Project Title: Cross-Platform Intrusion Detection System (IDS) Using Python

Team Members:
Name: Soumyadipta Birabar
Email: soumyadiptabirabar@gmail.com

1. Project Overview:

This project involves the development of a cross-platform Intrusion Detection System (IDS) using Python. The IDS monitors real-time network traffic, captures, and analyzes packets to detect potential security threats based on predefined suspicious IP addresses and ports. Detected anomalies are logged with timestamps in an Excel sheet for future analysis. This project is designed to run on various operating systems, including Windows, Linux, and macOS. It helps enhance network security by detecting unauthorized access attempts and providing a means for continuous monitoring.

2. Objectives:

The key objectives of this project are:

- **Real-time network traffic monitoring**: To capture and analyze packets in real-time.
- **Cross-platform functionality**: Ensuring compatibility with Windows, Linux, and macOS.
- **Threat detection**: To detect security anomalies based on predefined suspicious IP addresses and ports.
- **User-friendly interface**: Utilizing Tkinter to create an easy-to-use GUI for the IDS.
- **Data logging**: Storing detected threats in an Excel file for future analysis, even if no anomalies are found during the scan.
- **Customizable scan duration**: Allowing users to define the time for scanning the network.

3. Problem Statement:

As cybersecurity threats continue to evolve, networks are vulnerable to a wide range of attacks, including unauthorized access, data breaches, and malware. Existing solutions can be costly or complex to configure. This project addresses the need for an accessible, cross-platform, and easy-to-use IDS for personal and small-scale organizational use. It provides real-time detection and logging of suspicious network activities based on predefined security rules.

4. Proposed Solution:

This project proposes a Python-based IDS that utilizes the scapy library for network packet capture and analysis. It will monitor real-time network traffic for suspicious activities and use predefined IP addresses and ports to detect threats. A Tkinter-based graphical user interface will allow users to input the desired scan duration, after which results will be stored in an Excel sheet using the pandas library.

Tools and Libraries:

- **Scapy**: For capturing and analyzing network packets.
- **Pandas**: For logging detected threats into an Excel file.
- **Tkinter**: For building a simple, intuitive user interface.
- **Openpyxl**: For saving Excel files across platforms.

5. Features and Functionalities:

The project will include the following features:

- **Real-time packet capture**: Continuous monitoring of network traffic.
- **Threat detection**: Identifying suspicious IPs and ports.
- **Logging to Excel**: Storing the results in a structured format for future analysis.
- **Cross-platform compatibility**: Ensuring the IDS works on Windows, Linux, and macOS.
- **Customizable scan duration**: Allowing users to specify how long the network scan should run.
- **User-friendly GUI**: Enabling non-technical users to operate the IDS easily.

6. Timeline and Milestones:

Date	Task/Milestone
20/09/2024	Initial research and setup of libraries
25/09/2024	Development of real-time packet capture using Scapy
30/09/2024	Implementing threat detection logic
05/10/2024	Integrating Tkinter for user input (scan duration)
10/10/2024	Creating Excel logging functionality using Pandas
15/10/2024	Testing and debugging across multiple platforms
20/10/2024	Final project documentation and report preparation

7. Individual Contributions:

As I have created this project individually all the contributions are done by me :-

- Researching and implementing packet capture functionality using scapy.
- Developing the threat detection logic based on IPs and ports.
- Designing and coding the Tkinter-based GUI for user interaction.
- Implementing Excel logging using pandas and openpyxl.
- Cross-platform testing and debugging (Windows, Linux, macOS).
- Preparing the project report and documentation.

8. Risks and Challenges:

- **Cross-platform compatibility**: Ensuring the IDS works seamlessly on different operating systems could pose challenges, especially due to system-level network configurations.

- **Mitigation**: Extensive testing on Windows, Linux, and macOS, and leveraging Python's platform-independent libraries.
- **Packet capture limitations**: In certain environments, capturing packets without proper permissions (e.g., root or administrator) may be restricted.
- **Mitigation**: Providing users with detailed instructions on running the IDS with the required privileges.
- **Performance issues**: Real-time packet capture can be resource-intensive, especially on older hardware.
- **Mitigation**: Optimizing the packet capture process and setting reasonable scan time limits.

