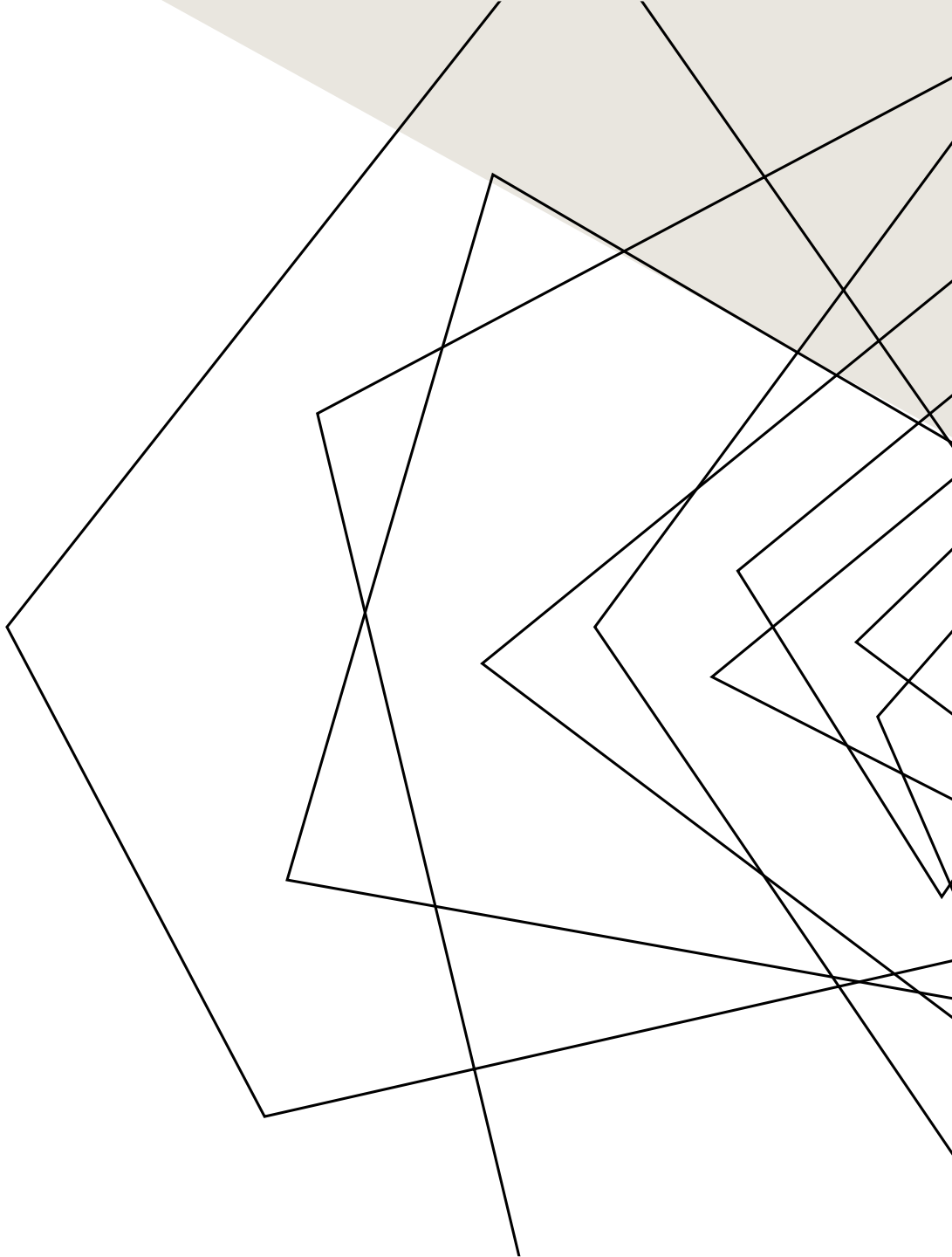
NETWORK TRAFFIC ANALYSIS TOOL

ABOUT US

We are a team of cybersecurity learners passionate about building tools that enhance digital safety. Our project, **Network Traffic Analysis Tool**, helps monitor and visualize network activity to detect potential threats, making cybersecurity more accessible and proactive.



