

실습목표

- 강의에서 활용된 자료 기반 직접 sniffing 공격을 따라해 봄으로써 공격 원리를 학습함

유의사항

- 모든 동작 결과 화면은 full screen 을 캡쳐한다. 가상 환경 사용자 이름은 학번으로 대체 하라

제출물

- 동작 원리와 각 동작 화면 및 공격 결과를 포함하는 보고서를 제출한다. 또한 동작화면과 결과에는 분석이 반드시 함께 서술되어야 한다.

실습 1

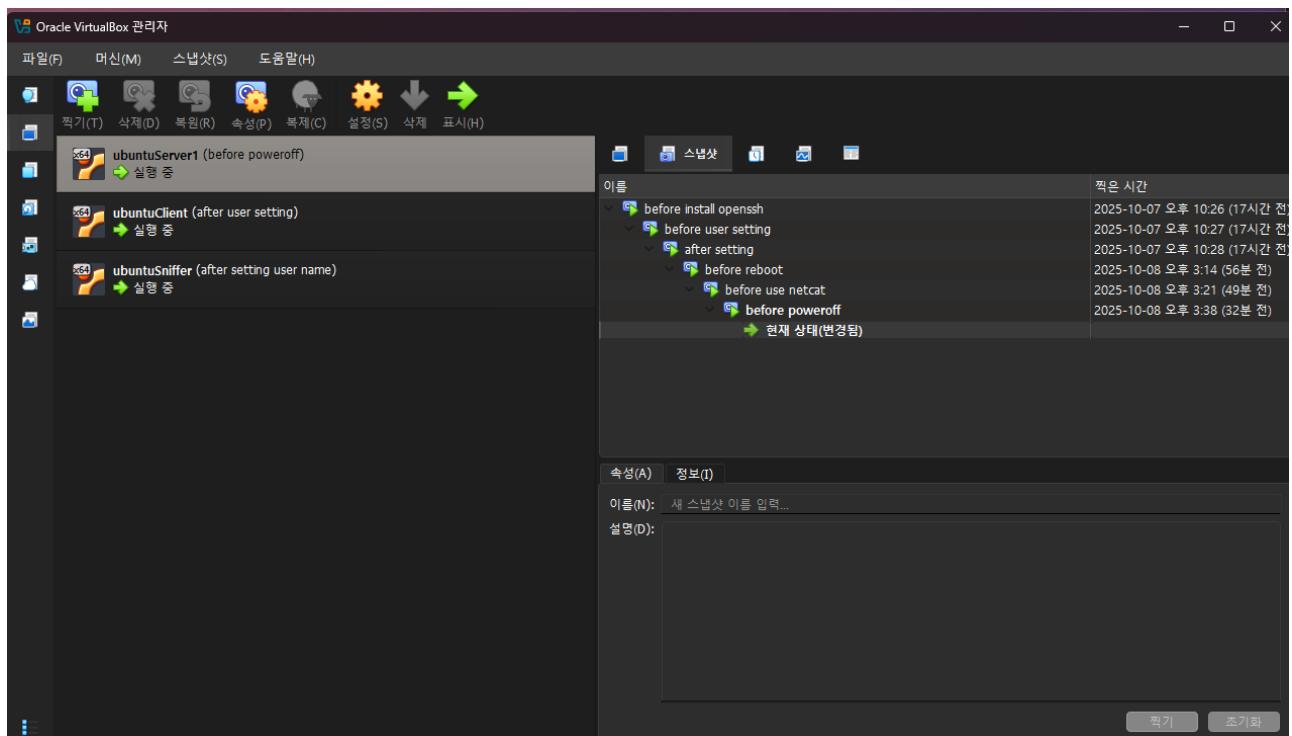
- 강의자료 3p~10p를 참고하여 TCP Dump를 사용해 sniffing 공격을 수행하라

실습 준비

hypervisor

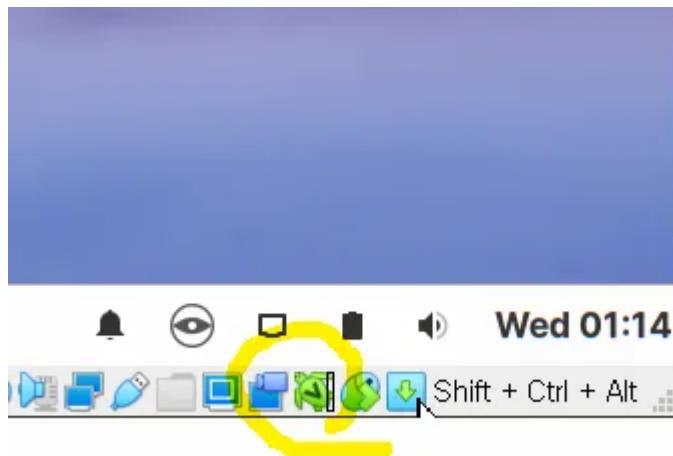
virtual machine 구성

- virtualbox 이용 가상화를 통해 윈도우 기반 랩톱 위에 ubuntu 등 운영체제를 올려서 사용
- 실습 1을 위해 3개의 가상머신 ubuntu server, ubuntu client, ubuntu sniffer 를 사용