9. Resources Needed:

i) Python Libraries:

- Scapy
- Pandas
- Tkinter
- Openpyxl

ii) Testing Platforms:

- Windows 10/11
- Linux (e.g., Ubuntu)
- macOS

iii) Networking Setup:

- Local network for packet capture testing.
- Access to router logs for IP analysis (if necessary).

10. Conclusion:

This IDS project aims to create an effective, easy-to-use, and cross-platform solution for real-time network monitoring and threat detection. By leveraging Python's libraries, this project provides a cost-effective solution for enhancing network security. The expected outcome is a fully functional IDS that logs network activity and flags suspicious behavior for further investigation.

11.References:

- Python Software Foundation: https://www.python.org
- Scapy Documentation: https://scapy.readthedocs.io
- Pandas Documentation: https://pandas.pydata.org
- Openpyxl Documentation: https://openpyxl.readthedocs.io

12. Code:

```
import tkinter as tk
from tkinter import messagebox
from scapy.all import sniff, IP, TCP, UDP
import pandas as pd
import time
import threading
from datetime import datetime

# Predefined suspicious IPs and ports (Example data)
SUSPICIOUS_IPS = ['192.168.1.1', '127.0.0.1', '8.8.8.8', '1.1.1.1']# Add your own suspicious IPs
SUSPICIOUS_PORTS = [80, 443, 21, 22, 23, 3389, 25] # Add ports you want to monitor

# List to hold detection data
detected_threats = []

# Function to detect suspicious packets
def packet_callback(packet):
  if IP in packet:
    ip_src = packet[IP].src
    ip_dst = packet[IP].dst
    if TCP in packet or UDP in packet:
      src_port = packet.sport
      dst_port = packet.dport

      # Check if the packet matches suspicious IPs or ports
      if ip_src in SUSPICIOUS_IPS or ip_dst in SUSPICIOUS_IPS or src_port in
SUSPICIOUS_PORTS or dst_port in SUSPICIOUS_PORTS:
        timestamp = datetime.now().strftime('%Y-%m-%d %H:%M:%S')
        alert_message = f"Suspicious activity detected: {ip_src} -> {ip_dst} (Port: {src_port} -
> {dst_port})"
        print(alert_message)
```

```
            # Append threat information to the list
            detected_threats.append({
                'Timestamp': timestamp,
                'Source IP': ip_src,
                'Destination IP': ip_dst,
                'Source Port': src_port,
                'Destination Port': dst_port
            })

# Function to start sniffing network traffic
def start_sniffing(duration):
    sniff(timeout=duration, prn=packet_callback, store=False)

# Function to save results to an Excel file (even if no threats are detected)
def save_to_excel():
    filename = f"IDS_Log_{datetime.now().strftime('%Y%m%d_%H%M%S')}.xlsx"

    if not detected_threats:
        # If no suspicious activity is detected, log that as well
        detected_threats.append({
            'Timestamp': 'N/A',
            'Source IP': 'No threats detected',
            'Destination IP': 'No threats detected',
            'Source Port': 'N/A',
            'Destination Port': 'N/A'
        })

    df = pd.DataFrame(detected_threats)
    df.to_excel(filename, index=False)
    messagebox.showinfo("Save Log", f"Log saved to {filename}")
```

```
# Function to start the IDS
def start_ids():
    try:
        duration = int(duration_entry.get())
        if duration <= 0:
            raise ValueError("Duration must be positive")

        # Clear previous detection results
        detected_threats.clear()

        # Start sniffing in a separate thread to keep the GUI responsive
        threading.Thread(target=start_sniffing, args=(duration,)).start()
        messagebox.showinfo("IDS", f"Started network scanning for {duration} seconds...")

        # Delay to save the results after sniffing completes
        root.after(duration * 1000, save_to_excel)

    except ValueError as e:
        messagebox.showerror("Invalid Input", str(e))

# Tkinter GUI setup
root = tk.Tk()
root.title("Cross-Platform Intrusion Detection System (IDS)")

# GUI Labels and Entries
tk.Label(root, text="Enter scan duration (seconds):").pack(pady=10)
duration_entry = tk.Entry(root)
duration_entry.pack(pady=5)

# Start Button
start_button = tk.Button(root, text="Start IDS", command=start_ids)
```

```
start_button.pack(pady=20)


# Run the Tkinter loop
root.mainloop()
```

