LABWORK
COURSE: DISTRIBUTED SYSTEMS
CHAPTER 5: NAMING

1. ARP Spoofing: Man in the midle

1.1. Content

The Address Resolution Protocol (ARP) is a communication protocol used for discovering the MAC address, associated with a given IP address, typically an IPv4 address. After the theoretical lesson, we know that ARP based on a traditional naming mechanism *broadcasting* to locate a host with its IP address. The ARP protocol works efficiently in LAN network but it consists some drawbacks including ARP spoofing-based attacks like Man in the middle (MITM). In this labwork, you will use a tool, called MITMf, to realize a MITM attack.

1.2. Requirements

1.2.1. Theory

- ARP protocol
- Naming System in Distributed Systems

1.2.2. Hardwares

- 2 Laptop/PC: 1 on Windows (victim) and 1 on Kali Linux (attacker)

1.2.3. Softwares

- Ubuntu
- Kali OS
- MITMf

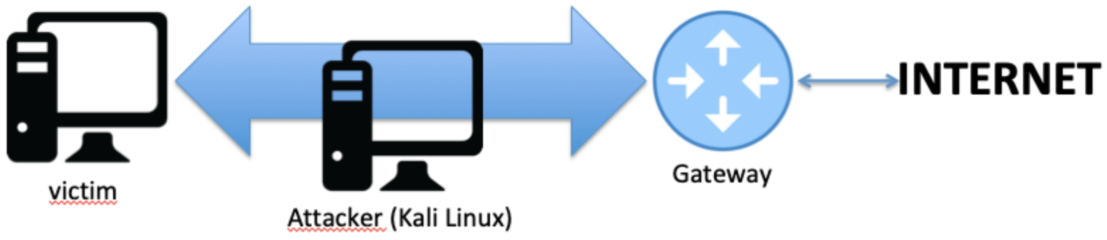
1.3. PRACTICAL STEPS

For installation steps, you have two options to choose:
1. You download the *iso* file to install a new virtual machine KALI Linux OS in VirtualBox:
https://www.kali.org/downloads/

2. You can also download available VirtualBox image for Kali Linux (and open it directly with VirutalBox without installation procedure), use this link: https://www.offensive-security.com/kali-linux-vm-vmware-virtualbox-image-download/

Choose the tab Kali Linux VirutalBox Images, then download the appropriate file for your PC. Remember this login information:
Username: *root*
Pass: *toor*

The testbed of this first part of the labwork is described as follows:



In this testbed, we will use a Kali Linux machine to play the role of an attacker. The latter will conduct an ARP spoofing MITM to eavesdrop on all communications between the victim and the gateway. Hence, the attacker will catch all the information the victim sends out to the Internet or it receives from the Internet.
Use *ifconfig* (on Linux) and *ipconfig* (on Windows) to determine the IP addresses of the attacker and the victim. Make sure that both of them are in the same network.
Now, you have to determine the IP of the Gateway:

On Windows:
Type this command:
```
>ipconfig | findstr /i "Gateway"
```

You should see something like this:
```
Default Gateway . . . . . . . . : 192.168.1.1
```

That means your Gateway’s IP is 192.168.1.1.

Or if you want to do it on Linux:
Type this command:
```
$ip route | grep default
```

The output will be like this:
```
default via 192.168.1.1 dev eth0 proto static
```

That means your Gateway’s IP is 192.168.1.1.

Or the command for Mac OS:

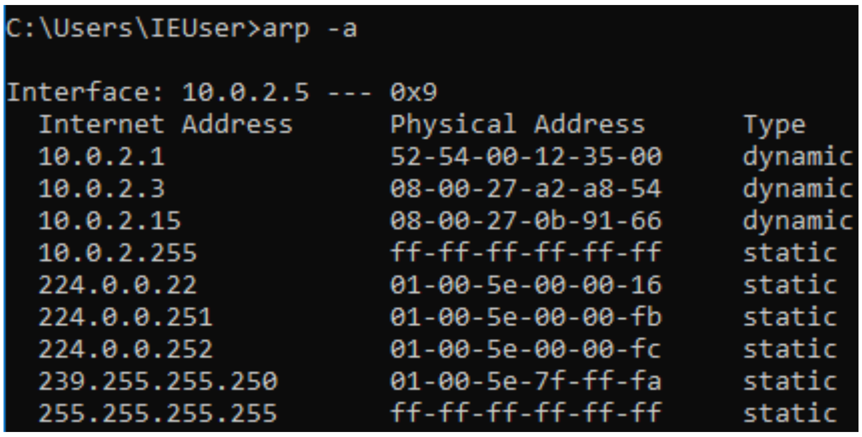
```
$netstat -nr | grep default
```

Question 1: What are the IP addresses of the Victim's PC, Attacker's PC, and the Gateway?

