HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

Bài tập lớn Giám sát an toàn mạng
Lập trình bắt và phân tích gói tin bằng Python

Giảng viên: **Nguyễn Ngọc Điệp**

Nhóm sinh viên: **Nguyễn Duy Tú Anh – B13DCAT048**

Ngô Đức Bắc – B13DCAT003

Phạm Trung Đức – B13DCAT056

Nguyễn Quốc Hoàn – B13DCAT057

Nguyễn Minh Ngọc – B13DCAT034

Tên lớp: **Nhóm 2 – học sáng thứ sáu.**

Hà nội, ngày 3 tháng 5 năm 2017

Mục lục

DANH MỤC CÁC CHỮ VIẾT TẮT

Viết tắt	Ý nghĩa
UDP	User Datagram Protocol
TCP	Transmision Control Protocol
IP	Internet Protocol
URG	Urgent Pointer field significant
ACK	Acknowledgment field significant
PSH	Push Function
RST	Reset the connection
SYN	Synchronize sequence numbers

DANH SÁCH CÁC BẢNG

DANH SÁCH HÌNH VẼ

Chương 1 – Tổng quan về lí thuyết giao thức

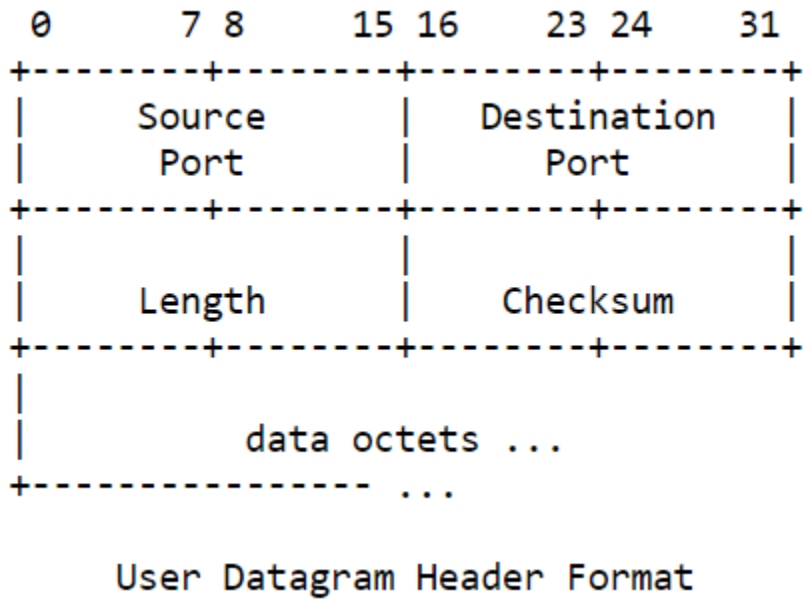
1.1 Dẫn chương:

Trong chương đầu tiên, 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, gói gọn, nhằm đặt nền tảng vững chắc cho chương hai khi phân tích cách lập trình bắt nội dung gói tin.

1.2 UDP Header

Giao thức User Datagram Protocol (UDP) 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.

Giao thức này cung cấp một thủ tục cho các chương trình ứng dụng để gửi tin nhắn đến các chương trình khác với cơ chế giao thức tối thiểu. Giao thức có hướng giao dịch, và truyền tin và bảo vệ bản sao không được đảm bảo. Các ứng dụng đòi hỏi yêu cầu phân phối luồng dữ liệu đáng tin cậy theo yêu cầu nên sử dụng TCP Control Protocol.



Hình 1.1: Các thành phần của header trong gói tin giao thức UDP

Dựa trên hình minh họa 1.1, 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 tùy chọn, khi có ý nghĩa, nó cho biết cổng của quá trình gửi, và có thể được giả định là cổng mà một trả lời nên được giải quyết trong trường hợp không có bất kỳ thông tin khác. Nếu không được sử dụng, một giá trị bằng không được chèn vào.
2	**Destination Port**	**16 bits**	Có ý nghĩa trong ngữ cảnh của một địa chỉ đích đến của Internet cụ thể.
3	**Length**	**16 bits**	Chiều dài là độ dài theo octet của gói tin người dùng này bao gồm tiêu đề và dữ liệu này. (Điều này có nghĩa là giá trị tối thiểu của chiều dài là tám).
4	**Checksum**	**16 bits**	Checksum là phần bổ sung của một phần của tiêu đề giả của thông tin từ tiêu đề IP, phần đầu UDP và dữ liệu, đệm với octet không ở cuối (nếu cần) để tạo ra một bội số của hai octet .

