A

PROJECT REPORT ON

# “Network Packet Capture and Analysis Using Python

SUBMITTED BY

Mast. INJMAM ANSARI - CA/DE1/7097

Under the guidance of

CodeAlpha


CYBERSECURITY

CodeAlpha

CODEALPHA

LUCKNOW - U.P

# Index

# 1. Introduction

In modern computer networks, data is transmitted in the form of packets. Each packet carries important information such as the source address, destination address, protocol type, and payload data. Understanding how these packets travel through a network is essential for networking and cybersecurity professionals, as it helps in monitoring communication, troubleshooting issues, and identifying security threats.

With the rapid growth of internet usage and digital communication, networks have become more complex and vulnerable to various security risks. Packet analysis plays a crucial role in understanding network behavior, detecting anomalies, and ensuring secure data transmission. By examining packet structure and protocol information, one can gain deeper insight into how devices communicate over a network.

This project focuses on building a Python-based program to capture network traffic packets and analyze their structure and content. Python is chosen due to its simplicity and powerful networking libraries. Through this project, real-time packet capturing and analysis are performed to study data flow across different layers of the network.

The project also aims to provide a practical understanding of fundamental network protocols such as TCP, UDP, IP, HTTP, and DNS. By analyzing live network packets, learners can bridge the gap between theoretical networking concepts and real-world network communication.

Overall, this project helps in understanding how data flows through a network, how protocols control communication, and how packet analysis is used in network monitoring and cybersecurity applications.

## 2. Objectives of the Project

The main objectives of this project are:

To build a Python program to capture network traffic packets

To analyze captured packets to understand their structure and content

To learn how data flows through the network

To understand the basics of network protocols

To display useful packet information such as source IP, destination IP, protocol, and payload

## 3. Tools and Technologies Used

**Programming Language:** Python

**Library Used:** Scapy

**Operating System:** Windows / Linux

**Protocols Analyzed:** TCP, UDP, IP, HTTP, DNS

## 4. Methodology

The project captures live network traffic using a Python script. Each captured packet is analyzed layer by layer. The information from different layers such as Ethernet, IP, and Transport layer is extracted and displayed to the user.

The packet analysis follows the TCP/IP model, where data passes through multiple layers before reaching its destination.

## 5. Python Code for Packet Capture and Analysis

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

def analyze_packet(packet):

print("\n================ PACKET DETAILS ================")


# Ethernet Layer

if packet.haslayer(Ether):

eth = packet[Ether]

print(f"Ethernet Layer:")

print(f"  Source MAC      : {eth.src}")

print(f"  Destination MAC : {eth.dst}")


# IP Layer

if packet.haslayer(IP):

ip = packet[IP]

print(f"IP Layer:")

print(f"  Source IP       : {ip.src}")

print(f"  Destination IP  : {ip.dst}")

print(f"  Protocol        : {ip.proto}")
```

```
# TCP Layer
if packet.haslayer(TCP):
tcp = packet[TCP]
print(f"TCP Layer:")
print(f"  Source Port     : {tcp.sport}")
print(f"  Destination Port: {tcp.dport}")
print(f"  Flags           : {tcp.flags}")

# UDP Layer
elif packet.haslayer(UDP):
udp = packet[UDP]
print(f"UDP Layer:")
print(f"  Source Port     : {udp.sport}")
print(f"  Destination Port: {udp.dport}")
```

```
# Payload

if packet.haslayer(Raw):

print("Payload:")

print(packet[Raw].load)

def main():

print("                          Welcome \n ")

print("==============================================")

print("  Python Network Packet Analyzer Started      ")

print("==============================================")

print("Press CTRL+C to stop capturing packets...\n")


sniff(prn=analyze_packet, store=False)


if __name__ == "__main__":

