Course	COMP 7003
Program	Bachelor of Science in Applied Computer Science
Term	January 2024

- This is an individual [programming](#) assignment.

Objective

- Develop the ability to capture and analyze network traffic at the packet level using Scapy in Python.
- Understand packet structures by converting raw data to a hex dump, parsing the fields of various network layers, and displaying the results in a clear and organized format.

Learning Outcomes

- Technical Skill in Packet Analysis: Gain proficiency in capturing, filtering, and analyzing network packets using Scapy and Python.
- Understanding Network Protocols: Strengthen knowledge of Ethernet, IPv4, ICMP, TCP, and UDP protocols by dissecting packet fields and identifying key components.
- Data Handling and Parsing: Improve the ability to convert raw packet data into a human-readable form (e.g., hex dumps) and extract relevant header information.
- Software Development Practice: Enhance coding skills and experience implementing structured, maintainable Python solutions that meet specified requirements.
- Problem-Solving and Debugging: Develop the capacity to troubleshoot, refine, and test code to ensure accurate packet capture and analysis.

Assignment Details

- You will receive a starter code template that uses Scapy to capture network traffic.
- Your task is to:
    - Capture packets on a specified interface.
    - Filter and identify packets using the required protocols: Ethernet, IPv4, ICMP, TCP, and UDP.
    - Convert each captured packet into a hex dump.
    - Parse the packet from the hex dump and display its fields in a format matching the provided reference screenshots.
    - Test your program with multiple packets for each protocol type, ensuring it accurately identifies and displays their details.
    - Screenshots of the expected output are at the end of this document.

- ○ The starting source code is provided.

Requirements

- Protocols: Must support Ethernet, IPv4, IPv6, ICMP, ICMPv6, TCP, UDP, and DNS.
- Hex Dump: Implement functionality to produce a hex dump of each packet's raw data.
- Field Extraction: Accurately parse and display relevant fields (e.g., source/destination MAC and IP addresses, protocol fields, source/destination ports for TCP/UDP details).
- Output Format: Match the style and clarity of the provided screenshots. Maintain consistent and organized formatting.
- Code Quality: Write clean, commented code that follows best practices in Python programming.
- Testing: Collect and analyze multiple packets from each supported protocol to verify that your program works correctly.

Constraints

- Ensure your code runs on the lab environment's standard Python installation with Scapy pre-installed.

Resources

- Official Scapy [documentation](#)
- Course materials and lecture notes on packet structure and protocols
- Provided starter code and reference screenshots

Submission

- Ensure your submission meets all the [guidelines](#), including formatting, file type, and [submission](#).
- Follow the [AI usage guidelines](#).
- Be aware of the [late submission policy](#) to avoid losing marks.
- ***Note: Please strictly adhere to the submission requirements to ensure you don't lose any marks.***

Evaluation

Topic	Value
Correct Output	50%
Design	25%

Testing	25%
Total	100%

Hints

- Review Scapy's layer structure and methods for filtering and dissecting packets. Understanding show() and layer fields will help determine which fields to print.
- Use the provided starter code as a template. Focus on integrating your parsing logic into the given structure.
- Experiment with capturing traffic from different sources (pinging hosts, making DNS queries, etc.) to generate test packets.
- Start testing and debugging early. Make incremental changes and confirm that each protocol layer parses correctly before moving on.
- To capture and generate traffic:

Protocol	Filter	Capture Command	Traffic Generation Command
ARP	arp	`sudo python3 main.py -i any -c 1 -f arp`	`arping -c 1 <IP_ADDRESS>`
UDP	udp	`sudo python3 main.py -i any -c 1 -f udp`	`echo "Hello, World!"	ncat --udp 192.168.0.1 12345`
TCP	tcp	`sudo python3 main.py -i any -c 1 -f tcp`	`curl http://<IP_ADDRESS> or telnet <IP_ADDRESS> <PORT>`
ICMP	icmp	`sudo python3 main.py -i any -c 1 -f icmp`	`ping -c 1 <IP_ADDRESS>`

