

ADJEI YAW OSEI

CODE ALPHA PROJECT

NETWORK SECURITY ASSESSMENT REPORT

TABLE OF CONTENT

SN	TITLE	PAGE NO
1.	Build a Basic Network Sniffer in Python	4
2.	Phishing Awareness Training	10
3.	Secure Coding Review Using Bandit on Kali Linux Secure Coding Review Using Bandit on Kali Linux	12
4.	Conclusion and Recommendations	15
5.	Final Recommendations	16

LIST OF FIGURES

FIG NO	NAME	PAGE NO
Fig 1.0	Python Installation	5
Fig 1.1	Necessary Libraries (scapy & pip3) installation	6
Fig 1.2	Source Code	8
Fig 1.3	Running Sniff	9
Fig 1.4	Output of Sniff	9
Fig 1.5	Phishing Email Sample	11
Fig 1.6	Bandit Installation	13
Fig 1.7	Running Bandit on code	14
Fig 1.8	Bandit Output	15

TASK 1

1. Build a Basic Network Sniffer in Python

Goal: Create a simple program that captures and displays network data flowing to and from your computer.

Step 1: Install Python

1. Check if Python is installed:

- Open your terminal in Kali Linux.
- Type the following command and press Enter:

```
python3 --version
```

2. Install Python:

- Python should already be installed on Kali Linux. If not, install it using:

```
sudo apt update
```

```
sudo apt install python3
```

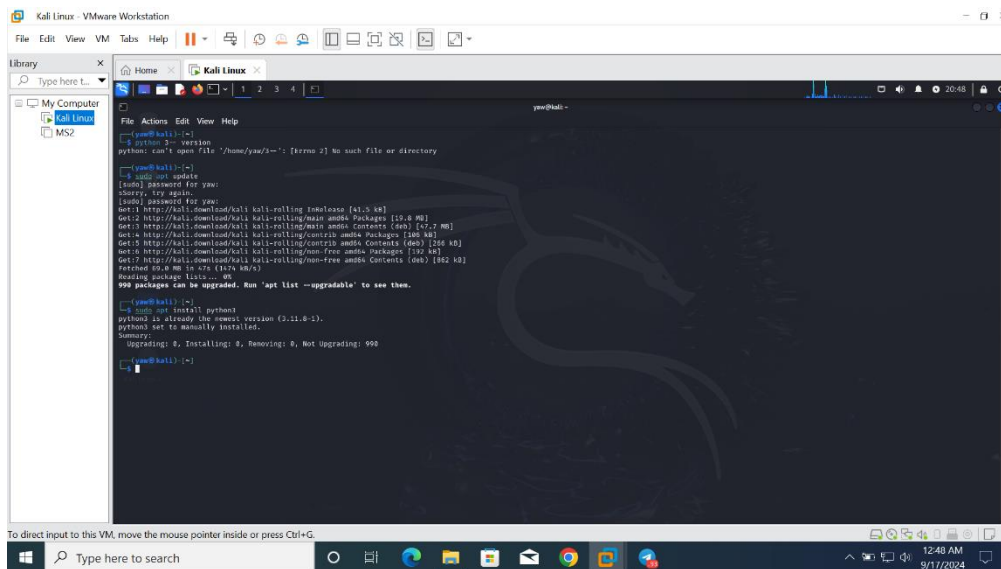


Fig 1.0

Step 2: Install Necessary Libraries (scapy)

1. Open your terminal.
2. Install scapy:
 - First, ensure you have pip for Python 3:
sudo apt install python3-pip
 - Install the scapy library using pip:
pip3 install scapy

