COMP 7003
Assignment 2
User Guide

Parth Chaturvedi
A01256537
09/29/2025

1. Purpose

This packet capture and analysis tool uses Scapy to capture network traffic and manually parse protocol headers from hex data. This program supports ARP, IPv4, IPv6, ICMP, ICMPv6, TCP, UDP, and DNS protocols, allowing analyzing network packets and displaying detailed information in both hexadecimal and decimal formats.

---

2. Installing

Obtaining

```
git clone https://github.com/ParthCv/Hex-Code-Dump-Packet-Analysis.git
```

Building

```
cd Hex-Code-Dump-Packet-Analysis
# Install required packages
pip install scapy psutil
```

Running

```
sudo python3 main.py -i <interface> -c <count> -f <filter>
```

---

3. Command Line Arguments

The program accepts the following arguments:

Argument	Purpose	Default
`-h / --help`	Display help message	N/A
`-i / --interface`	Interface to capture on (e.g., eth0, wlan0, any)	**Required**
`-c / --count`	Number of packets to capture	**Required**

`-f /` `--filter`	Filter expression (e.g., 'tcp', 'udp and port 53')	None (captures all)

Examples:

```
# Capture 5 ICMP packets on any interface
sudo python3 main.py -i any -c 5 -f icmp

# Capture 1 ARP packet on eth0
sudo python3 main.py -i eth0 -c 1 -f arp

# Capture 10 TCP packets
sudo python3 main.py -i any -c 10 -f tcp
```

---

4. Features

- **Multi-protocol support**: Parses ARP, IPv4, IPv6, ICMP, ICMPv6, TCP, UDP, and DNS
- **Hex parsing**: Extracts fields directly from hexadecimal data without relying on Scapy's built-in dissection
- **Packet Filtering**: Supports packet filtering for targeted packet capture
- **Multiple interfaces**: Can capture on specific interfaces or all interfaces simultaneously
- **Protocol routing**: Routes packets through appropriate protocol parsers based on header fields
- **Detailed output**: Displays packet fields in both hex and decoded formats

---

5. Limitations

Platform-Specific Issues:

- **Interface names vary by OS**:
  - Linux: `eth0`, `wlan0`, `eno1`
  - macOS: `en0`, `en1`, `lo0`
  - Windows: Long device paths like `\Device\NPF_{GUID}`

Filter Syntax:

- Complex filters must be quoted: `-f "tcp and port 80”`
- Some filters may not work on all interfaces (e.g., `arp` on loopback)

Known Issues:

- IPv6 capture may be limited on some systems
- Loopback interface may not capture all traffic types

Unsupported Features:

- Does not parse TCP options beyond basic header
- Does not decode DNS question/answer sections (only header)
- Does not reassemble fragmented packets

---

6. Examples

Example 1: Capture ARP Traffic

```
# Terminal 1: Start capture
sudo python3 main.py -i any -c 1 -f arp

# Terminal 2: Generate ARP traffic
sudo arping -c 1 192.168.1.1
```

Example 2: Capture ICMP (Ping)

bash

```
sudo python3 main.py -i any -c 1 -f icmp
# Then: ping -c 1 8.8.8.8
```

Example 3: Capture DNS over UDP

bash

```
sudo python3 main.py -i any -c 1 -f "udp and port 53"
# Then: nslookup google.com
```

Example 4: Capture TCP Traffic

bash

```
sudo python3 main.py -i any -c 1 -f tcp
# Then: curl http://example.com
```

Example 5: Test Fragmentation

bash

```
sudo python3 main.py -i any -c 3 -f ip
# Then: ping -c 1 -s 2000 8.8.8.8
```

Example 6: Invalid Arguments

bash

```
# Missing required argument
python3 main.py -i any
# Error: the following arguments are required: -c/--count

# Invalid interface
sudo python3 main.py -i invalid_interface -c 1
# Error: Interface 'invalid_interface' not found
```

Example 7: Using "any" Interface

bash

```
sudo python3 main.py -i any -c 5
# Captures on all available non-loopback interfaces
```

Troubleshooting

No packets captured:

- Ensure you're running with sudo/root privileges
- Check that the interface is active and has traffic, check with wireshark to confirm
- Try using `-i any` to capture on all interfaces
- Generate traffic while capture is running
- Try disabling the firewall for a while

Permission denied:

- Always run with `sudo` on Linux/macOS
- Run as Administrator on Windows, in IDEs even launching them as admin resulted in no captures detected