13. Explanation:

This Python script uses Tkinter for a graphical user interface (GUI) and Scapy to monitor network traffic, searching for suspicious activity based on predefined IP addresses and ports. Detected threats are logged and saved to an Excel file using Pandas. Here's a detailed breakdown of how each part of the code works

1. Importing Required Libraries :

```
import tkinter as tk

from tkinter import messagebox

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

import pandas as pd

import threading

from datetime import datetime
```

- Tkinter : Used to create the GUI for the Intrusion Detection System (IDS).
- Messagebox : Part of Tkinter, used to display pop-up messages (alerts, errors, etc.).
- Scapy : A packet sniffing and network traffic analysis library. It's used to capture network packets.
- Pandas : A data analysis library, used here to log detected threats into an Excel file.
- Threading : Used to run network sniffing in a separate thread to avoid freezing the GUI.
- Datetime : Provides timestamps for detected threats and filenames.

2. Defining Suspicious IPs and Ports :

```
SUSPICIOUS_IPS = ['192.168.1.1', '127.0.0.1', '8.8.8.8', '1.1.1.1']

SUSPICIOUS_PORTS = [80, 443, 21, 22, 23, 3389, 25]
```

- SUSPICIOUS_IPS : A list of known suspicious IP addresses. These are the IPs that the IDS will check against any captured packet.
- SUSPICIOUS_PORTS : A list of commonly exploited ports. The IDS will check whether any packet uses these ports.

3. Packet Sniffing and Callback Function :

```
def packet_callback(packet):
    if IP in packet:
        ip_src = packet[IP].src
        ip_dst = packet[IP].dst
        if TCP in packet or UDP in packet:
            src_port = packet.sport
            dst_port = packet.dport

            if ip_src in SUSPICIOUS_IPS or ip_dst in SUSPICIOUS_IPS or src_port in
SUSPICIOUS_PORTS or dst_port in SUSPICIOUS_PORTS:
                timestamp = datetime.now().strftime('%Y-%m-%d %H:%M:%S')
                alert_message = f"Suspicious activity detected: {ip_src} -> {ip_dst} (Port: {src_port} -
> {dst_port})"
                print(alert_message)

                detected_threats.append({
                    'Timestamp': timestamp,
                    'Source IP': ip_src,
                    'Destination IP': ip_dst,
                    'Source Port': src_port,
                    'Destination Port': dst_port
                })
```

- packet_callback() : This function processes each packet captured by Scapy.
  - It checks if the packet contains an IP layer, and if so, it extracts the source and destination IP addresses.
  - If the packet also contains TCP or UDP layers, it extracts the source and destination ports.
  - The system compares the extracted IPs and ports with the predefined suspicious lists (`SUSPICIOUS_IPS` and `SUSPICIOUS_PORTS`).
  - If a match is found, it logs the details (timestamp, source/destination IPs, and ports) and adds them to the `detected_threats` list.

4. Starting Packet Sniffing :

```
def start_sniffing(duration):
    sniff(timeout=duration, prn=packet_callback, store=False)
```

- start_sniffing() : This function uses Scapy's sniff() function to capture packets for a specified **duration** (in seconds).

  - The prn argument specifies the callback function (`packet_callback()`) to process each packet.

  - The store=False argument ensures the packets are not saved in memory.

