Part 2: Python Network Scanner

Contents

Time required: 60 minutes

Python Tabs and Spaces Issue

Visual Studio Code automatically changes a tab into four spaces. Other editors, like geany and nano in Linux, do not. You can end up with a combination of spaces and tabs. Python doesn't like a combination, it wants either one or the other. The preferred method is spaces.

Recommendation:

1. Create your Python files in Visual Studio Code in Windows.
2. Copy and paste the code into either nano or geany in Linux.

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.

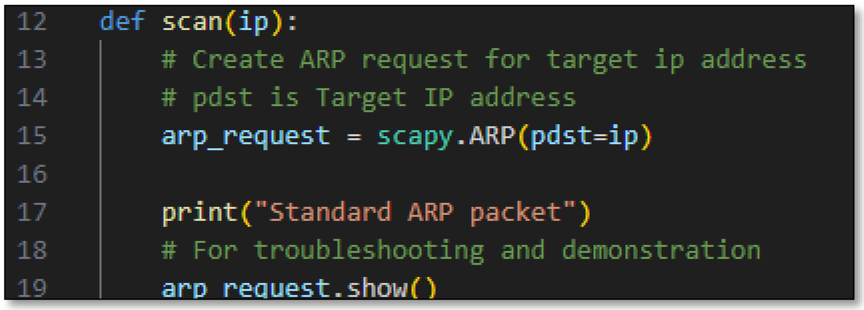
Instead of scanning the network, we are going to create an ARP packet that broadcasts an ARP request to the entire network for the MAC address of the default gateway (router).

This is how the arp command works. Instead of using a built-in arp command, we are going to build our own.

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

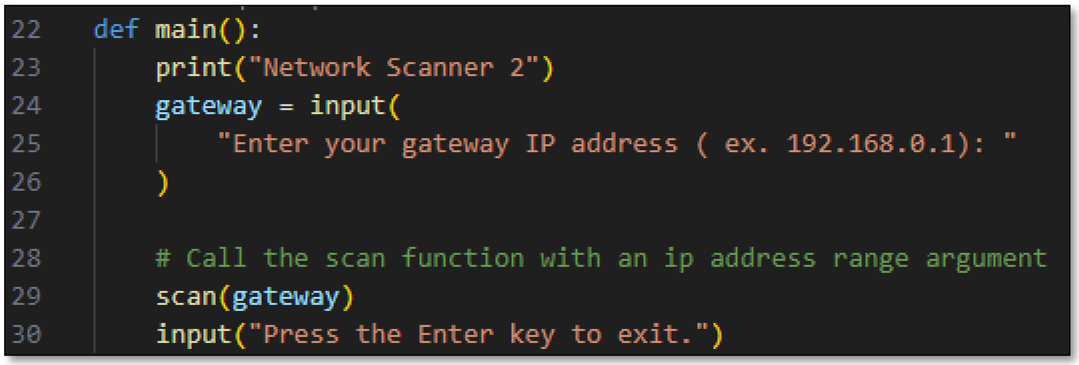
Standard ARP Packet

1. Save **network_scanner_1.py** as **network_scanner_2.py**
2. Modify your scan function to the following.



```python
def scan(ip):
    # Create ARP request for target ip address
    # pdst is Target IP address
    arp_request = scapy.ARP(pdst=ip)

    print("Standard ARP packet")
    # For troubleshooting and demonstration
    arp_request.show()
```

3. Modify main to the following.



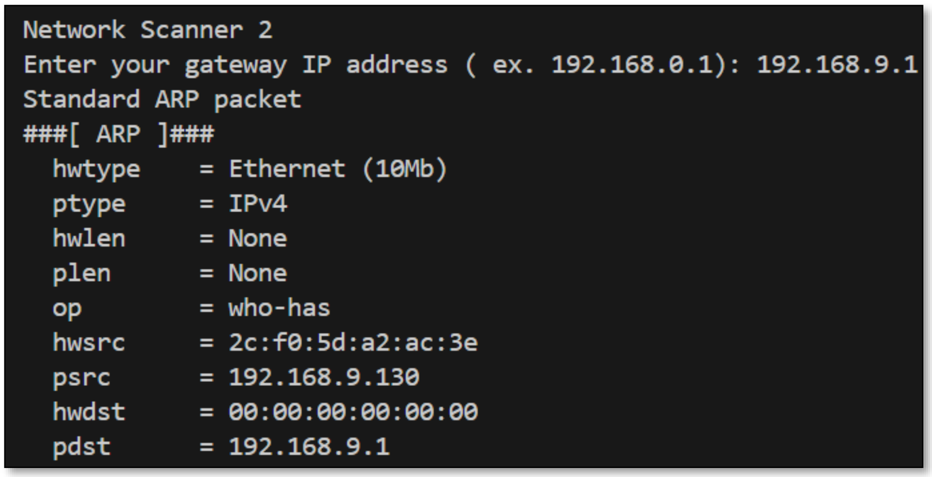
```python
def main():
    print("Network Scanner 2")
    gateway = input(
        "Enter your gateway IP address ( ex. 192.168.0.1): "
    )

    # Call the scan function with an ip address range argument
    scan(gateway)
    input("Press the Enter key to exit.")
```

4. Execute **network_scanner_2.py** at the command line. Use your default gateway as the destination.
5. 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:

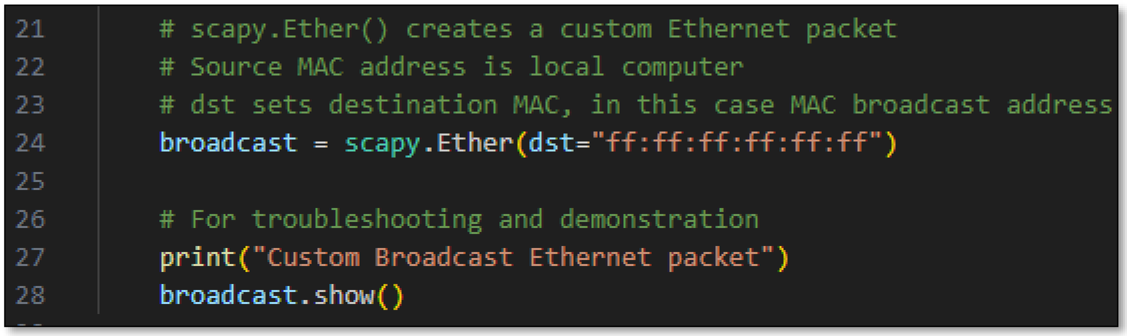


```
Network Scanner 2
Enter your gateway IP address ( ex. 192.168.0.1): 192.168.9.1
Standard ARP packet
###[ ARP ]###
  hwtype    = Ethernet (10Mb)
  ptype     = IPv4
  hwlen     = None
  plen      = None
  op        = who-has
  hwsrc     = 2c:f0:5d:a2:ac:3e
  psrc      = 192.168.9.130
  hwdst     = 00:00:00:00:00:00
  pdst      = 192.168.9.1
```

Custom Ethernet Broadcast Packet

We are going to create an ethernet broadcast packet. The MAC address ff:ff etc is a broadcast MAC address.

Add the following code at the beginning of the code in the scan() function.



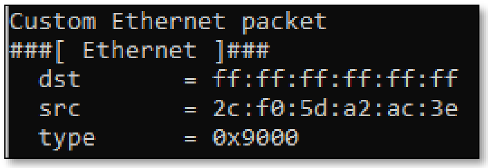
```python
    # scapy.Ether() creates a custom Ethernet packet
    # Source MAC address is local computer
    # dst sets destination MAC, in this case MAC broadcast address
    broadcast = scapy.Ether(dst="ff:ff:ff:ff:ff:ff")

    # For troubleshooting and demonstration
    print("Custom Broadcast 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:

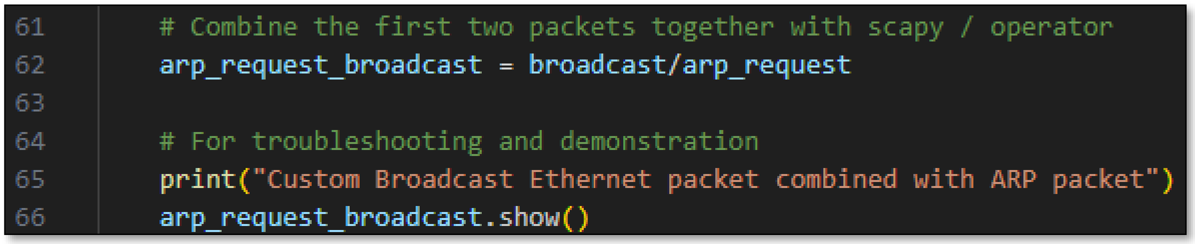
Execute **network_scanner_2.py** at the command line. Use your default gateway as the destination.


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

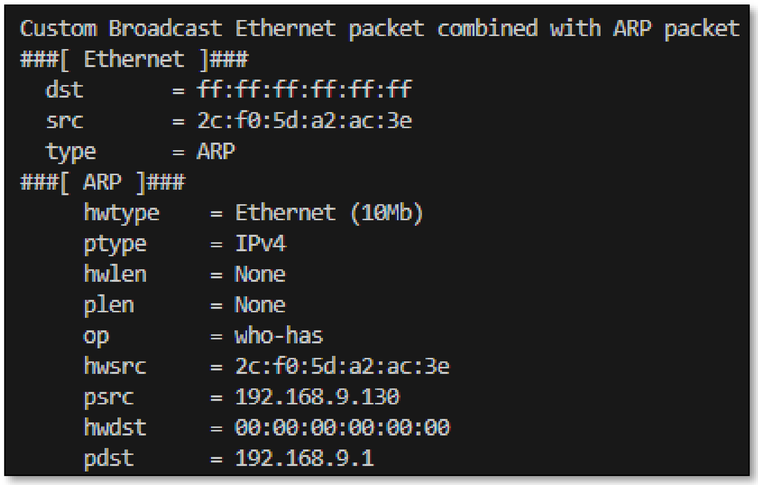
Combine Both Packets

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

Add this code at the end of the scan() function.


```python
    # Combine the first two packets together with scapy / operator
    arp_request_broadcast = broadcast/arp_request

    # For troubleshooting and demonstration
    print("Custom Broadcast Ethernet packet combined with ARP packet")
    arp_request_broadcast.show()
```

Example run:


```
Custom Broadcast Ethernet packet combined with ARP packet
###[ Ethernet ]###
  dst       = ff:ff:ff:ff:ff:ff
  src       = 2c:f0:5d:a2:ac:3e
  type      = ARP
###[ ARP ]###
     hwtype    = Ethernet (10Mb)
     ptype     = IPv4
     hwlen     = None
     plen      = None
     op        = who-has
     hwsrc     = 2c:f0:5d:a2:ac:3e
     psrc      = 192.168.9.130
     hwdst     = 00:00:00:00:00:00
     pdst      = 192.168.9.1
```

Success! We combined a MAC broadcast address with a standard ARP packet to create our own ARP command.

Execute **network_scanner_2.py** at the command line. Use your default gateway as the destination.

Test your Python file on Windows and Kali Linux. Make sure Kali Linux is on a bridged adapter.

Assignment Submission

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