Go to the Windows PC (Victim), use the following command to see the ARP table of the Victim's machine:

```
>arp -a
```

You will get something like that:



```
C:\Users\IEUser>arp -a

Interface: 10.0.2.5 --- 0x9
  Internet Address      Physical Address      Type
  10.0.2.1              52-54-00-12-35-00     dynamic
  10.0.2.3              08-00-27-a2-a8-54     dynamic
  10.0.2.15             08-00-27-0b-91-66     dynamic
  10.0.2.255            ff-ff-ff-ff-ff-ff     static
  224.0.0.22            01-00-5e-00-00-16     static
  224.0.0.251           01-00-5e-00-00-fb     static
  224.0.0.252           01-00-5e-00-00-fc     static
  239.255.255.250       01-00-5e-7f-ff-fa     static
  255.255.255.255       ff-ff-ff-ff-ff-ff     static
```

So, now you can determine the MAC address of these 3 components.

Question 2: What are the MAC addresses of the Victim's PC, Attacker's PC, and the Gateway?

On the Attacker machine, use this command to conduct the MITM attack:

```
$mitmf --arp --spoof --gateway G-IP --target V-IP -i eth0
```

Where: **G-IP** is the Gateway's IP and **V-IP** is the Victim's IP. **eth0** is the network interface of the attacker. Replace the right values to the command above.

If everything is OK, on the Victim's machine you use the command `arp -a` to re-check the ARP table.

Question 3: What changes do you see in term of MAC addresses?

Now go to an HTTP website to see how the Attacker captures a username and password. On the Victim's machine, go to the website *Hack.me*; then, go to the login page to log into an account (make sure that the tool *MITMf* on the

Attacker's machine is still running). Enter your Email Address and your Password.

Question 4: Take a look at the console of the tool MITMf in the Attacker's machine. What do you see? What information of the above login do you get?

2. Set up your own DNS server

2.1. Contents

The process of DNS resolution involves converting a hostname (such as www.example.com) into a computer-friendly IP address (such as 192.168.1.1). In this labwork, you will set up your own DNS for your private network. It is a great way to improve the management of your servers. Concretely, you will go over how to set up an internal DNS server, using the BIND name server software (BIND9) on Ubuntu 18.04 that can be used by your servers to resolve private hostnames and private IP addresses. This provides a central way to manage your internal hostnames and private IP addresses, which is indispensable when your environment expands to more than a few hosts.

2.2. Requirements

2.2.1. Theory

- DNS
- Naming System in Distributed Systems

2.2.2. Hardwares

- Laptop/PC on Ubuntu 18.04

2.2.3. Softwares

- Ubuntu OS 18.04
- bind9

2.3. PRACTICAL STEPS

In this labwork, we'll work on 4 host on Ubuntu 18.04 described as below (you can use virtual machines if you have not enough of physical PC).

Host	Role	Private FQDN	Private IP Address
ns1	Primary DNS Server	ns1.ds.soict.hust.com	192.168.1.20
ns2	Secondary DNS Server	ns2.ds.soict.hust.com	192.168.1.21
host1	Generic Host 1	host1.ds.soict.hust.com	192.168.1.100
host2	Generic Host 2	host2.ds.soict.hust.com	192.168.1.101

In this labwork, *ns1* and *ns2* are two DNS servers. You have also two additional clients, *host1* and *host2*, that will be using the DNS infrastructure you create. You can add as many as you'd like for your infrastructure, but in this labwork you just use 2.
You assume that all of these servers exist in the same datacenter *ds*. You will use a naming scheme that uses "*ds.soict.hust.com*" to refer to your private subnet or zone.
The IP addresses of 4 hosts can be changed as you like, but they have to be modified in all configuration files of this labwork.