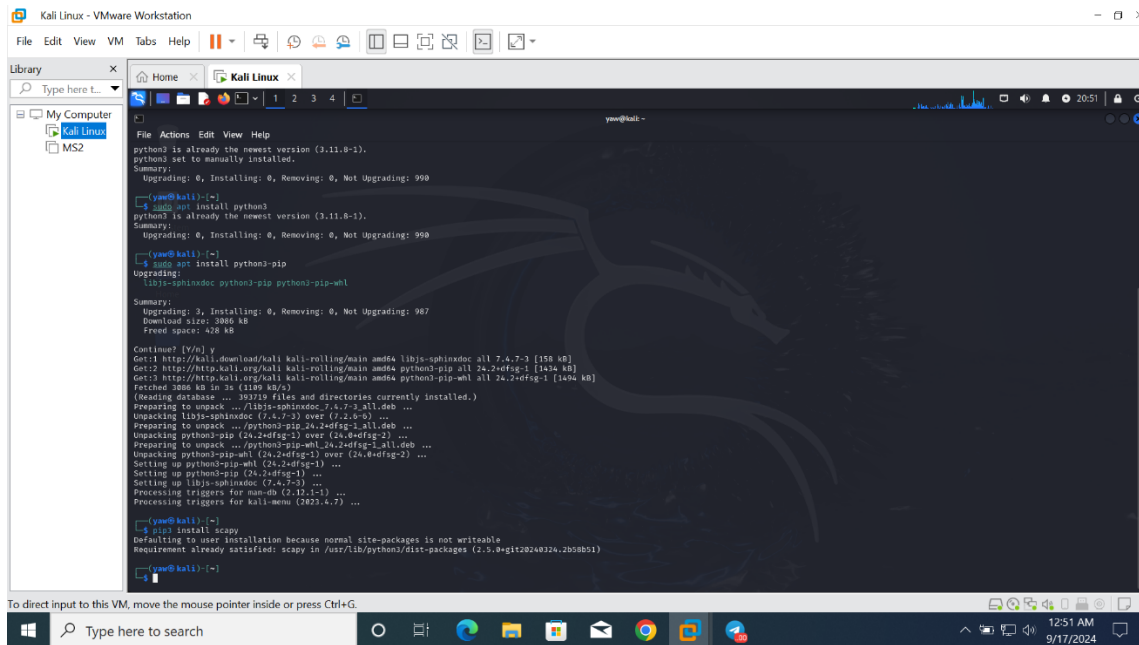


Fig 1.1

Step 3: Create a Basic Sniffer Program

1. Open a text editor:

- On Kali Linux, you can use a text editor like nano, vim, or a graphical editor like gedit.
- For nano, open the terminal and type:

```
nano network_sniffer.py
```

2. Write the program:

- In the editor, enter the following code:

```
from scapy.all import sniff, Ether, IP, TCP, UDP
```

```
# Function to handle and analyze each packet
```

```
def packet_handler(packet):
```

```
    # Print general packet summary
```

```
    print(packet.summary())
```

```

# Check if the packet has an Ethernet layer

if packet.haslayer(Ether):

    print(f"Ethernet Frame: {packet[Ether].summary()}")


# Check if the packet has an IP layer

if packet.haslayer(IP):

    ip_layer = packet[IP]

    print(f"IP Packet: {ip_layer.src} -> {ip_layer.dst}")

    print(f"Protocol: {ip_layer.proto}")


# Check if the packet has a TCP layer

if packet.haslayer(TCP):

    tcp_layer = packet[TCP]

    print(f"TCP Segment: Src Port: {tcp_layer.sport}, Dst Port: {tcp_layer.dport}")

    print(f"Flags: {tcp_layer.flags}")


# Check if the packet has a UDP layer

elif packet.haslayer(UDP):

    udp_layer = packet[UDP]

    print(f"UDP Datagram: Src Port: {udp_layer.sport}, Dst Port: {udp_layer.dport}")


# Print a separator line

print("-" * 50)


# Function to start sniffing packets

def start_sniffing(interface):

    print(f"Starting packet sniffing on {interface}...")

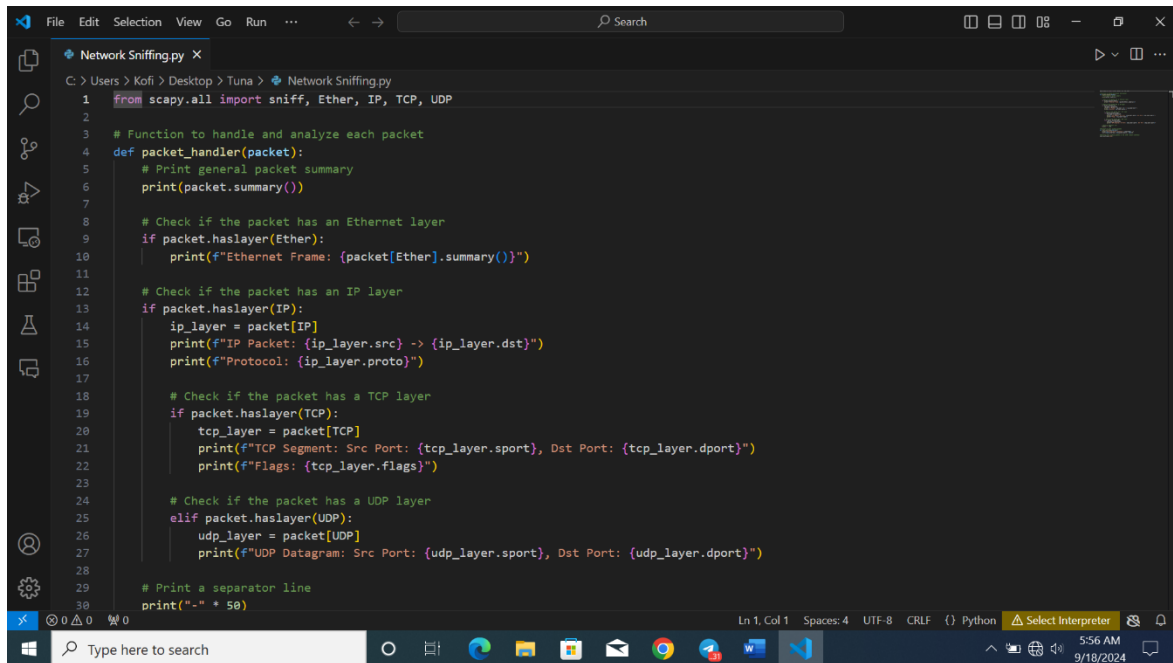
    sniff(iface=interface, prn=packet_handler, store=0)

