Computer Network – Project: Implementation of HTTP Sniffer

J201568008 김양선

- Introduction
+ code language: python3(>=3.4)
+ OS: Ubuntu 14.04
+ Library: Pcapy(>= 0.11)

-Block

Function name	Function describe	Variables	Exception handle
main	User must choose Network device name in main function. Open file for write a HTTP header, pcapy start live packet capture.	Input - argv	KeyboardInterrupt : It needs when stop this process by Keyboard Interrupt. It will saves data file.
payload_parser	Main function gives it a byte string. Then payload_parser function parsing a byte string to readable string list that split by (b'\r\n\r\n') and ('\r\n') after do this, encoding 'UTF-8'	Input - hdrpkt payload - file data_file - int idx output - a string data in file data_file return - int idx	UnicodeDecodeError : It needs when gets a image byte string.

- "Block by Block" Comments

```
import socket
from struct import unpack
import os
import pcapy
import sys



def payload_parser(payload, data_file, idx):
    try:
        eth_length = 14
        eth = unpack('!6s6sH', payload[:eth_length])
        eth_protocol = socket.ntohs(eth[2])
```

```
    if eth_protocol == 8:
        ip_header = payload[eth_length:20+eth_length]
        ip_header = unpack('!BBHHHBBH4s4s', ip_header)

        version_ih = ip_header[0]
        ip_header_length = (version_ih & 0xF) * 4
        protocol = ip_header[6]
        source_address = socket.inet_ntoa(ip_header[8])
        dest_address = socket.inet_ntoa(ip_header[9])

        if protocol == 6:
            tcp_header_index = ip_header_length + eth_length
            tcp_header = payload[tcp_header_index:tcp_header_index+20]
            tcp_header = unpack('!HHLLBBHHH', tcp_header)

            source_port = tcp_header[0]
            dest_port = tcp_header[1]
            data_offset_reserved = tcp_header[4]
            tcp_header_length = data_offset_reserved >> 4

            header_size = eth_length + ip_header_length +\
                tcp_header_length * 4

            data = payload[header_size:]
            data_list = data.partition(b'\r\n\r\n')[0]

            data_splited = str(data_list, 'utf-8').split('\r\n')
            if "HTTP/1.1" in data_splited[0]:
                data_file.write("#" + str(idx) + " " +
                            str(source_address) + ":" +
                            str(source_port) + " " +
                            str(dest_address) + ":" +
                            str(dest_port) + "\n")
                idx = idx + 1
                for string in data_splited:
                    data_file.write(string + "\n")

                data_file.write("\n")

except UnicodeDecodeError:
    pass
```

eth_protocol ==8:
means IP Protocol.

Calculate its header size.

IP header has source address & destination adress

Protocol == 6:
Means TCP Protocol.
TCP Protocol is also IP Protocol's body.

It Splits header and body.

TCP's body is http header.

Header size will calculated by byte.