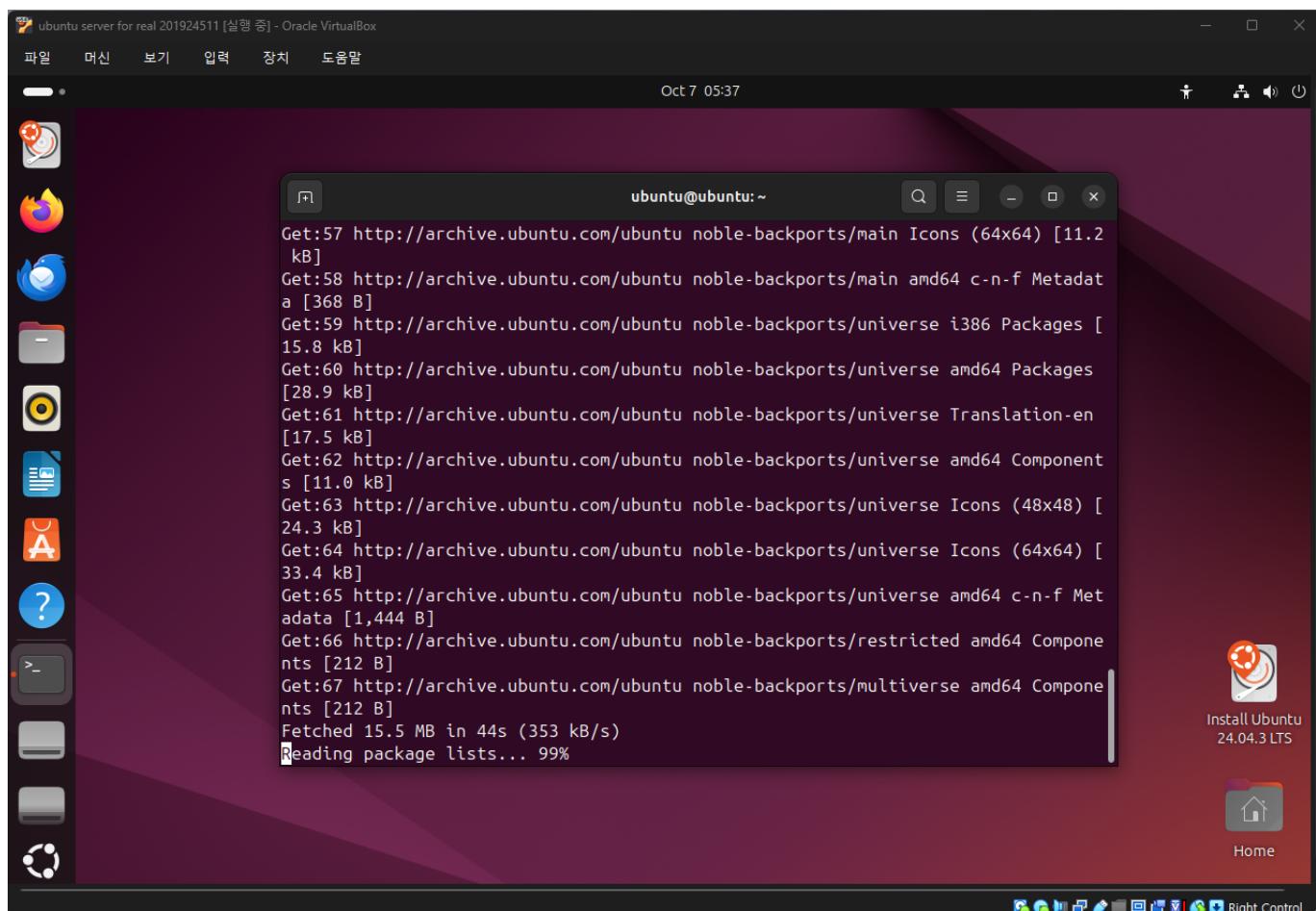


virtualbox

- oracle 에서 만든 무료 가상화 툴
- 윈도우 호스트에서 거북이 문제 발생 시 다음 방법으로 해결



turtle solution



- 문제해결 시 위와같이 V 아이콘이 뜨며 멈춤현상이 줄어드는 모습을 볼 수 있음

virtual machine 하드웨어 설정

- 램: 2GB , CPU: 2GHz, HDD: 20GB 사용

virtual machine 네트워크 설정

- NAT: 10.0.2.15 고정. server-2222/22, client-2223/22, sniffer-2224/22 (호스트-게스트 통신에 사용)
 - 호스트 전용 어댑터: server-192.158.56.102, client-192.168.56.105, sniffer-192.158.56.107 (게스트-게스트 간 통신에 사용)

ssh

- virtual box 무료버전의 한계로 인해 GUI에서 작업 시 작업 멈춤현상 발생
 - openssh + putty를 이용해서 window 환경에서 cli를 통해 작업

운영체제

- promiscuous mode 지원
 - 강의자료에서는 client system을 ubuntu desktop 14로, telnet server system을 ubuntu server 16으로 사용
 - 실험에서는 모두 ubuntu 22.04 Its 버전 사용

tcp dump

- 기본적이고 강력한 리눅스 스니핑 도구. 네트워크 관리툴로 개발
 - 설치: sudo apt-get install tcpdump
 - 강의자료에서는 sudo tcpdump -i eth0 -xX host 192.168.0.2 로 dump
 - 실험에서는 eth0이 아닌 enp0s8을 호스트간 통신에 사용함으로 이를 통해 대체
 - ip주소는 server주소 사용. client 주소로 해도 동일할 것으로 추정

telnet

- server 와 원격으로 연결하는 웰. 패킷 메시지가 암호화되지 않아 보안 취약
 - 연결: telnet 192.168.0.2 (서버 주소)
 - 분석: telnet, ftp 같은 초기 서비스들은 account, password 등의 정보를 plain text 형태로 전송