```

Directly start sniffing packets on the 'eth0' network interface

start_sniffing("eth0")

- Save the file and exit the editor. In nano, you can save by pressing CTRL + O, then exit with CTRL + X.

A screenshot of a Visual Studio Code editor window. The title bar says "Network Sniffing.py X". The file explorer on the left shows the file path "C:\Users> Kofi > Desktop > Tuna > Network Sniffing.py". The editor displays a Python script with the following content:

```
1 from scapy.all import sniff, Ether, IP, TCP, UDP
2
3 # Function to handle and analyze each packet
4 def packet_handler(packet):
5     # Print general packet summary
6     print(packet.summary())
7
8     # Check if the packet has an Ethernet layer
9     if packet.haslayer(Ether):
10        print(f"Ethernet Frame: {packet[Ether].summary()}")
11
12    # Check if the packet has an IP layer
13    if packet.haslayer(IP):
14        ip_layer = packet[IP]
15        print(f"IP Packet: {ip_layer.src} -> {ip_layer.dst}")
16        print(f"Protocol: {ip_layer.proto}")
17
18    # Check if the packet has a TCP layer
19    if packet.haslayer(TCP):
20        tcp_layer = packet[TCP]
21        print(f"TCP Segment: Src Port: {tcp_layer.sport}, Dst Port: {tcp_layer.dport}")
22        print(f"Flags: {tcp_layer.flags}")
23
24    # Check if the packet has a UDP layer
25    elif packet.haslayer(UDP):
26        udp_layer = packet[UDP]
27        print(f"UDP Datagram: Src Port: {udp_layer.sport}, Dst Port: {udp_layer.dport}")
28
29    # Print a separator line
30    print("-" * 50)
```

The status bar at the bottom indicates "Ln 1, Col 1", "Spaces: 4", "UTF-8", "CRLF", and "Python". The Windows taskbar is visible at the bottom with the search bar and several application icons.