Install BIND in both DNS servers *ns1* and *ns2*

On both DNS servers, *ns1* and *ns2*, update the apt package cache by typing:

```
$sudo apt-get update
```

Now install BIND:

```
$sudo apt-get install bind9 bind9utils bind9-doc
```

Before continuing, let's set BIND to IPv4 mode since our private networking uses IPv4 exclusively. On both servers, edit the *OPTIONS* feature in bind9 default settings file */etc/default/bind9*

```
OPTIONS="-u bind -4"
```

Restart BIND to implement the changes:

```
$sudo systemctl restart bind9
```

Configure the Primary DNS Server *ns1*

Now it’s time to configure the primary DNS server *ns1*.

On *ns1*, open the file */etc/bind/named.conf.options* for editing.
First, you insert the configuration block ACL (access control list). This block is inserted above the *options* block. This is where you will define a list of clients that you will allow recursive DNS queries from. In this labwork, you will add *ns1*, *ns2*, *host1*, and *host2* to your list of trusted clients:

```
acl "trusted" {
        192.168.1.20;     # ns1
        192.168.1.21;     # ns2
        192.168.1.100;   # host1
        192.168.1.101;   # host2
};

options {

        . . .
```

Then, you have to edit the *options* block as below:

```
options {
      directory "/var/cache/bind";

      recursion yes;                 # enables recursive queries
      allow-recursion { trusted; };  # allows recursive queries
      listen-on { 192.168.1.20; };   # ns1 private IP address
      allow-transfer { none; };   # disable zone transfers by default

      forwarders {
              8.8.8.8;
              8.8.4.4;
      };

        . . .
};
```

Question 5: What is the role of the block *forwarders* inside the block *options*?

The above configuration specifies that only your own hosts (the "trusted" ones) will be able to query your DNS server for outside domains.

Next, you will configure the local file (*/etc/bind/named.conf.local*), to specify our DNS zones.
Configuring the local file is to specify your forward and reverse zones. DNS zones designate a specific scope for managing and defining DNS records. Since your domains will all be within the "*ds.soict.hust.com*" subdomain, you will use that as your forward zone. Because your servers' private IP addresses are each in the 192.168.1.0/24 IP space, you will set up a reverse zone so that you can define reverse lookups within that range.

Question 6: What is *forward lookup* and *reverse lookup* in DNS?

Open the file */etc/bind/named.conf.local* and add the forward zone with the following lines:

```
zone "ds.soict.hust.com" {
    type master;
    file "/etc/bind/zones/db.ds.soict.hust.com"; # zone file path
    allow-transfer { 192.168.1.21; };           # ns2 IP
};
```

Assuming that your private subnet is 192.168.1.0/24, add the reverse zone by with the following lines:

```
zone "1.168.192.in-addr.arpa" {
    type master;
    file "/etc/bind/zones/db.192.168.1";  # 192.168.1.0/24 subnet
    allow-transfer { 192.168.1.21; };  # ns2 IP
};
```

Question 7: What are the two files *db.ds.soict.hust.com* and *db.192.168.1*? What are they used for?

After, it's time to create the forward zone file.
You create a folder named zones and create a new forward zone file in making a copy of the *db.local* file as below:

```
$sudo mkdir /etc/bind/zones
$sudo cp /etc/bind/db.local /etc/bind/zones/db.ds.soict.hust.com
```

Now, you open the file *db.ds.soict.hust.com,* delete all the content and replace with the text below:

```
$TTL    604800
@     IN      SOA   ns1.ds.soict.hust.com. admin.ds.soict.hust.com. (
                 3     ; Serial
             604800     ; Refresh
              86400     ; Retry
            2419200     ; Expire
             604800 )   ; Negative Cache TTL
;
; name servers - NS records
     IN      NS      ns1.ds.soict.hust.com.
     IN      NS      ns2.ds.soict.hust.com.

; name servers - A records
ns1.ds.soict.hust.com.         IN      A       192.168.1.20
ns2.ds.soict.hust.com.         IN      A       192.168.1.21

; 192.168.1.0/24 - A records
host1.ds.soict.hust.com.        IN      A      192.168.1.100
host2.ds.soict.hust.com.        IN      A      192.168.1.101
```

Question 8: Explain three types of DNS record: *SOA*, *NS*, and *A*.

Then, you have to create the reverse zone file.
Reverse zone files are where we define DNS PTR records for reverse DNS lookups. That is, when the DNS receives a query by IP address, "192.168.1.100" for example, it will look in the reverse zone file(s) to resolve the corresponding FQDN, "*host1.ds.soict.hust.com*" in this case.