Bảng 1.1: Phân tích các thành phần của header trong giao thức UDP

1.3 TCP Header

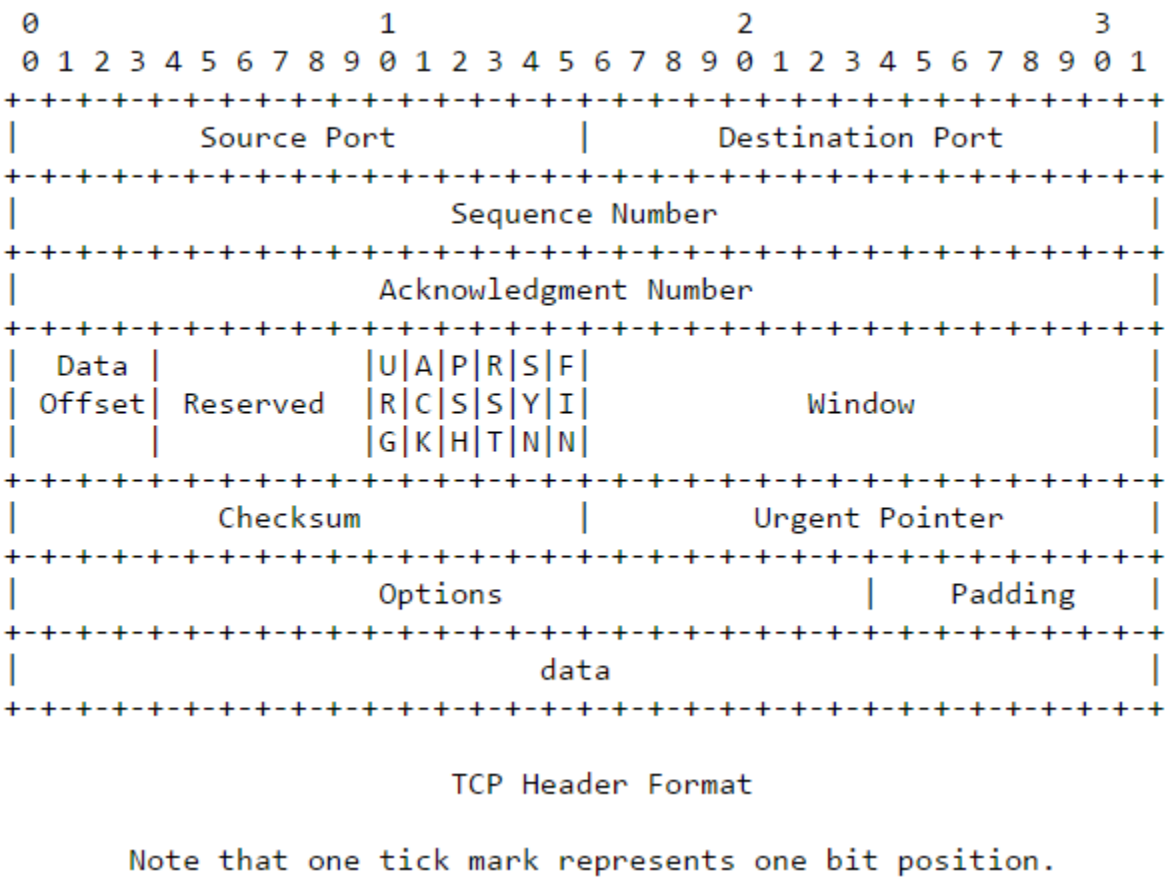
TCP là một giao thức tin cậy định hướng kết nối, đầu cuối được thiết kế để phù hợp với một hệ thống phân cấp các giao thức có hỗ trợ các ứng dụng đa mạng.

TCP cung cấp giao tiếp giữa quá trình đáng tin cậy giữa các cặp quy trình trong các máy chủ lưu trữ gắn với các mạng truyền thông máy tính riêng biệt nhưng kết nối với nhau. Rất ít giả định được thực hiện đối với độ tin cậy của các giao thức truyền thông dưới lớp TCP.

TCP giả sử nó có thể có được một dịch vụ datagram đơn giản, không đáng tin cậy từ các giao thức cấp thấp hơn. Về nguyên tắc, TCP nên có khả

năng hoạt động trên nhiều hệ thống truyền thông khác nhau, từ các kết nối cứng đến các mạng chuyển mạch gói hoặc mạng chuyển mạch.

Các phân đoạn TCP được gửi dưới dạng các gói tin internet. Tiêu đề Giao thức Internet chứa nhiều trường thông tin, bao gồm cả địa chỉ nguồn và đích. Một tiêu đề TCP đi theo tiêu đề internet, cung cấp thông tin cụ thể cho giao thức TCP. Bộ phận này cho phép tồn tại các giao thức mức lưu trữ khác với TCP.



Hình 1.2: Các thành phần header trong gói tin giao thức TCP

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

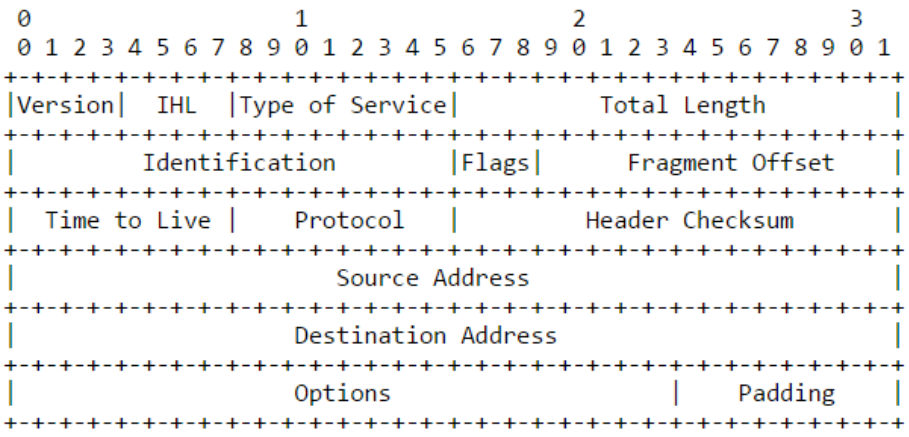
Bảng 1.2: Phân tích các thành phần của header trong giao thức TCP

STT	Tên thành phần	Độ dài thành phần	Đặc điểm chức năng
1	**Source Port**	16 bits	Số cổng nguồn
2	**Destination Port**	16 bits	Số cổng đích
3	**Sequence number**	32 bits	Số thứ tự của octet dữ liệu đầu tiên trong phân đoạn này (trừ khi có SYN). Nếu SYN có mặt, số thứ tự là số thứ tự ban đầu (ISN) và octet dữ liệu đầu tiên là ISN + 1.
4	**Acknowledgment Number**	32 bits	Nếu bit điều khiển ACK được thiết lập, trường này chứa giá trị của số thứ tự tiếp theo mà người gửi của phân đoạn này mong đợi nhận. Một khi kết nối được thiết lập, điều này luôn được gửi.
5	**Data offset**	4 bits	Số lượng các từ 32 bit trong Tiêu đề TCP, cho biết nơi dữ liệu bắt đầu. Tiêu đề TCP (ngay cả một bao gồm các tùy chọn) là một số không tách rời dài 32 bit.

