HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

BÀI TẬP LỚN

Môn: Kĩ Thuật Theo Dõi Và Giám Sát An Toàn Mạng

Đề Tài: Xây Dựng Một Công Cụ Phân Tích Gói Tin Sử Dụng Python

Giảng Viên : **Ninh Thị Thu Trang**

Nhóm Sinh Viên : **Hoàng Thạch Hùng - B20DCAT074**

Nguyễn Huy Hoàng - B20DCAT070

Trần Quốc Huy - B20DCAT086

Nguyễn Viết Hoàng Huy - B20DCAT082

Nhóm btl : **05**

Mã đề : **09**

Hà Nội, tháng 3 năm 2024

MỤC LỤC

- **Nội dung cần tìm hiểu:**

Tìm hiểu lý thuyết các giao thức, phân tích các thành phần header trong gói tin của các giao thức, hàm phân tích gói tin giao thức, các thư viện phân tích dữ liệu trong python.

- UDP Header
- TCP Header
- IP Header
- Ethernet Header.

- **Bảng phân công nhiệm vụ:**

STT	Họ tên	Mã SV	Nhiệm vụ	Deadline
1	**Hoàng Thạch Hùng**	**B20DCAT074**	**- UDP - Demo - Các thư viện**	**12h00 15/3(trưa)**
2	**Nguyễn Huy Hoàng**	**B20DCAT070**	**- TCP - Slide - Các thư viện**	**12h00 15/3(trưa)**
3	**Trần Quốc Huy**	**B20DCAT086**	**- IP - Báo cáo - Các thư viện**	**12h00 15/3(trưa)**
4	**Nguyễn Viết Hoàng Huy**	**B20DCAT082**	**- Ethernet - Demo - Các thư viện**	**12h00 15/3(trưa)**

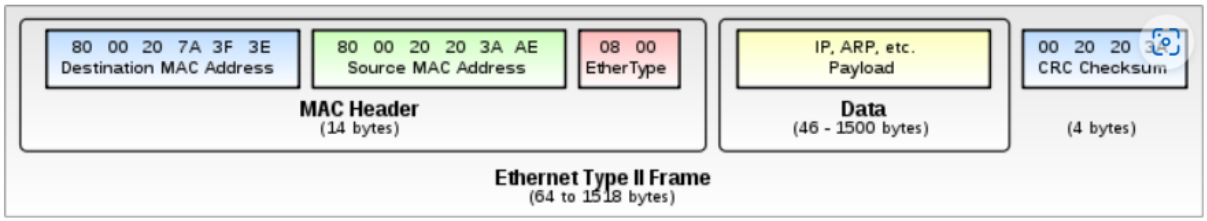
I. Tổng quan về lí thuyết giao thức:

A. Giới thiệu:

Trong phần lí thuyết giao thức, chúng ta sẽ tìm hiểu sơ lược về lí thuyết các header của gói tin giao thức UDP, TCP, IP và Ethernet. Các kiến thức được súc tích, gọn, nhằm đặt nền tảng vững chắc cho phần tổng quan về kĩ thuật khi phân tích cách lập trình bắt nội dung gói tin.

B. Ethernet Header:

Một gói tin được đóng gói từ tầng ứng dụng, sau đó xuống các tầng thấp hơn. Do đó khi bóc tách gói tin, phần đầu tiên chính là tiêu đề của giao thức cuối cùng được thêm vào ở tầng liên kết dữ liệu. Giao thức liên kết dữ liệu phổ biến nhất là Ethernet. Do đó, phần đầu tiên này thường là một tiêu đề Ethernet.



Cấu trúc của 1 khung Ethernet:

STT	Tên thành phần	Độ dài thành phần	Đặc điểm, chức năng
1	Destination MAC Address	6 bytes	Trường này cho biết địa chỉ MAC của máy đích.
2	Source MAC Address	6 bytes	Trường này cho biết địa chỉ MAC của máy nguồn.
3	Type	2 bytes	Trường này chỉ ra giao thức sau phần tiêu đề Ethernet.
4	Payload	45 - 1500 bytes	Chứa các thông tin được đóng gói ở các tầng phía trên.
5	CRC Checksum	4 bytes	Sử dụng thuật toán CRC để kiểm tra lỗi nhằm đảm bảo tính toàn vẹn của dữ liệu.

Ở đây phần tiêu đề Ethernet là 14 bytes đầu tiên.