Now, you create the reverse zone file in basing on the sample *db.127* zone file:

```
$sudo cp /etc/bind/db.127 /etc/bind/zones/db.192.168.1
```

Open the file *db.192.168.1* , delete all the existing content and insert the content as below:

```
$TTL    604800
@      IN      SOA     ds.soict.hust.com. admin.ds.soict.hust.com. (
                             3         ; Serial
                        604800         ; Refresh
                         86400         ; Retry
                       2419200         ; Expire
                        604800 )       ; Negative Cache TTL
```

```
; name servers
      IN      NS      ns1.ds.soict.hust.com.
      IN      NS      ns2.ds.soict.hust.com.

; PTR Records
20    IN      PTR     ns1.ds.soict.hust.com.    ; 192.168.1.20
21    IN      PTR     ns2.ds.soict.hust.com.    ; 192.168.1.21
100   IN      PTR     host1.ds.soict.hust.com.  ; 192.168.1.100
101   IN      PTR     host2.ds.soict.hust.com.  ; 192.168.1.101
```

After, you Run the following command to check the syntax of the *named.conf** files:

```
$sudo named-checkconf
```

If your named configuration files have no syntax errors, you will return to your shell prompt and see no error messages.

You use *named-checkzone* command to check the "*ds.soict.hust.com*" forward zone configuration, run the following command:

```
$sudo named-checkzone ds.soict.hust.com
/etc/bind/zones/db.ds.soict.hust.com
```

And you check also the reverse zone configuration:

```
$sudo named-checkzone 1.168.192.in-addr.arpa
/etc/bind/zones/db.192.168.1
```

Question 9: What is the output of the above commands? Explain it!

When all of your configuration and zone files have no errors in them, you should be ready to restart the BIND service with the following commands:

```
$sudo systemctl restart bind9
$sudo ufw allow bind9
```

Question 10: which command you can use to make sure that the *bind9* is running?

Your primary DNS server is now setup and ready to respond to DNS queries. Let's move on to creating the secondary DNS server.

Configure the Secondary DSN Server *ns2*

In most environments, it is a good idea to set up a secondary DNS server that will respond to requests if the primary becomes unavailable.

Edit the file /etc/bind/named.conf.options as below:

```
acl "trusted" {
        192.168.1.20;    # ns1
        192.168.1.21;    # ns2
        192.168.1.100;  # host1
        192.168.1.101;  # host2
};

options {
      directory "/var/cache/bind";

      recursion yes;                 # enables resursive queries
      allow-recursion { trusted; };  # allows recursive queries
      listen-on { 192.168.1.21; };   # ns2 private IP address
      allow-transfer { none; };      # disable zone transfers by
      default

      forwarders {
              8.8.8.8;
              8.8.4.4;
      };
};
```

Now open the file /etc/bind/named.conf.local and insert the content as below:

```
zone "ds.soict.hust.com" {
    type slave;
    file "db.ds.soict.hust.com";
    masters { 192.168.1.20; };  # ns1 private IP
};

zone "1.168.192.in-addr.arpa" {
    type slave;
    file "db.192.168.1";
    masters { 192.168.1.20; };  # ns1 private IP
};
```

Run the following command to check the validity of your configuration files:

```
$sudo named-checkconf
```

Once that checks out, restart BIND:

```
$sudo systemctl restart bind9
$sudo ufw allow Bind9
```

The Client host1 and host1

Now, you work on the client pc (host1 or host2).

First, find the device associated with your private network by querying the private subnet with the *ip address* command:

```
$ip address show to 192.168.1.0/24
```

You will see the output like that:

```
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc
fq_codel state UP group default qlen 1000
...
```

So, in this example, the private interface is *eth1*. (but your private interface may be different).

Now, you configure the DNS server in open the file /etc/resolv.conf and add some entries as below:

```
nameserver 192.168.1.20
nameserver 192.168.1.21
search ds.soict.hust.com
```

Now, you can use the *nslookup* command to perform a forward lookup:

```
$nslookup host1
```

or

```
$nslookup host2
```

Question 11: What is the output you received after the command above? Explain how it works.

You do the same thing to perform a reverse lookup:

```
$nslookup 192.168.1.100
```

or

```
$nslookup 192.168.1.101
```

Question 12: What is the output you received after the command above? Explain how it works.

Question 13: If you want to add a host to your environment, you will want to add it to DNS. List of steps you have to do in order to perform that!