6	**Reserved**	6 bits	Mục đích của nó là để sử dụng cho tương lai, giá trị phải là 0.
7	**Control bit**	6 bits	Từ trái sang phải: • URG: Trường con trỏ khẩn cấp • ACK: Xác nhận các trường đáng kể • PSH: Chức năng đẩy • RST: Đặt lại kết nối • SYN: Đồng bộ hóa số thứ tự • FIN: Không có thêm dữ liệu từ người gửi.
8	**Window**	16 bits	Số lượng octet dữ liệu bắt đầu với số được chỉ ra trong trường xác nhận mà người gửi của phân đoạn này sẵn sàng chấp nhận.
9	**Checksum**	16 bits	Trường tổng kiểm tra là phần bổ sung của bit 16 bit cho tổng số tất cả các từ 16 bit trong tiêu đề và văn bản. Nếu một phân đoạn chứa một số lẻ của phần đầu và phần tử văn bản được kiểm tra, thì octet cuối cùng được thêm vào bên phải bằng số không để tạo thành một từ 16 bit cho các mục đích kiểm tra. Phần đệm không được truyền đi như một phần của đoạn. Trong khi tính tổng kiểm tra, trường kiểm tra sẽ được thay thế bằng các số không.

			Chiều dài TCP là chiều dài tiêu đề TCP cộng với độ dài dữ liệu trong octet (đây không phải là số lượng truyền rõ ràng, nhưng được tính), và nó không tính 12 octet của tiêu đề giả.
10	**Urgent Pointer**	16 bits	Trường này truyền tải giá trị hiện tại của con trỏ khẩn cấp như là một số dương bù đắp từ số thứ tự trong đoạn này. Con trỏ khẩn cấp chỉ ra số thứ tự của octet sau dữ liệu khẩn cấp. Trường này chỉ được hiểu trong các phân đoạn với thiết lập bit điều khiển URG.
11	**Padding**	Tùy biến	Phần đệm tiêu đề TCP được sử dụng để đảm bảo rằng tiêu đề TCP kết thúc và dữ liệu bắt đầu trên ranh giới 32 bit. Phần đệm bao gồm các giá trị 0.

1.4 IP Header



Example Internet Datagram Header

Figure 4.

Hình 1.3: Minh họa các thành phần của header giao thức IP.

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

Bảng 1.3: Phân tích các thành phần của header trong giao thức IP

STT	Tên thành phần	Độ dài thành phần	Đặc điểm
1	**Version**	4 bits	Trường Phiên bản chỉ định định dạng của tiêu đề internet. Ví dụ như hiển thị 6 thì giao thức của nó là TCP, hiển thị 17 thì là UDP.
2	**IHL (Internet Headers Length)**	4 bits	Chiều dài tiêu đề Internet là chiều dài của tiêu đề internet bằng các từ 32 bit, và do đó chỉ ra sự bắt đầu của dữ liệu. Lưu ý rằng giá trị tối thiểu cho một tiêu đề chính xác là 5

3	**Type of Services**	8 bits	Loại Dịch vụ cung cấp một chỉ dẫn về các tham số trừu tượng về chất lượng dịch vụ mong muốn. Các tham số này sẽ được sử dụng để hướng dẫn lựa chọn các tham số dịch vụ thực tế khi truyền một gói tin thông qua một mạng lưới cụ thể. Một số mạng cung cấp dịch vụ ưu tiên, bằng cách nào đó xử lý lưu lượng truy cập cao quan trọng hơn các lưu lượng truy cập khác (thông thường chỉ chấp nhận lưu lượng truy cập trên một độ ưu tiên nhất định tại thời điểm tải cao). Sự lựa chọn chính là sự cân bằng ba chiều giữa độ trễ thấp, độ tin cậy cao và thông lượng cao.
4	**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 một 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.
5	**Identification**	16 bits	Một giá trị nhận dạng được chỉ định bởi người gửi để giúp lắp ráp các mảnh của một gói tin.
6	**Flags**	3 bits	Các cờ điều khiển khác nhau. • Bit 0: dự trữ, phải bằng không • Bit 1: (DF) 0 = Có thể phân mảnh, 1 = Không phân mảnh. • Bit 2: (MF) 0 = Đoạn cuối, 1 = Các phân đoạn khác.

7	**Fragment Offset**	13 bits	Trường này cho biết trong datagram đoạn này thuộc về datagram. Phần bù đắp được đo bằng đơn vị 8 octet (64 bit). Đoạn đầu tiên đã được bù đắp bằng không.
8	**Time to Live**	8 bits	Trường này cho biết thời gian tối đa cho phép datagram được chấp nhận trong hệ thống internet. Nếu trường này chứa giá trị 0, thì gói tin đó phải bị hủy. Trường này được sửa đổi trong quá trình xử lý phần đầu internet. Thời gian được đo bằng đơn vị giây, nhưng vì mỗi mô-đun xử lý một gói tin phải giảm TTL xuống ít nhất một, thậm chí nếu nó xử lý gói tin trong vòng chưa đầy một giây, thì TTL phải được coi như là một giới hạn trên trên Thời gian một datagram có thể tồn tại. Mục đích là làm cho các gói tin không gửi đi được sẽ bị loại bỏ, và để ràng buộc datagram tối đa.
9	**Protocol**	8 bits	Trường này cho biết giao thức mức tiếp theo được sử dụng trong phần dữ liệu của gói tin internet.
10	**Header Checksum**	16 bits	Checksum chỉ trên tiêu đề. 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.
11	**Source Address**	32 bits	Địa chỉ nguồn
12	**Destination Address**	32 bits	Địa chỉ đích

13	**Options**	Tùy biến	
14	**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.