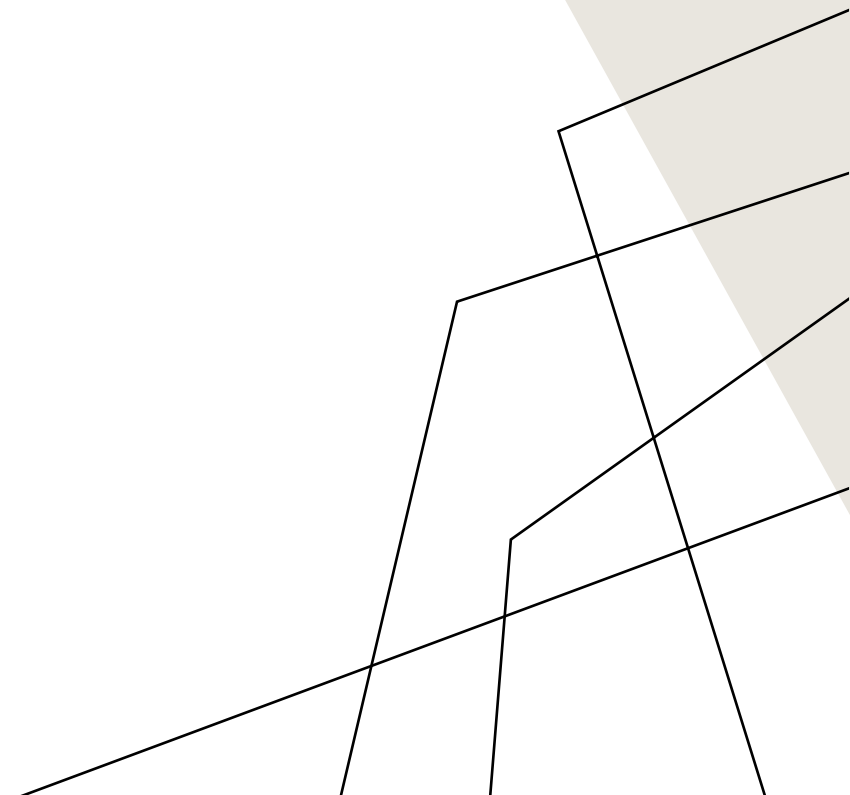
PROJECT OVERVIEW

OVERVIEW

- The Network Traffic Analysis Tool is a Python-based cybersecurity project that captures and analyzes network packets to detect suspicious activity. It uses packet sniffing techniques and protocol inspection to monitor real-time traffic and identify potential threats like port scans or abnormal spikes.
- To make insights more accessible, the tool visualizes traffic patterns using pie charts and graphs, helping users quickly understand network behavior. It's a lightweight, user-friendly solution ideal for students, analysts, and IT professionals aiming to enhance digital security.

PROBLEM

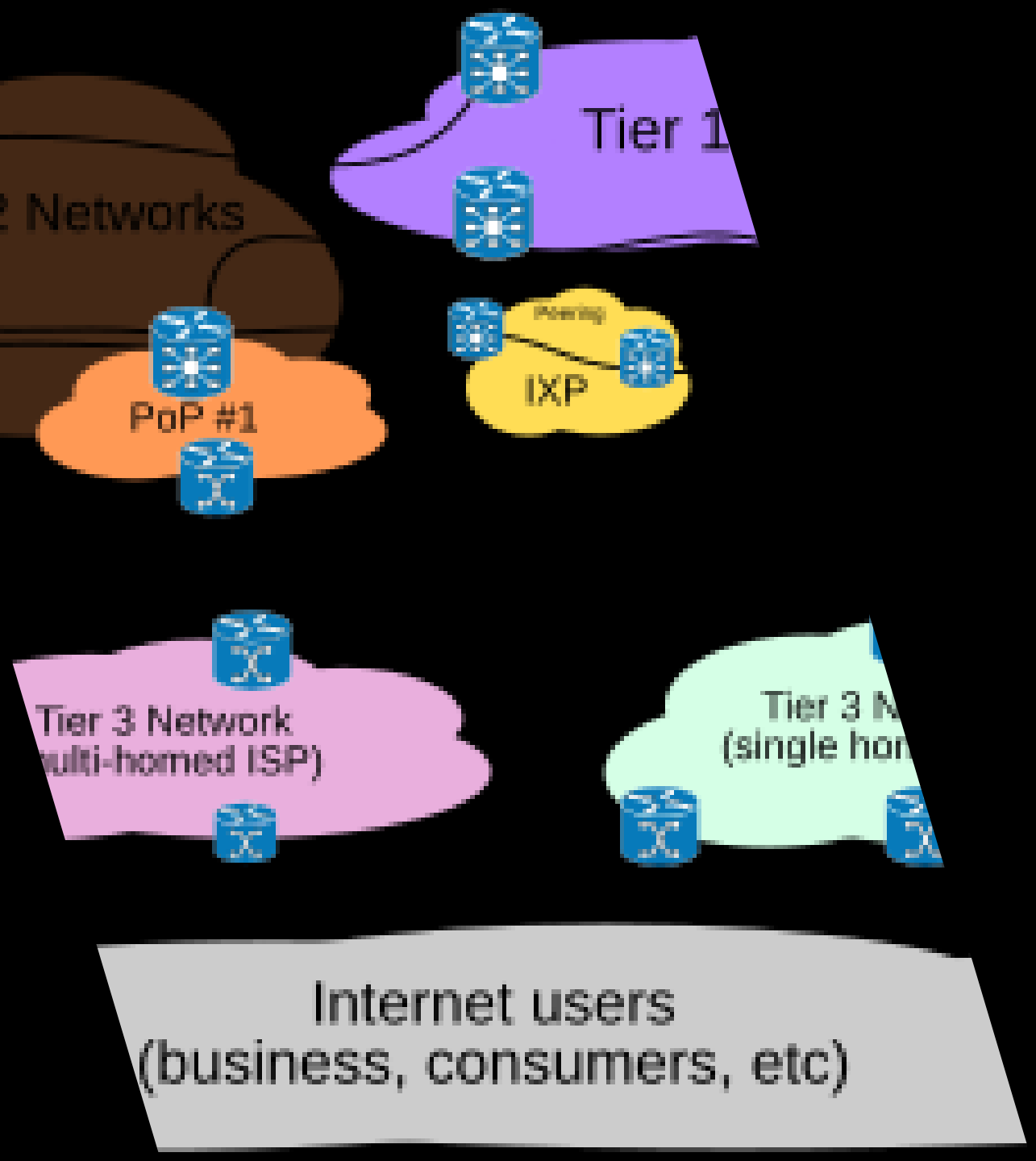
- The basic problem this project addresses is the **lack of real-time visibility into network activity**, which makes it difficult to detect cyber threats like unauthorized access, data exfiltration, or port scanning.
- Most users and even small organizations don’t have tools to monitor what’s happening on their networks. This creates a blind spot that attackers can exploit. Your **Network Traffic Analysis Tool** helps fill that gap by capturing and analyzing traffic patterns, making it easier to spot suspicious behavior before it becomes a serious breach.