실습 과정

virtual machine setting 공통

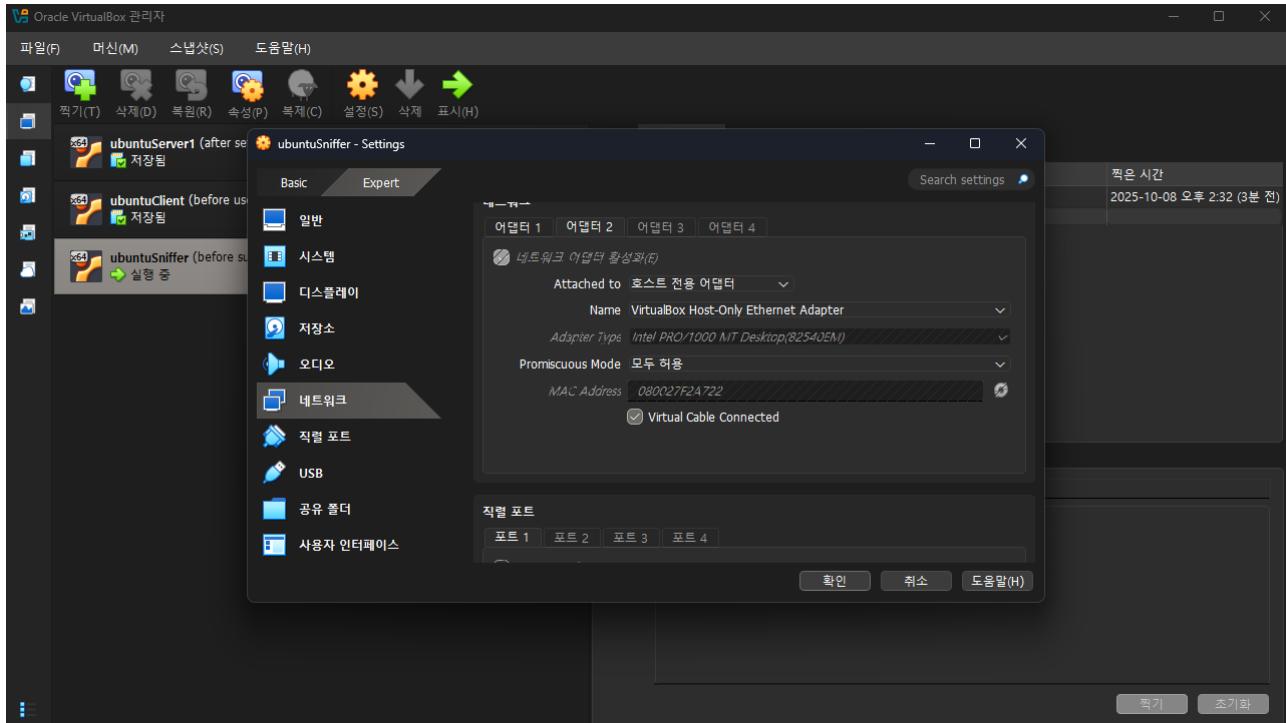
- 각 virtual machine 에서 sudo apt update 및 sudo apt install openssh-server 동작

```
ubuntu@ubuntu:~$ ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:f3:cd:76 brd ff:ff:ff:ff:ff:ff
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:b0:cc:9b brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.102/24 brd 192.168.56.255 scope global dynamic noprefixroute enp0s8
        valid_lft 400sec preferred_lft 400sec
    inet6 fe80::a00:27ff:feb0:cc9b/64 scope link
        valid_lft forever preferred_lft forever
```

- 위와같이 enp0s3에 ip 주소가 할당되지 않은 경우 sudo apt update 등이 제대로 동작하지 않음
- 이 때 nat 네트워크 수동설정 by sudo ip link set dev enp0s3 up << 요걸로 enp0s3 활성화, sudo ip addr add 10.0.2.15/24 dev enp0s3, sudo ip route add default via 10.0.2.2, echo "nameserver 8.8.8.8" | sudo tee /etc/resolv.conf, echo "nameserver 8.8.4.4" | sudo tee -a /etc/resolv.conf
- 과제 요구사항에 맞게 사용자 이름 설정 by sudo adduser ubuntuclient201924511, sudo usermod -aG sudo ubuntuclient201924511

virtual machine setting sniffer

- sniffer의 경우 다른 ip로 가는 패킷을 sniffing하기 위해 promiscuous mode 를 켜야함



- 먼저 virtualbox-네트워크-어댑터2(호스트전용)-promiscuous 부분에서 모두허용으로 설정
- vm 내부에서 promiscuous on by sudo ip link set enp0s8 promisc on

virtual machine setting server

- telnet 설정: server 23번 포트 사용
 - telnet /usr/sbin/in.telnetd 관련 문제: 설정파일이 없다고 뜨는 경우 sbin에서 ls 후 비슷한 이름을 찾아 대체해볼 것. (ex. /usr/sbin/telnetd)

실습 결과

서버 로그인

account dump

- 실습에 사용된 계정은 ubuntuuser 계정인 ubuntu201924511

- 위의 결과를 분석해보면 ack166~186 까지 계정명이 plain text로 출력된 것을 볼 수 있음

password dump

- 실습에 사용된 계정 계정인 `ubuntuserver201924511` 의 password 는 `so` 로 시작함

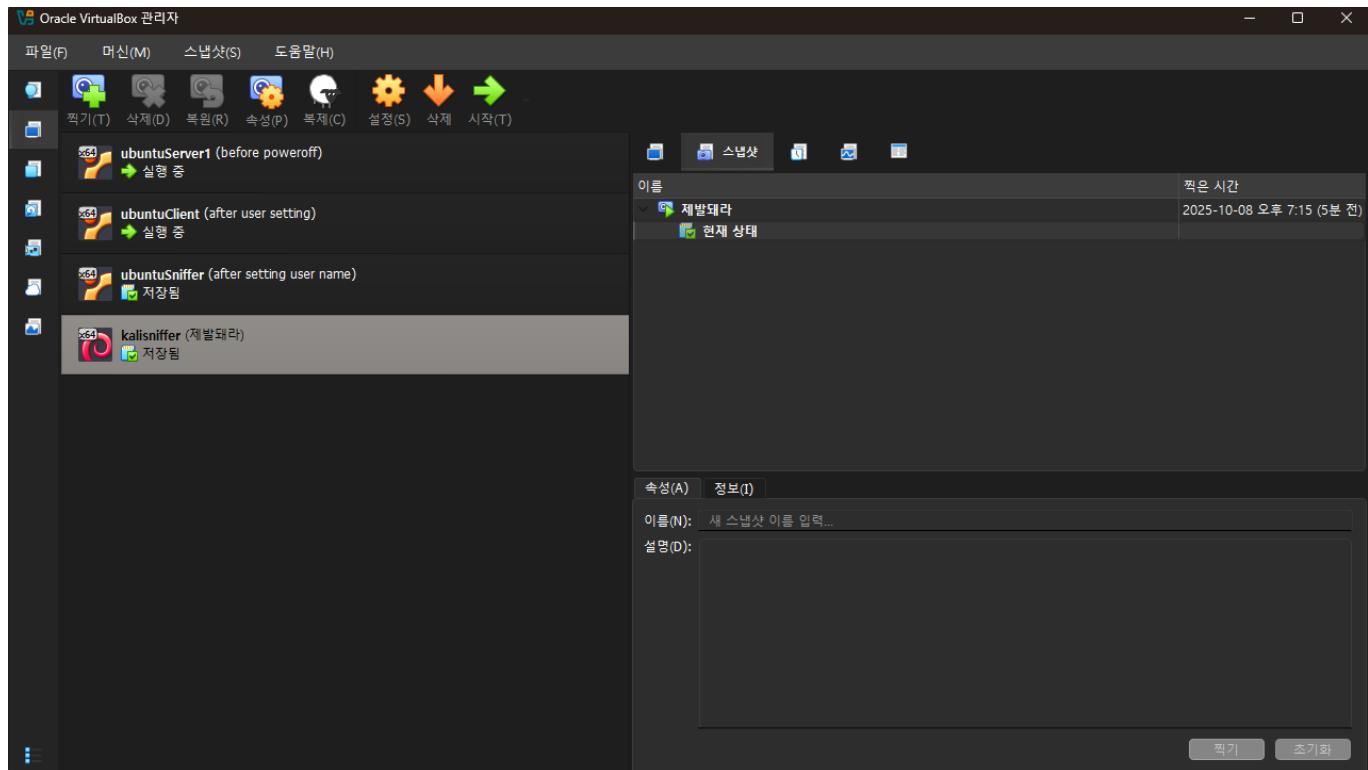
- ack 240부터 password가 plain text로 출력된 것을 볼 수 있음

