Part 2: Python Network Scanner

Contents

Time required: 60 minutes

Create a Custom MAC Destination Address

The first step to manually building a network scanner is to create a custom Ethernet packet. **scapy** has a function called **Ether** that we will use to create a custom Ethernet packet with a destination broadcast MAC address.

We won't build the custom packet all at once, but one piece at a time. First, we will set the destination MAC broadcast address. ff:ff:ff:ff:ff:ff

Standard ARP Packet

1. Save **network_scanner_1.py** as **network_scanner_2.py**
2. Modify your code to include the following.

```
def scan(ip):
    # Create ARP request for targeted ip address
    # pdst is Target protocol address
    arp_request = scapy.ARP(pdst=ip)
    print('Standard ARP packet')
    arp_request.show()
```

3. Execute **network_scanner_2.py** at the command line. Your output should look like the example run below.

How it Works

1. The **arp_request** variable captures and stores the results of the ARP request.
2. **arp_request.show()** shows a standard ARP request packet. We are showing this to figure out what to add to our custom packet.

Example run:

```
D:\Temp>python network_scanner2.py
Standard ARP packet
WARNING: More than one possible route for Net("192.168.9.1/24")
###[ ARP ]###
  hwtype    = 0x1
  ptype     = IPv4
  hwlen     = None
  plen      = None
  op        = who-has
  hwsrc     = 2c:f0:5d:a2:ac:3e
  psrc      = 192.168.9.101
  hwdst     = 00:00:00:00:00:00
  pdst      = Net("192.168.9.1/24")
```

Custom Ethernet Packet

Add the following code to your program.

```
# scapy.Ether() creates a custom Ethernet packet
# Source MAC address is local computer
# dst sets destination MAC, in this case MAC broadcast address
broadcast = scapy.Ether(dst='ff:ff:ff:ff:ff:ff')
print('Custom Ethernet packet')
broadcast.show()
```

How it Works

1. **scapy.Ether** creates a **dst** (Destination) broadcast MAC address.
2. **src** (Source) MAC address is automatically included.

Example run:

```
Custom Ethernet packet
###[ Ethernet ]###
  dst       = ff:ff:ff:ff:ff:ff
  src       = 2c:f0:5d:a2:ac:3e
  type      = 0x9000
```

Combine Both Packets

Combining both packets will create an ARP packet with a broadcast address.

```
# Combining the first two packets together with scapy / operator
arp_request_broadcast = broadcast/arp_request
print('Custom Ethernet packet combined with ARP')
arp_request_broadcast.show()
```

Example run:

```
Custom Ethernet packet combined with ARP
WARNING: More than one possible route for Net("192.168.9.1/24")
###[ Ethernet ]###
  dst       = ff:ff:ff:ff:ff:ff
  src       = 2c:f0:5d:a2:ac:3e
  type      = ARP
###[ ARP ]###
     hwtype    = 0x1
     ptype     = IPv4
     hwlen     = None
     plen      = None
     op        = who-has
     hwsrc     = 2c:f0:5d:a2:ac:3e
     psrc      = 192.168.9.101
     hwdst     = 00:00:00:00:00:00
     pdst      = Net("192.168.9.1/24")
```

Success! We combined a MAC broadcast address with a standard ARP packet.

NOTE: Test your Python file on Windows and Kali Linux.

Assignment Submission

Attach all program files and screenshots of your results from both operating systems to the assignment in BlackBoard.