C. IP Header:

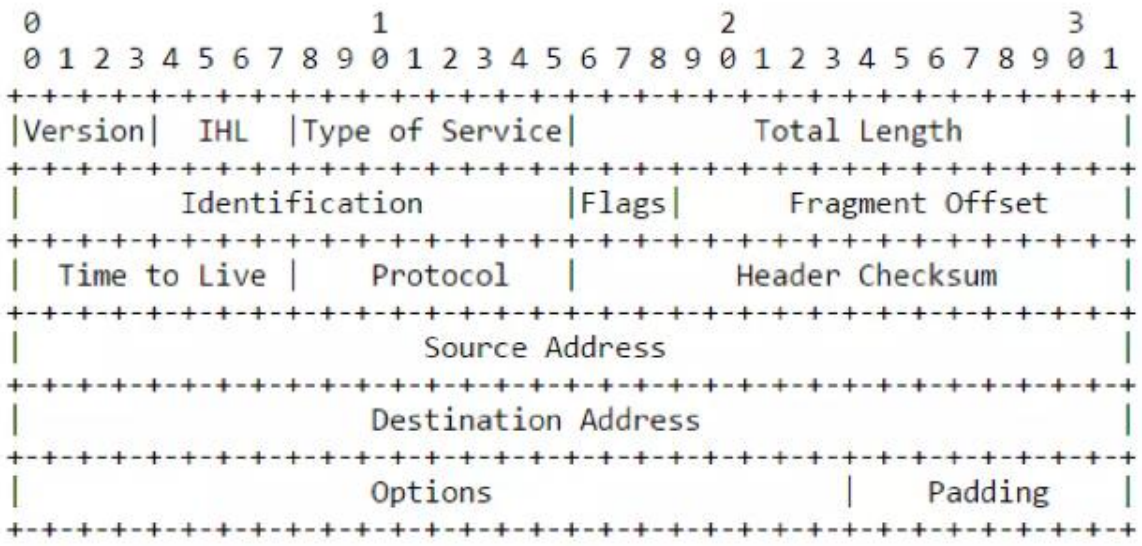
Giao thức IP là một giao thức của chồng giao thức TCP/IP thuộc tầng mạng.

Đặc điểm của giao thức IP:

- Là 1 trong những giao thức quan trọng nhất của bộ giao thức TCP/IP.
- Là giao thức hướng không liên kết (connectionless): dữ liệu của IP được truyền đi ngay lập tức nếu có thể (best effort), không có bất kì cơ chế thiết lập kết nối, không có cơ chế báo nhận hay điều khiển luồng nào được sử dụng với IP, các gói tin IP cũng không được đánh số thứ tự khi trao đổi trên mạng…
- Mỗi gói tin IP được xử lý một cách hoàn toàn độc lập với các gói tin IP khác.
- Giao thức IP sử dụng cơ chế định địa chỉ theo kiểu phân cấp, trong đó phần NetworkId của địa chỉ giống như tên của một con đường và phần hostId của địa chỉ sẽ là số nhà của một căn nhà trên con đường ấy.
- Không có cơ chế khôi phục lại gói tin bị mất trên đường truyền. Việc này được giao lại cho các giao thức tầng trên để đảm bảo độ tin cậy (TCP).

Cấu trúc gói tin:

Gồm 2 phần là Header và data. *Header* chứa thông tin quản lý của gói tin, *data* là phần dữ liệu cần truyền tải được đóng gói trong gói tin IP.



Example Internet Datagram Header

Figure 4.

Dựa trên hình minh họa, header của gói tin trong giao thức IP được phân tích như sau:

STT	Tên thành phần	Độ dài thành phần	Đặc điểm
1	Version	4 bits	Trường version cho biết phiên bản của giao thức IP.
2	IHL(Internet Headers Length)	4 bits	Chiều dài tiêu đề Internet là chiều dài của tiêu đề internet bằng các từ 32 bits, và do đó chỉ ra sự bắt đầu của dữ liệu. Lưu ý rằng giá trị tối thiểu cho 1 tiêu đề chính xác là 5
3	Total Length	16 bits	Tổng chiều dài là độ dài của datagram, được đo bằng octet, bao gồm tiêu đề internet và dữ liệu. Trường này cho phép độ dài của gói tin có thể lên tới 65.535 octet. Các datagram dài như vậy là không thực tế đối với hầu hết các máy chủ và mạng.
4	Identification	16 bits	Một giá trị nhận dạng được chỉ định bởi người gửi để giúp lắp ráp các mảnh của 1 gói tin.
5	Flags	3 bits	Các cờ điều khiển khác nhau. • Bit 0: dữ trữ, phải bằng 0 • Bit 1:(DF) 0 = Có thể phân mảnh, 1= Không phân mảnh. • Bit 2: (MF) 0 = Đoạn cuối, 1 = Các phân đoạn khác.
6	Protocol	8 bits	Trường này cho biết giao thức mức tiếp theo được sử dụng trong phần dữ liệu của gói tin internet.
7	Header Checksum	16 bits	Checksum chỉ trên tiêu đê. Do một số trường tiêu đề thay đổi (ví dụ: Time to live), tính năng này được tính lại và xác minh tại mỗi điểm mà tiêu đề internet được xử lý.
8	Source Address	32 bits	Địa chỉ nguồn
9	Destination Address	32 bits	Địa chỉ đích

10	Padding	Tùy biến	Phần đệm Internet header được sử dụng để đảm bảo đầu cuối của internet header kết thúc trên ranh giới 32 bit. Phần đệm có giá trị 0.