1.5 Ethernet Header

Một khung Ethernet được nối trước bởi một dấu phân cách đầu và khung bắt đầu frame (SFD), là một phần của gói Ethernet ở lớp vật lý. Mỗi khung Ethernet bắt đầu với một tiêu đề Ethernet, chứa địa chỉ MAC đích và nguồn như hai trường đầu tiên. Phần giữa của khung là dữ liệu tải trọng bao gồm các tiêu đề cho các giao thức khác (ví dụ như Giao thức Internet) được thực hiện trong khung. Khung kết thúc bằng một chuỗi kiểm tra khung (FCS), là kiểm tra dự phòng chu kỳ 32 bit được sử dụng để phát hiện bất kỳ sự truyền dữ liệu nào trong quá trình chuyển.

802.3 Ethernet packet and frame structure

Layer	Preamble	Start of frame delimiter	MAC destination	MAC source	802.1Q tag (optional)	Ethertype (Ethernet II) or length (IEEE 802.3)	Payload	Frame check sequence (32-bit CRC)	Interpacket gap
	7 octets	1 octet	6 octets	6 octets	(4 octets)	2 octets	46–1500 octets	4 octets	12 octets
Layer 2 Ethernet frame			← 64–1522 octets →						
Layer 1 Ethernet packet & IPG	← 72–1530 octets →								← 12 oct. →

Hình 1.4: Minh họa thành phần header giao thức IP.

Chương 2 – Tổng quan về kĩ thuật

2.1 Dẫn chương:

Trong chương hai của bài tiểu luận, chúng ta sẽ tìm hiểu về việc áp dụng ngôn ngữ lập trình Python vào việc bắt và phân tích các gói tin.

2.2 Giới thiệu về Python

Phiên bản Python: Python 2.7.12+ [GCC 6.2.0 20160822] on linux2

Bảng 2.1: Các thư viện sử dụng trong chương trình Python.

STT	Tên thư viện	Chức năng
1	Struct	Mô-đun này thực hiện chuyển đổi giữa các giá trị Python và các cấu trúc C được biểu diễn dưới dạng các chuỗi Python. Điều này có thể được sử dụng trong việc xử lý dữ liệu nhị phân được lưu trữ trong các tệp tin hoặc từ các kết nối mạng, trong số các nguồn khác. Nó sử dụng định dạng chuỗi như mô tả gọn gàng của cách bố trí của cấu trúc C và chuyển đổi dự định đến / từ các giá trị Python.
2	Socket	Mô đun này cung cấp truy cập vào giao diện ổ cắm BSD. Nó có sẵn trên tất cả các hệ thống Unix hiện đại, Windows, Mac OS X, BeOS, OS / 2, và có lẽ các nền tảng bổ sung.
3	Binascii	Mô-đun binascii chứa một số phương pháp để chuyển đổi giữa các tập tin nhị phân và mã hóa ASCII khác nhau.
4	OS, SYS	Mô-đun này cung cấp cách sử dụng chức năng hệ điều hành phụ thuộc
5	Time	Mô-đun này dùng để dừng màn hình chương trình trong một thời gian nhất định (Để tiện việc theo dõi các gói tin)

2.3 Lập trình bắt và phân tích các gói tin

Protocol Number	Hex	Keyword	Protocol
0	0x00	HOPOPT	IPv6 Hop-by-Hop Option
1	0x01	ICMP	Internet Control Message Protocol
2	0x02	IGMP	Internet Group Management Protocol
3	0x03	GGP	Gateway-to-Gateway Protocol
6	0x06	TCP	Transmission Control Protocol
17	0x11	UDP	User Datagram Protocol
18	0x12	MUX	Multiplexing
19	0x13	DCN-MEAS	DCN Measurement Subsystems

Hình 2.1: Số thứ tự chỉ định hai thức UDP và TCP.

Bảng 2.2: Quy đổi các giá trị trong thư viện Struct của ngôn ngữ Python

Format	C Type	Python type	Standard size
x	pad byte	no value	
c	**char**	string of length 1	1
b	**signed char**	integer	1
B	**unsigned char**	integer	1
?	**_Bool**	bool	1
h	**short**	integer	2
H	**unsigned short**	integer	2
i	**int**	integer	4
I	**unsigned int**	integer	4
l	**long**	integer	4
L	**unsigned long**	integer	4
q	**long long**	integer	8
Q	**unsigned long long**	integer	8
f	**float**	float	4
d	**double**	float	8
s	**char[]**	string	
p	**char[]**	string	
P	**void ***	integer	

Lưu ý: Header field luôn sử dụng hàm giá trị unsign có giá trị dương.

