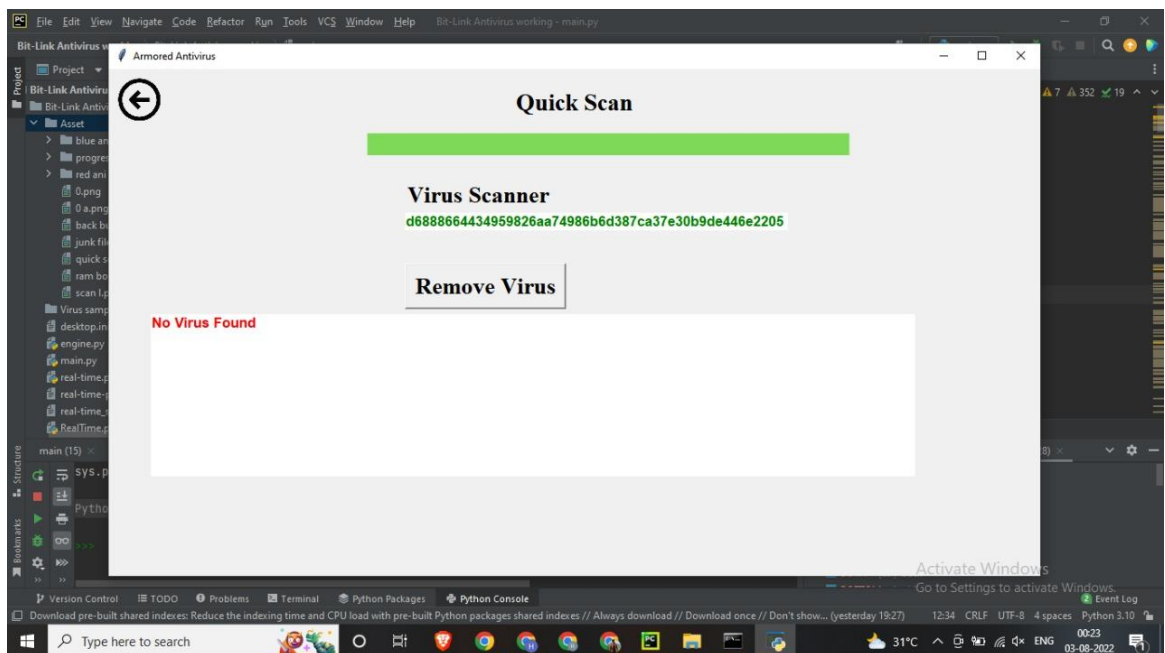
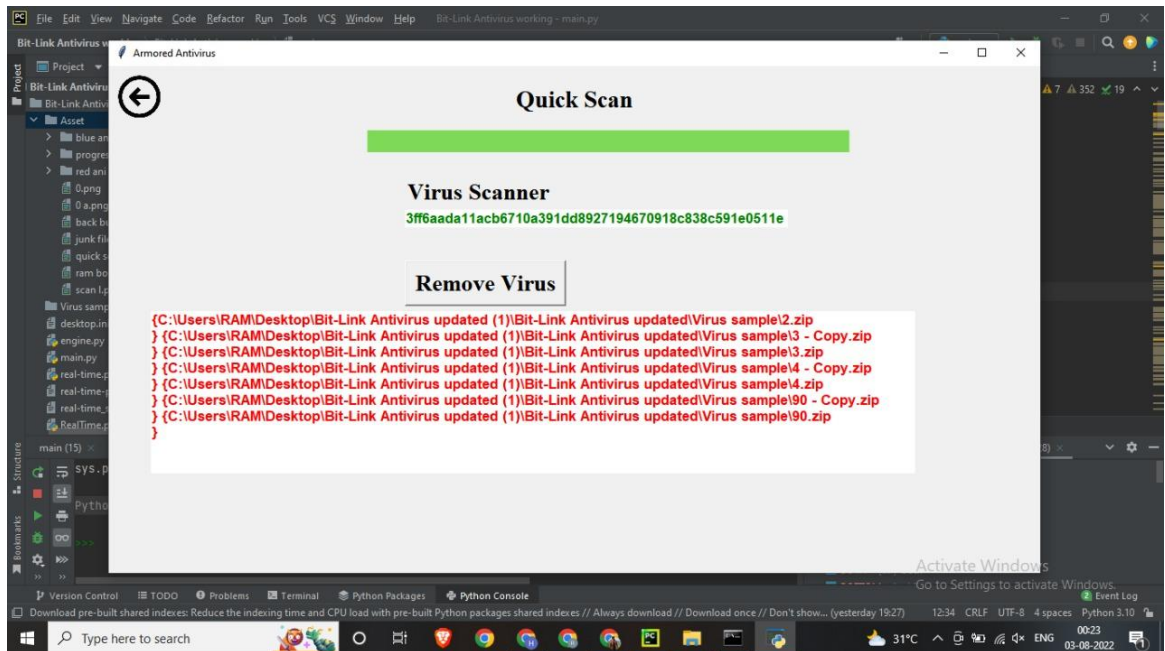
☐ **Hardware:-**

- Operating system: Windows XP, 2000,98.
- Processor: Pentium 3.0GHz or higher
- RAM: 256 MB or more.



CHAPTER 4

4. RESULTS (SCREENSHOTS, GRAPHS Etc.):-



CHAPTER 5

5.1 CONCLUSION:-

The “**Armored Antivirus**” is a generic antivirus approach that will detect the suspicious behaviors of the files that are scanned and it avoids the pitfalls of the signature scanning method and provides full security to the user.

This project has dropped a small stone in water, by designing an application that provides a generic antivirus approach that is used to scan the files efficiently. “**Armored Antivirus**” being developed by restricting the present technology available in our college meets the desired needs of the requirements completely.

5.2 FUTURE SCOPE:-

At present in our system only the files that were scanned and reported as affected can be deleted or can be moved to a vault to delete in future. So the only option provided for the user is to delete the affected file. Moreover the affected file can be repaired by deleting the virus code that was matched from the disassembled code and restoring the new file from the repaired code.

REFERENCES:-

- ☐ Alex korthny description paper for malware detection based on system events trace collected during exhibitions of the malware program.
- ☐ Anic Data set for all malware hashes and some tutorials for malware detections.