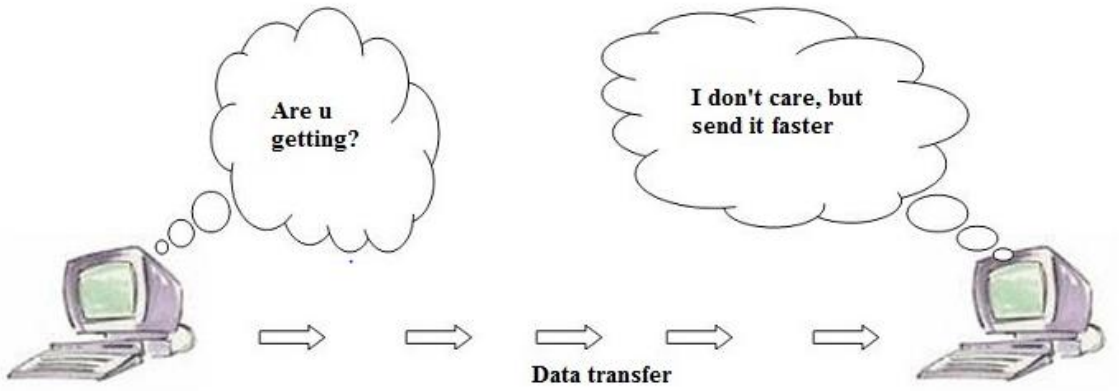
D. UDP Header:

Khái niệm về giao thức UDP

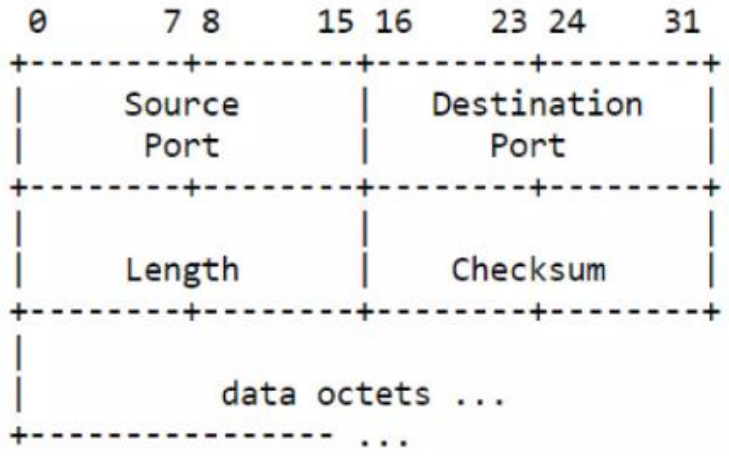
Giao thức UDP (User Datagram Protocol) là giao thức cung cấp gói tin datagram cho việc giao tiếp giữa các máy tính chuyển mạch trong môi trường của một tập hợp các mạng máy tính kết nối. Giao thức này giả định rằng Internet Protocol (IP) được sử dụng làm giao thức cơ bản.

UDP không cung cấp sự tin cậy và thứ tự truyền nhận mà TCP làm; các gói dữ liệu có thể đến không đúng thứ tự hoặc bị mất mà không có thông báo. Tuy nhiên UDP nhanh và hiệu quả hơn đối với các mục tiêu như kích thước nhỏ và yêu cầu khắt khe về thời gian. Do bản chất không trạng thái của UDP nên UDP hữu dụng đối với việc trả lời các truy vấn nhỏ với số lượng lớn người yêu cầu.

UDP



Cấu trúc gói tin của UDP:



User Datagram Header Format

Dựa trên hình mình hoạ, header của gói tin trong giao thức UDP được phân tích như sau:

STT	Tên thành phần	Độ dài thành phần	Đặc điểm, chức năng
1	Source Port	16 bits	Là một trường tuỳ chọn, nó cho biết cổng của thiết bị gửi. Trường này có thể đặt là 0 nếu máy tính đích đến không cần trả lời người gửi.
2	Destination Port	16 bits	Nó cho biết cổng mà diagram được gửi đến
3	Length	16 bits	Xác định chiều dài của toàn bộ datagram: phần header và dữ liệu. Chiều dài tối thiểu là 8 byte khi gói tin không có dữ liệu, chỉ có header.
4	Checksum	16 bits	Là thuật toán kiểm tra lỗi đảm bảo tính toàn vẹn của dữ liệu

