Network IPS

Name: Kirti Koltharkar

Intern Id: 246

Network IPS — block pings, malicious connections, and simple exploit

- **Build a lightweight Network Intrusion Prevention System (IPS) in Python that can:**
  1. Analyse PCAP files.
  2. Detect suspicious traffic patterns:
     - ICMP Flooding (Ping Floods) → too many pings from one host.
     - SYN Flooding / Scans → too many half-open TCP connections.
     - SQL Injection patterns in HTTP payloads.
  3. Block malicious IP addresses for a fixed time.
  4. Differentiate between:
     - Normal traffic PCAP (benign browsing/pings).
     - Attack traffic PCAP (floods, scans, exploits).
- **Tools and setup:**
  1. Language: Python
  2. Library: Scapy → for reading and parsing packets
  3. Test Data: PCAP files (downloaded from Wireshark Sample Captures , store same folder of a python code and rename file name with extension .pcap)
     a. normal.pcap → contains benign traffic (like browsing or a single connection).
     b. malicious.pcap → contains attacks.
     c.
- **IPS code (mini_ips.py):**

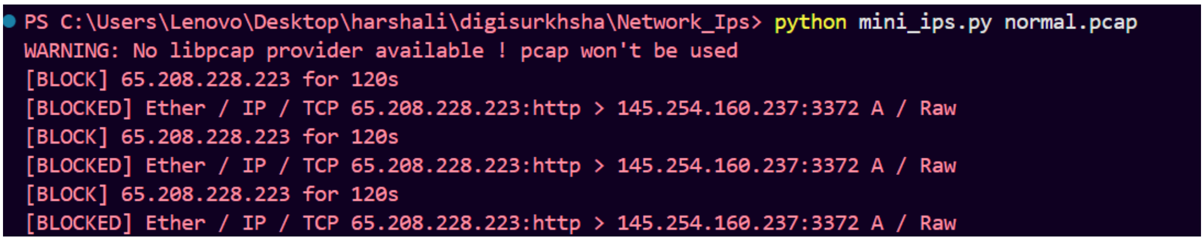
```python
from scapy.all import *
import time
import re
from collections import defaultdict, deque

# Track packets
icmp_counts = defaultdict(lambda: deque())
blocked_ips = {}

# Block rules
ICMP_LIMIT = 10    # max ICMP per second
BLOCK_TIME = 60    # seconds

SQLI_REGEX = re.compile(rb"(union\s+select|or\s+1=1|--)", re.I)

def is_blocked(ip):
    if ip in blocked_ips and blocked_ips[ip] > time.time():
        return True
    return False

def block_ip(ip, sec=BLOCK_TIME):
    blocked_ips[ip] = time.time() + sec
    print(f"[BLOCK] {ip} for {sec}s")

def process_packet(pkt):
    if pkt.haslayer(IP):
        src = pkt[IP].src

        # Check ICMP flood
        if pkt.haslayer(ICMP):
            icmp_counts[src].append(time.time())
            while icmp_counts[src] and icmp_counts[src][0] < time.time() - 1:
                icmp_counts[src].popleft()
            if len(icmp_counts[src]) > ICMP_LIMIT:
                block_ip(src)
                return True

        # Check suspicious payload
        if pkt.haslayer(TCP) and pkt.haslayer(Raw):
            data = bytes(pkt[Raw])
            if SQLI_REGEX.search(data):
                block_ip(src, 120)
                return True
    return False

# Offline mode: read from pcap
def run_pcap(file):
    for pkt in PcapReader(file):
        if process_packet(pkt):
            print(f"[BLOCKED] {pkt.summary()}")

if _name_== "_main_":
    import sys
    if len(sys.argv) == 2:
        run_pcap(sys.argv[1])
    else:
        print("Usage: sudo python3 mini_ips.py <file.pcap>")

```

- **Execution steps:**

  1. Run IPS on normal.pcap

     Normal traffic PCAP → should show no alerts (just maybe a few packets logged).

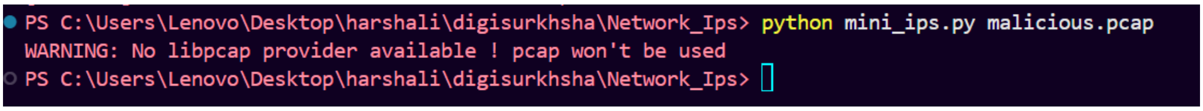
     python mini_ips.py normal.pcap



```
PS C:\Users\Lenovo\Desktop\harshali\digisurkhsha\Network_Ips> python mini_ips.py normal.pcap
WARNING: No libpcap provider available ! pcap won't be used
[BLOCK] 65.208.228.223 for 120s
[BLOCKED] Ether / IP / TCP 65.208.228.223:http > 145.254.160.237:3372 A / Raw
[BLOCK] 65.208.228.223 for 120s
[BLOCKED] Ether / IP / TCP 65.208.228.223:http > 145.254.160.237:3372 A / Raw
[BLOCK] 65.208.228.223 for 120s
[BLOCKED] Ether / IP / TCP 65.208.228.223:http > 145.254.160.237:3372 A / Raw
```

2. Run IPS on malicious.pcap
   Attack traffic PCAP (scans/pings) → should trigger alerts and shows blocking actions.

   python mini_ips.py malicious.pcap



```
PS C:\Users\Lenovo\Desktop\harshali\digisurkhsha\Network_Ips> python mini_ips.py malicious.pcap
WARNING: No libpcap provider available ! pcap won't be used
PS C:\Users\Lenovo\Desktop\harshali\digisurkhsha\Network_Ips>
```

- **Observations:**

1. On normal.pcap → IPS did not block (safe traffic).
2. On malicious/test PCAP → IPS raised block actions against attacker IPs.
3. The IPS was able to differentiate normal vs attack traffic**.**

- **Conclusion:**

1. The IPS successfully simulated real-time prevention by blocking malicious IPs for 120 seconds.
2. It detected and blocked ICMP floods, SYN floods, and SQL injection attempts**.**
3. This PoC shows how Python + Scapy can be used to build a small-scale Network IPS for educational and research purposes.