
Packet Sniffing 구현

목차

- 목표
 - 설계 환경
 - 설계 내용
 - 결과
 - 기대효과
-

목표

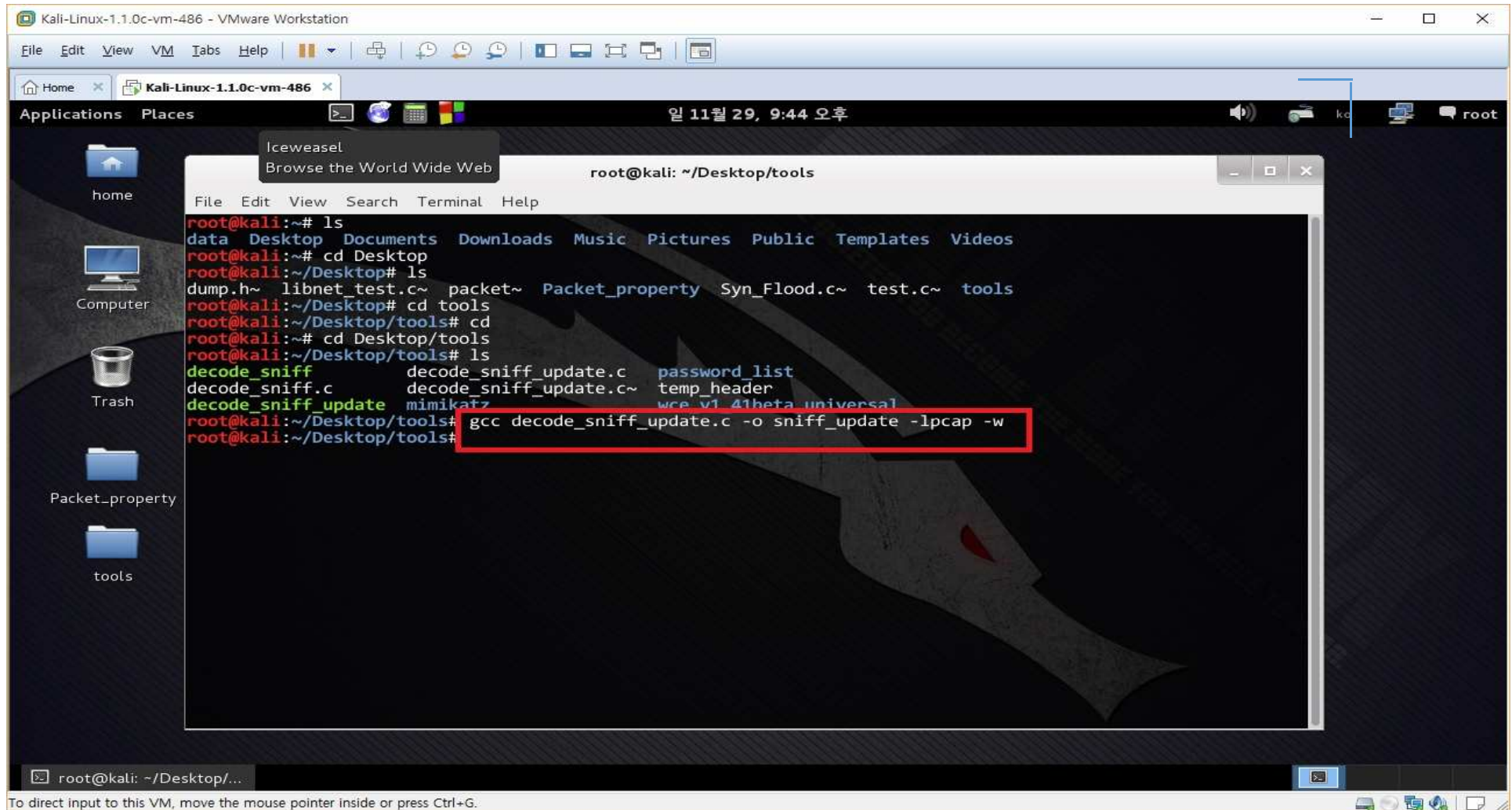
- ✓ Packet Sniffing 프로그램 구현
 - Wireshark에서 캡처되는 패킷 내용 중 일부분 출력
 - 각 계층 Header Size
 - 2계층 (출발/목적지 Mac Address)
 - 3계층 (출발/목적지 IP, Type, ID와 Length)
 - 4계층 (송/수신지 포트번호, Seq #, Ack #, Flags)
 - App data와 App data size 정보 출력
 - ✓ TCP기반의 패킷 정보 확인 및 TCP IP 3way handshaking 확인하기(http port 80)
 - ✓ UDP기반의 패킷 정보 확인(Nslookup 사용, Dns port 53)
 - ✓ WireShark와 Packet sniff Program 패킷 비교분석
-