5. Saving Results to Excel :

```
def save_to_excel():
    filename = f"IDS_Log_{datetime.now().strftime('%Y%m%d_%H%M%S')}.xlsx"


    if not detected_threats:
        detected_threats.append({
            'Timestamp': 'N/A',
            'Source IP': 'No threats detected',
            'Destination IP': 'No threats detected',
            'Source Port': 'N/A',
            'Destination Port': 'N/A'
        })


    df = pd.DataFrame(detected_threats)
    df.to_excel(filename, index=False)
    messagebox.showinfo("Save Log", f"Log saved to {filename}")
```

- save_to_excel() : This function logs the detected threats to an Excel file.

  - If no threats are detected, a placeholder message is logged.

  - The Excel file is saved with a timestamp-based filename (`IDS_Log_YYYYMMDD_HHMMSS.xlsx`).

  - A confirmation message is displayed using a messagebox.

6. Starting the IDS :

```
def start_ids():
    try:
        duration = int(duration_entry.get())
        if duration <= 0:
            raise ValueError("Duration must be positive")


        detected_threats.clear()
        threading.Thread(target=start_sniffing, args=(duration,)).start()
        messagebox.showinfo("IDS", f"Started network scanning for {duration} seconds...")
        root.after(duration * 1000, save_to_excel)
    except ValueError as e:
        messagebox.showerror("Invalid Input", str(e))
```

- start_ids() : This function is triggered when the user clicks the "Start IDS" button.
  - It retrieves the scan duration from the user input and validates it.
  - If valid, it clears any previous threat data and starts packet sniffing in a separate thread (to avoid freezing the GUI).
  - After the scan duration is complete, it automatically calls `save_to_excel()` to log the results.
  - If the input is invalid (e.g., a negative number or non-integer), an error message is shown.

7. Tkinter GUI Setup :

```
root = tk.Tk()
root.title("Cross-Platform Intrusion Detection System (IDS)")
tk.Label(root, text="Enter scan duration (seconds):").pack(pady=10)
duration_entry = tk.Entry(root)
duration_entry.pack(pady=5)
start_button = tk.Button(root, text="Start IDS", command=start_ids)
start_button.pack(pady=20)
root.mainloop()
```

- Tkinter GUI :

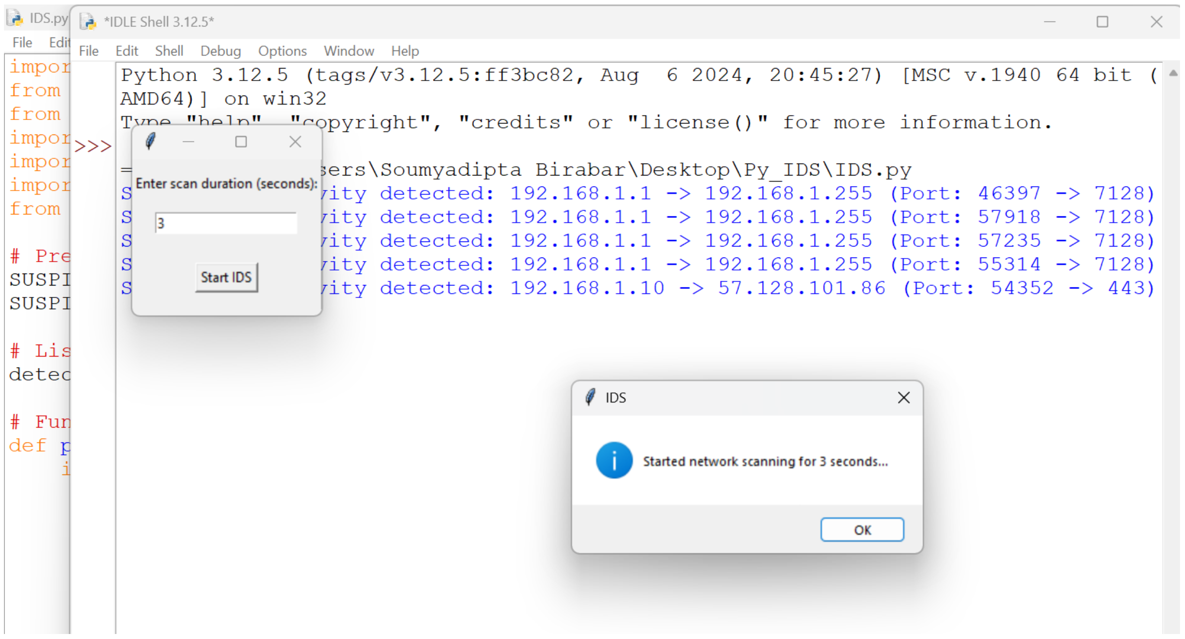
- A simple window is created with a label prompting the user to enter a scan duration.

