COMP 7003
Assignment 2
Report

Parth Chaturvedi
A01256537
09/29/2025

Table of Contents

1. Purpose

This script demonstrates network packet capture and analysis using the Scapy library. The tool captures real live network traffic and then manually parses protocol headers from raw hexadecimal data dumps. The script can handle multiple network protocols including Ethernet, ARP, IPv4, IPv6, ICMP, ICMPv6, TCP, UDP, and DNS. The output of the script is the parsed hex data to the correct protocol with all its fields displayed.

---

2. Requirements

Task	Status
Capture packets through Scapy	Fully implemented
Manually parse Ethernet headers	Fully implemented
Manually parse ARP packet fields	Fully implemented
Manually parse IPv4 packet fields	Fully implemented
Manually parse IPv6 packet fields	Fully implemented
Manually parse ICMP packet fields	Fully implemented
Manually parse ICMPv6 packet fields	Fully implemented
Manually parse TCP packet fields	Fully implemented
Manually parse UDP packet fields	Fully implemented
Manually parse DNS packet fields	Fully implemented

---

3. Platforms

This packet analysis tool has been tested on and works on:

- **Manjaro Linux** (primary development environment)
- **MacOS**
- **Windows**

The tool requires root/administrator privileges for packet capture operations on all platforms.

4. Language

Programming Language: Python 3.10+

Required Libraries:

- `scapy` - For packet capture functionality
- `psutil` - For network interface enumeration
- `argparse` - For command-line argument parsing

5. Findings

The development of this packet analysis tool provided valuable insights into network protocol structures and the challenges of low-level packet parsing. By manually extracting fields from hex dump data rather than relying on any built-in dissection functions I gained a deep understanding of how protocols are layered..

One of the most significant learning experiences was implementing the protocol routing logic. Each protocol layer must correctly identify and parse the next layer based on some fields like EtherType for Ethernet, Protocol field for IPv4, Next Header for IPv6, and port numbers for application protocols. This layered approach helped better my understanding of the OSI model and how encapsulation works in practice.

Parsing TCP was particularly challenging due to the complexity of its header structure. TCP includes basic fields like ports and sequence numbers, but also a bunch of flags for connection management (SYN, ACK, FIN, RST, PSH, URG) and a variable-length header with optional fields. Understanding how to extract individual bits from the flags byte and interpret the data offset field to handle options was complete but once I understood the interpretation it was easy to understand.

Throughout this project, the importance of bitwise operations became clear. Extracting individual flags required right-shift operations and masking with bitwise AND, while handling multi-byte fields required proper conversion from hexadecimal strings to integers.

The project successfully achieved all assignment objectives, implementing complete parsers for 8 network protocols with proper field extraction and display formatting. The experience of building this tool rather than using existing parsing libraries, provided hands-on understanding of network protocols that will be beneficial for future work in network programming.