Network Scanning using scapy module – Python

Difficulty Level : Medium • Last Updated : 01 Mar, 2020

Scapy is a library supported by both Python2 and Python3. It is used for interacting with the packets on the network. It has several functionalities through which we can easily forge and manipulate the packet. Through scapy module we can create different network tools like ARP Spoofer, Network Scanner, packet dumpers etc. This module can be used to create more advanced tools related network security and ethical hacking.

Installation of scapy module:
As scapy module is not included in Python3 library by default, we have to add it into our Python library using pip. Execute this command in your Linux terminal to get the scapy module for Python3.

```
pip3 install scapy-python3
```

What is network scanning ?
Network scanning refers to scanning of whole network to which we are connected and try to find out what are all the clients connected to our network. We can identify each and every client using their IP and MAC address. We can use ARP ping to find out the alive systems in our network.

Some important functions for creating Network

scanner –

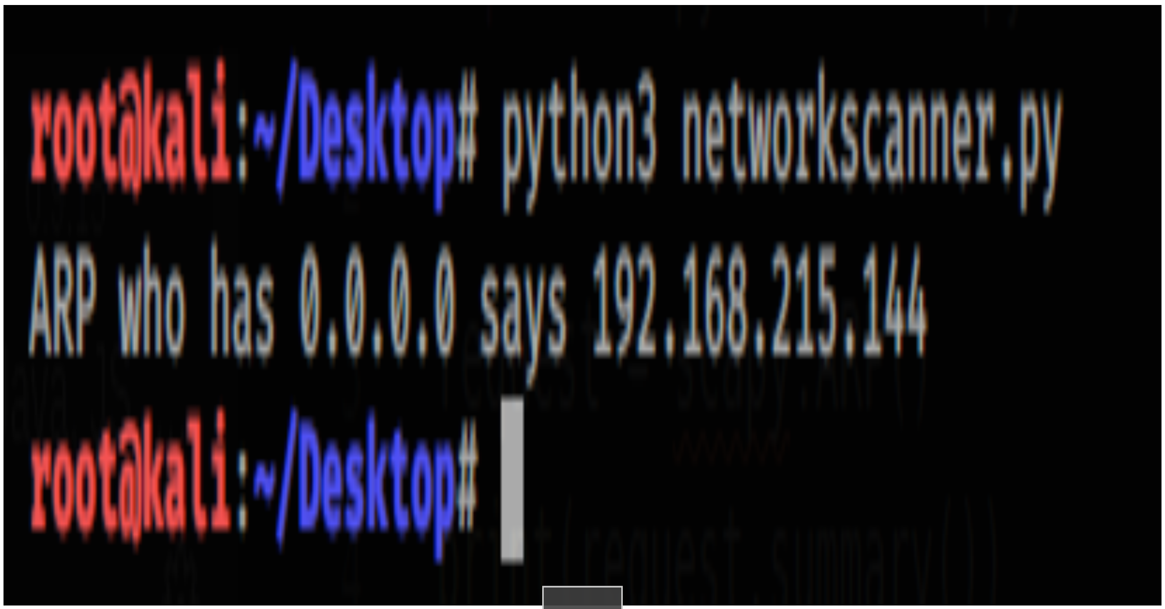
ARP(): This function defined in scapy module which allows us to create ARP packets (request or response). By default, if we are calling it, it will create an ARP request packet for us.

```
import scapy.all as scapy

request = scapy.ARP()
```

summary(): This method provide us the status of the packet that we have created. It does not provide the detailed information about the packet, it just gives us the basic idea like what is the type of packet, what is the destination of the packet etc.
For example if we want to create an ARP packet using `ARP()` method which is present in the scapy module and want to see the summary of the packet then we can do this by creating the object of ARP class.

```
import scapy.all as scapy

request = scapy.ARP()
print(request.summary())
```