실습 2

- 강의자료 11p~15p를 참고하여 dsniff를 사용해 sniffing 공격을 수행하라

실습 준비

hypervisor 설정 동일



운영체제

- kali linux 2025.3 사용
- dsniff, arpspoof 등 다양한 툴 기본제공

telnet 사용

실습 과정

virtual machine setting

```

$ sudo ip link set eth1 promisc on
(kalisniffer201924511@kalisniffer201924511)-[~]
$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
      inet 10.0.2.15 netmask 255.255.255.0 broadcast 10.0.2.255
      inet6 fd17:625c:f037:2:a00:27ff:fe67:817 prefixlen 64 scopeid 0x0<global>
      inet6 fd17:625c:f037:2:6127:c989:7fc5:8c80 prefixlen 64 scopeid 0x0<global>
      inet6 fe80::a00:27ff:fe67:817 prefixlen 64 scopeid 0x20<link>
      ether 08:00:27:67:08:17 txqueuelen 1000 (Ethernet)
      RX packets 87 bytes 37931 (37.0 KiB)
      RX errors 0 dropped 0 overruns 0 frame 0
      TX packets 145 bytes 14159 (13.8 KiB)
      TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

eth1: flags=4419<UP,BROADCAST,RUNNING,PROMISC,MULTICAST> mtu 1500
      inet 192.168.56.109 netmask 255.255.255.255 broadcast 0.0.0.0
      ether 08:00:27:7c:ac:7c txqueuelen 1000 (Ethernet)
      RX packets 113 bytes 10251 (10.0 KiB)
      RX errors 0 dropped 0 overruns 0 frame 0
      TX packets 0 bytes 0 (0.0 B)
      TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
      inet 127.0.0.1 netmask 255.0.0.0
      inet6 ::1 prefixlen 128 scopeid 0x10<host>

```

- ubuntu 와 다르게 GUI환경 사용이 원활하여 GUI를 사용하여 진행
- eth0: 호스트와 통신. 10.0.2.15 사용. 따로 설정없이 virtualbox 네트워크 란에서 포트포워딩 룰만 추가. (2225 > 22)
- eth1: 게스트간 통신. 192.168.56.109. vm 내부에서 192.168.56.109로 수동 설정.

virtual machine setting server 동일

실험결과

```

ubuntuServer1 login: ^CConnection closed by foreign host.
ubuntuclient201924511@ubuntuClient:~$ telnet 192.168.56.102
Trying 192.168.56.102...
Connected to 192.168.56.102.
Escape character is '^J'.
Linux 6.14.0-27-generic (ubuntuServer1) (pts/4)

ubuntuServer1 login: ubuntuserver201924511
Password:
Login incorrect
ubuntuServer1 login: Connection closed by foreign host.
ubuntuclient201924511@ubuntuclient:~$ 

```

- 실습 1과 같이 client에서 server로 telnet을 이용하여 login 을 시도

The screenshot shows a terminal window titled "kalisniffer201924511@kalisniffer201924511: ~". The terminal displays the following command-line session:

```

inet6 fe80::a00:27ff:fe67:817/64 scope link noprefixroute
    valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
        link/ether 08:00:27:7c:ac:7c brd ff:ff:ff:ff:ff:ff
        inet 192.168.56.109/32 scope global eth1
            valid_lft forever preferred_lft forever

(kalisniffer201924511@kalisniffer201924511) [~]
$ ip neigh
10.0.2.0 dev eth0 lladdr 52:55:0a:00:02:02 STALE
fd17:625c:f037:2::2 dev eth0 lladdr 52:56:00:00:00:02 router STALE
fe80::2 dev eth0 lladdr 52:56:00:00:00:02 router STALE

(kalisniffer201924511@kalisniffer201924511) [~]
$ sudo dsniff -i 192.168.56.102
dsniff: nids_init: 192.168.56.102: No such device exists (No such device exists)

(kalisniffer201924511@kalisniffer201924511) [~]
$ sudo dsniff -i eth1
dsniff: listening on eth1

10/08/25 19:13:15 tcp 192.168.56.105.34856 → 192.168.56.102.23 (telnet)
ubuntu server201924511
helloworld

```

- eth1 과 연결된 host-only 회선을 사용함으로 prom mode 로 패킷수집 및 정제

실습목표

- ARP spoofing 공격을 통해 희생자 노드의 패킷을 sniffing하고 패킷을 변조시켜 전파한다. 또한, 패킷 분석 툴을 이용해 동작 과정을 관찰하고 ARP spoofing 방지 기법에 대해 학습한다.

유의사항

- 모든 동작 결과 화면은 full screen 을 캡쳐한다. 가상 환경 사용자 이름은 학번으로 대체 하라

제출물

- 동작 원리와 각 동작 화면 및 공격 결과를 포함하는 보고서를 제출한다.