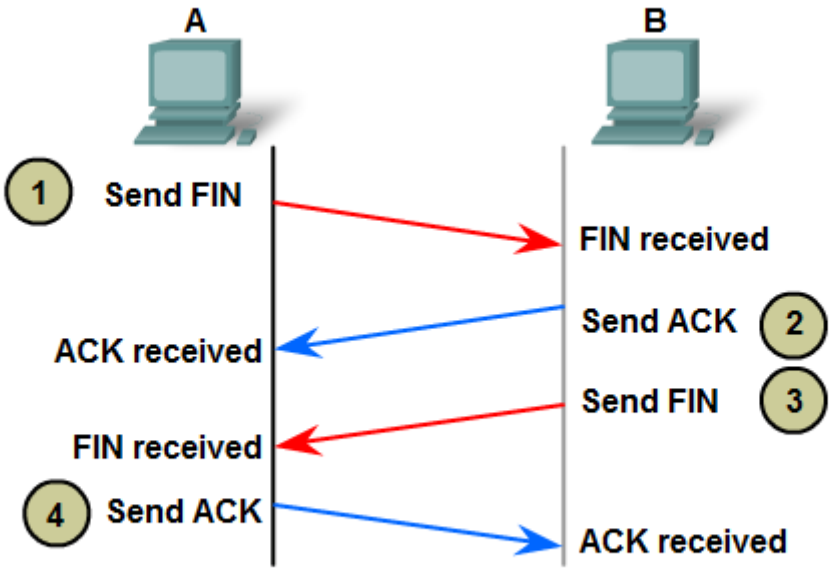
E. TCP Header:

Khái niệm về giao thức TCP

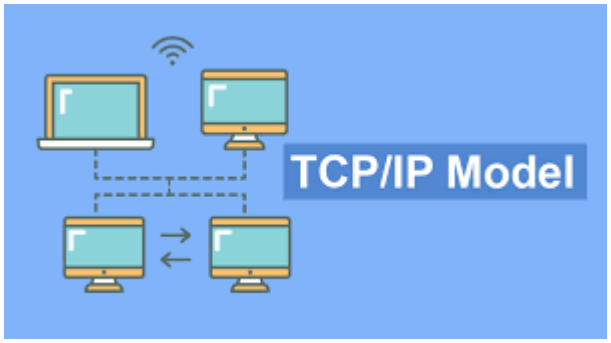
TCP là một giao thức kết nối đường dẫn, điều này có nghĩa là trước khi truyền dữ liệu, hai thiết bị trên mạng cần thiết lập một kết nối TCP. Quá trình này bao gồm ba bước: thiết lập kết nối (connection establishment), truyền dữ liệu, và kết thúc kết nối (connection termination).

TCP đảm bảo tính tin cậy của dữ liệu bằng cách sử dụng các cơ chế như xác nhận (acknowledgment), số thứ tự (sequence number), và kiểm tra hợp pháp (checksum). Nó đảm bảo rằng dữ liệu được truyền được nhận đúng thứ tự và không bị mất hoặc bị lỗi trong quá trình truyền.

TCP Connection Establishment and Termination



A sends ACK response to B.



Cấu trúc gói tin của TCP:

+	Bit 0 - 3	4 - 9	10 - 15	16 - 31
0	Source Port			Destination Port
32	Sequence Number			
64	Acknowledgement Number			
96	Data Offset	Reserved	Flags	Window
128	Checksum			Urgent Pointer
160	Options (optional)			
160/192+	Data			

Dựa trên hình mình hoạ, header của gói tin trong giao thức TCP được phân tích như sau:

STT	Tên thành phần	Độ dài thành phần	Đặc điểm, chức năng
1	Source Port	16 bits	Xác định cổng nguồn của gói tin, tức là cổng từ thiết bị gửi dữ liệu.
2	Destination Port	16 bits	Xác định cổng đích của gói tin, tức là cổng mà dữ liệu sẽ được gửi đến.
3	Sequence number	32 bits	Xác định số thứ tự của byte đầu tiên trong dữ liệu được truyền qua gói tin.
4	Acknowledgment number	32 bits	Xác định số thứ tự mà thiết bị nhận được mong đợi nhận được tiếp theo từ thiết bị gửi.
5	Data Offset	4 bits	Xác định kích thước của tiêu đề TCP, thường được tính bằng số lượng từ 32-bit trong tiêu đề.
6	Reserved	6 bits	đặt trước để dùng về sau, nên để 0.

7	Flags	6 bits	Bao gồm các cờ như URG, ACK, PSH, RST, SYN, và FIN, được sử dụng để quản lý các trạng thái và thao tác của kết nối TCP.
8	Window	16 bits	Xác định kích thước của cửa sổ trượt, cho biết số lượng byte mà thiết bị gửi có thể gửi mà không cần phải chờ phản hồi xác nhận từ thiết bị đích.
9	Checksum	16 bits	Giá trị kiểm tra hợp pháp được sử dụng để kiểm tra tính toàn vẹn của dữ liệu trong gói tin.
10	Urgent Pointer	16 bits	Chỉ ra vị trí của dữ liệu khẩn cấp trong gói tin.
11	Options	Biến 0-320 bits, chia hết cho 32	Có thể bao gồm các tùy chọn bổ sung như tùy chỉnh kích thước cửa sổ, mã hóa, hoặc timestamp.

