컴퓨터 네트워크 project1

2017110006 박지우

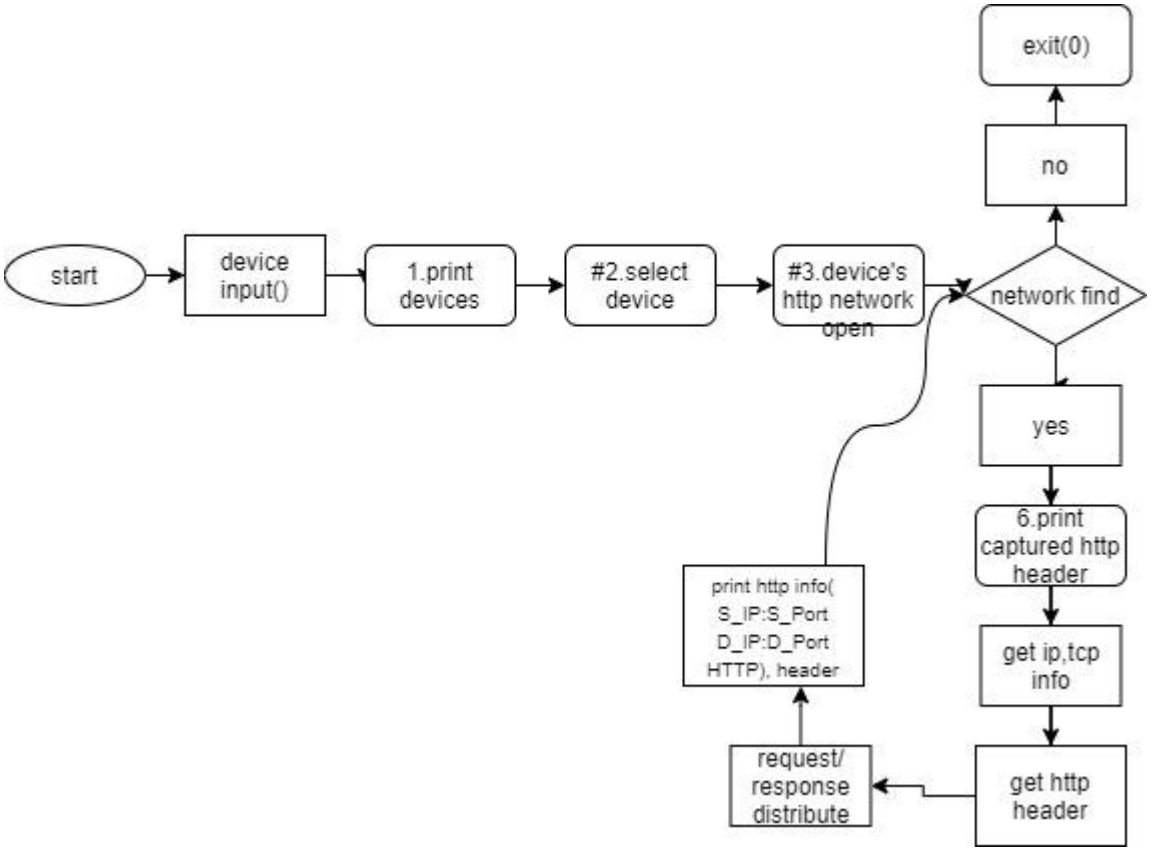
Software environment :
-windows
-vmware 16 player
-ubuntu18.04
-module: pip pcapy, pip struct

programming language you used:
Python

version: Python 3.8

reference:
struct : https://docs.python.org/ko/3/library/struct.html
pcapy: https://rawgit.com/CoreSecurity/pcapy/master/pcapy.html

Flow chart



•Logical explanations block by block in detail

```
from pcapy import *
from struct import *
```

-pcapy, struct 모듈 가져오기

```
# http number
no = 0
```

-각 헤더별 넘버 정의

```
# return data list to ints
def IP_addr(a) :
        b = []
        for i in range(len(a)):
                b.append("%.2d" % a[i])
        return ".".join(b)
```

-ip주소를 바이트에서 정수로 변환해 양식에 맞게 바꿈

```
# 6.print captured http header
```

-캡쳐된 패킷들을 재조립하는 함수

```
def cap_print(header, packet):

        # if not http, func close
        if packet.find(b'HTTP') == -1:
                return None
```

-패킷이 http가 아니면 넘어가기

```
global no
no += 1
```
-글로벌 변수선언, 패킷별로 넘버 다르게 지정

```
# -----Ethernet----------------------------------------
start_len = 0

eth_len = 14
```
-패킷의 레이어별 길이를 미리 정의

```
# -----IP----------------------------------------------

start_len += eth_len
```
-패킷 레이어별 길이를 하나씩 더함

```
ip_len = 20
ip_header = packet[start_len:start_len + ip_len]
ip = unpack('!BBHHBBBBH8B', ip_header)
```
-ip 헤더를 부위별로 미리 쪼개놓음

```
ip_len = (ip[0] & 0xF)*4
```
-ip 헤더를 판독해 ip의 길이 확인

```
# -----TCP---------------------------------------------
start_len += ip_len

tcp_len = 20
tcp_header = packet[start_len:start_len + tcp_len]
tcp = unpack('!HHLLBBHHH', tcp_header)
tcp_len = (tcp[4]>>4)*4
```
-마찬가지로 tcp도 동일하게 진행

```
# -----HTTP--------------------------------------------
start_len += tcp_len

#get http header
http = packet[start_len:]
rn_i = http.find(b'₩r₩n₩r₩n')
http = http[:rn_i]
```

-http 헤더에 해당하는 부분을 앞,뒤로 자르기

```
        # ------------------------------------------------------
        # select request/response
        state = ''
        if http.find(b'GET') != -1:
                state = 'Request'
                http = http[http.find(b'GET'):]
        elif http.find(b'POST') != -1:
                state = 'Request'
                http = http[http.find(b'POST'):]
        else:
                state = 'Response'
                http = http[http.find(b'HTTP'):]
```

-http 헤더의 종류에 따라 요청/응답 구별하기

-혹시 모르니 헤더 앞 부분 정리하기

```
        # print: ip protocol, sip,spt,dip,dpt
        print('%s %s:%s %s:%s HTTP %s' % (no, IP_addr(ip[9:13]), tcp[0], IP_addr(ip[13:17]), tcp[1],
state) )
        # print: http
        http = str(http, 'utf-8')
        print(http)

        print('₩r₩n₩r₩n')
```

-#No S_IP:S_Port D_IP:D_Port HTTP [Request|Response]₩r₩n [Request Line](or [Status Line])₩r₩n [Header Lines] ₩r₩n₩r₩n

의 양식에 맞게 필요한 정보들 프린트

```
#0.device search
devices = findalldevs()

#1.print devices
print('devices: ')
for idx in range(len(devices)):
        print("%d : %s" % (idx, devices[idx]))
```

-주변 디바이스 검색 후 차례로 보여주기

```
#2.select device
```

```
dv_idx = input('select device: ')
```
-input 으로 디바이스 숫자 고르기

```
#3.device's http network open
cap = open_live(devices[int(dv_idx)], 15000, 1, 20)
cap.setfilter('tcp port 80') #HTTP filter
```
-네트워크를 연 뒤, http 패킷을 골라내기

```
#4.exception : no device network
if cap is None:
        print('No Open Live!')
        exit(0)
```
-라이브가 존재하지 않으면 종료

```
print('₩r₩n')

#5.in captured http...
cap.loop(-1, cap_print)
```
-해당 네트워크 라이브 안에서 패킷 계속 가져오는거 반복