실습 1: 공격 발생 전 서버에 ICMP 패킷 전송

- 공격 수행 전, 모든 노드의 network를 "NAT"에서 "Host-only"로 변경하여 고립시킨다
- 클라이언트에서 ping을 통해 서버와 통신한다
- 서버 노드에서 wireshark를 통해 클라이언트 ping packet의 mac address를 확인한다

1번 실습준비

hypervisor

virtual machine 구성

- client, server 용 ubuntu 가상머신 2대

virtual machine network

- NAT + host-only 구성
- 과제 지시사항에서는 host-only의 고립된 환경
- 하지만 virtualbox 특성 상 상태가 저장된 vm의 네트워크 설정을 변경할 수 없음
- 초기부터 host-only 상태라면 wireshark 등 툴을 활용할 수 없음

tshark

- 통신중인 패킷 흐름을 살펴보는 툴인 wire shark cli 버전
- 해당 실습에서는 packet mac 주소를 분석하기 위해 사용

1번 실습과정

tshark

- 설치 by sudo apt install tshark -y

The screenshot shows two terminal windows side-by-side. The left window is titled 'ubuntuServer201924511@ubuntuServer:' and displays a tshark capture of ICMP traffic between two hosts. The right window is titled 'ubuntuClient201924511@ubuntuClient:' and shows ping statistics from the client to the server.

```

ubuntuServer201924511@ubuntuServer:~ 
1d=0x4f46, seq=13/320, ttl=64
 30 12.273774320 192.168.56.108 -> 192.168.56.105 ICMP 98 Echo (ping) reply
1d=0x4f46, seq=13/320, ttl=64 (request in 29)
 31 13.297767858 192.168.56.105 -> 192.168.56.108 ICMP 98 Echo (ping) request
1d=0x4f46, seq=14/3584, ttl=64
 32 14.314551994 192.168.56.108 -> 192.168.56.105 ICMP 98 Echo (ping) reply
1d=0x4f46, seq=14/3584, ttl=64 (request in 31)
 33 14.321437677 192.168.56.105 -> 192.168.56.108 ICMP 98 Echo (ping) request
1d=0x4f46, seq=15/3940, ttl=64
 34 14.321459867 192.168.56.105 -> 192.168.56.105 ICMP 98 Echo (ping) reply
1d=0x4f46, seq=15/3940 (request in 33)
 35 15.345549710 192.168.56.105 -> 192.168.56.108 ICMP 98 Echo (ping) request
1d=0x4f46, seq=16/4096, ttl=64
 36 15.34551994 192.168.56.108 -> 192.168.56.105 ICMP 98 Echo (ping) reply
1d=0x4f46, seq=16/4096 (request in 35)
 37 16.365088231 192.168.56.105 -> 192.168.56.108 ICMP 98 Echo (ping) request
1d=0x4f46, seq=17/4352, ttl=64
 38 16.365081284 192.168.56.108 -> 192.168.56.105 ICMP 98 Echo (ping) reply
1d=0x4f46, seq=17/4352, ttl=64 (request in 37)
*C88 packets captured
ubuntuServer201924511@ubuntuServer:~ $ sudo tshark -i enp0s8 -f "icmp" -T fields
- e frame -e eth.dst -e ip.src -e ip.dst
Running on interface 'enp0s8' (group "dangerous")
Capturing on 'enp0s8'
1 08:00:27:00:99:64 08:00:27:bb:f3:13 192.168.56.105 192.168.
56.108
2 08:00:27:bb:f3:13 08:00:27:00:99:64 192.168.56.108 192.168.
56.105
3 08:00:27:00:99:64 08:00:27:bb:f3:13 192.168.56.105 192.168.
56.108
4 08:00:27:bb:f3:13 08:00:27:00:99:64 192.168.56.108 192.168.
56.105
5 08:00:27:00:99:64 08:00:27:bb:f3:13 192.168.56.105 192.168.
56.108
6 08:00:27:bb:f3:13 08:00:27:00:99:64 192.168.56.108 192.168.
56.105
7 08:00:27:00:99:64 08:00:27:bb:f3:13 192.168.56.105 192.168.
56.108
8 08:00:27:bb:f3:13 08:00:27:00:99:64 192.168.56.108 192.168.
56.105
*C
 192.168.56.108 ping statistics ---
 7 packets transmitted, 7 received, 0% packet loss, time 16370ms
rtt min/avg/max/mdev = 0.176/0.196/0.272/0.021 ms
ubuntuClient201924511@ubuntuClient:~$ ping 3 192.168.56.108
PING 192.168.56.108 (192.168.56.108) 56(84) bytes of data.
3 192.168.56.108 ping statistics ---
 9 packets transmitted, 0 received, 100% packet loss, time 18439ms
ubuntuClient201924511@ubuntuClient:~$ ping 192.168.56.108
PING 192.168.56.108 (192.168.56.108) 56(84) bytes of data.
4 192.168.56.108 ping statistics ---
 4 packets transmitted, 4 received, 0% packet loss, time 3061ms
rtt min/avg/max/mdev = 0.187/0.197/0.224/0.015 ms
ubuntuClient201924511@ubuntuClient:~$ 

```

- 데이터 추출 by sudo tshark -i enp0s8 -f "icmp" -T fields -e eth.src -e eth.dst -e ip.src -e ip.dst
- 순서대로 client mac, server mac, client ip, server ip
- client mac(08:00:27:00:99:64), server mac(08:00:27:bb:f3:13) 확인 가능

실습 2: ettercap 툴을 통한 ARP Spoofing 공격 수행

- 공격자 노드에서 기존에 설치했던 ettercap graphical 툴을 실행한다
- 클라이언트에서 Mac Address Table 을 확인하고, 서버로 ping을 전송한다
- 공격자 노드에서 wireshark를 통해 수집한 ping packet의 mac address 변화를 확인한다