2.3.1 Hàm phân tích gói tin giao thức UDP:

```
def analyze_udp_header(data):
    udp_hdr     = struct.unpack("!4H", data[:8])
    src_port    = udp_hdr[0]
    dst_port    = udp_hdr[1]
    length      = udp_hdr[2]
    chk_sum     = udp_hdr[3]

    data        = data[8:]
```

Hình 2.2: Hàm phân tích gói tin UDP

2.3.2 Hàm phân tích gói tin giao thức TCP:

```
def analyze_tcp_header(data):
    tcp_hdr = struct.unpack("!2H2I4H", data[:20])
    src_port = tcp_hdr[0]
    dst_port = tcp_hdr[1]
    seq_num = tcp_hdr[2]
    ack_num = tcp_hdr[3]
    data_offset = tcp_hdr[4] >> 12
    reserved = (tcp_hdr[4] >> 6) & 0x03ff
    flags = tcp_hdr[4] & 0x003f

    urg = flags & 0x0020
    ack = flags & 0x0010
    psh = flags & 0x0008
    rst = flags & 0x0004
    syn = flags & 0x0002
    fin = flags & 0x0001

    window = tcp_hdr[5]
    checksum = tcp_hdr[6]
    urg_ptr = tcp_hdr[7]
    data = data[20:]
```

Hình 2.3: Hàm phân tích giao thức TCP

2.3.3 Hàm phân tích gói tin giao thức IP:

```
def analyze_ip_header(data):
    ip_hdr    = struct.unpack("!6H4s4s", data[:20])
    ver       = ip_hdr[0] >> 12
    ihl = (ip_hdr[0] >> 8) & 0x0f
    tos = ip_hdr[0] & 0x00ff
    tot_len = ip_hdr[1]
    ip_id = ip_hdr[2]
    flags = ip_hdr[3] >> 13
    frag_offset = ip_hdr[3] & 0x1fff
    ip_ttl = ip_hdr[4] >> 8
    ip_proto = ip_hdr[4] & 0x00ff
    chk_sum = ip_hdr[5]
    src_addr = socket.inet_ntoa(ip_hdr[6]) #first 4s
    dst_addr = socket.inet_ntoa(ip_hdr[7]) #second 4s

    no_frag = flags >> 1
    more_frag = flags & 0x1
```

Hình 2.4: Hàm phân tích giao thức IP

2.3.4 Hàm phân tích gói tin giao thức Ethernet:

```
def analyze_ether_header(data):
    ip_bool = False
    eth_hdr  = struct.unpack("!6s6sH", data[:14])
    dest_mac = binascii.hexlify(eth_hdr[0]) #Desti
    src_mac  = binascii.hexlify(eth_hdr[1]) #Sourc
    proto    = eth_hdr[2] >> 8 #NextProtocol
```

Hình 2.5: Hàm phân tích giao thức Ethernet

Chương 3 – Lập trình việc phân tích và bắt gói tin bằng Python

3.1 Dẫn chương:

Trong chương số ba của bài tiểu luận, chúng ta ứng dụng thực tế script Python PacketSniffer.py vào thực tế.

3.2 Chương trình đầy đủ:

PacketSniffer.py

```
## Lap trinh phan tich va bat goi tin tren Python
import struct
import socket
import binascii
import os, sys
import time
def analyze_udp_header(data):
        udp_hdr    = struct.unpack("!4H", data[:8])
        src_port   = udp_hdr[0]
        dst_port   = udp_hdr[1]
        length     = udp_hdr[2]
        chk_sum    = udp_hdr[3]
        data       = data[8:]
        print "!_____UDP HEADER_____!"
        print "|\t\tSource:\t\t%hu" % src_port
        print "|\t\tDest:\t\t%hu" % dst_port
        print "|\t\tLength:\t\t%hu" % length
        print "|\t\tChecksum:\t%hu" % chk_sum
        return data
```

```
def analyze_tcp_header(data):
        tcp_hdr = struct.unpack("!2H2I4H", data[:20])
        src_port = tcp_hdr[0]
        dst_port = tcp_hdr[1]
        seq_num = tcp_hdr[2]
        ack_num = tcp_hdr[3]
        data_offset = tcp_hdr[4] >> 12
        reserved = (tcp_hdr[4] >> 6) & 0x03ff
        flags = tcp_hdr[4] & 0x003f

        urg = flags & 0x0020
        ack = flags & 0x0010
        psh = flags & 0x0008
        rst = flags & 0x0004
        syn = flags & 0x0002
        fin = flags & 0x0001
        window = tcp_hdr[5]
        checksum = tcp_hdr[6]
        urg_ptr = tcp_hdr[7]
        data = data[20:]
        print "!_____TCP HEADER_____!"
        print "|\t\tSource: \t%hu" % src_port
        print "|\t\tDest: \t\t%hu" % dst_port
        print "|\t\tSeq: \t\t%u" % seq_num
        print "|\t\tAck: \t\t%u" % ack_num
        print "|\t\tFlags: "
```

```
        print "|\t\t\t URG:%d" % urg
        print "|\t\t\t ACK:%d" % ack
        print "|\t\t\t PSH:%d" % psh
        print "|\t\t\t RST:%d" % rst
        print "|\t\t\t SYN:%d" % syn
        print "|\t\t\t FIN:%d" % fin
        print "|\t\t Window:\t%hu" % window
        print "|\t\t Checksum:\t%hu" % checksum
        return data


def analyze_ip_header(data):
        ip_hdr      = struct.unpack("!6H4s4s", data[:20])
        ver         = ip_hdr[0] >> 12
        ihl = (ip_hdr[0] >> 8) & 0x0f
        tos = ip_hdr[0] & 0x00ff
        tot_len = ip_hdr[1]
        ip_id = ip_hdr[2]
        flags = ip_hdr[3] >> 13
        frag_offset = ip_hdr[3] & 0x1fff
        ip_ttl = ip_hdr[4] >> 8
        ip_proto = ip_hdr[4] & 0x00ff
        chk_sum = ip_hdr[5]
        src_addr = socket.inet_ntoa(ip_hdr[6]) #first 4s
        dst_addr = socket.inet_ntoa(ip_hdr[7]) #second 4s
        no_frag = flags >> 1
        more_frag = flags & 0x1
```

```
        print "!_____IP HEADER_____!"
        print "|\t\tVersion:\t%hu" % ver
        print "|\t\tIHL:\t\t%hu" % ihl
        print "|\t\tTOS:\t\t%hu" % tos
        print "|\t\tLength:\t\t%hu" % tot_len
        print "|\t\tID:\t\t%hu" % ip_id
        print "|\t\tFlags:\t\t%hu" % flags
        print "|\t\tOffset:\t\t%hu" % frag_offset
        print "|\t\tTTL:\t\t%hu" % ip_ttl
        print "|\t\tNext Proto:\t%hu" % ip_proto
        print "|\t\tChecksum:\t%hu" % chk_sum
        print "|\t\tSource IP:\t%s" % src_addr
        print "|\t\tDest IP:\t%s" % dst_addr

        if ip_proto == 6:
                next_proto = "TCP"
        elif ip_proto == 17:
                next_proto = "UDP"
        else:
                next_proto = "Other"
        data = data[20:]
        return data, next_proto

def analyze_ether_header(data):
        ip_bool = False
        eth_hdr  = struct.unpack("!6s6sH", data[:14]) #IPv4 = 0x0800
        dest_mac = binascii.hexlify(eth_hdr[0]) #Destination Address
```