II. Tổng quan về kĩ thuật:

A. Giới thiệu:

Trong phần tổng quan về kĩ thuật, chúng ta sẽ tìm hiểu về việc áp dụng ngôn ngữ lập trình Python vào việc bắt và phân tích các gói tin.

B. Giới thiệu về Python

Các thư viện được sử dụng trong chương trình Python:

STT	Tên thư viện	Chức năng
1	struct	Thư viện struct trong Python cung cấp các chức năng để thực hiện việc đóng gói và giải nén các dữ liệu cấu trúc thành và từ chuỗi byte, dựa trên một định dạng cụ thể.
2	socket	Thư viện socket trong Python là một phần của thư viện tiêu chuẩn (standard library) của Python và được sử dụng để thực hiện các kết nối mạng, gửi và nhận dữ liệu qua mạng sử dụng giao thức TCP/IP hoặc UDP. Cụ thể, nó cung cấp các công cụ để tạo ra các socket, làm việc

		với các địa chỉ IP, cổng, và thực hiện các hoạt động liên quan đến mạng như kết nối, gửi và nhận dữ liệu qua mạng.
3	textwrap	Thư viện textwrap trong Python cung cấp các công cụ để xử lý văn bản và định dạng các đoạn văn bản theo các quy tắc cụ thể

C. Lập trình bắt và phân tích các gói tin:

1. Hàm phân tích gói tin giao thức UDP:

```
def udp_segment(data):
    src_port, dest_port, length, checksum = struct.unpack('! H H H H', data[:8])
    return src_port, dest_port, length, checksum, data[8:]
```

2. Hàm phân tích gói tin giao thức TCP:

```
def tcp_segment(data):
    src_port, dest_port, seq, ack, o_r_f, window, checksum, urg_ptr = (
        struct.unpack('! H H L L H H H H', data[:20]))
    offset = (o_r_f >> 12) * 4
    flag_urg = (o_r_f & 32) >> 5
    flag_ack = (o_r_f & 16) >> 4
    flag_psh = (o_r_f & 8) >> 3
    flag_rst = (o_r_f & 4) >> 2
    flag_syn = (o_r_f & 2) >> 1
    flag_fin = o_r_f & 1
    return (src_port, dest_port, seq, ack, offset, flag_urg, flag_ack, flag_psh, flag_rst, flag_syn, flag_fin,
            checksum, urg_ptr, data[offset:])
```

3. Hàm phân tích gói tin giao thức IP:

```
def ipv4_packet(data):
    version_header_length = data[0]
    version = version_header_length >> 4
    header_length = (version_header_length & 15) * 4
    (tos, total_length, ip_id, flag_fragment, ttl, proto, checksum, src,
     target) = struct.unpack('! x B H H H B B H 4s 4s', data[:20])
    flags = flag_fragment >> 13
    reversed_bit = flags >> 2
    no_frag = (flags & 2) >> 1
    more_frag = flags & 1
    fragment_offset = flag_fragment & 0x1fff    #keep 13 bits on the most right
    return (version, header_length, tos, total_length, ip_id, reversed_bit, no_frag, more_frag,
            fragment_offset, ttl, proto, checksum, get_ipv4(src), get_ipv4(target), data[header_length:])
```

4. Hàm phân tích gói tin giao thức Ethernet:

```python
def ethernet_frame(data):
    dest_mac, src_mac, proto = struct.unpack('! 6s 6s H', data[:14])
    return get_mac_addr(dest_mac), get_mac_addr(src_mac), socket.htons(proto), data[14:]
```

III. Lập trình phân tích và bắt các gói tin bằng Python:

A. Giới thiệu:

Trong phần này, chúng ta sẽ ứng dụng thực tế script Python PacketSniffer.py vào thực thế.

