Python Make a DHCP Listener using Scapy

Contents

Time required: 60 minutes

What is DHCP?

Dynamic Host Configuration Protocol (DHCP) is a network protocol that allows clients connected to a network to obtain TCP/IP configuration information (such as the private IP address) from a DHCP server.

A DHCP server (this can be an access point, router, or configured in a server) dynamically assigns an IP address and other configuration parameters to each device connected to the network.

The DHCP protocol uses User Datagram Protocol (UDP) to perform the communication between the server and clients. It is implemented with two port numbers:

- Server: UDP port number 67
- Client: UDP port number 68

Let’s make a simple Python DHCP listener using the **Scapy** library. We'll be able to listen for DHCP packets in the network and extract valuable information whenever a device connects to the network we're in.

Setup scapy in Windows

Windows needs three parts to use scapy.

1. Python 3 www.python.org
2. Npcap (For capturing and sending packets.)
3. scapy (Python library for manipulating packets.)

Install Npcap

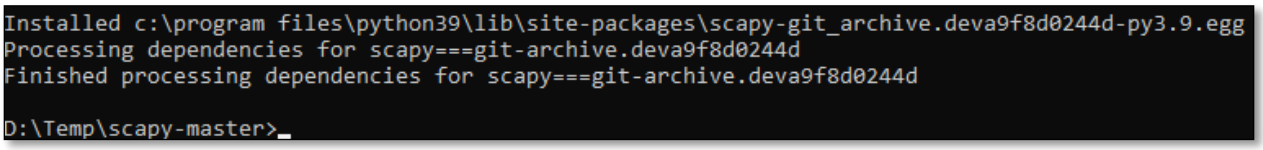
Npcap is the Nmap Project's packet sniffing (and sending) library for Windows. This is needed for **scapy** to communicate with the network.

1. Go to https://npcap.com/#download
2. Download and install the latest **Npcap** for Windows.

Install scapy in Windows

In Windows, **scapy** can't be installed using **pip**. We must download and install **scapy** from **github**.

1. Click https://github.com/secdev/scapy/archive/master.zip to download **master.zip**.
2. Unzip **master.zip** into a folder you can access using the command prompt.
3. Using the command prompt → Navigate to the **scapy-master** folder with all the files in it.
4. Run the command → **python setup.py install**
5. Your command should be successful and show something like this at the end.



Setup scapy in Linux

Kali Linux already has Python 3 and libcap for capturing and sending packets. We need to install **scapy**.

1. Open a terminal

2. Run the command → **pip3 install scapy**

Edit Python in Linux with geany (Optional)

Geany is a lightweight code editor that works very well in Linux. This following will install geany on Linux.

1. **sudo apt update**
2. **sudo apt upgrade**
3. **sudo apt install geany**

To use geany from the terminal:

1. Create and edit a Python file: **geany dhcp_listener.py**

Scapy sniff()

The **sniff()** function in Scapy is responsible for sniffing any type of packet that can be monitored. Sniff listens to or monitors the network interface on your computer. It is continuously listening.

To remove other packets that we're not interested in, we use the filter parameter in the **sniff()** function:

```
1 from scapy.all import *
2 import time
3
4
5 def listen_dhcp():
6     # Set the filter options for DHCP
7     sniff(prn=print_packet, filter="udp and (port 67 or port 68)")
8
```

In the sniff function, the first argument is **prn=print_packet**. **prn** passes the packet that has been captured to the **print_packet()** function. Each packet that is captured by sniff is passed to the **print_packet()** function.

The second argument is a **filter** for only DHCP packets. We only capture UDP packets with port 67 or 68 in their attributes.

Let's define the **print_packet()** function:

```python
def print_packet(packet):
    # initialize these variables to None at first
    target_mac, requested_ip, hostname, vendor_id = [None] * 4
    # get the MAC address of the requester
    if packet.haslayer(Ether):
        target_mac = packet.getlayer(Ether).src
    # get the DHCP options
    dhcp_options = packet[DHCP].options
    for item in dhcp_options:
        try:
            label, value = item
        except ValueError:
            continue
        if label == 'requested_addr':
            # get the requested IP
            requested_ip = value
        elif label == 'hostname':
            # get the hostname of the device
            hostname = value.decode()
        elif label == 'vendor_class_id':
            # get the vendor ID
            vendor_id = value.decode()
    if target_mac and vendor_id and hostname and requested_ip:
        # if all variables are not None, print the device details
        time_now = time.strftime("[%m-%d-%Y - %H:%M:%S]")
        print(f"{time_now} : {target_mac} - {hostname}")
        print(f"{vendor_id} requested {requested_ip}")
```

First, we extract the MAC address from the **src** attribute of the **Ether** packet layer.

If there are DHCP options included in the packet, we iterate over them and extract the **requested_addr** (which is the requested IP address), **hostname** (the hostname of the requester), and the **vendor_class_id** (DHCP vendor client ID). After that, we get the current time and print the details.

This line starts sniffing/listening:

```python
if __name__ == "__main__":
    listen_dhcp()
```

Before running the script, make sure you're connected to your own network for testing purposes.

Run the script. Connect with another device to the network and see the output. When a host starts up and connects to the network, it checks in with a dhcp server.

Example run:

```
[05-19-2022 - 17:01:55] : 82:ca:13:1a:ed:58 - Galaxy-Note9
android-dhcp-10 requested 192.168.9.111
[05-19-2022 - 17:02:55] : dc:41:a9:e4:9d:eb - Yoga940
MSFT 5.0 requested 192.168.9.110
```

Conclusion

Awesome! Now you have a quick DHCP listener in Python that you can extend, I suggest you print the **dhcp_options** variable in the **print_packet()** function to see what that object looks like.

Assignment Submission

1. Attach the pseudocode.
2. Attach the program files.
3. Attach screenshots showing the successful operation of the program.
4. Submit in Blackboard.