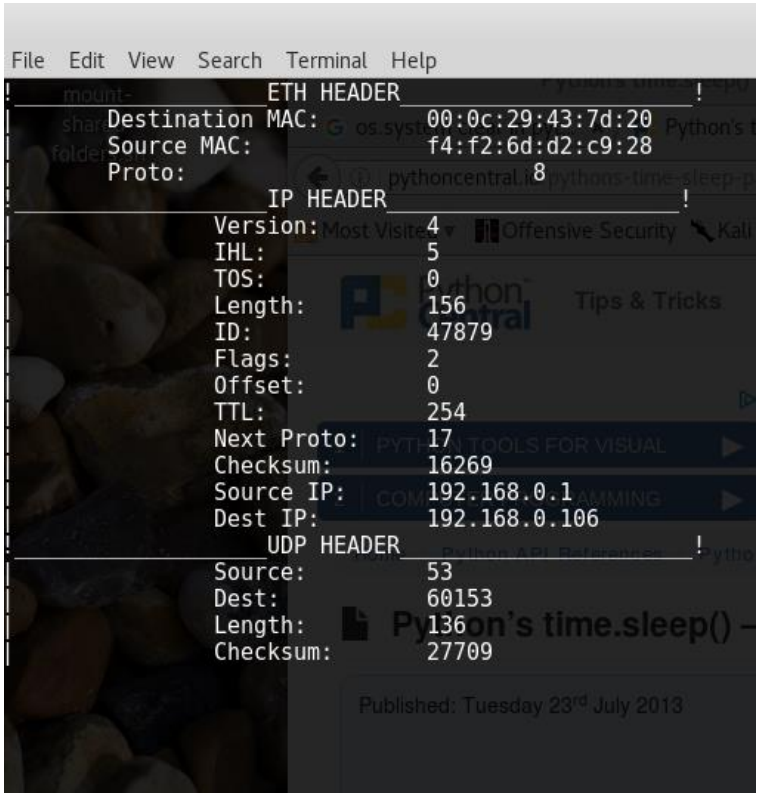
```
        src_mac  = binascii.hexlify(eth_hdr[1]) #Source Address
        proto    = eth_hdr[2] >> 8 #NextProtocol
        print "!_____ETH HEADER_____!"
        print "|\tDestination MAC: \t%s:%s:%s:%s:%s:%s " % (dest_mac[0:2],
dest_mac[2:4],
                dest_mac[4:6], dest_mac[6:8], dest_mac[8:10], dest_mac[10:12])
        print "|\tSource MAC: \t\t%s:%s:%s:%s:%s:%s" %
(src_mac[0:2],src_mac[2:4],src_mac[4:6],
                src_mac[6:8], src_mac[8:10], src_mac[10:12])
        print "|\tProto:\t\t\t\t%hu" % proto
        if proto == 0x08:
                ip_bool = True
        data = data[14:]
        return data, ip_bool


def main():
        sniffer_socket = socket.socket(socket.PF_PACKET, socket.SOCK_RAW,
socket.ntohs(0x0003))
        recv_data = sniffer_socket.recv(2048)
        time.sleep(1)
        os.system("clear")
        data, ip_bool = analyze_ether_header(recv_data)
        if ip_bool:
                data, next_proto = analyze_ip_header(data)
        else:
                return
        if next_proto == "TCP":
                data = analyze_tcp_header(data)
```

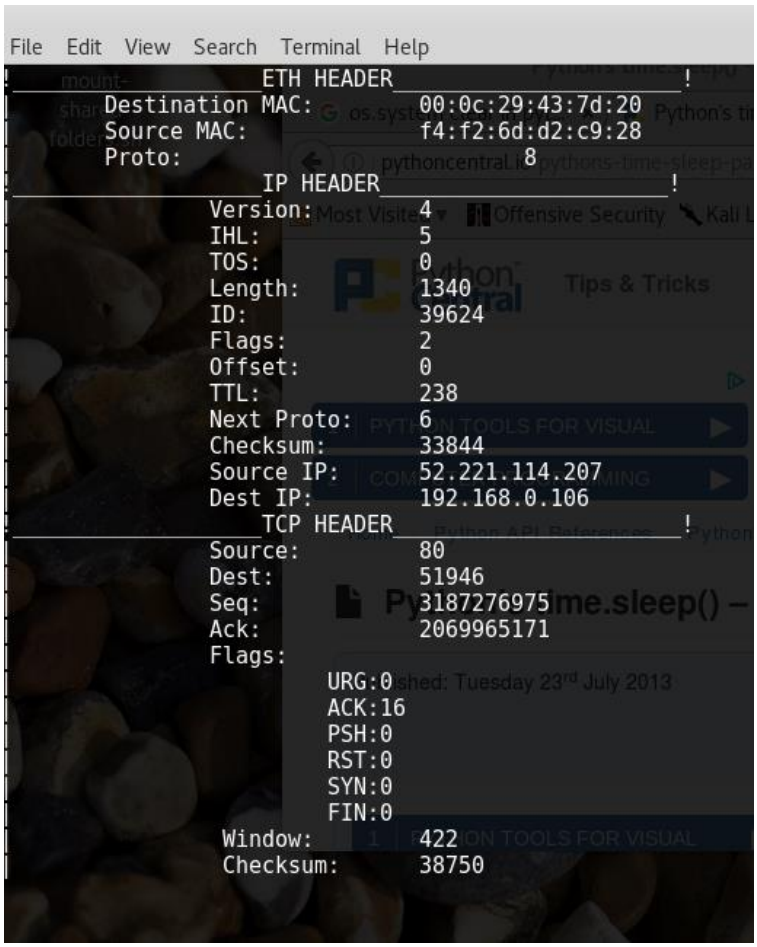
```
        elif next_proto == "UDP":
                data = analyze_udp_header(data)
        else:
                return
while True:
        main()
```