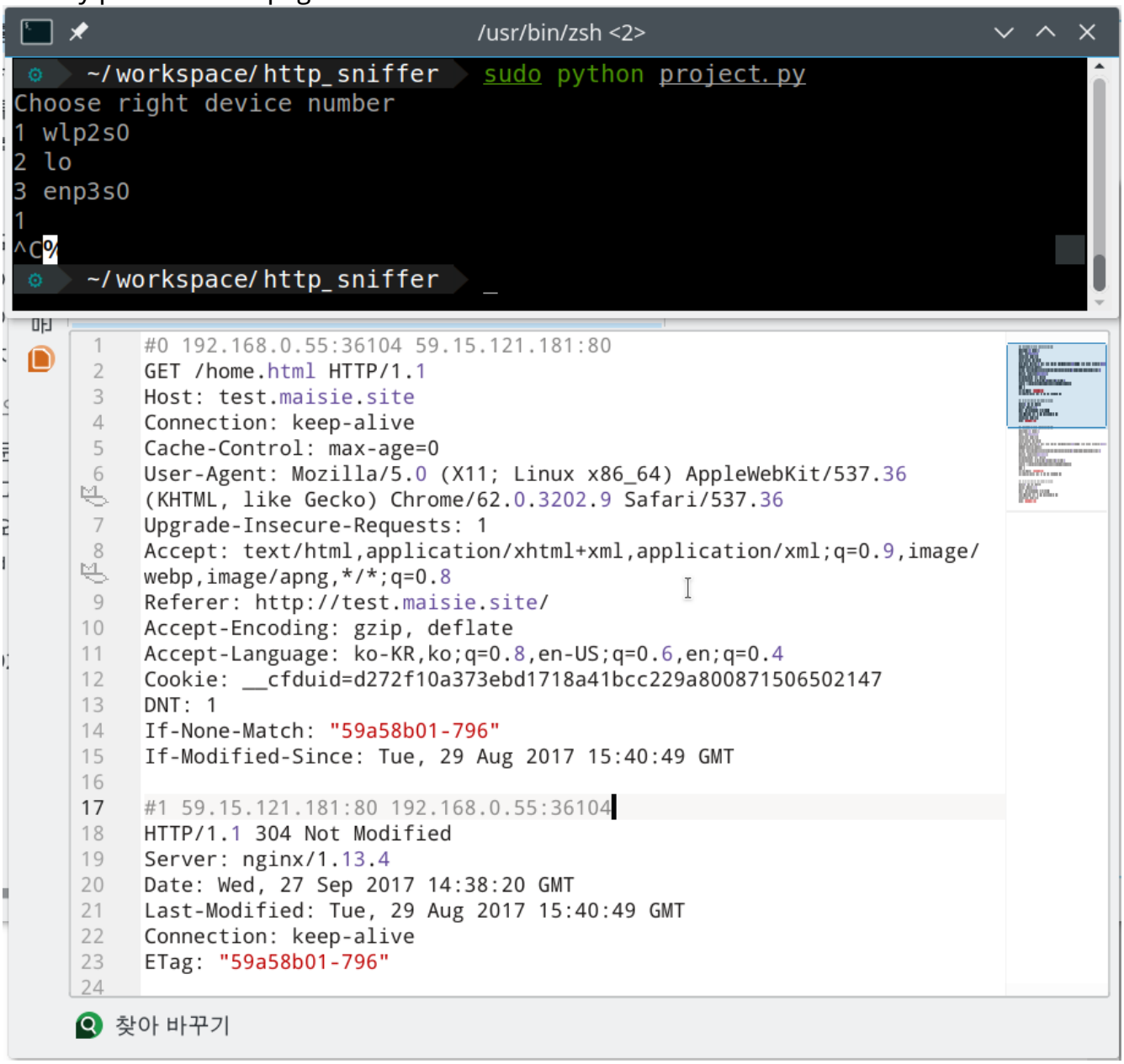
Protocol == 6:
Means TCP Protocol
It Splits header and body.
TCP's body is http header.
Header size will calculated by byte.

```
    return idx


def main(argv):
    print("Choose right device number")
    devices = os.listdir('/sys/class/net/')

    for index, device in enumerate(devices):
        print("{0} {1}".format(index+1, device))

    device_name = devices[int(input())-1]
    captured = pcapy.open_live(device_name, 65536, 1, 1)
    idx = 0

    (header, packet) = captured.next()
    data_file = open("./result.txt", "w")
    try:
        while header is not None:
            (header, packet) = captured.next()
            idx = payload_parser(packet, data_file, idx)
    except KeyboardInterrupt:
        data_file.close()
        pass


if __name__ == "__main__":
    main(sys.argv)
```