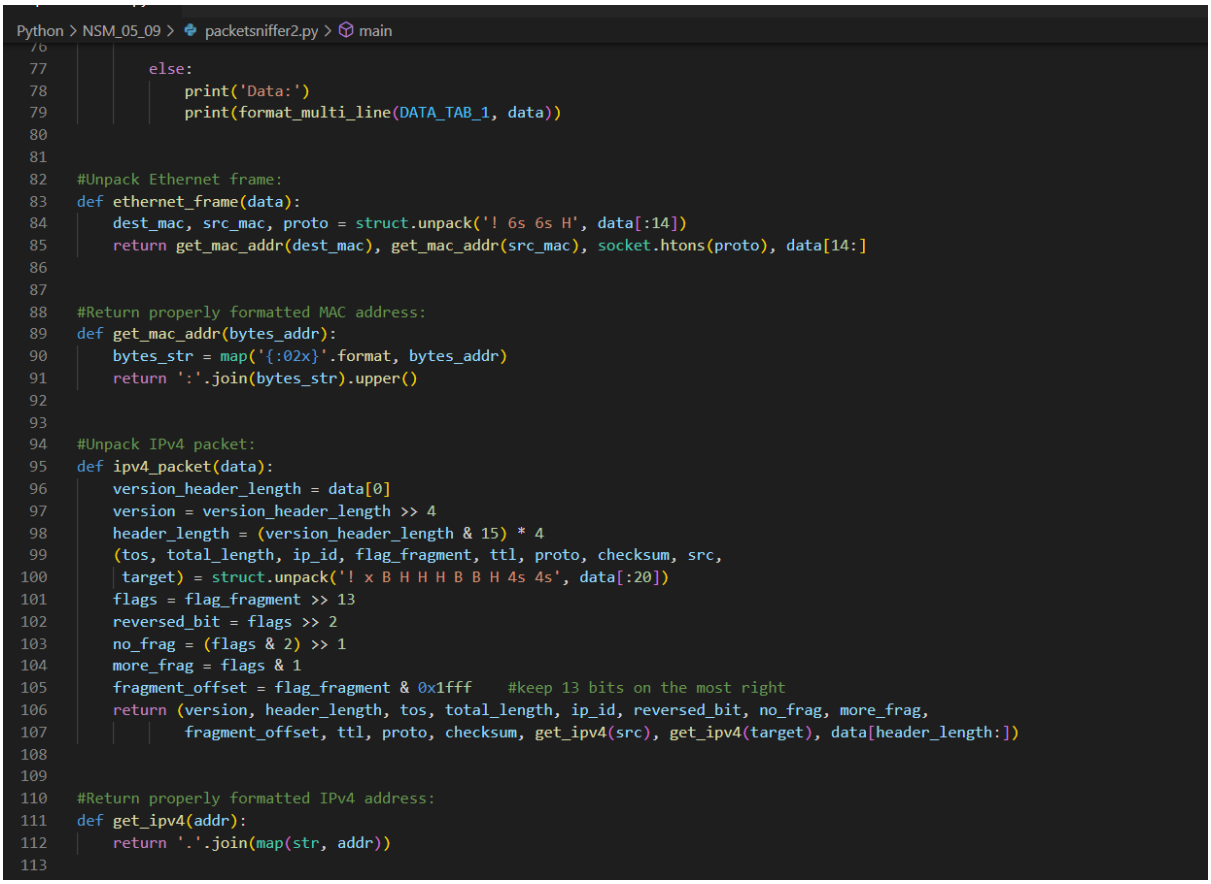
B. Chương trình đầy đủ:

Python > NSM_05_09 > packetsniffer2.py > icmp_packet

```python
import socket
import struct
import textwrap

TAB_1 = '\t - '
TAB_2 = '\t\t - '
TAB_3 = '\t\t\t - '
TAB_4 = '\t\t\t\t - '

DATA_TAB_1 = '\t '
DATA_TAB_2 = '\t\t '
DATA_TAB_3 = '\t\t\t '
DATA_TAB_4 = '\t\t\t\t '

def main():
    conn = socket.socket(socket.AF_PACKET, socket.SOCK_RAW, socket.ntohs(3))

    while True:
        raw_data, addr = conn.recvfrom(65536)
        dest_mac, src_mac, eth_proto, data = ethernet_frame(raw_data)
        print('\nEthernet Frame:')
        print(TAB_1 + 'Destination: {}, Source: {}, Protocol: {}'.format(dest_mac, src_mac, eth_proto))

        #eth_proto = 8 is IPv4:
        if eth_proto == 8:
            (version, header_length, tos, total_length, ip_id, reversed_bit, no_frag, more_frag,
             fragment_offset, ttl, proto, checksum, src, target, data) = ipv4_packet(data)
            print(TAB_1 + 'IPv4 Packet:')
            print(TAB_2 + 'Verssion: {}, Header Length: {}, Type of Service: {}'.format(
                version, header_length, tos))
            print(TAB_2 + 'Total Length: {}, Identification: {}'.format(
                total_length, ip_id))
            print(TAB_2 + 'Flags:')
            print(TAB_3 + 'Res: {}, DF: {}, MF: {}'.format(
                reversed_bit, no_frag, more_frag))
            print(TAB_2 + 'Fragment Offset: {}, TTL: {}, Protocol: {}, Checksum: {}'.format(
                fragment_offset, ttl, proto, checksum))
            print(TAB_2 + 'Source: {}, Destination: {}'.format(
                src, target))