main()
```

## 6. Packet Structure Analysis

Each captured packet consists of multiple layers:

### 6.1 Ethernet Layer

Contains source and destination MAC addresses

Responsible for local network communication

### 6.2 IP Layer

Contains source and destination IP addresses

Responsible for routing packets across networks

### 6.3 Transport Layer

**TCP:** Reliable data transfer with error checking

**UDP:** Fast data transfer without reliability

### 6.4 Application Layer

Contains actual data such as HTTP requests or DNS queries

## 7. Data Flow in a Network

When a user accesses a website:

The browser sends an HTTP request

The request is encapsulated inside a TCP segment

TCP segment is placed inside an IP packet

IP packet is sent as an Ethernet frame

At the destination, the process is reversed

This layered approach ensures reliable and structured data communication.

## 8. Result and Output

The Python program successfully captured live network packets and displayed:

Source and destination MAC addresses

Source and destination IP addresses

Protocol information (TCP/UDP)

Port numbers

Packet payload data

This helped in understanding packet structure and protocol behavior in real-time.

## 9. Snapshots


Python Network Packet Analyzer Started
Press CTRL+C to stop capturing packets...
PACKET DETAILS


PACKET DETAILS
Ethernet Layer:
IP Layer:
UDP Layer:


Apps Places
Dec 10 10:27
Destination Port: 50208
Flags : A
---------- PACKET DETAILS ----------
Ethernet Layer:
Source MAC : 00:50:56:f9:9b:00
Destination MAC : 00:0c:29:f9:2a:05
IP Layer:
Source IP : 34.36.137.203
Destination IP : 192.168.217.130
Protocol : 6
TCP Layer:
Source Port : 443
Destination Port: 50208
Flags : PPA
---------- PACKET DETAILS ----------
Ethernet Layer:
Source MAC : 00:0c:29:f9:2a:05
Destination MAC : 00:50:56:f9:9b:00
IP Layer:
Source IP : 192.168.217.130
Destination IP : 34.36.137.203
Protocol : 6
TCP Layer:
Source Port : 50208
Destination Port: 443
Flags : FA
---------- PACKET DETAILS ----------
Ethernet Layer:
Source MAC : 00:50:56:f9:9b:00
Destination MAC : 00:0c:29:f9:2a:05
IP Layer:
Source IP : 34.36.137.203
Destination IP : 192.168.217.130
Protocol : 6
TCP Layer:
Source Port : 443
Destination Port: 50208
Flags : A

## 10. Conclusion

This project successfully demonstrated the process of capturing and analyzing network packets using Python, providing valuable insight into how data flows across a network. By examining packet structures such as IP headers, protocol types, source and destination addresses, and payload data, the project helped in understanding the working principles of common network protocols like TCP, UDP, and ICMP.

The hands-on implementation enhanced practical knowledge of network monitoring and traffic analysis, which is crucial for identifying performance issues, troubleshooting network problems, and detecting suspicious or malicious activities. Additionally, this project highlighted the importance of packet analysis tools in real-world cybersecurity scenarios, such as intrusion detection and network forensics.

Overall, this project serves as a strong foundation for further exploration in advanced networking and cybersecurity topics. It can be extended in the future by adding features such as real-time traffic visualization, protocol-based filtering, logging packet data to files, and integrating intrusion detection mechanisms. The skills gained through this project are highly relevant for careers in networking, system administration, and ethical hacking.

## 11. Future Scope

This project can be further enhanced in several ways to make it more powerful and applicable to real-world networking and cybersecurity scenarios:

1. **Detection of Malicious Network Traffic**
   Advanced algorithms and rule-based detection techniques can be implemented to identify suspicious or malicious packets such as DoS attacks, port scanning, and packet flooding.
2. **Analysis of Encrypted HTTPS Packets**
   Although HTTPS packets are encrypted, metadata analysis such as packet size, timing, and frequency can be used to understand traffic behavior and detect anomalies.
3. **Automation of Packet Analysis**
   The packet capture and analysis process can be fully automated using scripts and schedulers, enabling continuous network monitoring without manual intervention.
4. **Integration with Visualization Tools**
   The analyzed packet data can be integrated with visualization libraries such as Matplotlib or tools like Wireshark dashboards to present traffic patterns in graphical form.
5. **Real-Time Network Monitoring**
   The project can be extended to support real-time monitoring with alerts for abnormal network activity.
6. **Machine Learning-Based Traffic Classification**
   Machine learning models can be trained to classify normal and abnormal traffic patterns for intelligent intrusion detection.
7. **Cross-Platform Support**
   Enhancing compatibility for Linux, Windows, and macOS systems will make the tool more versatile.
8. **Cloud Network Traffic Analysis**
   The project can be adapted to analyze traffic in cloud environments for better security and performance monitoring.

## 12. References


- Tanenbaum, A. S., & Wetherall, D. J., Computer Networks, Pearson Education.
- Scapy Official Documentation – Python Packet Manipulation Tool.
- Kurose, J. F., & Ross, K. W., Computer Networking: A Top-Down Approach.
- Wireshark User Guide and Documentation.
- Networking Fundamentals – Online Tutorials and Research Articles.
- Python Official Documentation – https://docs.python.org
- Cybersecurity and Network Analysis Research Papers (Online Sources).