Part 1: Python Network Scanner

Contents

Time required: 60 minutes

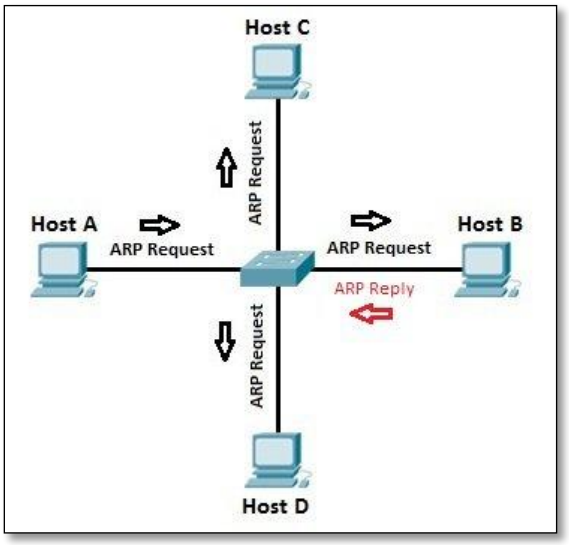
Python Tabs and Spaces Issue

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 prefered 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.

What is ARP?



Most computer programs/applications use **logical addresses (IP address)** to send/receive messages. The actual communication happens over the **physical address (MAC address)** i.e from layer 2 of OSI model. **Address Resolution Protocol (ARP)** translates **Internet Protocol (IP)** addresses to **Media Access Control (MAC)** addresses.

1. Host A wants to communicate with Host B.
2. Host A sends out a broadcast ARP request to all hosts on the network.
3. Host B replies with its IP address and MAC address.
4. Host A and Host B can communicate using MAC addresses.

What is scapy?

scapy is a powerful interactive packet manipulation program. It can forge or decode packets of a wide number of protocols, send them on the wire, capture them, match requests and replies, and much more. It can easily handle most classic tasks like scanning, tracerouting, probing, unit tests, attacks, or network discovery. It runs on Linux, Windows, and OSX. The code base runs on Python 2 and Python 3. We will use Python 3.

Setup scapy in Windows

Windows needs three parts to use scapy.

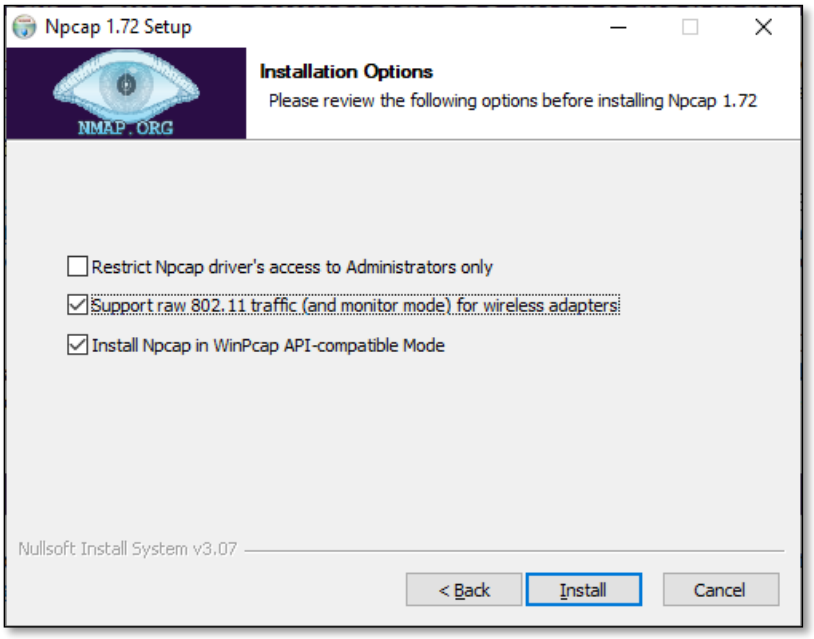
1. Python 3: www.python.org
2. Npcap: (For capturing and sending packets.)
3. scapy: (Python library for manipulating packets.)

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.

NOTE: If you have nmap or Wireshark installed, you already have npcap on your system.

1. Go to https://nmap.org/npcap
2. Download the latest **Npcap** for Windows.
3. When installing: add the following options.



Install scapy in Windows

In Windows, **scapy** is installed from an administrator command prompt.

1. Start an **administrative command prompt.**

```
# Install wheel setup package
pip install wheel
# Install scapy library
pip install scapy
```