2번 실습준비

hypervisor

virtual machine 구성

- client, server 용 ubuntu 가상머신 2대 구성 동일
- 추가로 spoofer 용 ubuntu 가상머신 1대

virtual machine network 구성 동일

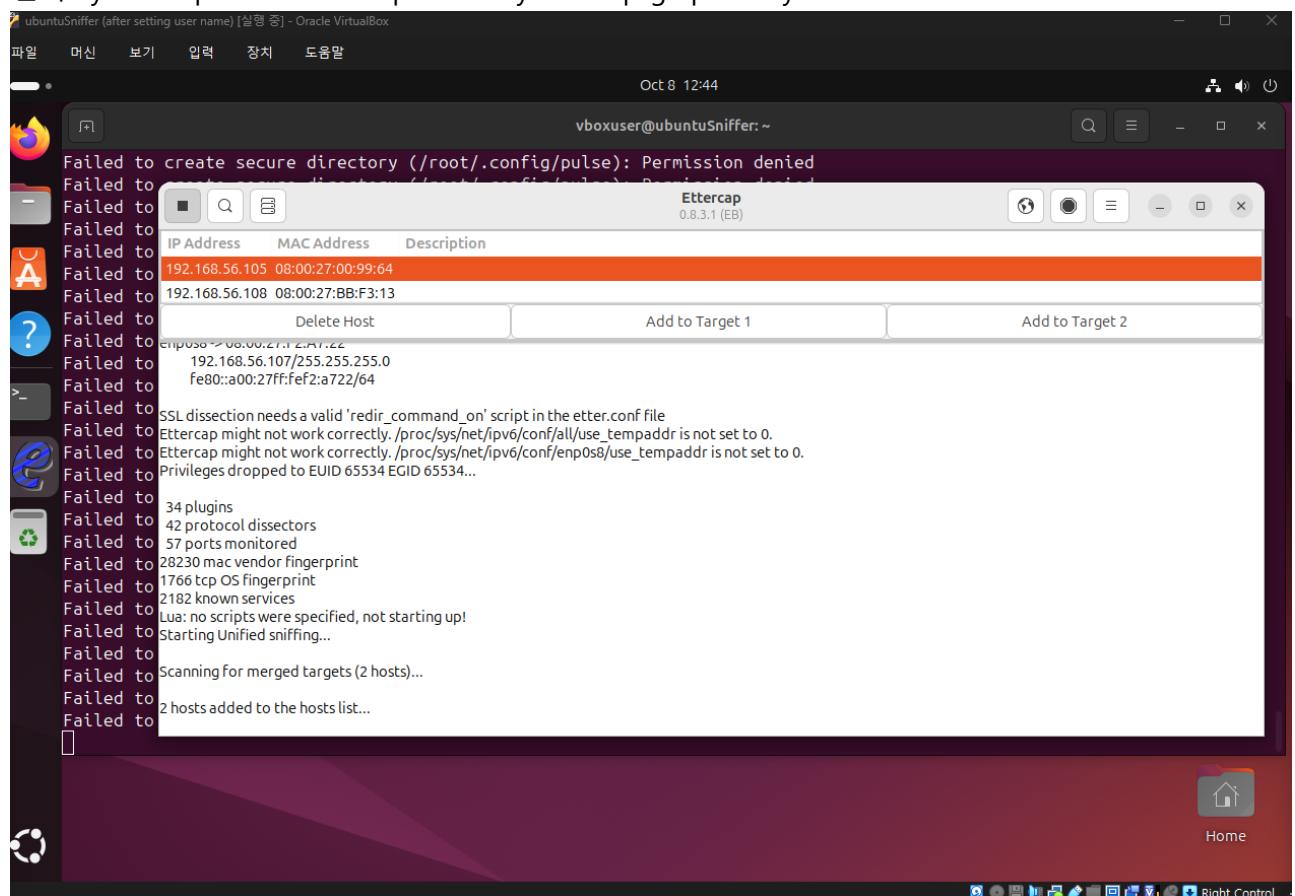
ettercat

- Ian에서 중간자공격을 위한 오픈소스 툴