3.3 Ứng dụng thực tế

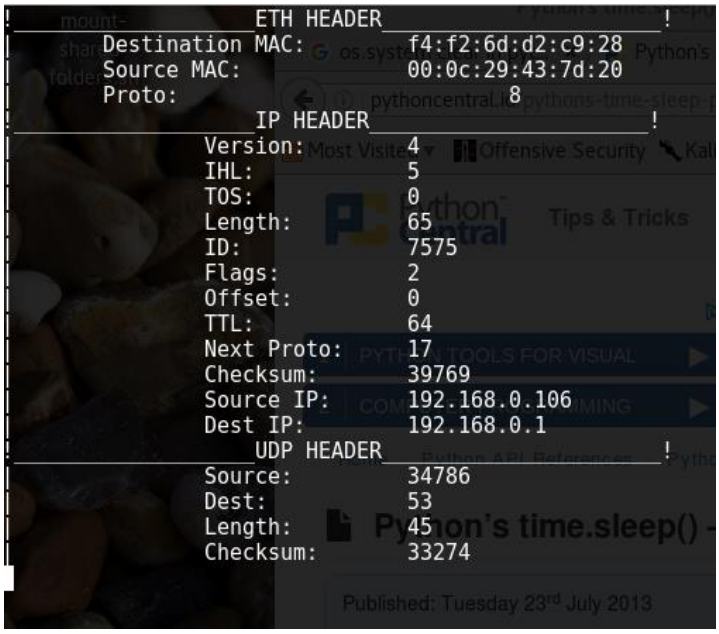
Thử chạy script PacketSniffer.py trên máy ảo Linux, ta được kết quả như sau:



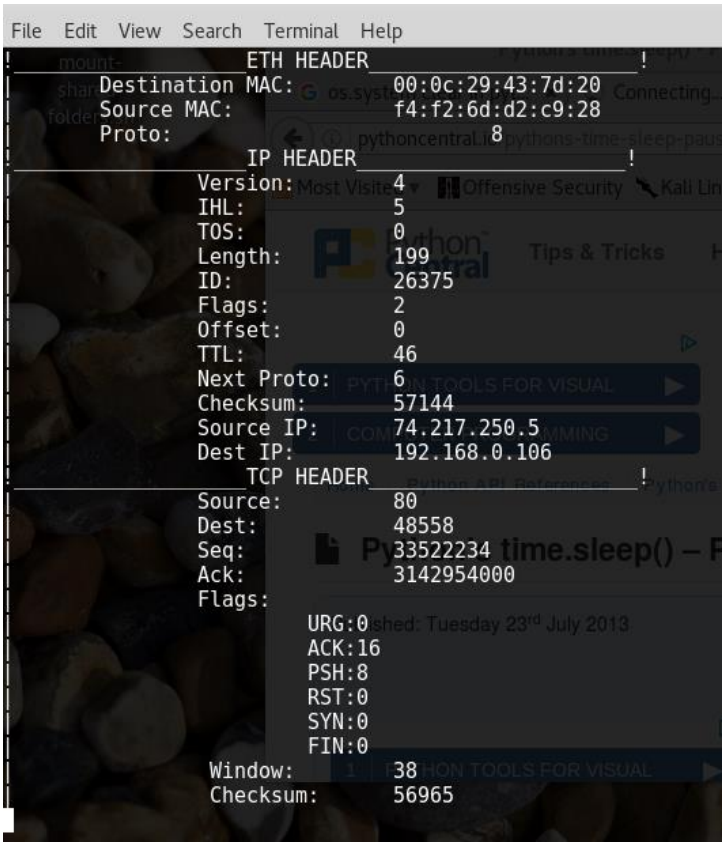
Hình 3.1: Chương trình PacketSniffer.py bắt gói tin UDP.