Fig 1.2

Step 4: Run the Sniffer

1. Open a terminal in the directory where your file is saved:

- If you used nano or another editor, you should already be in the terminal. Navigate to the folder where network_sniffer.py is located if needed:

cd /path/to/directory

2. Run the program:

- Execute the Python script using:

sudo python3 network_sniffer.py

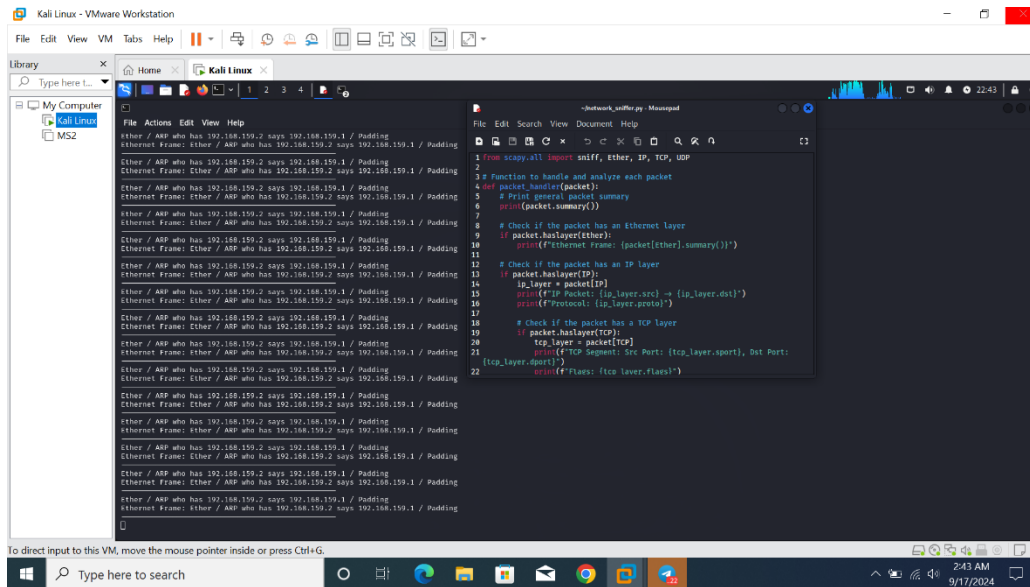


Fig 1.3

3. Stop the program:

- To stop the program, press CTRL + C in the terminal.

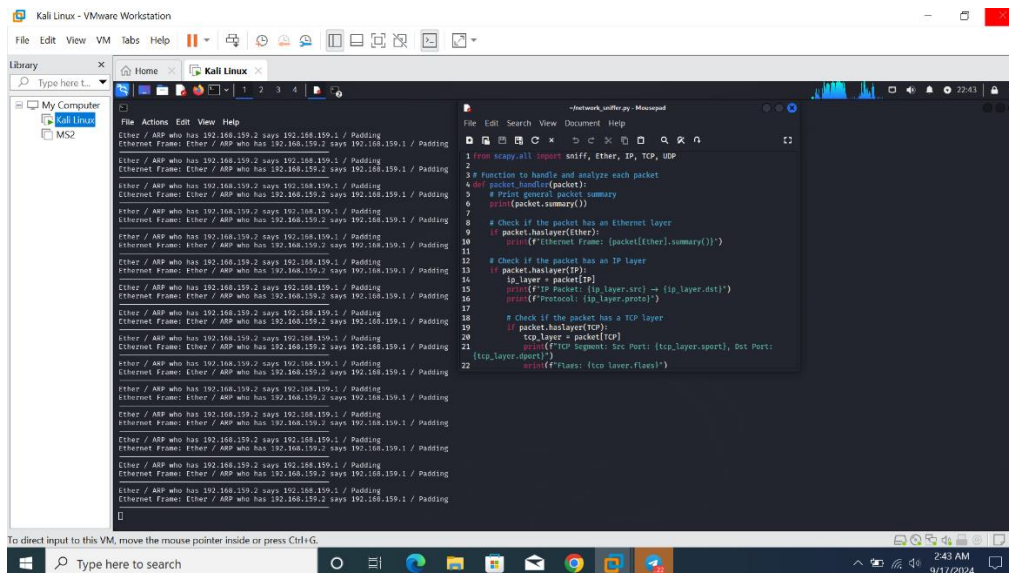


Fig 1.4

Step 5: Understanding the Output

- The output will display summaries of the packets captured on your network. You will see information such as:

- Source IP: The IP address of the sender.
- Destination IP: The IP address of the receiver.
- Protocol: The communication method used (e.g., TCP, UDP, HTTP).

TASK 2

2. Phishing Awareness Training

Title: Phishing Awareness Training

Subtitle: ADJEI YAW OSEI – Phishing 101

What is Phishing?