- 이번 실습에서는 ARP poisoning 용도로 사용

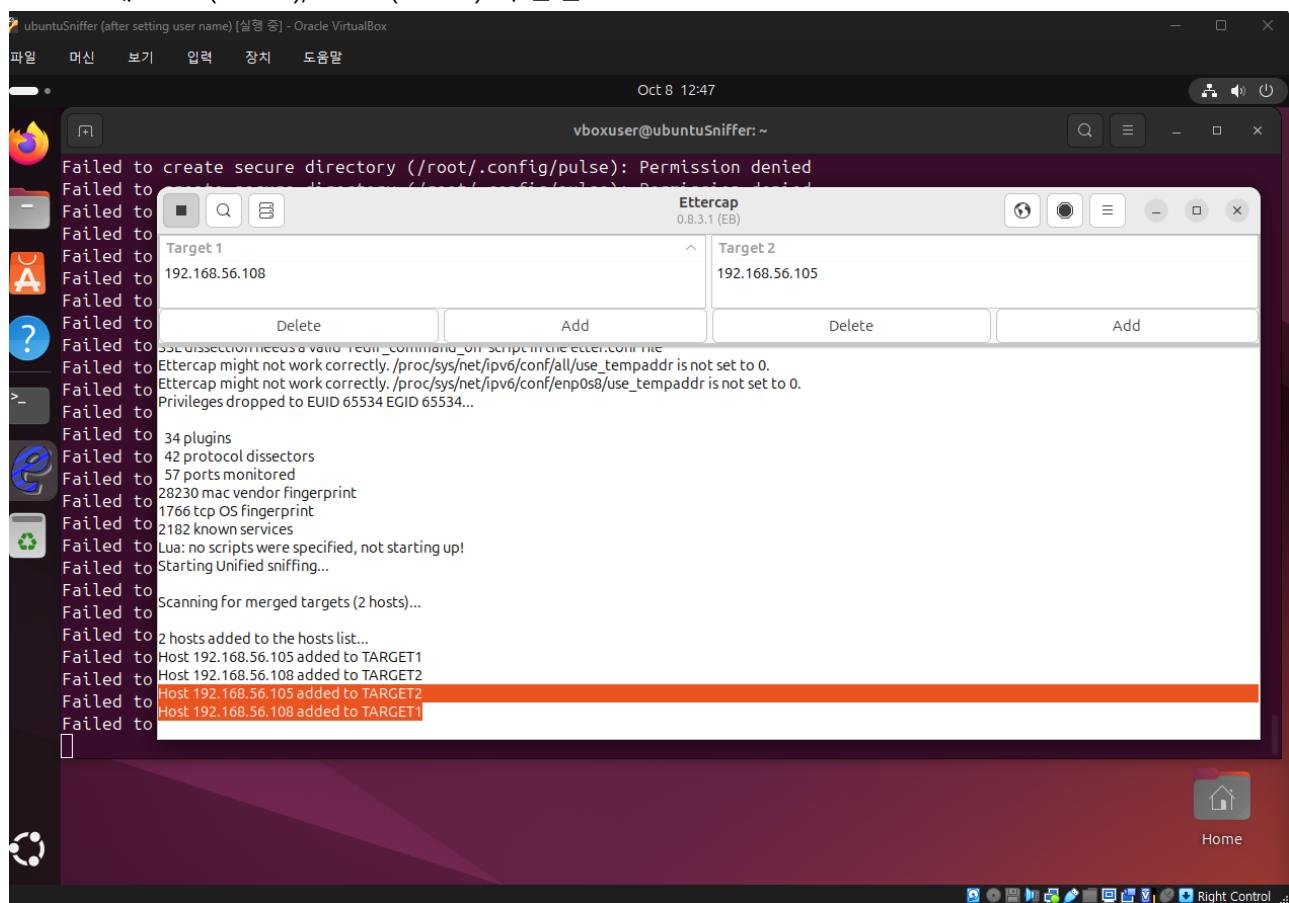
2번 실습과정

ettercat

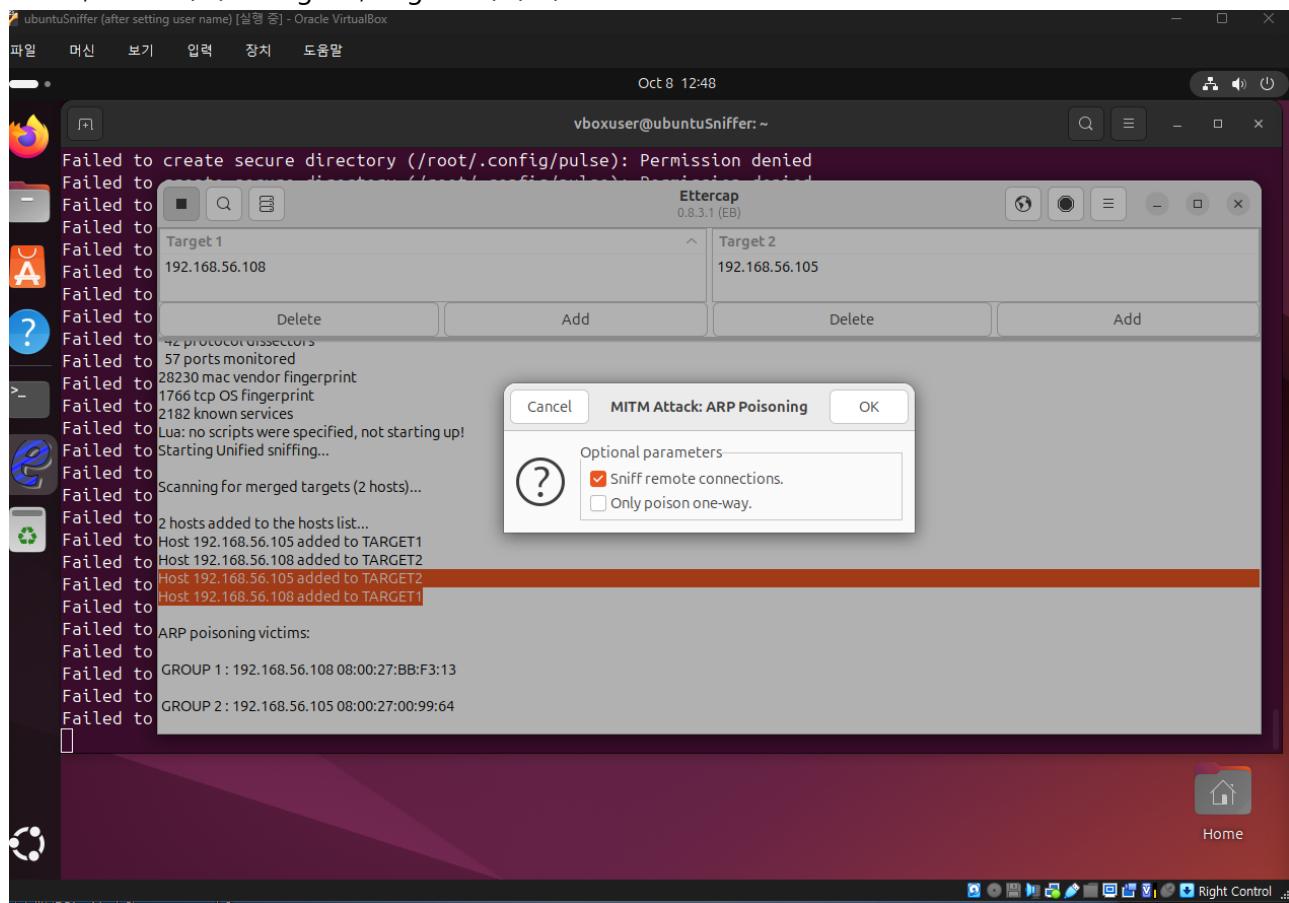
- 설치 by sudo apt install ettercap-text-only ettercap-graphical -y