ARP

```
● ● ●                    📁 assign-2 — -zsh — 95×26
(.venv) ds@chaos assign-2 % sudo python3 main.py -i any -c 1 -f arp
Password:
Available interfaces: ['lo0', 'gif0', 'stf0', 'anpi1', 'anpi2', 'anpi0', 'en4', 'en5', 'en6', '
en1', 'en2', 'en3', 'ap1', 'en0', 'bridge0', 'awdl0', 'llw0', 'utun0', 'utun1', 'utun2', 'utun3
', 'utun4', 'utun5', 'utun6', 'utun7', 'utun8', 'utun9']
Starting packet capture on en0
Starting packet capture on en0 with filter: arp

Captured Packet 1:
Ethernet Header:
  Destination MAC:        ffffffffffff        | ff:ff:ff:ff:ff:ff
  Source MAC:             cc96e52a1ea5        | cc:96:e5:2a:1e:a5
  EtherType:              0806                | 2054
ARP Header:
  Hardware Type:          0001                | 1
  Protocol Type:          0800                | 2048
  Hardware Size:          06                  | 6
  Protocol Size:          04                  | 4
  Operation:              0001                | 1
  Sender MAC:             cc96e52a1ea5        | cc:96:e5:2a:1e:a5
  Sender IP:              c0a80014            | 192.168.0.20
  Target MAC:             000000000000        | 00:00:00:00:00:00
  Target IP:              c0a80062            | 192.168.0.98
Packet capture completed on en0.
(.venv) ds@chaos assign-2 % ▊
```

UDP

```
● ● ●                    📁 assign-2 — -zsh — 95×32
[(.venv) ds@chaos assign-2 % sudo python3 main.py -i any -c 1 -f udp
Available interfaces: ['lo0', 'gif0', 'stf0', 'anpi1', 'anpi2', 'anpi0', 'en4', 'en5', 'en6', '
en1', 'en2', 'en3', 'ap1', 'en0', 'bridge0', 'awdl0', 'llw0', 'utun0', 'utun1', 'utun2', 'utun3
', 'utun4', 'utun5', 'utun6', 'utun7', 'utun8', 'utun9']
Starting packet capture on en0
Starting packet capture on en0 with filter: udp

Captured Packet 1:
Ethernet Header:
  Destination MAC:          dab3701e949f            | da:b3:70:1e:94:9f
  Source MAC:               e2842607c9b9            | e2:84:26:07:c9:b9
  EtherType:                0800                    | 2048
IPv4 Header:
  Version:                  4                       | 4
  Header Length:            5                       | 20 bytes
  Total Length:             0039                    | 57
  Flags & Frag Offset:      0000                    | 0b0
    Reserved:            0
    DF (Do not Fragment): 0
    MF (More Fragments): 0
    Fragment Offset:     0x0 | 0
  Protocol:                 11                      | 17
  Source IP:                c0a8003f                | 192.168.0.63
  Destination IP:           8efb216a                | 142.251.33.106
UDP Header:
  Source Port:              cd22                    | 52514
  Destination Port:         01bb                    | 443
  Length:                   0025                    | 37
  Checksum:                 2de5                    | 11749
  Payload (hex):            42ea88b1358becb33db421363f10b88fd0bf62dbb683a519cd1566e08a
Packet capture completed on en0.
(.venv) ds@chaos assign-2 % ▊
```

