Packet feature extractor

> ***Python for Security***

Mini Project
Week 1.3

26th of *August, 2025*

Lead: *Muhammad Tayyab*
Athar Imran (Buildables Cybersecurity Fellow)

Introduction

While packet-level data provides raw visibility, analysts and AI/ML models work better with **flow-level features**. A flow represents communication between a **source IP:port → destination IP:port** (with protocol). Extracting features such as **flow duration, total packets, TCP flags, and bytes transferred** allows for more meaningful **network behavior analysis** and anomaly detection.

This project builds a **Python packet feature extractor** that converts PCAPs into structured **flow-based CSV datasets**.

Objectives

- Parse a PCAP into **network flows**.
- Extract **flow features**:
    - Flow identifiers (src/dst IP, ports, protocol).
    - Start time, end time, duration.
    - Total packets, total bytes.
    - TCP flags (SYN, FIN, RST, PSH, ACK counts).
- Save flows as **CSV**.

Environment

- Python 3.10+
- Libraries: `scapy`, `pandas`

Methodology

Step 1: Imports

```
from scapy.all import *
import pandas as pd
from collections import defaultdict
```

Step 2: Define Flow Key

Flows are identified by **(src_ip, dst_ip, src_port, dst_port, protocol)**.

```
def get_flow_key(pkt):
    if IP in pkt:
        src_ip = pkt[IP].src
        dst_ip = pkt[IP].dst
        proto = pkt[IP].proto

        src_port = pkt.sport if hasattr(pkt, "sport") else 0
        dst_port = pkt.dport if hasattr(pkt, "dport") else 0

        return (src_ip, dst_ip, src_port, dst_port, proto)
    return None
```

This function **builds a unique 5-tuple flow key** from a packet:

- Source IP
- Destination IP
- Source port
- Destination port
- Protocol

If the packet doesn't have an IP layer, it returns `None`.

Step 3: Extract Flow Features

```
flows = defaultdict(lambda: {
    "start": None,
    "end": None,
    "pkt_count": 0,
    "byte_count": 0,
```

```
    "flags": {"SYN":0, "FIN":0, "RST":0, "PSH":0, "ACK":0}
})

pcap = rdpcap("traffic.pcap")

for pkt in pcap:
    key = get_flow_key(pkt)
    if not key:
        continue

    flow = flows[key]
    ts = pkt.time

    # Set start time
    if flow["start"] is None:
        flow["start"] = ts
    flow["end"] = ts

    # Count packets & bytes
    flow["pkt_count"] += 1
    flow["byte_count"] += len(pkt)

    # Extract TCP flags if present
    if TCP in pkt:
        tcp_flags = pkt[TCP].flags
        if tcp_flags & 0x02: flow["flags"]["SYN"] += 1
        if tcp_flags & 0x01: flow["flags"]["FIN"] += 1
        if tcp_flags & 0x04: flow["flags"]["RST"] += 1
        if tcp_flags & 0x08: flow["flags"]["PSH"] += 1
        if tcp_flags & 0x10: flow["flags"]["ACK"] += 1
```

1. **`flows` dictionary:** Keeps track of each unique network flow (identified by the 5-tuple from `get_flow_key`).
    - Stores start & end time, packet count, byte count, and TCP flag counters.
2. **Looping through packets (`pcap`):**
    - Extracts flow key, groups packets into flows.
    - Records **start & end time** of the flow.
    - Increments **packet count** and **total bytes**.
    - If it's a TCP packet, it counts occurrences of important **flags (SYN, FIN, RST, PSH, ACK)**.
3. **End result:**
    - You get structured statistics for every network flow in the PCAP (when it started/ended, how many packets/bytes, and TCP flag behavior).