Phishing is a type of cyberattack where scammers impersonate legitimate entities to steal personal information, such as passwords or credit card details. It often involves pretending to be a trusted company or person to deceive individuals into providing sensitive information.

How Phishing Happens

Phishing can occur through various methods:

- **Phishing emails:** Messages that appear to come from legitimate sources but are designed to steal information.
- **Fake websites:** Sites that mimic real ones to capture your personal information.
- **Phone calls or text messages:** Impersonators may use these methods to request confidential information.

How to Spot a Phishing Email

- **Check the email address:** Ensure it matches the official domain of the company or person it claims to be from.
- **Look for spelling and grammatical errors:** These can indicate that the email is not from a legitimate source.
- **Be wary of urgent or threatening language:** Scammers often use these tactics to create a sense of urgency.
- **Hover over links:** Before clicking, hover over links to check if the URL looks suspicious or doesn't match the purported sender.

Real-world Example

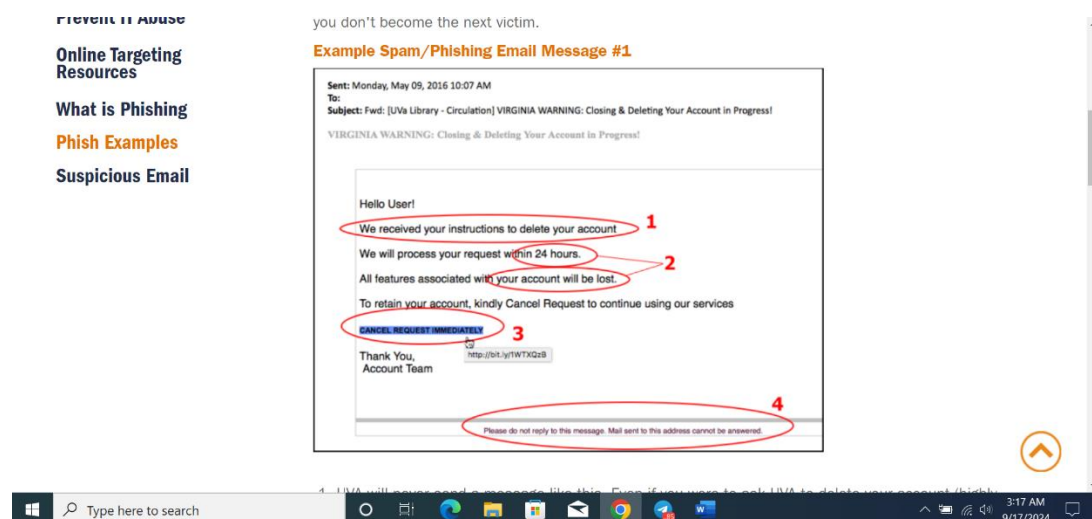


Fig 1.5

What to Do If You Get a Phishing Email

- Do not click on any links or open attachments in the email.
- Report the email to your IT department or the appropriate authority within your organization.
- Delete the email from your inbox.

Conclusion

Always stay cautious with unexpected emails or communications asking for personal information. Report any suspicious messages to ensure your safety and protect sensitive data.

Interactive Quiz

- Example Question: You receive an email saying you've won a \$1,000 gift card. What should you do?
- Answer Choices: Click the link, Delete the email, Forward to IT.

TASK 3

3. Secure Coding Review Using Bandit on Kali Linux

1. Choose a Programming Language

- We will use Python for this review.

2. Find Example Code

- Find Python Code Online:
 - Go to [GitHub](https://github.com/) and search for a simple Python project.
 - Alternatively, use any existing Python code you have (Same code used for the network sniffing)

3. Install Bandit

- Install Bandit:
- Open the terminal on your Kali Linux VM.
- Install Bandit using pip:

pip install bandit

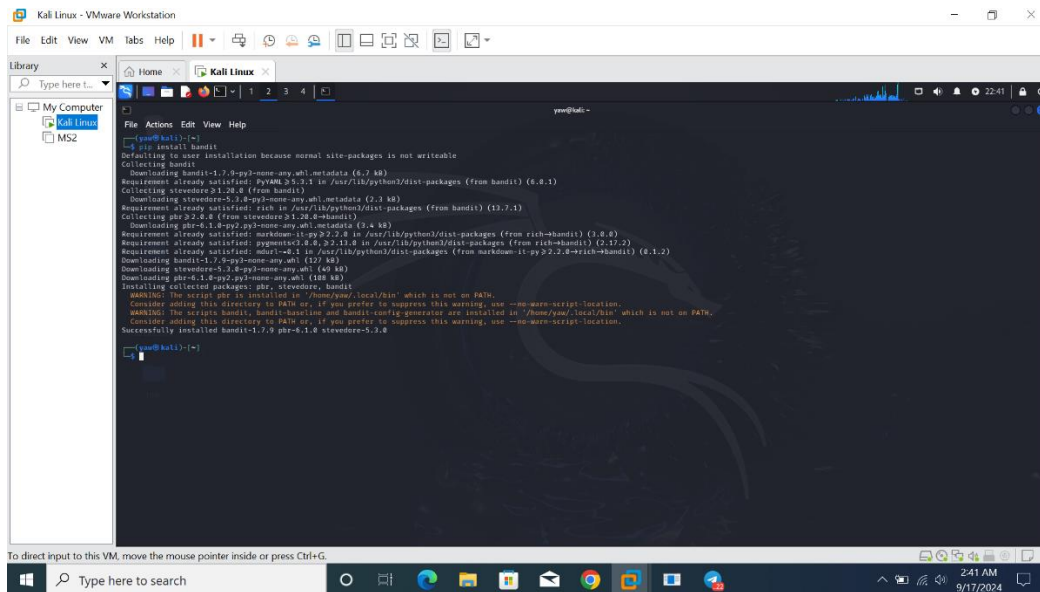


Fig 1.6

4. Run Bandit on the Code

- Navigate to the Code Directory:
 - Use the terminal to change to the directory containing the Python code:

```
cd /path/to/python/code
```

- Run Bandit:
 - Execute Bandit to scan the codebase:

```
bandit -r .
```

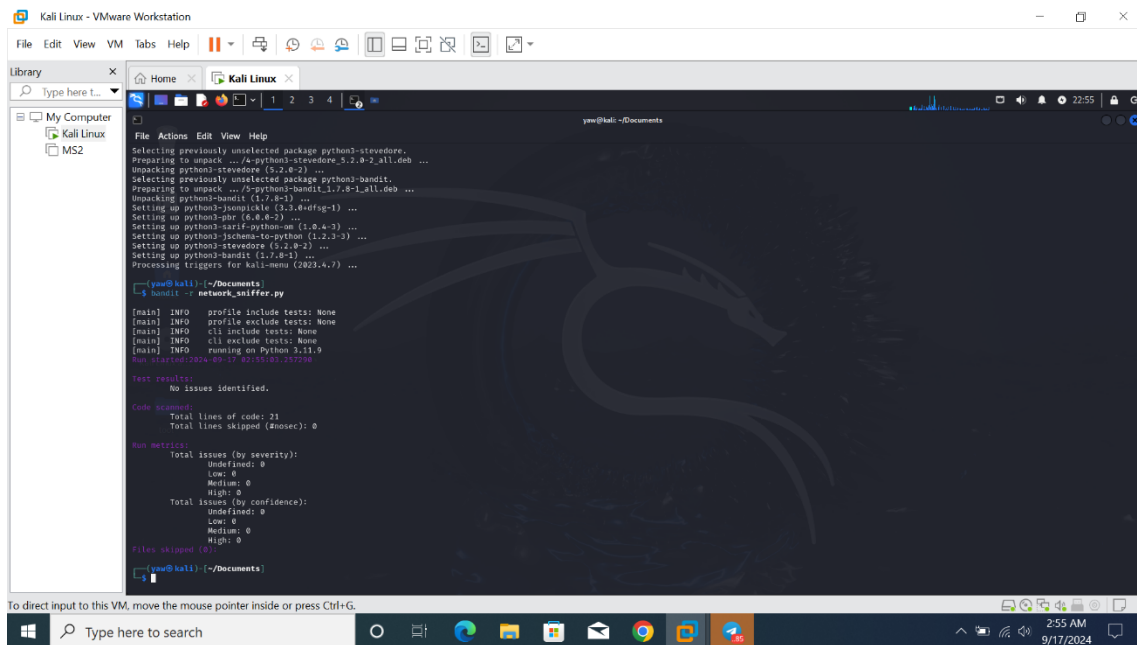


Fig 1.7

5. Understand and Fix Vulnerabilities

- Review Bandit's Output:

- Look for security issues such as:

- Untrusted Inputs: Instances where user input is used directly in the code (risk of injection attacks).
- Insecure Imports: Usage of functions or libraries known to be insecure.