설계 환경

- ✓ Kail Linux (Debian 계열) + Raw socket
 - ✓ Linux Version 3.18.0-kali3
 - ✓ Gcc Version 4.7.2(Debian 4.7.2.-5)
 - ✓ Debian 3.18.6-1~kail2(2015-03-02)
 - ✓ 패킷 캡처를 위한 pcap 라이브러리 사용
(sudo apt-get install libpcap-dev 명령어로 설치 후 사용)
-

설계내용

컴파일 화면

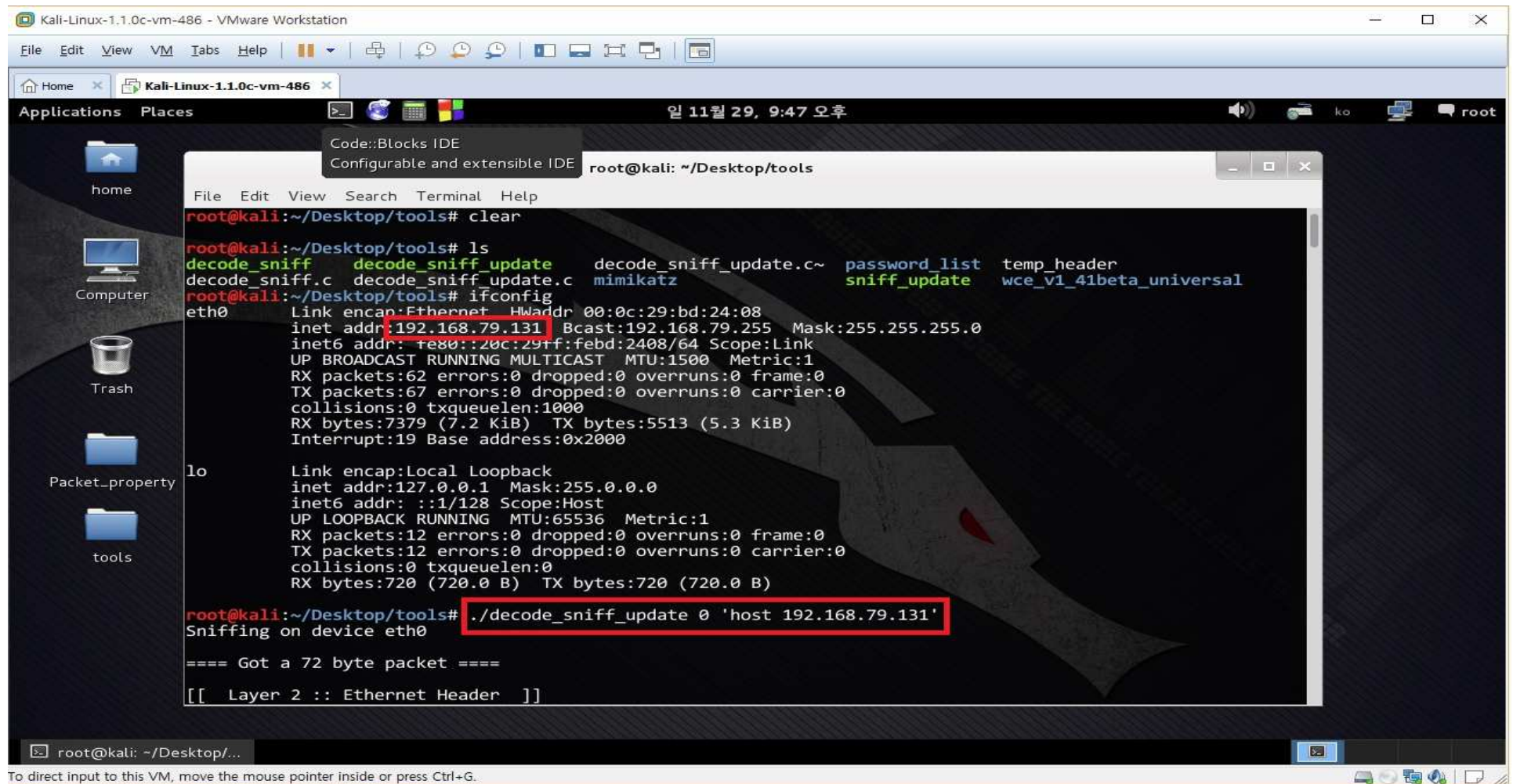


```
Kali-Linux-1.1.0c-vm-486 - VMware Workstation
File Edit View VM Tabs Help
Applications Places
home
Computer
Trash
Packet_property
tools
Iceweasel
Browse the World Wide Web
root@kali: ~/Desktop/tools
File Edit View Search Terminal Help
root@kali:~# ls
data Desktop Documents Downloads Music Pictures Public Templates Videos
root@kali:~# cd Desktop
root@kali:~/Desktop# ls
dump.h~ libnet_test.c~ packet~ Packet_property Syn_Flood.c~ test.c~ tools
root@kali:~/Desktop# cd tools
root@kali:~/Desktop/tools# cd
root@kali:~# cd Desktop/tools
root@kali:~/Desktop/tools# ls
decode_sniff decode_sniff_update.c password_list
decode_sniff.c decode_sniff_update.c~ temp_header
decode_sniff_update mimikatz wce_v1_41beta_universal
root@kali:~/Desktop/tools# gcc decode_sniff_update.c -o sniff_update -lpcap -w
root@kali:~/Desktop/tools#
```