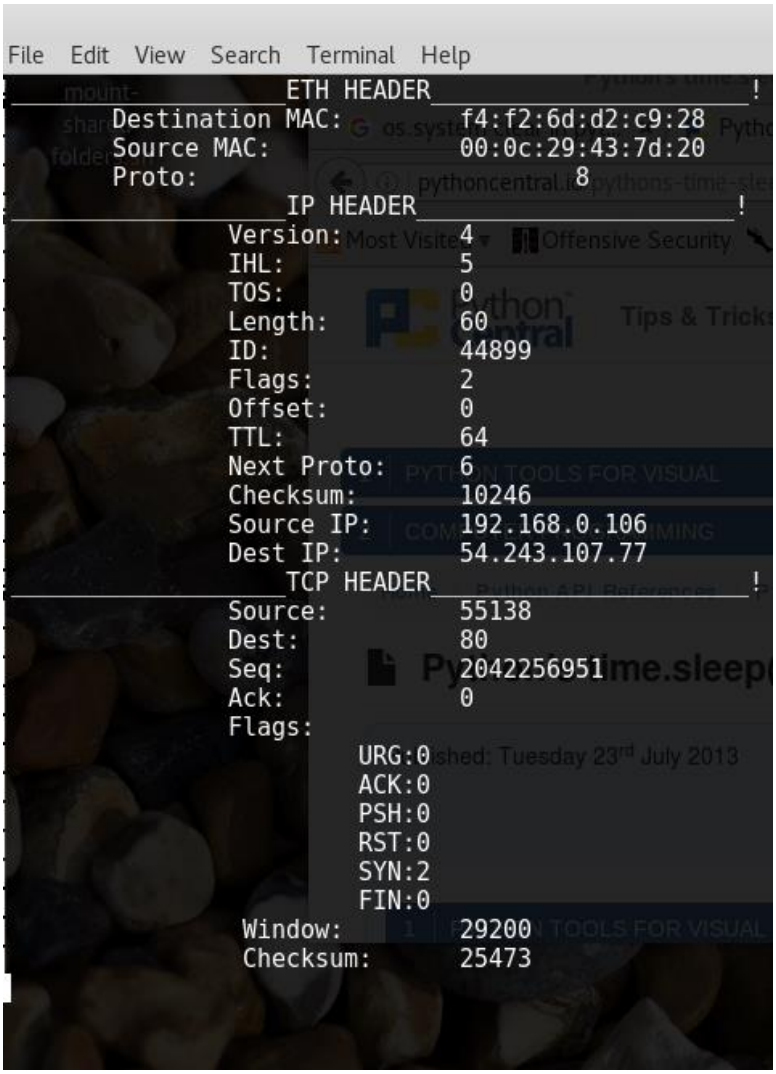
Hình 3.2: Chương trình PacketSniffer.py bắt gói tin TCP.



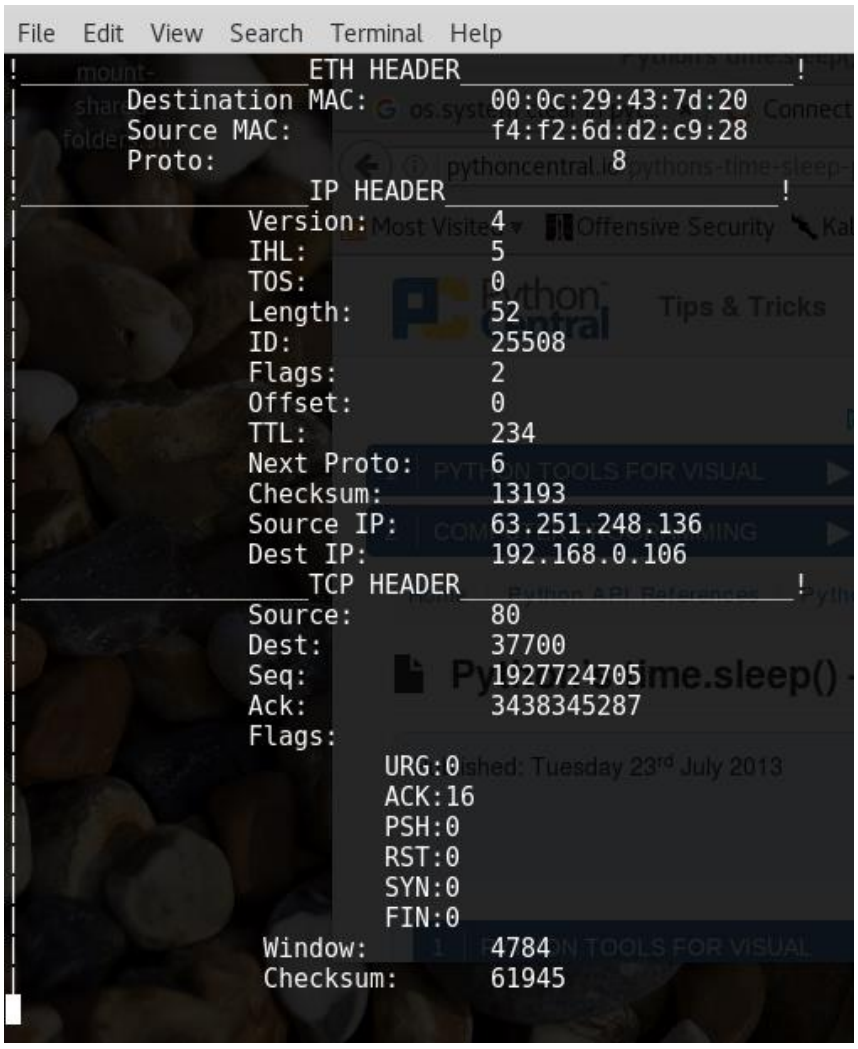
Hình 3.3: Chương trình PacketSniffer.py bắt gói tin UDP.



Hình 3.4: Chương trình PacketSniffer.py bắt gói tin TCP.



Hình 3.5: Chương trình PacketSniffer.py bắt gói tin TCP.



Hình 3.6: Chương trình PacketSniffer.py bắt gói tin TCP.

Danh mục tài liệu tham khảo

[1]. https://docs.python.org/2/library/os.html
[2]. https://tools.ietf.org/html/rfc793#page-15
[3]. https://www.ietf.org/rfc/rfc768.txt
[4]. https://en.wikipedia.org/wiki/Ethernet_frame
[5]. https://www.cybrary.it/video/packet-analyzer-part-2/