- host list에 client(xx.105), server(xx.108) 이 잡힘



- server, client 각각을 target 1, target 2에 추가



- ARP 공격 수행

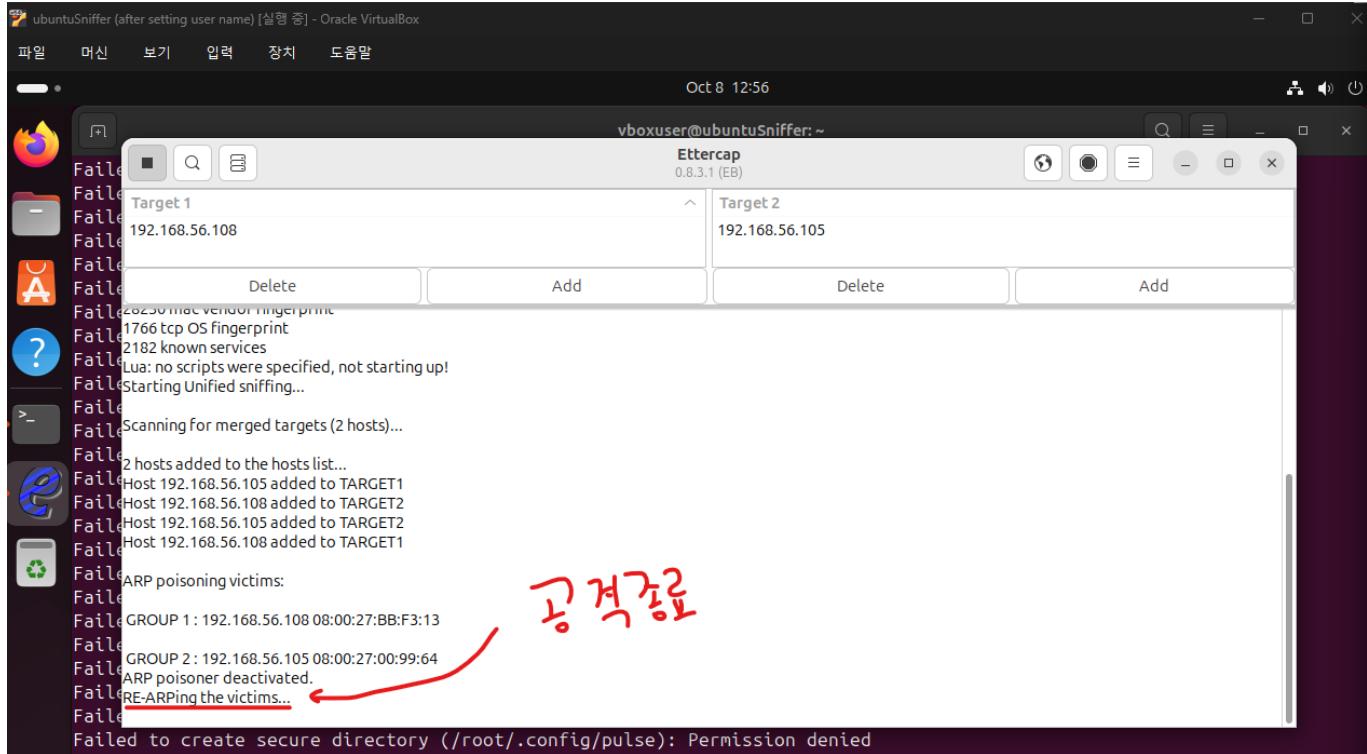
- server side에서 client의 mac 주소 변경 확인

실습3: ARP spoofing 방지기법을 적용한다

- 클라이언트 노드에서 네트워크 내의 Mac Address를 정적으로 설정하고, Mac Address Table을 관찰한다
 - 공격자 노드에서 ARP spoofing 공격을 수행 후, wireshark를 통해 클라이언트의 Mac Address를 확인한

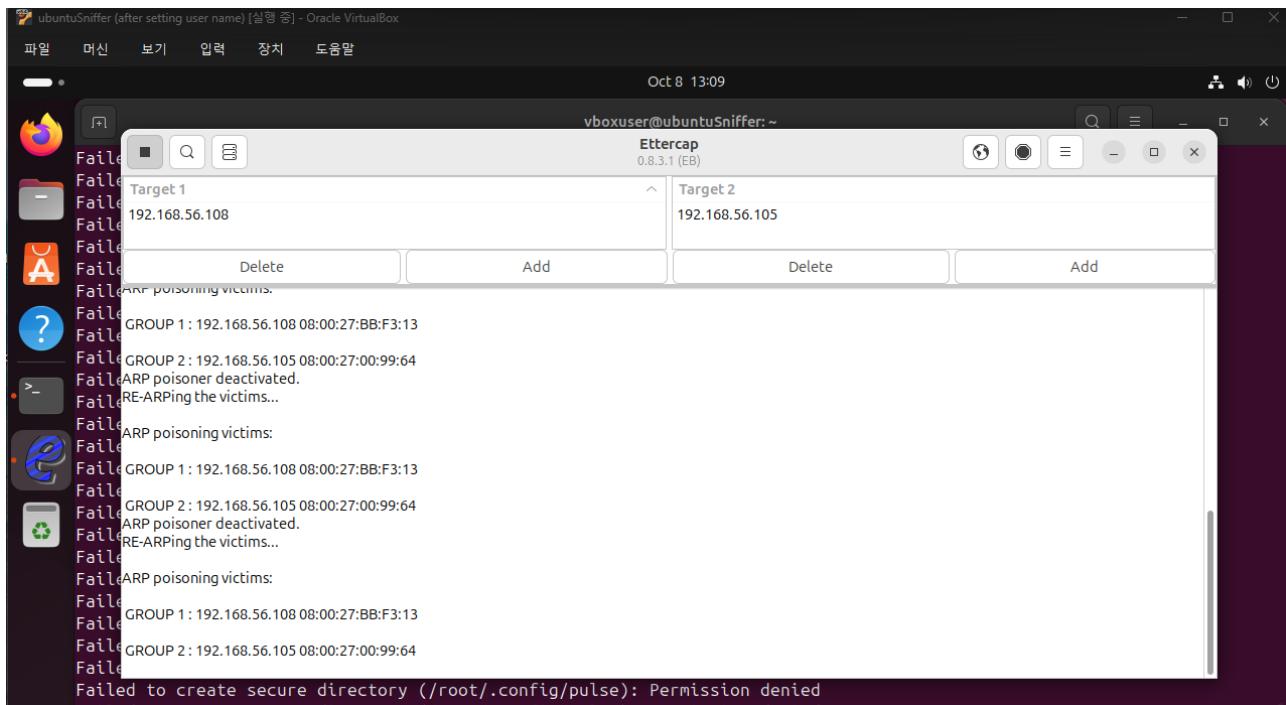
3번 실습 준비 2번과 동일

3번 실습 과정



- ARP 공격 종료

- server side에서 client의 mac 주소 원래대로 돌아옴 확인
 - server에서 sudo ip neigh replace 192.168.56.105 lladdr 08:00:27:00:99:64 dev enp0s8 nud permanent
 - client에서 sudo ip neigh replace 192.168.56.105 lladdr 08:00:27:bb:f3:13 dev enp0s8 nud permanent
 - 로 mac 주소 고정



- 다시 ARP 공격 수행

- 공격은 잘 수행되었지만, mac 주소는 바꿔지 않아 spoofing 공격으로부터 안전함 확인