Gcc decode_sniff_update.c -o sniff_update -lpcap -w

설계내용

Host IP로 스니핑하는 법



The screenshot shows a Kali Linux virtual machine running in VMware Workstation. The terminal window is open, displaying the following commands and output:

```
root@kali: ~/Desktop/tools
File Edit View Search Terminal Help
root@kali:~/Desktop/tools# clear
root@kali:~/Desktop/tools# ls
decode_sniff decode_sniff_update decode_sniff_update.c~ password_list temp_header
decode_sniff.c decode_sniff_update.c mimikatz sniff_update wce_v1_41beta_universal
root@kali:~/Desktop/tools# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:bd:24:08
          inet addr:192.168.79.131  Bcast:192.168.79.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:febd:2408/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:62 errors:0 dropped:0 overruns:0 frame:0
          TX packets:67 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:7379 (7.2 KiB)  TX bytes:5513 (5.3 KiB)
          Interrupt:19 Base address:0x2000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:12 errors:0 dropped:0 overruns:0 frame:0
          TX packets:12 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:720 (720.0 B)  TX bytes:720 (720.0 B)

root@kali:~/Desktop/tools# ./decode_sniff_update 0 'host 192.168.79.131'
Sniffing on device eth0

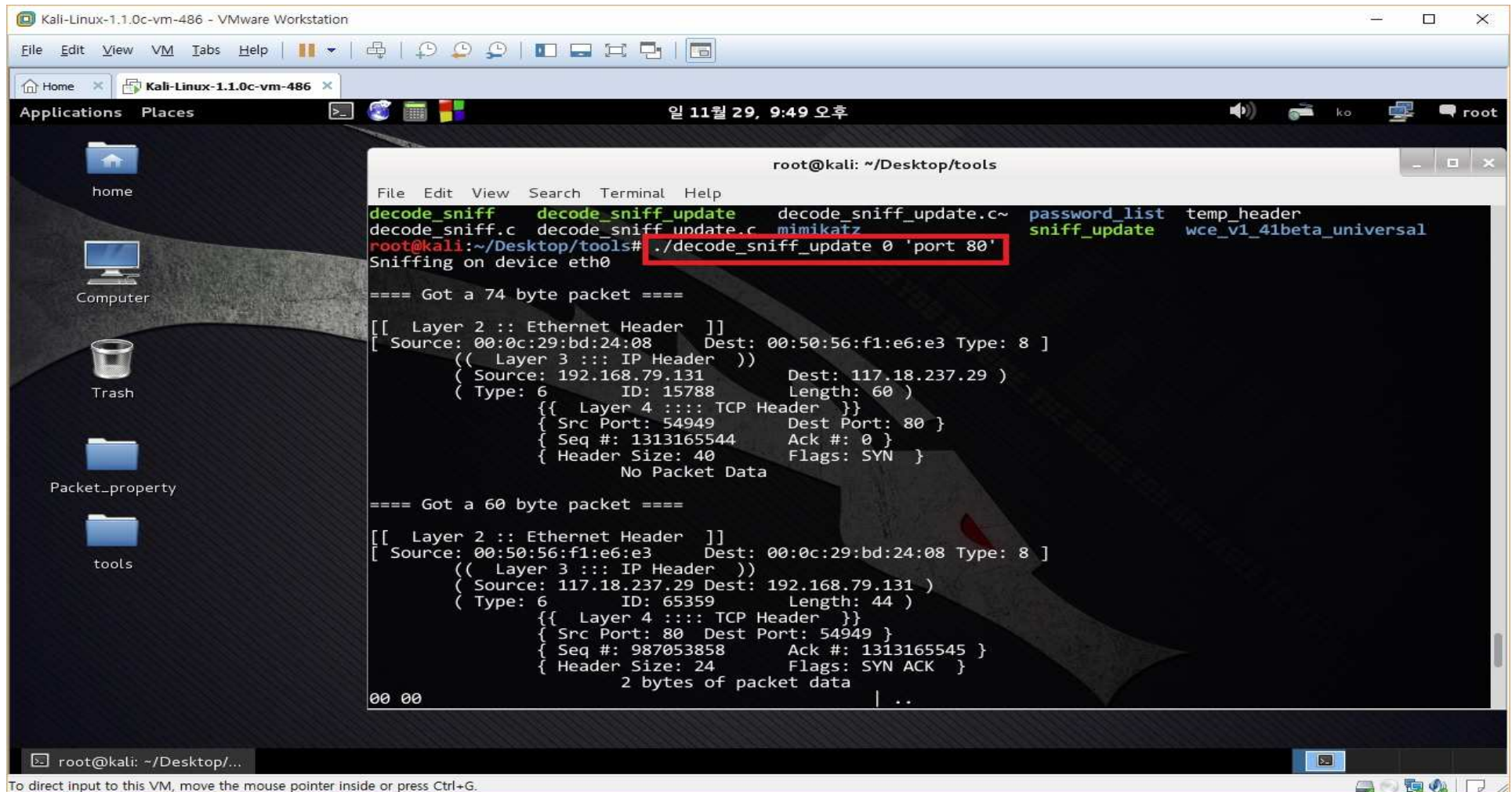
==== Got a 72 byte packet ====
[[ Layer 2 :: Ethernet Header ]]
```