SOLUTION

The solution is to develop a **Network Traffic Analysis Tool** that gives users real-time visibility into their network activity. By capturing and analyzing packets, the tool can detect anomalies like port scans, suspicious payloads, or traffic spikes—issues that often go unnoticed without proper monitoring.

This tool empowers users to take proactive steps in securing their networks by combining **packet sniffing**, **protocol analysis**, and **data visualization** into one lightweight, Python-based application. It bridges the gap between raw network data and actionable insights, making threat detection more accessible and efficient.



PROJECT BENEFITS

OUR BENEFITS

- **Real-Time Threat Detection**: It helps identify suspicious activities like port scans, data exfiltration, or malware communication as they happen.
- **Improved Network Visibility**: Users gain a clear view of which devices are communicating, what protocols are used, and how data flows across the network.
- **Faster Incident Response**: By visualizing traffic patterns and anomalies, users can respond to potential threats more quickly and effectively.

FRAMEWORKS OF PROJECT

🔧 1. **Set Up Your Environment**

•**Language**: Python is ideal due to libraries like scapy, pyshark, and matplotlib.

•**Tools**: Wireshark (for reference), TCPDump (for packet capture), and Jupyter Notebook (for visualization).

📡 2. **Capture Network Traffic (Packet Sniffing)**

Use scapy or pyshark to sniff packets:

from scapy.all import sniff def capture_packets(interface="eth0", packet_count=100) : packets = sniff(iface=interface, count=packet_count) packets.summary()

code:-

```
from scapy.all import sniffdef capture_packets(interface="eth0", packet_count=100):
    packets = sniff(iface=interface, count=packet_count)    packets.summary()
```

Or

use tcpdump in a shell script:

Code:-

```
sudo tcpdump -i any -w traffic.pcap
```

🧠 3. **Analyze the Traffic**

Parse .pcap files to extract:

- Source/Destination IPs
- Protocols used (TCP, UDP, ICMP)
- Port numbers
- Packet sizes and timestamps

Use pyshark:

import pyshark cap = pyshark.FileCapture('traffic.pcap') for packet in cap: print(packet.highest_layer, packet.ip.src, packet.ip.dst)

code:-

```
import pysharkcap = pyshark.FileCapture('traffic.pcap')for packet in cap:
print(packet.highest_layer, packet.ip.src, packet.ip.dst)
```

📊 4. **Visualize the Data**

Use matplotlib or seaborn to create:

- Pie charts for protocol distribution
- Line graphs for traffic over time
- Heatmaps for IP communication