2. Scapy should install successfully.

Setup scapy in Linux

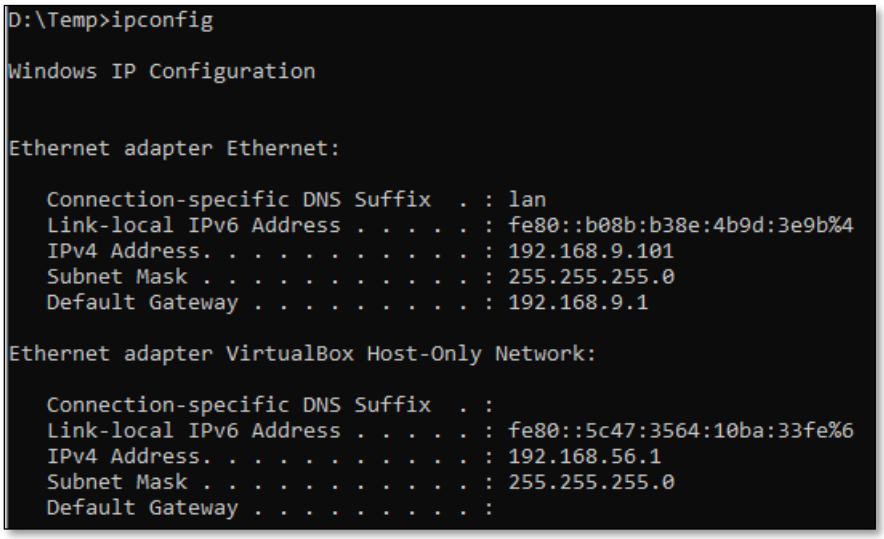
Kali Linux already has Python 3, libcap and scapy for capturing and sending packets.

Determine Your Network IP Range

The first step is to determine the address range of your local network.

1. Connect your Linux or any other VM being used in this assignment to the VirtualBox **Bridged Adapter**.

2. In Windows: At a command prompt → type **ipconfig**
   In Linux: **ifconfig** or **ip -a**



You will have one network adapter with a Default Gateway. That will help you identify your network.

Subnet Mask: This network has a class C subnet: **255.255.255.0** which can be designated as **/24**. If your subnet mask is different, Google the / address of your subnet mask.

Network Address: The first host address is **192.168.9.1**.

This network can be scanned as **192.168.9.1/24**

Network Scanner with scapy

We are going to manually build a network scanner in Python using the ARP protocol. We will build a custom ARP packet and display the results.

Before we start manually building a network scanner, we are going to use a built-in **scapy.arping() method** to show what our result will look like.

1. Create a Python program using a main function called **network_scanner1.py**

2. The first step is to import the **scapy** module.

```
#!/usr/bin/env python3
"""
    Name: network_scanner1.py
    Author:
    Created:
    Purpose: Send an ARP packet to all hosts on a network
"""

# Import the scapy module
import scapy.all as scapy
```

3. Define a **scan** function that accepts an **IP** address or **IP** address range as an argument. **scapy** uses the **arping** method to send an **arp** packet to all IP addresses in the range.

```
def scan(ip):
    """
        Send ARP packet to all hosts
        Receive and display ARP information
    """
    # Call the scapy.arping() method using an
    # IP address or IP range argument
    scapy.arping(ip)
```

4. Call the **scan** function from main(). Replace the IP range with your own.

```
def main():
    # IP range to scan
    # Substitute your network IP range
    scan("192.168.9.1/24")
    input("Press the Enter key to exit.")


# Call main function or use as module
if __name__ == "__main__":
    main()
```

5. The program will not work properly from an IDE. Start a **command prompt**.
6. Navigate to the folder that **network_scanner1.py** is located.
7. Windows: **python network_scanner1.py** at the command prompt as shown below.
8. Linux: **sudo python3 network_scanner1.py** at a terminal prompt as shown below

Example run Windows:

```
Z:\_WNCC\Ethical Hacking\Assignments\Network Scanner>python network_scanner1.py
Begin emission:
Finished sending 256 packets.
**********
Received 10 packets, got 10 answers, remaining 246 packets
  70:4f:57:33:05:b8 Tp-LinkT 192.168.9.1
  6c:0b:84:09:b4:a6 Universa 192.168.9.10
  2c:f0:5d:a2:ac:3e Micro-St 192.168.9.101
  0c:8b:7d:6c:3c:f5 Vizio    192.168.9.100
  4c:1b:86:9a:2b:3c Arcadyan 192.168.9.124
  82:ca:13:1a:ed:58 unknown  192.168.9.150
  88:c2:55:20:58:b4 TexasIns 192.168.9.110
  58:ef:68:ea:92:a1 BelkinIn 192.168.9.114
  5c:cf:7f:2c:31:9c Espressi 192.168.9.142
  10:2c:6b:be:c6:76 AMPAKTec 192.168.9.136
Press the Enter key to exit.
```

Example run Linux:

```
┌──(user㉿kali)-[~]
└─$ sudo python3 network_scanner1.py
Begin emission:
Finished sending 256 packets.
**********.*
Received 12 packets, got 11 answers, remaining 245 packets
  70:4f:57:33:05:b8 Tp-LinkT 192.168.9.1
  6c:0b:84:09:b4:a6 Universa 192.168.9.10
  2c:f0:5d:a2:ac:3e Micro-St 192.168.9.101
  0c:8b:7d:6c:3c:f5 Vizio    192.168.9.100
  4c:1b:86:9a:2b:3c Arcadyan 192.168.9.124
  82:ca:13:1a:ed:58 unknown  192.168.9.150
  88:c2:55:20:58:b4 TexasIns 192.168.9.110
  58:ef:68:ea:92:a1 BelkinIn 192.168.9.114
  48:a2:e6:1f:3d:0d Resideo  192.168.9.119
  10:2c:6b:be:c6:76 AMPAKTec 192.168.9.136
  a4:cf:12:b4:f4:79 Espressi 192.168.9.147
Press the Enter key to exit.
```

Test the program on Windows and Linux.

That's it, we are done. We can use this simple hand-built network scanner on any network that you have permission to scan.

Assignment Submission

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