The IP address 192.168.79.131 is highlighted in red in the terminal output, indicating the target host for sniffing.

— 호스트 IP를 확인 후 스니핑 대상을 선정

설계내용

Port number로 스니핑 하는 법



```
root@kali: ~/Desktop/tools
File Edit View Search Terminal Help
decode_sniff decode_sniff_update decode_sniff_update.c~ password_list temp_header
decode_sniff.c decode_sniff_update.c mimikatz sniff_update wce_v1_41beta_universal
root@kali:~/Desktop/tools# ./decode_sniff_update 0 'port 80'
Sniffing on device eth0

==== Got a 74 byte packet ====

[[ Layer 2 :: Ethernet Header ]]
[ Source: 00:0c:29:bd:24:08 Dest: 00:50:56:f1:e6:e3 Type: 8 ]
(( Layer 3 :: IP Header ))
( Source: 192.168.79.131 Dest: 117.18.237.29 )
( Type: 6 ID: 15788 Length: 60 )
{{ Layer 4 :: TCP Header }}
{ Src Port: 54949 Dest Port: 80 }
{ Seq #: 1313165544 Ack #: 0 }
{ Header Size: 40 Flags: SYN }
No Packet Data

==== Got a 60 byte packet ====

[[ Layer 2 :: Ethernet Header ]]
[ Source: 00:50:56:f1:e6:e3 Dest: 00:0c:29:bd:24:08 Type: 8 ]
(( Layer 3 :: IP Header ))
( Source: 117.18.237.29 Dest: 192.168.79.131 )
( Type: 6 ID: 65359 Length: 44 )
{{ Layer 4 :: TCP Header }}
{ Src Port: 80 Dest Port: 54949 }
{ Seq #: 987053858 Ack #: 1313165545 }
{ Header Size: 24 Flags: SYN ACK }
2 bytes of packet data
00 00 | ..
```