Step 4: Convert to DataFrame

```
rows = []
for key, feat in flows.items():
    src_ip, dst_ip, src_port, dst_port, proto = key
    duration = feat["end"] - feat["start"] if feat["end"] and
feat["start"] else 0

    row = {
        "src_ip": src_ip,
        "dst_ip": dst_ip,
        "src_port": src_port,
        "dst_port": dst_port,
        "protocol": proto,
        "duration": duration,
        "pkt_count": feat["pkt_count"],
        "byte_count": feat["byte_count"],
        "syn_count": feat["flags"]["SYN"],
        "fin_count": feat["flags"]["FIN"],
        "rst_count": feat["flags"]["RST"],
        "psh_count": feat["flags"]["PSH"],
        "ack_count": feat["flags"]["ACK"]
    }
    rows.append(row)

df = pd.DataFrame(rows)
df.to_csv("flow_features.csv", index=False)
print("Flow features exported to flow_features.csv")
```

1. **Iterates over all flows** collected earlier.
    - Extracts source/destination IP, ports, and protocol.
    - Computes flow **duration** (end time – start time).
    - Collects **packet count, byte count, and TCP flag counts**.
2. **Builds rows of dictionaries** with these flow features.
3. **Converts rows -> Pandas DataFrame.**
4. **Exports to CSV (`flow_features.csv`)** -> each row = one flow, each column = feature.

Results

1.3 Mini Project - Packet feature extractor .py 9 | flow_features.csv

```
src_ip,dst_ip,src_port,dst_port,protocol,duration,pkt_count,byte_count,syn_count,fin_count,rst_count,psh_count,ack_count
192.168.56.1,192.168.56.103,55470,22,6,8.219647,26,4613,1,1,0,11,25
192.168.56.103,192.168.56.1,22,55470,6,8.219106,19,3710,1,1,0,10,19
192.168.56.1,192.168.56.103,55471,22,6,4.696945,22,4045,1,1,0,9,21
192.168.56.103,192.168.56.1,22,55471,6,4.696341,15,3334,1,1,0,8,15
192.168.56.1,192.168.56.103,55472,22,6,3.641563,22,4045,1,1,0,9,21
192.168.56.103,192.168.56.1,22,55472,6,3.640521,15,3334,1,1,0,8,15
192.168.56.1,192.168.56.103,55473,22,6,3.576275,22,4045,1,1,0,9,21
192.168.56.103,192.168.56.1,22,55473,6,3.575605,15,3334,1,1,0,8,15
192.168.56.1,192.168.56.103,55474,22,6,6.225861,24,4329,1,1,0,10,23
192.168.56.103,192.168.56.1,22,55474,6,6.225217,17,3522,1,1,0,9,17
192.168.56.1,192.168.56.103,55475,22,6,10.018158,26,4613,1,1,0,11,25
192.168.56.103,192.168.56.1,22,55475,6,10.017744,19,3710,1,1,0,10,19
192.168.56.1,192.168.56.103,55476,22,6,3.969061,22,4045,1,1,0,9,21
192.168.56.103,192.168.56.1,22,55476,6,3.968636,15,3334,1,1,0,8,15
192.168.56.1,192.168.56.103,55477,22,6,4.071990,22,4045,1,1,0,9,21
192.168.56.103,192.168.56.1,22,55477,6,4.071371,15,3334,1,1,0,8,15
192.168.56.1,192.168.56.103,55478,22,6,7.979809,24,4329,1,1,0,10,23
192.168.56.103,192.168.56.1,22,55478,6,7.979085,17,3522,1,1,0,9,17
192.168.56.1,192.168.56.103,55479,22,6,4.306853,22,4045,1,1,0,9,21
192.168.56.103,192.168.56.1,22,55479,6,4.305717,15,3334,1,1,0,8,15
192.168.56.1,192.168.56.103,55480,22,6,3.744409,22,4045,1,1,0,9,21
192.168.56.103,192.168.56.1,22,55480,6,3.743460,15,3334,1,1,0,8,15
```

Conclusion

This project demonstrates how to extract **flow-level features from PCAPs**, a key step in building **AI-powered intrusion detection systems (IDS)**. Structured CSV flow features can be directly used in **machine learning models** for classification (benign vs malicious traffic).