```

Python > NSM_05_09 > packetsniffer2.py > main

```
41              #proto = 1 is ICMP:
42              if proto == 1:
43                  icmp_type, code, checksum, data = icmp_packet(data)
44                  print(TAB_1 + 'ICMP Packet:')
45                  print(TAB_2 + 'Type: {}, Code: {}, Checksum: {}'.format(icmp_type, code, checksum))
46                  print(TAB_2 + 'Data:')
47                  print(format_multi_line(DATA_TAB_3, data))
48
49              #proto = 6 is TCP:
50              elif proto == 6:
51                  (src_port, dest_port, seq, ack, offset, flag_urg, flag_ack, flag_psh, flag_rst, flag_syn,
52                   flag_fin, checksum, urg_ptr, data) = tcp_segment(data)
53                  print(TAB_1 + 'TCP Segment:')
54                  print(TAB_2 + 'Source Port: {}, Destination Port: {}'.format(src_port, dest_port))
55                  print(TAB_2 + 'Sequence: {}, Acknowledgment: {}'.format(seq, ack))
56                  print(TAB_2 + 'Flags:')
57                  print(TAB_3 + 'URG: {}, ACK: {}, PSH: {}, RST: {}, SYN: {}, FIN: {}'.format(
58                      flag_urg, flag_ack, flag_psh, flag_rst, flag_syn, flag_fin))
59                  print(TAB_2 + 'Checksum: {}, Urgent Pointer: {}'.format(checksum, urg_ptr))
60                  print(TAB_2 + 'Data:')
61                  print(format_multi_line(DATA_TAB_3, data))
62
63              #proto= 17 is UDP:
64              elif proto == 17:
65                  src_port, dest_port, length, checksum, data = udp_segment(data)
66                  print(TAB_1 + 'UDP Segment:')
67                  print(TAB_2 + 'Source Port: {}, Destination Port: {}'.format(src_port, dest_port))
68                  print(TAB_2 + 'Length: {}, Checksum: {}'.format(length, checksum))
69                  print(TAB_2 + 'Data:')
70                  print(format_multi_line(DATA_TAB_3, data))
71
72              #Other:
73              else:
74                  print(TAB_1 + 'Data:')
75                  print(format_multi_line(DATA_TAB_2, data))
76
```


Python > NSM_05_09 > packetsniffer2.py > main
else:
print('Data:')
print(format_multi_line(DATA_TAB_1, data))
#Unpack Ethernet frame:
def ethernet_frame(data):
dest_mac, src_mac, proto = struct.unpack('! 6s 6s H', data[:14])
return get_mac_addr(dest_mac), get_mac_addr(src_mac), socket.htons(proto), data[14:]
#Return properly formatted MAC address:
def get_mac_addr(bytes_addr):
bytes_str = map('{:02x}'.format, bytes_addr)
return ':'.join(bytes_str).upper()
#Unpack IPv4 packet:
def ipv4_packet(data):
version_header_length = data[0]
version = version_header_length >> 4
header_length = (version_header_length & 15) * 4
(tos, total_length, ip_id, flag_fragment, ttl, proto, checksum, src,
target) = struct.unpack('! x B H H H B B H 4s 4s', data[:20])
flags = flag_fragment >> 13
reversed_bit = flags >> 2
no_frag = (flags & 2) >> 1
more_frag = flags & 1
fragment_offset = flag_fragment & 0x1fff #keep 13 bits on the most right
return (version, header_length, tos, total_length, ip_id, reversed_bit, no_frag, more_frag,
fragment_offset, ttl, proto, checksum, get_ipv4(src), get_ipv4(target), data[header_length:])
#Return properly formatted IPv4 address:
def get_ipv4(addr):
return '.'.join(map(str, addr))

Python > NSM_05_09 > packetsniffer2.py > ...

```
114
115 #Unpack ICMP packet:
116 def icmp_packet(data):
117     icmp_type, code, checksum = struct.unpack('! B B H', data[:4])
118     return icmp_type, code, checksum, data[4:]
119
120
121 #Unpack TCP segment:
122 def tcp_segment(data):
123     src_port, dest_port, seq, ack, o_r_f, window, checksum, urg_ptr = (
124         struct.unpack('! H H L L H H H H', data[:20]))
125     offset = (o_r_f >> 12) * 4
126     flag_urg = (o_r_f & 32) >> 5
127     flag_ack = (o_r_f & 16) >> 4
128     flag_psh = (o_r_f & 8) >> 3
129     flag_rst = (o_r_f & 4) >> 2
130     flag_syn = (o_r_f & 2) >> 1
131     flag_fin = o_r_f & 1
132     return (src_port, dest_port, seq, ack, offset, flag_urg, flag_ack, flag_psh, flag_rst, flag_syn, flag_fin,
133             checksum, urg_ptr, data[offset:])
134
135 #Unpack UDP segment:
136 def udp_segment(data):
137     src_port, dest_port, length, checksum = struct.unpack('! H H H H', data[:8])
138     return src_port, dest_port, length, checksum, data[8:]
139
140
141 #Format multi-line data (make data more readable to humman :D):
142 def format_multi_line(prefix, string, size=80):
143     size -= len(prefix)
144     if isinstance(string, bytes):
145         string = ''.join(r'\x{:02x}'.format(byte) for byte in string)
146         if size % 2:
147             size -= 1
148     return '\n'.join([prefix + line for line in textwrap.wrap(string, size)])
149
150
151 main()
```