– 스니핑 대상을 80번 Port로 설정

```
==== Got a 74 byte packet ====
[[ Layer 2 :: Ethernet Header ]]
[ Source: 00:0c:29:bd:24:08 Dest: 00:50:56:f1:e6:e3 Type: 8 ]
  ( Layer 3 :: IP Header )
  ( Source: 192.168.79.131 Dest: 117.18.237.29 )
  ( Type: 6 ID: 15788 Length: 60 )
    {{ Layer 4 :: TCP Header }}
    { Src Port: 54949 Dest Port: 80 }
    { Seq #: 1313165544 Ack #: 0 }
    { Header Size: 40 Flags: SYN }
    No Packet Data

==== Got a 60 byte packet ====
[[ Layer 2 :: Ethernet Header ]]
[ Source: 00:50:56:f1:e6:e3 Dest: 00:0c:29:bd:24:08 Type: 8 ]
  ( Layer 3 :: IP Header )
  ( Source: 117.18.237.29 Dest: 192.168.79.131 )
  ( Type: 6 ID: 65359 Length: 44 )
    {{ Layer 4 :: TCP Header }}
    { Src Port: 80 Dest Port: 54949 }
    { Seq #: 987053858 Ack #: 1313165545 }
    { Header Size: 24 Flags: SYN ACK }
    2 bytes of packet data
00 00 | ..

==== Got a 54 byte packet ====
[[ Layer 2 :: Ethernet Header ]]
[ Source: 00:0c:29:bd:24:08 Dest: 00:50:56:f1:e6:e3 Type: 8 ]
  ( Layer 3 :: IP Header )
  ( Source: 192.168.79.131 Dest: 117.18.237.29 )
  ( Type: 6 ID: 15789 Length: 40 )
    {{ Layer 4 :: TCP Header }}
    { Src Port: 54949 Dest Port: 80 }
    { Seq #: 1313165545 Ack #: 987053859 }
    { Header Size: 20 Flags: ACK }
    No Packet Data
```

Flags 정보로부터 3-way임을 알 수 있다.

설계내용

UDP Packet Sniffing

The screenshot shows a Kali Linux virtual machine interface. On the left, a terminal window displays the output of the `nslookup www.google.com` command. The output lists multiple IP addresses for `www.google.com`, with the first one, `192.168.79.2`, highlighted by a red box. On the right, another terminal window shows the output of a packet capture tool, displaying the details of two captured UDP packets. The first packet is 74 bytes long, and the second is 330 bytes long. Both packets show the source and destination IP addresses and ports, along with the packet data in hexadecimal and ASCII format.

```
root@kali: ~# nslookup www.google.com
Server:      192.168.79.2
Address:     192.168.79.2#53

Non-authoritative answer:
Name:   www.google.com
Address: 1.255.33.231
Name:   www.google.com
Address: 1.255.33.251
Name:   www.google.com
Address: 1.255.33.222
Name:   www.google.com
Address: 1.255.33.221
Name:   www.google.com
Address: 1.255.33.236
Name:   www.google.com
Address: 1.255.33.242
Name:   www.google.com
Address: 1.255.33.226
Name:   www.google.com
Address: 1.255.33.216
Name:   www.google.com
Address: 1.255.33.237
Name:   www.google.com
Address: 1.255.33.217
Name:   www.google.com
Address: 1.255.33.246
Name:   www.google.com
Address: 1.255.33.212
Name:   www.google.com
Address: 1.255.33.247
Name:   www.google.com
```

```
root@kali: ~/Desktop/tools
==== Got a 74 byte packet ====
[[ Layer 2 :: Ethernet Header ]]
[ Source: 00:0c:29:bd:24:08 Dest: 00:50:56:f1:e6:e3 Type: 8 ]
(( Layer 3 :: IP Header ))
( Source: 192.168.79.131 Dest: 192.168.79.2 )
( Type: 17 ID: 62042 Length: 60 )
{{ Layer 4 :: UDP Header }}
{ Src Port: 39646 Dest Port: 53 }
{ Header Size: 8 }
32 bytes of packet data
78 c0 01 00 00 01 00 00 00 00 03 77 77 77 | X.....WWW
06 67 6f 6f 67 6c 65 03 63 6f 6d 00 00 01 00 01 | .google.com.....

==== Got a 330 byte packet ====
[[ Layer 2 :: Ethernet Header ]]
[ Source: 00:50:56:f1:e6:e3 Dest: 00:0c:29:bd:24:08 Type: 8 ]
(( Layer 3 :: IP Header ))
( Source: 192.168.79.2 Dest: 192.168.79.131 )
( Type: 17 ID: 65409 Length: 316 )
{{ Layer 4 :: UDP Header }}
{ Src Port: 53 Dest Port: 39646 }
{ Header Size: 8 }
288 bytes of packet data
78 c0 81 80 00 01 00 10 00 00 00 00 03 77 77 77 | X.....WWW
06 67 6f 6f 67 6c 65 03 63 6f 6d 00 00 01 00 01 | .google.com.....
c0 0c 00 01 00 01 00 00 00 05 00 04 01 ff 21 e7 | .....!..
c0 0c 00 01 00 01 00 00 00 05 00 04 01 ff 21 fb | .....!..
c0 0c 00 01 00 01 00 00 00 05 00 04 01 ff 21 de | .....!..
c0 0c 00 01 00 01 00 00 00 05 00 04 01 ff 21 dd | .....!..
c0 0c 00 01 00 01 00 00 00 05 00 04 01 ff 21 ec | .....!..
c0 0c 00 01 00 01 00 00 00 05 00 04 01 ff 21 f2 | .....!..
c0 0c 00 01 00 01 00 00 00 05 00 04 01 ff 21 e2 | .....!..
c0 0c 00 01 00 01 00 00 00 05 00 04 01 ff 21 d8 | .....!..
c0 0c 00 01 00 01 00 00 00 05 00 04 01 ff 21 ed | .....!..
c0 0c 00 01 00 01 00 00 00 05 00 04 01 ff 21 d9 | .....!..
```