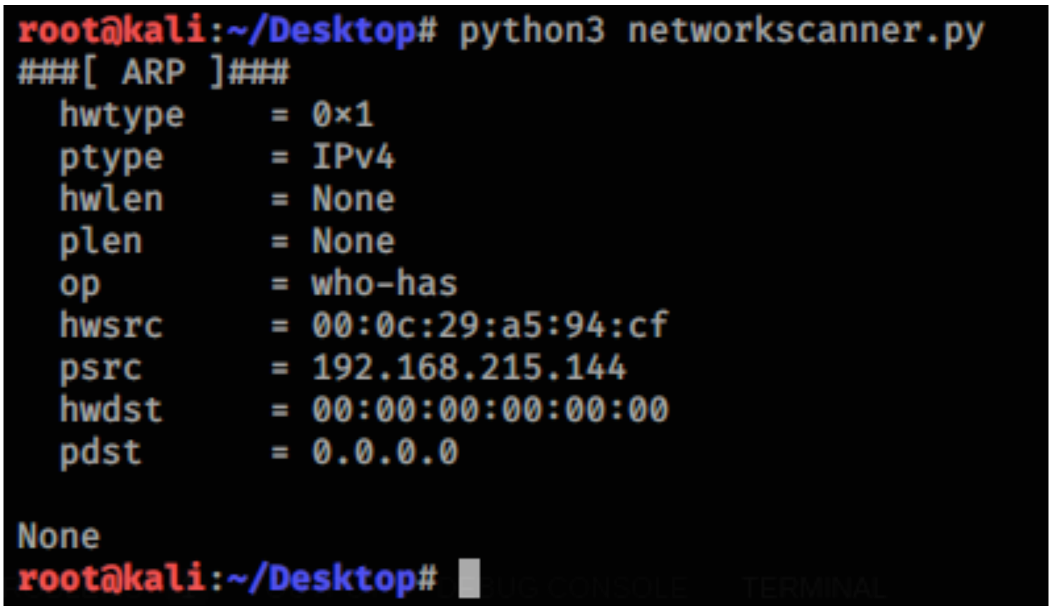
Now we have created a request packet of ARP. Here the output of the program will be like this –



```
root@kali:~/Desktop# python3 networkscanner.py
ARP who has 0.0.0.0 says 192.168.215.144
root@kali:~/Desktop# 
```

show() method: This method is very similar to `summary()` method. It gives more detailed information about the packet. The usage of this function is also much similar to as `summary()` method.

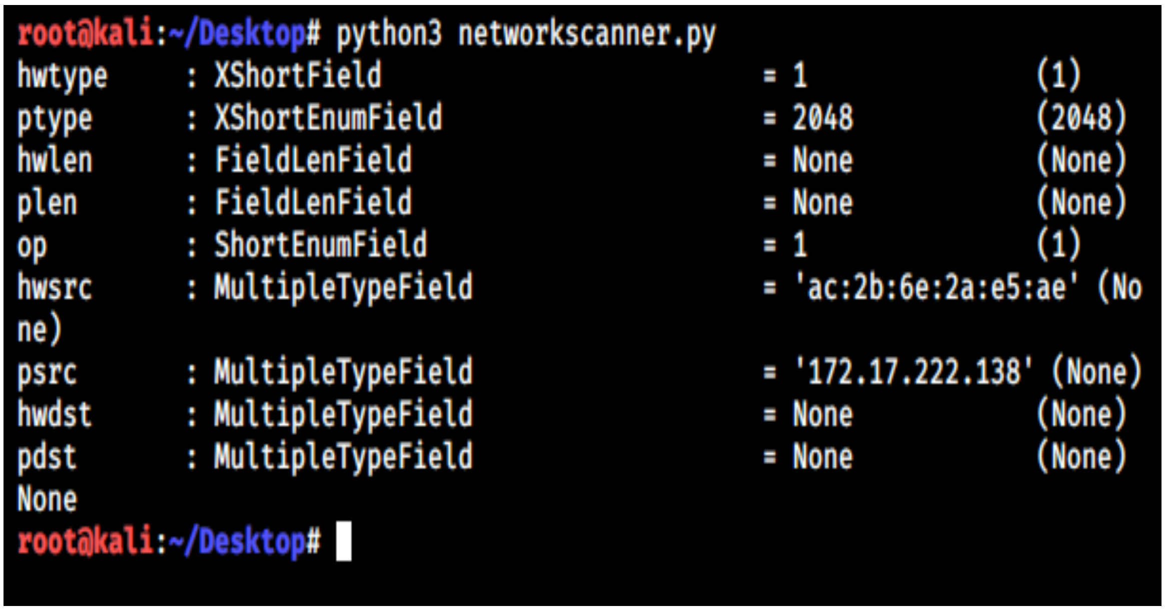
```
import scapy.all as scapy

request = scapy.ARP()
print(request.show())
```



```
root@kali:~/Desktop# python3 networkscanner.py
###[ ARP ]###
  hwtype    = 0x1
  ptype     = IPv4
  hwlen     = None
  plen      = None
  op        = who-has
  hwsrc     = 00:0c:29:a5:94:cf
  psrc      = 192.168.215.144
  hwdst     = 00:00:00:00:00:00
  pdst      = 0.0.0.0

None
root@kali:~/Desktop# 
```