C. Ứng dụng thực tế:

Thử chạy script PacketSniffer.py trên máy ảo Linux. Ta thu được kết quả như sau:

Chương trình PacketSniffer.py bắt gói tin ICMP.

```
Ethernet Frame:
     - Destination: 00:50:56:F2:F0:90, Source: 00:0C:29:8D:4E:47, Protocol: 8
     - IPv4 Packet:
         - Version: 4, Header Length: 20, Type of Service: 0
         - Total Length: 84, Identification: 18853
         - Flags:
             - Res: 0, DF: 1, MF: 0
         - Fragment Offset: 0, TTL: 64, Protocol: 1, Checksum: 64147
         - Source: 192.168.112.128, Destination: 172.217.24.110
     - ICMP Packet:
         - Type: 8, Code: 0, Checksum: 29096
         - Data:
             \xbf\x06\x00\x02\x0b\xcc\xf5\x65\x00\x00\x00\x00\xf9\x49\x0e\x00\x00\x00\x00
             \x00\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f\x20\x21
             \x22\x23\x24\x25\x26\x27\x28\x29\x2a\x2b\x2c\x2d\x2e\x2f\x30\x31\x32\x33\x34
             \x35\x36\x37
```

Chương trình PacketSniffer.py bắt gói tin UDP.

```
Ethernet Frame:
     - Destination: 00:50:56:F2:F0:90, Source: 00:0C:29:8D:4E:47, Protocol: 8
     - IPv4 Packet:
         - Version: 4, Header Length: 20, Type of Service: 0
         - Total Length: 73, Identification: 38779
         - Flags:
             - Res: 0, DF: 1, MF: 0
         - Fragment Offset: 0, TTL: 64, Protocol: 17, Checksum: 16725
         - Source: 192.168.112.128, Destination: 192.168.112.2
     - UDP Segment:
         - Source Port: 35149, Destination Port: 53
         - Length: 53, Checksum: 25114
         - Data:
             \xd9\xaa\x01\x00\x00\x01\x00\x00\x00\x00\x00\x00\x03\x31\x31\x30\x02\x32\x34
             \x03\x32\x31\x37\x03\x31\x37\x32\x07\x69\x6e\x2d\x61\x64\x64\x72\x04\x61\x72
             \x70\x61\x00\x00\x0c\x00\x01
```

Chương trình PacketSniffer.py bắt gói tin TCP.

```
Ethernet Frame:
    - Destination: 00:50:56:F2:F0:90, Source: 00:0C:29:8D:4E:47, Protocol: 8
    - IPv4 Packet:
        - Version: 4, Header Length: 20, Type of Service: 0
        - Total Length: 79, Identification: 35992
        - Flags:
            - Res: 0, DF: 1, MF: 0
        - Fragment Offset: 0, TTL: 64, Protocol: 6, Checksum: 32579
        - Source: 192.168.112.128, Destination: 216.239.36.181
    - TCP Segment:
        - Source Port: 36640, Destination Port: 443
        - Sequence: 2073466549, Acknowledgment: 793594539
        - Flags:
            - URG: 0, ACK: 1, PSH: 1, RST: 0, SYN: 0, FIN: 0
        - Checksum: 12047, Urgent Pointer: 0
        - Data:
            \x17\x03\x03\x00\x22\x39\x19\x3a\x55\x7b\x94\x63\xe8\xa3\x51\xa8\xa8\x5d\x44
            \xb1\xbc\xdc\xad\x5f\x0b\x07\xe6\xfe\xd1\x30\x92\xd8\x40\xcc\x80\x13\x13\xfa
            \x63
```

IV. Danh mục tài liệu tham khảo:

[1].https://viblo.asia/p/tim-hieu-giao-thuc-tcp-va-udp-jvEla11xlkw

[2].https://bizflycloud.vn/tin-tuc/user-datagram-protocol-udp-la-gi-20181029112323536.htm

[3].https://r.search.yahoo.com/_ylt=Awrjav4ko_ZlJ6oM_qtXNyoA;_ylu=Y29sbwNncTEEcG9zAzIEdnRpZAMEc2VjA3Ny/RV=2/RE=1711872037/RO=10/RU=https%3a%2f%2fviblo.asia%2fp%2ftim-hieu-giao-thuc-ip-phan-1-bJzKmxer59N/RK=2/RS=iEMFPnwQ2pz3Dal5fGlypEW0C5c-

[4]. https://uzivoz.wordpress.com/2011/10/27/giao-th%E1%BB%A9c-tcp/