CodeAlpha Cybersecurity Internship

Task 1: Basic Network Sniffer – Final Report

Intern: SEKABERA Ibrahim
Task: Task 1 – Basic Network Sniffer
Repository: CodeAlpha_NetworkSniffer
Platform: Kali Linux
Technologies: Python 3, Scapy

EXECUTIVE SUMMARY

Project Status: ✓ COMPLETE AND SUCCESSFUL

Task 1 Requirements Achievement

✓ Developed a Python-based program to capture live network traffic packets

✓ Analyzed captured packets to understand structure, headers, and payloads

✓ Demonstrated understanding of data flow and core networking protocols

✓ Utilized the Scapy library for professional-grade packet capture and analysis

✓ Displayed source/destination IP addresses, ports, protocols, and payload previews

Bonus Achievements

Real-time statistics and packet filtering

Clear demonstration of HTTP vs HTTPS security differences

Analysis of modern web tracking and background network activity

Professional documentation, setup scripts, and usage instructions

Strong educational value with cybersecurity awareness insights

---

TECHNICAL REPORT

Project Overview

This project implements a comprehensive network packet sniffer using Python 3 and the Scapy library. The tool captures, analyzes, and displays network traffic in real time, providing visibility into protocol behavior, packet structure, and the security implications of modern network communications.

The solution was designed for educational and ethical cybersecurity purposes, aligned with CodeAlpha internship objectives.

Technical Implementation

Architecture Overview

```
Network Interface
        ↓
Raw Packet Capture
        ↓
Protocol Analysis
        ↓
Display Output
        ↓
Statistics & Logging (Real-time)
```

Core Components

1. Packet Capture Engine

Library: Scapy (professional packet manipulation framework)

Capture Method: Raw socket capture with optional BPF filtering

Capabilities:

Multi-interface support

Custom protocol filters

Packet counting and logging

2. Protocol Analysis Module

Supported Protocols:

Layer 2: Ethernet, ARP

Layer 3: IP, ICMP

Layer 4: TCP, UDP

Application Layer: HTTP, HTTPS, DNS, DHCP

3. Real-time Display System

Timestamped packet capture

Source and destination IP/port analysis

Safe payload preview (security-aware decoding)

Protocol stack visualization

Live packet statistics

TESTING AND ANALYSIS RESULTS

Test Environment

Operating System: Kali Linux (Virtual Machine)

Network Type: NAT configuration (10.0.2.x subnet)

Interface Used: eth0

DNS Server: 10.0.0.138

Captured Traffic Analysis

1. ICMP Protocol Analysis

Test Command: `ping -c 5 8.8.8.8`

Observed Results:

Echo Request (Type 8, Code 0): `10.0.2.4 → 8.8.8.8`

Echo Reply (Type 0, Code 0): `8.8.8.8 → 10.0.2.4`

Packet Size: 98 bytes (consistent)

Layer Structure: `Ether → IP → ICMP → Raw`

---

2. DNS Protocol Analysis

Test Commands:

```
nslookup google.com
nslookup github.com
```

Observed Results:

Query Types: A (IPv4) and AAAA (IPv6)

DNS Server: `10.0.0.138:53`

Response Size Range: 86–309 bytes

Demonstrated dual-stack (IPv4/IPv6) networking

---

3. HTTP vs HTTPS Security Comparison

HTTP Traffic (Unencrypted):

```
GET / HTTP/1.1

Host: example.com

User-Agent: curl
```

Full request and response content visible

High risk of data exposure

HTTPS Traffic (Encrypted):

```
Encrypted TLS handshake and application data
```

Payload unreadable due to TLS encryption

Demonstrates confidentiality and integrity protection

Critical Finding: HTTP exposes all transmitted data in plain text, while HTTPS protects content using TLS encryption.

---

4. Background Network Activity Analysis

A mixed 30-packet capture revealed:

DHCP lease renewal traffic

ARP address resolution requests

Browser background communication

Advertising and tracking domains, including:

```
easyupload.io

eb2.3lift.com

cookies.nextmillmedia.com

acdn.adnxs.com
```

This highlights the extent of background network activity even during minimal user interaction.

SECURITY IMPLICATIONS AND FINDINGS

1. Protocol Security Assessment

Protocol	Encryption	Risk Level	Notes
HTTP	None	HIGH	Complete data exposure
HTTPS	TLS 1.3	LOW	Metadata only visible
DNS	None	MEDIUM	Queries can be monitored
ARP	None	MEDIUM	Susceptible to spoofing

---

2. Identified Attack Vectors

Man-in-the-Middle attacks on HTTP traffic

DNS query monitoring and profiling

Traffic pattern analysis revealing user behavior

ARP spoofing attacks on local networks

---

3. Defensive Recommendations

Enforce HTTPS for all web communications

Use DNS over HTTPS (DoH) or DNS over TLS

Employ VPNs on untrusted networks

Apply network segmentation for critical assets

EDUCATIONAL VALUE AND SKILLS DEVELOPED

Technical Skills

Network Protocol Knowledge

TCP/IP stack understanding

Packet structure and flow analysis

Python Programming

Advanced use of Scapy

Modular and object-oriented design

Error handling and logging

Cybersecurity Awareness

Identification of attack vectors

Evaluation of protocol security

Privacy and data protection considerations

CONCLUSION

This project successfully demonstrates strong practical understanding of networking fundamentals, packet analysis, and cybersecurity principles. The developed network sniffer effectively captures and analyzes real-world traffic, emphasizing the importance of encryption and secure communication.

Key Learning Outcomes

Deep understanding of network protocols from Layer 2 to Application Layer

Increased security awareness regarding unencrypted traffic

Hands-on experience with real-time traffic analysis

Professional documentation and ethical cybersecurity practices

Prepared by: Sekabera Ibrahim
CodeAlpha Cybersecurity Internship – Task 1