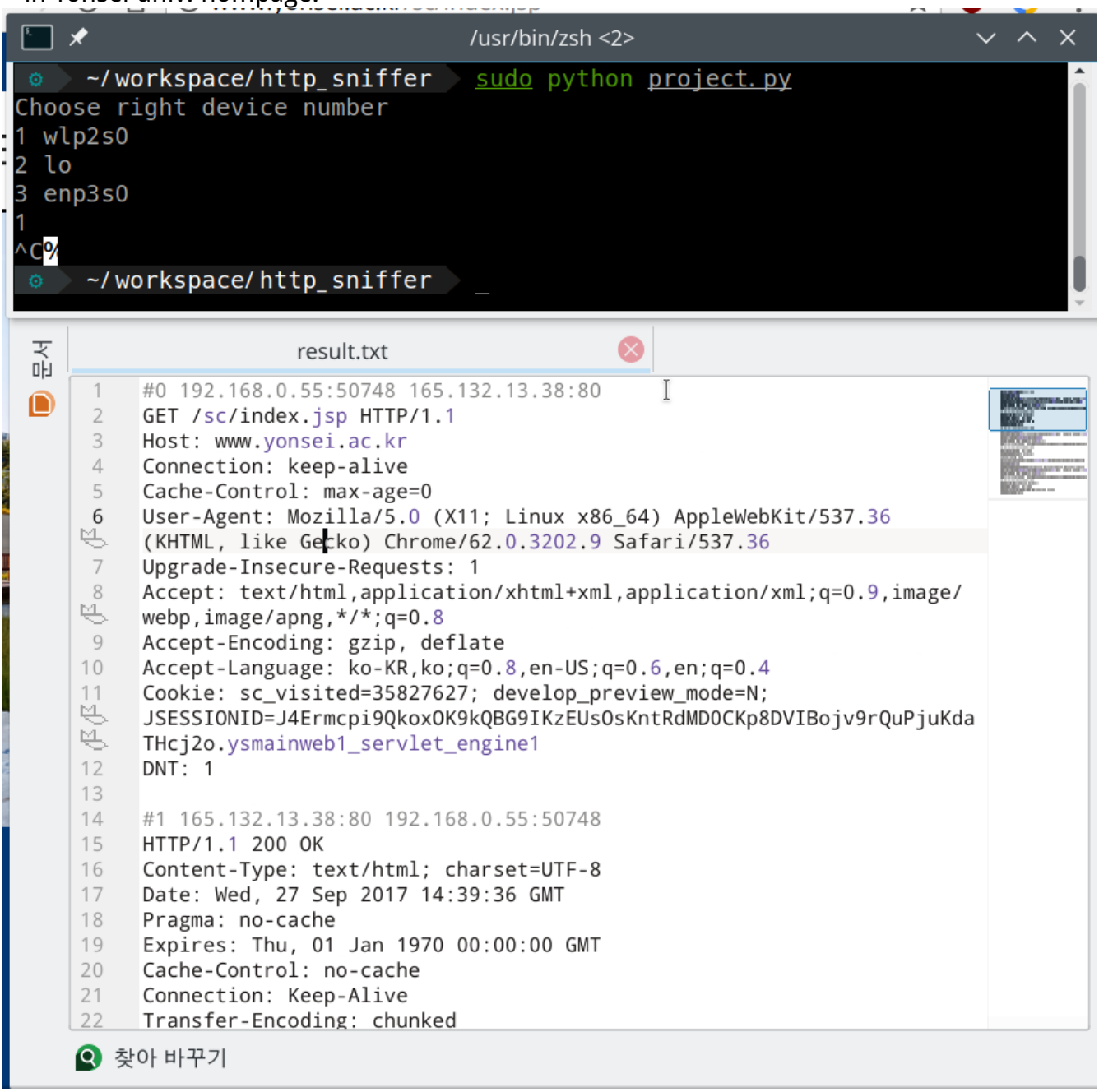
Read Network Interface's name.
User can choose device that want to sniff.

Device captured from here.
Pcapy.open_live needs device name, buffer byte, promisc, to_ms

- Screen Shots
+ in my personal hompage.



+ in Yonsei univ. hompage.



- References

+ Official Documentation(included in pcapy library file)
+ Pcapy github (https://github.com/CoreSecurity/pcapy)
+ Mozilla Developer Network(https://developer.mozilla.org/ko/docs/Web/HTTP/Basics_of_HTTP/MIME_types)