ls() function: This method is present in the scapy class. By using this method, we can see what are the fields that we can set for a specific packet.
In our example we will create an ARP packet and the with the help of ls() function, we will see what are the available fields for this packet.

```
import scapy.all as scapy

request = scapy.ARP()
print(scapy.ls(scapy.ARP()))
```



```
root@kali:~/Desktop# python3 networkscanner.py
hwtype     : XShortField                          = 1                (1)
ptype      : XShortEnumField                      = 2048             (2048)
hwlen      : FieldLenField                        = None             (None)
plen       : FieldLenField                        = None             (None)
op         : ShortEnumField                       = 1                (1)
hwsrc      : MultipleTypeField                    = 'ac:2b:6e:2a:e5:ae' (No
ne)
psrc       : MultipleTypeField                    = '172.17.222.138' (None)
hwdst      : MultipleTypeField                    = None             (None)
pdst       : MultipleTypeField                    = None             (None)
None
root@kali:~/Desktop#
```

Steps for creating Network Scanner –

1. Create an ARP packet using ARP() method.

2. Set the network range using variable.

3. Create an Ethernet packet using Ether() method.

4. Set the destination to broadcast using variable hwdst.

5. Combine ARP request packet and Ethernet frame using '/'.

6. Send this to your network and capture the response from different devices.

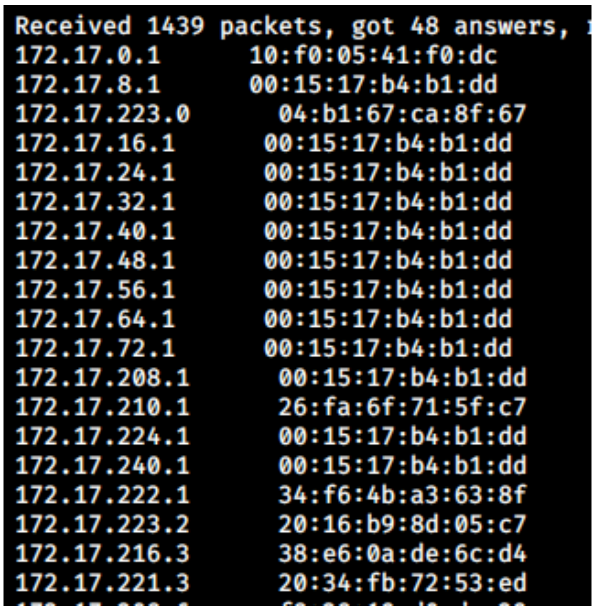
7. Print the IP and MAC address from the response packets.

Below is the Python implementation –

```
import scapy.all as scapy

request = scapy.ARP()

request.pdst = 'x'
broadcast = scapy.Ether()

broadcast.dst = 'ff:ff:ff:ff:ff:ff'

request_broadcast = broadcast / request
clients = scapy.srp(request_broadcast, timeout = 1)[0]
for element in clients:
    print(element[1].psrc + "       " + element[1].hwsrc)
```

Here x = Network range. For example x = 192.168.1.1/24, 172.16.5.1/16 etc

Output:



```
Received 1439 packets, got 48 answers,
172.17.0.1      10:f0:05:41:f0:dc
172.17.8.1      00:15:17:b4:b1:dd
172.17.223.0      04:b1:67:ca:8f:67
172.17.16.1      00:15:17:b4:b1:dd
172.17.24.1      00:15:17:b4:b1:dd
172.17.32.1      00:15:17:b4:b1:dd
172.17.40.1      00:15:17:b4:b1:dd
172.17.48.1      00:15:17:b4:b1:dd
172.17.56.1      00:15:17:b4:b1:dd
172.17.64.1      00:15:17:b4:b1:dd
172.17.72.1      00:15:17:b4:b1:dd
172.17.208.1      00:15:17:b4:b1:dd
172.17.210.1      26:fa:6f:71:5f:c7
172.17.224.1      00:15:17:b4:b1:dd
172.17.240.1      00:15:17:b4:b1:dd
172.17.222.1      34:f6:4b:a3:63:8f
172.17.223.2      20:16:b9:8d:05:c7
172.17.216.3      38:e6:0a:de:6c:d4
172.17.221.3      20:34:fb:72:53:ed
```

Like 0

Previous
Python - How to create an ARP Spoofer using Scapy?

Next
Port Scanner using Python

▲