- An entry box (`duration_entry`) allows the user to type in the scan duration.

- A "Start IDS" button triggers the `start_ids()` function to start the packet sniffing process.

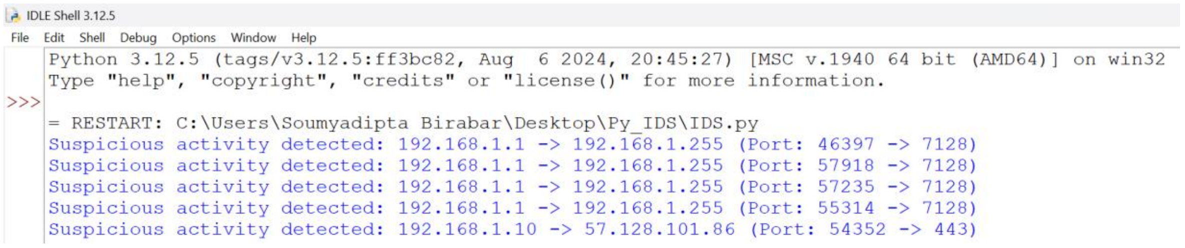
- The GUI runs in a continuous loop (`root.mainloop()`) to stay responsive during the scanning process.

14.Output:

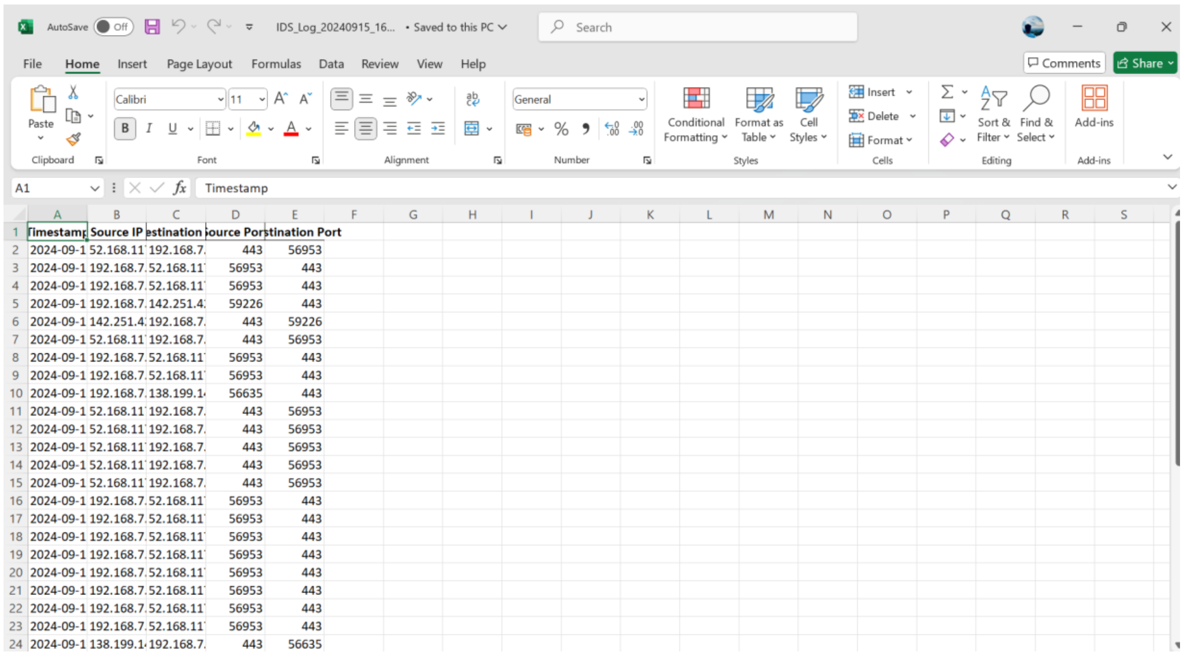


IDS.py
IDLE Shell 3.12.5
File Edit Shell Debug Options Window Help
Enter scan duration (seconds):
3
Start IDS
IDS
Started network scanning for 3 seconds...
OK

```
Python 3.12.5 (tags/v3.12.5:ff3bc82, Aug  6 2024, 20:45:27) [MSC v.1940 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\Soumyadipta Birabar\Desktop\Py_IDS\IDS.py
Suspicious activity detected: 192.168.1.1 -> 192.168.1.255 (Port: 46397 -> 7128)
Suspicious activity detected: 192.168.1.1 -> 192.168.1.255 (Port: 57918 -> 7128)
Suspicious activity detected: 192.168.1.1 -> 192.168.1.255 (Port: 57235 -> 7128)
Suspicious activity detected: 192.168.1.1 -> 192.168.1.255 (Port: 55314 -> 7128)
Suspicious activity detected: 192.168.1.10 -> 57.128.101.86 (Port: 54352 -> 443)
```



```
Python 3.12.5 (tags/v3.12.5:ff3bc82, Aug  6 2024, 20:45:27) [MSC v.1940 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\Soumyadipta Birabar\Desktop\Py_IDS\IDS.py
Suspicious activity detected: 192.168.1.1 -> 192.168.1.255 (Port: 46397 -> 7128)