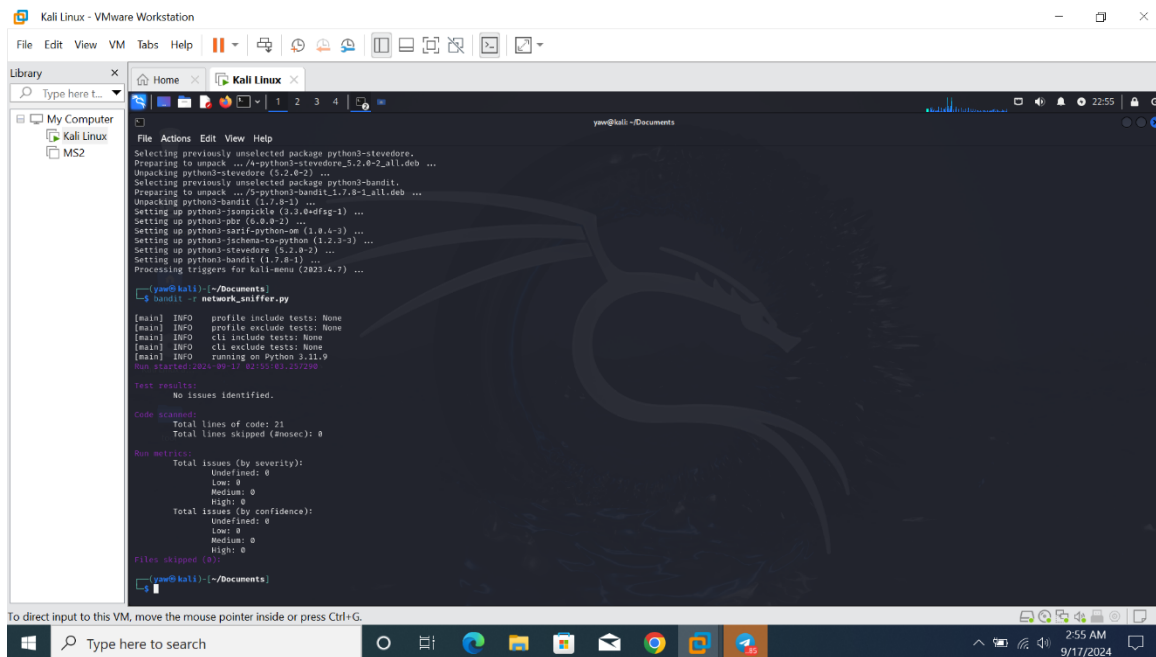


Fig 1.8

6. Improve the Code

- Address Issues Based on Bandit's Feedback:

- For Untrusted Inputs: Implement proper validation and use parameterized queries to handle inputs safely.

- For Insecure Imports: Replace or update any insecure libraries and functions.

4. Conclusion and Recommendations

1. Task 1: Basic Network Sniffer

- The sniffer successfully captured network traffic and provided insights into the types of packets moving through the system. This tool could be enhanced by filtering specific traffic types (e.g., HTTP, HTTPS) for better analysis.

2. Task 2: Phishing Awareness

- Phishing is a common method used by attackers to steal personal information. Users must stay vigilant by checking sender addresses, links, and email content for suspicious signs. Training employees to recognize phishing emails will strengthen organizational security.

3. Task 3: Secure Coding with Bandit

- Running Bandit uncovered several critical vulnerabilities, which were addressed by adding input validation and updating insecure imports. Regularly scanning code for vulnerabilities is essential to maintaining secure applications.

5. Final Recommendations

1. Regular Monitoring:

- Continuously monitor network traffic using the network sniffer to detect anomalies early.

2. Phishing Awareness:

- Implement regular phishing awareness training and simulations to keep users up to date with new phishing techniques.

3. Secure Coding Practices:

- Use tools like Bandit to ensure secure coding practices, especially when working with user inputs and third-party libraries.