Nslookup에 따른 UDP Packet 캡처

설계내용

Wireshark와 Packet sniff Program 캡처 정보 비교

The screenshot displays a Kali Linux virtual machine environment. On the left, Wireshark is running, capturing traffic on the eth0 interface. The filter is set to 'ip.addr == 192.168.79.131 && udp.port == 53'. The packet list shows two packets: a DNS standard query (73 bytes) and a DNS standard query response (139 bytes). The packet details pane shows the structure of the second packet, including Ethernet II, IP, and UDP headers, and the domain name system response.

On the right, a terminal window shows the output of a packet sniffing program. It displays the raw packet data in hexadecimal and ASCII, along with the packet structure details, matching the information shown in Wireshark.

```
==== Got a 73 byte packet ====
[[ Layer 2 :: Ethernet Header ]]
[ Source: 00:0c:29:bd:24:08 Dest: 00:50:56:f1:e6:e3 Type: 8 ]
( Layer 3 :: IP Header )
( Source: 192.168.79.131 Dest: 192.168.79.2 )
( Type: 17 ID: 4979 Length: 59 )
{{ Layer 4 :: UDP Header }}
{ Src Port: 50578 Dest Port: 53 }
{ Header Size: 8 }
31 bytes of packet data
a5 c3 01 00 00 01 00 00 00 00 00 03 77 77 77 .....WWW
05 6e 61 76 65 72 03 63 6f 6d 00 00 01 00 01 .naver.com....

==== Got a 139 byte packet ====
[[ Layer 2 :: Ethernet Header ]]
[ Source: 00:50:56:f1:e6:e3 Dest: 00:0c:29:bd:24:08 Type: 8 ]
( Layer 3 :: IP Header )
( Source: 192.168.79.2 Dest: 192.168.79.131 )
( Type: 17 ID: 55646 Length: 125 )
{{ Layer 4 :: UDP Header }}
{ Src Port: 53 Dest Port: 50578 }
{ Header Size: 8 }
97 bytes of packet data
a5 c3 81 80 00 01 00 03 00 00 00 00 03 77 77 77 .....WWW
05 6e 61 76 65 72 03 63 6f 6d 00 00 01 00 01 c0 .naver.com.....
0c 00 05 00 01 00 00 05 00 16 03 77 77 77 05 .....WWW.
6e 61 76 65 72 03 63 6f 6d 05 6e 68 65 6f 73 c0 naver.com.nheos.
16 c0 2b 00 01 00 01 00 00 05 00 04 7d d1 de ..+.....}..
8d c0 2b 00 01 00 01 00 00 05 00 04 ca b3 b1 ..+.....}..
16 ..+.....}..

```

— 헤더 정보들을 제외한 나머지 정보들이 같음을 알 수 있다.

결과

- ✓ 각 계층별로 요약된 정보를 확인 할 수 있음
 - ✓ UDP와 TCP에서의 패킷 정보를 확인할 수 있음
 - ✓ WireShark와 Packet sniff Program이 스니핑한 패킷 정보가 같다는 것을 알 수 있음.
-

기대 효과

- ✓ 실시간으로 Packet 캡처
 - ✓ 스니핑 대상은 현재 실행하는 컴퓨터가 속해있는 네트워크 내 모든 컴퓨터가 대상이 될 수 있음.
 - ✓ 특정 호스트 및 특정 포트에 대한 패킷 필터링 가능
 - ✓ TCP의 경우 Flags를 바탕으로 Packet 형태, 상태 등을 알 수 있음
-

감사합니다