TCP

```
● ● ●                        📁 assign-2 — -zsh — 95×48

(.venv) ds@chaos assign-2 % sudo python3 main.py -i any -c 1 -f tcp
Available interfaces: ['lo0', 'gif0', 'stf0', 'anpi1', 'anpi2', 'anpi0', 'en4', 'en5', 'en6', '
en1', 'en2', 'en3', 'ap1', 'en0', 'bridge0', 'awdl0', 'llw0', 'utun0', 'utun1', 'utun2', 'utun3
', 'utun4', 'utun5', 'utun6', 'utun7', 'utun8', 'utun9']
Starting packet capture on en0
Starting packet capture on en0 with filter: tcp

Captured Packet 1:
Ethernet Header:
  Destination MAC:         e2842607c9b9         | e2:84:26:07:c9:b9
  Source MAC:              dab3701e949f         | da:b3:70:1e:94:9f
  EtherType:               0800                 | 2048
IPv4 Header:
  Version:                 4                    | 4
  Header Length:           5                    | 20 bytes
  Total Length:            007d                 | 125
  Flags & Frag Offset:     da79                 | 0b1101101001111001
    Reserved:        1
    DF (Do not Fragment): 1
    MF (More Fragments): 0
    Fragment Offset:     0x1a79 | 6777
  Protocol:                06                   | 6
  Source IP:               8efbd3e6             | 142.251.211.230
  Destination IP:          c0a8003f             | 192.168.0.63
TCP Header:
  Source Port:             01bb                 | 443
  Destination Port:        c0cc                 | 49356
  Sequence Number:         da96f530             | 3667326256
  Acknowledgment Number:   44a0227a             | 1151345274
  Data Offset:             8                    | 32 bytes
  Reserved:                0b0                  | 0
  Flags:                   0b000011000          | 24
    NS:              0
    CWR:             0
    ECE:             0
    URG:             0
    ACK:             1
    PSH:             1
    RST:             0
    SYN:             0
    FIN:             0
  Window Size:             041a                 | 1050
  Checksum:                0807                 | 2055
  Urgent Pointer:          0000                 | 0
  Payload (hex):           1703030044190968a9a1df104f6472e4949de47ec4fd0a606630b77a96b052fd41f
02b5ce5839a8f06ca3d200ce0ed36302f65114df9b7becae0b0819df1ab9696a1383d97cbfc7cd7
Packet capture completed on en0.
(.venv) ds@chaos assign-2 % ▮
```

ICMP

```
(.venv) ds@chaos assign-2 % sudo python3 main.py -i any -c 1 -f icmp
Available interfaces: ['lo0', 'gif0', 'stf0', 'anpi1', 'anpi2', 'anpi0', 'en4', 'en5', 'en6', '
en1', 'en2', 'en3', 'ap1', 'en0', 'bridge0', 'awdl0', 'llw0', 'utun0', 'utun1', 'utun2', 'utun3
', 'utun4', 'utun5', 'utun6', 'utun7', 'utun8', 'utun9']
Starting packet capture on en0
Starting packet capture on en0 with filter: icmp

Captured Packet 1:
Ethernet Header:
  Destination MAC:       6c5ab03de75c          | 6c:5a:b0:3d:e7:5c
  Source MAC:            ea6f69a682c7          | ea:6f:69:a6:82:c7
  EtherType:             0800                  | 2048
IPv4 Header:
  Version:               4                     | 4
  Header Length:         5                     | 20 bytes
  Total Length:          0054                  | 84
  Flags & Frag Offset:   ebde                  | 0b1110101111011110
    Reserved:            1
    DF (Do not Fragment): 1
    MF (More Fragments): 1
    Fragment Offset:     0xbde | 3038
  Protocol:              01                    | 1
  Source IP:             c0a800f1              | 192.168.0.241
  Destination IP:        ac43c328              | 172.67.195.40
ICMP Header:
  Type:                  08                    | 8
  Code:                  00                    | 0
  Checksum:              464d                  | 17997
  Payload (hex):         d0790b29678fd9a10002a9d908090a0b0c0d0e0f101112131415161718191a1b1c1
d1e1f202122232425262728292a2b2c2d2e2f3031323334353637
Packet capture completed on en0.
(.venv) ds@chaos assign-2 %
```