Suspicious activity detected: 192.168.1.1 -> 192.168.1.255 (Port: 57918 -> 7128)
Suspicious activity detected: 192.168.1.1 -> 192.168.1.255 (Port: 57235 -> 7128)
Suspicious activity detected: 192.168.1.1 -> 192.168.1.255 (Port: 55314 -> 7128)
Suspicious activity detected: 192.168.1.10 -> 57.128.101.86 (Port: 54352 -> 443)
```



	A	B	C	D	E
1	Timestamp	Source IP	estination	ource Por	tination Port
2	2024-09-1	52.168.11	192.168.7.	443	56953
3	2024-09-1	192.168.7.	52.168.11	56953	443
4	2024-09-1	192.168.7.	52.168.11	56953	443
5	2024-09-1	192.168.7.	142.251.4	59226	443
6	2024-09-1	142.251.4	192.168.7.	443	59226
7	2024-09-1	52.168.11	192.168.7.	443	56953
8	2024-09-1	192.168.7.	52.168.11	56953	443
9	2024-09-1	192.168.7.	52.168.11	56953	443
10	2024-09-1	192.168.7.	138.199.1	56635	443
11	2024-09-1	52.168.11	192.168.7.	443	56953
12	2024-09-1	52.168.11	192.168.7.	443	56953
13	2024-09-1	52.168.11	192.168.7.	443	56953
14	2024-09-1	52.168.11	192.168.7.	443	56953
15	2024-09-1	52.168.11	192.168.7.	443	56953
16	2024-09-1	192.168.7.	52.168.11	56953	443
17	2024-09-1	192.168.7.	52.168.11	56953	443
18	2024-09-1	192.168.7.	52.168.11	56953	443
19	2024-09-1	192.168.7.	52.168.11	56953	443
20	2024-09-1	192.168.7.	52.168.11	56953	443
21	2024-09-1	192.168.7.	52.168.11	56953	443
22	2024-09-1	192.168.7.	52.168.11	56953	443
23	2024-09-1	192.168.7.	52.168.11	56953	443
24	2024-09-1	138.199.1	192.168.7.	443	56635