CODE:-

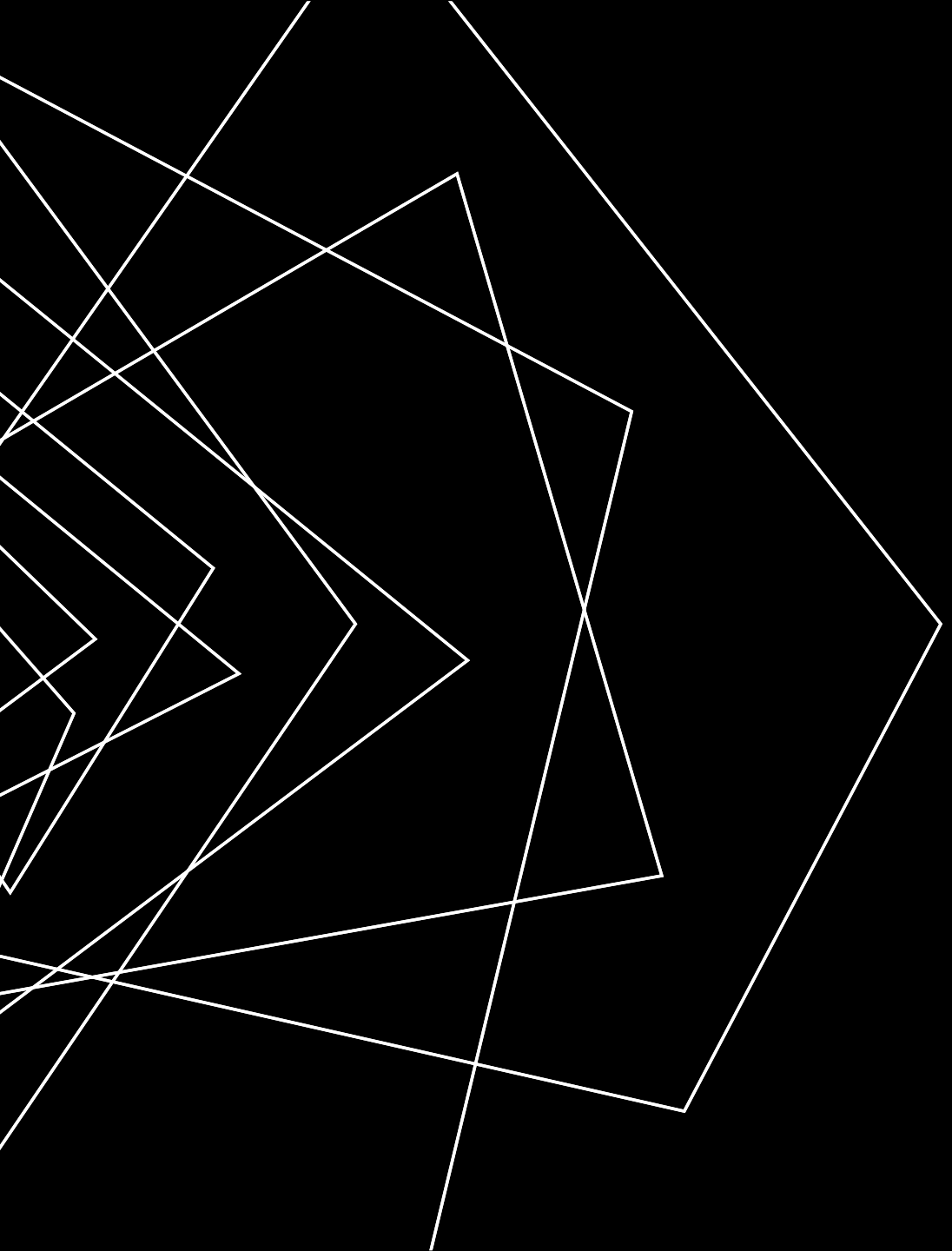
```
IMPORT MATPLOTLIB.PYPLOT AS
PLTPROTOCOLS = ['TCP', 'UDP',
'ICMP']COUNTS = [120, 80,
30]PLT.PIE(COUNTS, LABELS=PROTOCOLS,
AUTOPCT='%1.1F%%')PLT.TITLE('PROTOCOL
DISTRIBUTION')PLT.SHOW()
```

🛡️ 5. DETECT SUSPICIOUS PATTERNS

IMPLEMENT BASIC THREAT DETECTION:
PORT SCANNING (MANY PORTS FROM ONE IP)
UNUSUAL TRAFFIC SPIKES
PACKETS WITH SUSPICIOUS PAYLOADS
YOU CAN SET THRESHOLDS AND FLAG ANOMALIES USING SIMPLE LOGIC OR INTEGRATE WITH ML MODELS LATER.

📁 6. Optional: Build a GUI

Use Tkinter or Streamlit to make your tool user-friendly.



THANK YOU

SHALINI ANAND
EMAIL- shalini.24bai10849@vitbhopal.ac.in