

***CSE 406 COMPUTER SECURITY SESSIONAL***

***Dos Attack ON Dns Server (using ip Spoofing)***

***Md Hasebul Hasan***

***Sid: 1605040***

## **Part (a) . Steps of attacks, snapshots, victim screen, etc.**

### **Overview:**

Here I implement Dos attack on Dns server (using ip spoofing) . For this attack I need three machines. One machine works as Dns server and others are the attacker machine and the normal user machine .Then From the attacker machine I execute a c code which performs a query flood attack to our DNS server . This dos attack affects our DNS server.DNS server stops to receive requests .So normal user can't visit there requested page because the ip address of their requested page can't be translated.

### **Step 1 : Setting Up a Local DNS Server and other machine**

The lab environment needs three separate machines: one for the normal user, one for the DNS server, and the other for the attacker. We will run these three virtual machines on one physical machine. All these VMs will run our pre-built Ubuntu VM image.For the VM network setting,we use "NAT Network" as the network adapter for each VM. We put all these VMs on the same network. We set the IP address of each machine using following mapping :

- (1) Dns server ip address 10.0.2.6
- (2) Normal user machine's ip address 10.0.2.5
- (3) Attacker machine's ip address 10.0.2.15

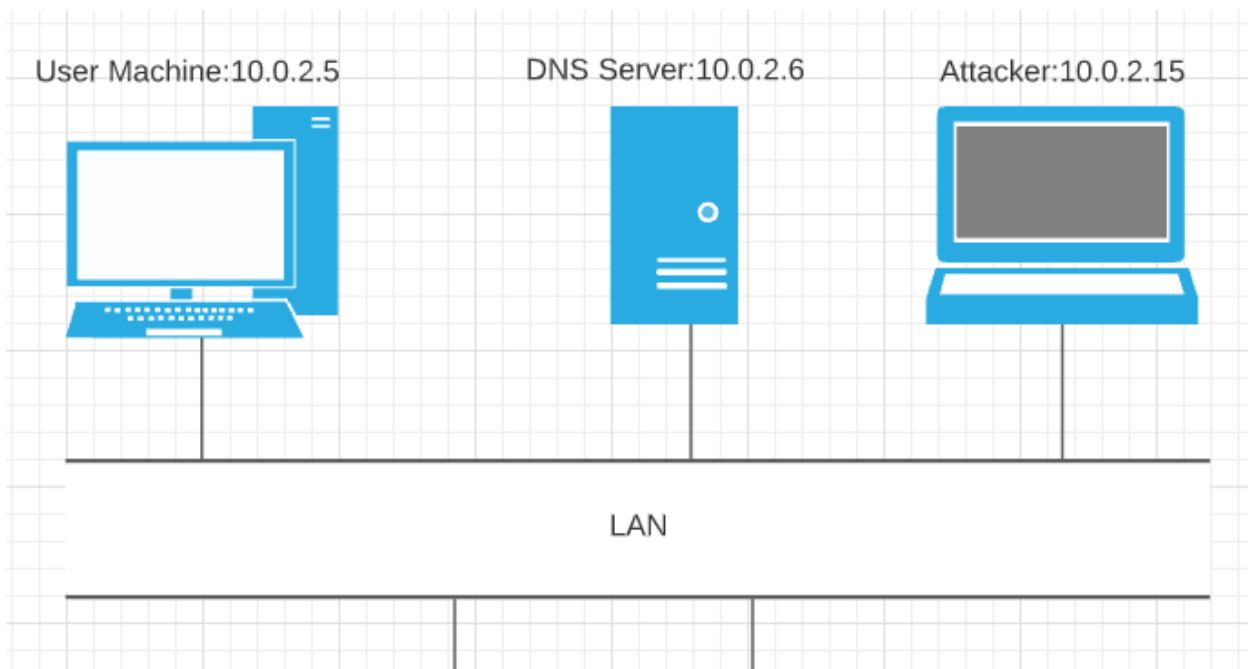


Fig 1: Network Topology

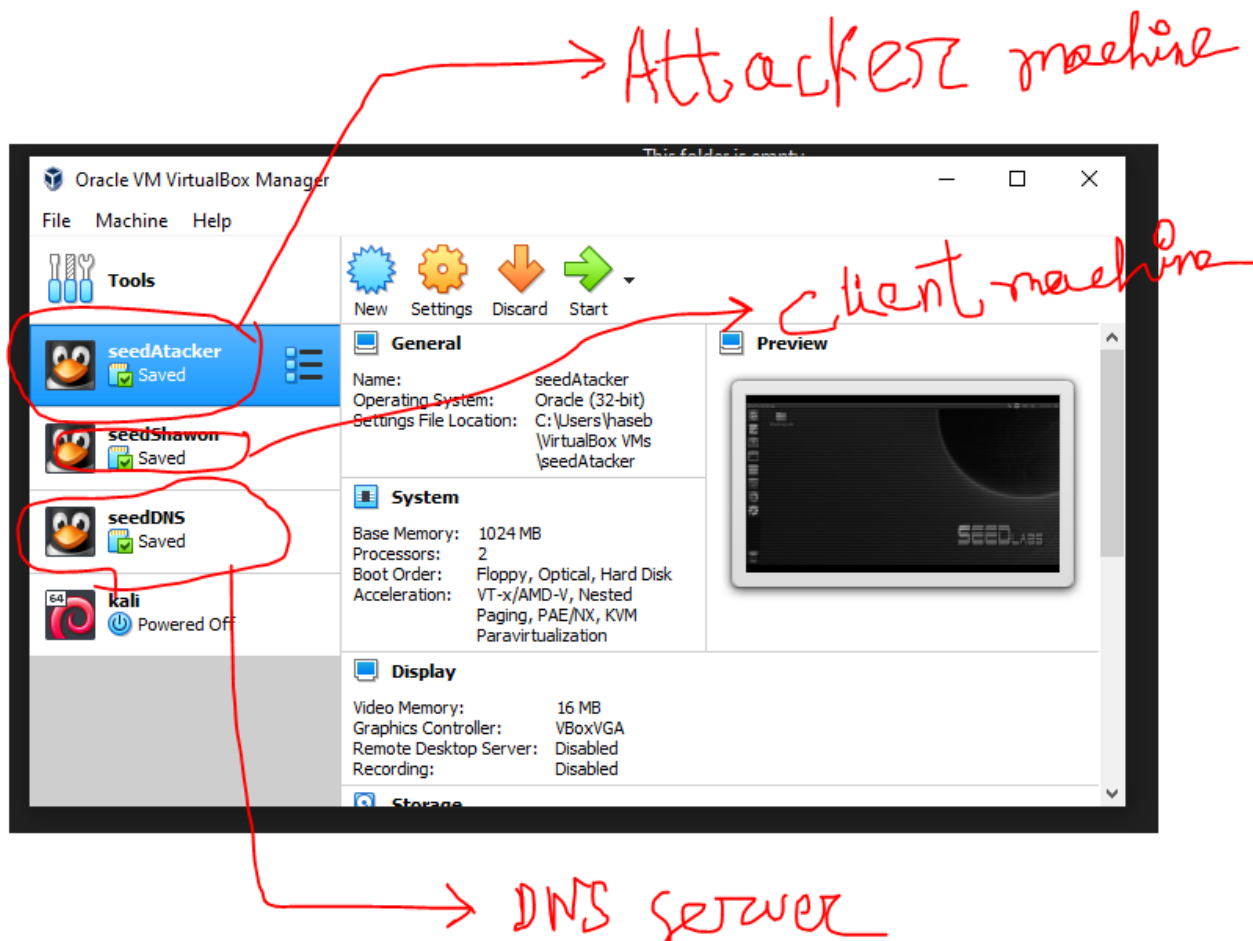


Fig 2:Setting up machine

Now we need to configure the user machine and the local DNS server; for the attacker machine, the default setup in the VM should be sufficient.

### **Step 1.1: Set up a Local DNS Server:**

For the local DNS server, we need to run a DNS server program. The most widely used DNS server software is called BIND (Berkeley Internet Name Domain). The BIND 9 server program is already installed in our pre-built Ubuntu VM image.

#### **Step 1.1.(a): Configure the BIND 9 server:**

BIND 9 gets its configuration from a file called ***/etc/bind/named.conf***. This file is the primary configuration file, and it usually contains several "***include***" entries, i.e., the actual configurations are stored in those included files. One of the included files is called ***/etc/bind/named.conf.options***.

This is where I set up the configuration options. Let first set up an option related to DNS cache by adding a dump-file entry to the options block:

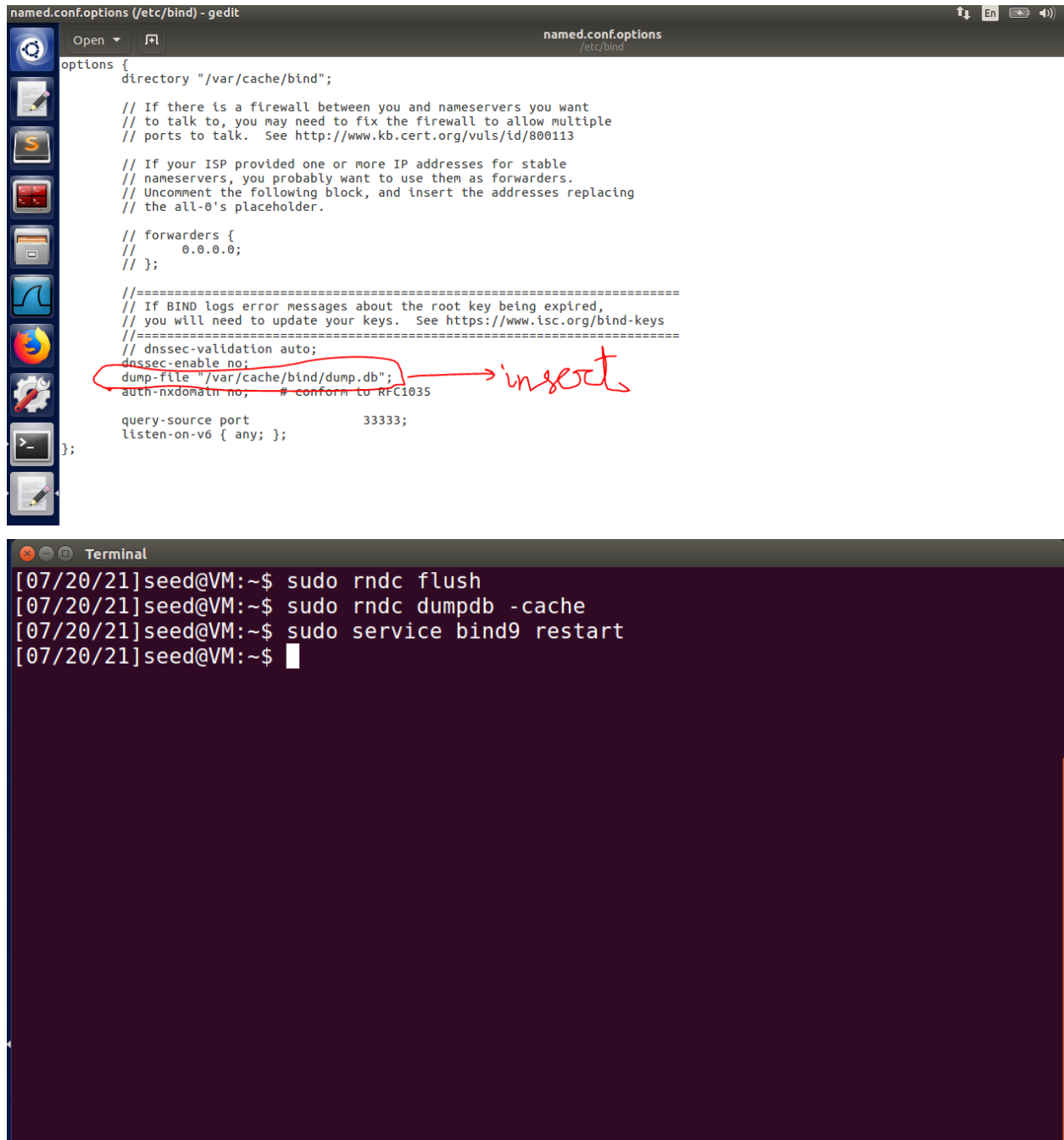
```
options {  
dump-file "/var/cache/bind/dump.db";  
};
```

The above option specifies where the cache content should be dumped to if BIND is asked to dump its cache. If this option is not specified, BIND dumps the cache to a default file called ***/var/cache/bind/named\_dump.db***. The two commands shown below are related to DNS cache. The first command dumps the content of the cache to the file specified above, and the second command clears the cache.

```
$ sudo rndc dumpdb -cache // Dump the cache to the specified file  
$ sudo rndc flush // Flush the DNS cache
```

```
Terminal
[07/20/21]seed@VM:~$ sudo apt-get install bind9
Reading package lists... Done
Building dependency tree
Reading state information... Done
bind9 is already the newest version (1:9.10.3.dfsg.P4-8ubuntu1.7).
0 upgraded, 0 newly installed, 0 to remove and 2 not upgraded.
[07/20/21]seed@VM:~$
```

```
[07/20/21]seed@VM:~$ sudo gedit /etc/bind/named.conf.options
```



The image shows a Gedit editor window titled "named.conf.options (/etc/bind) - gedit" with the file "named.conf.options" open at "/etc/bind". The editor displays the configuration for the BIND9 daemon. A red circle highlights the line `dnssec-validation auto;`, and a red arrow points from it to the word "insert" written in red. Below the editor is a terminal window titled "Terminal" showing the following commands and their output:

```
[07/20/21]seed@VM:~$ sudo rndc flush
[07/20/21]seed@VM:~$ sudo rndc dumpdb -cache
[07/20/21]seed@VM:~$ sudo service bind9 restart
[07/20/21]seed@VM:~$
```

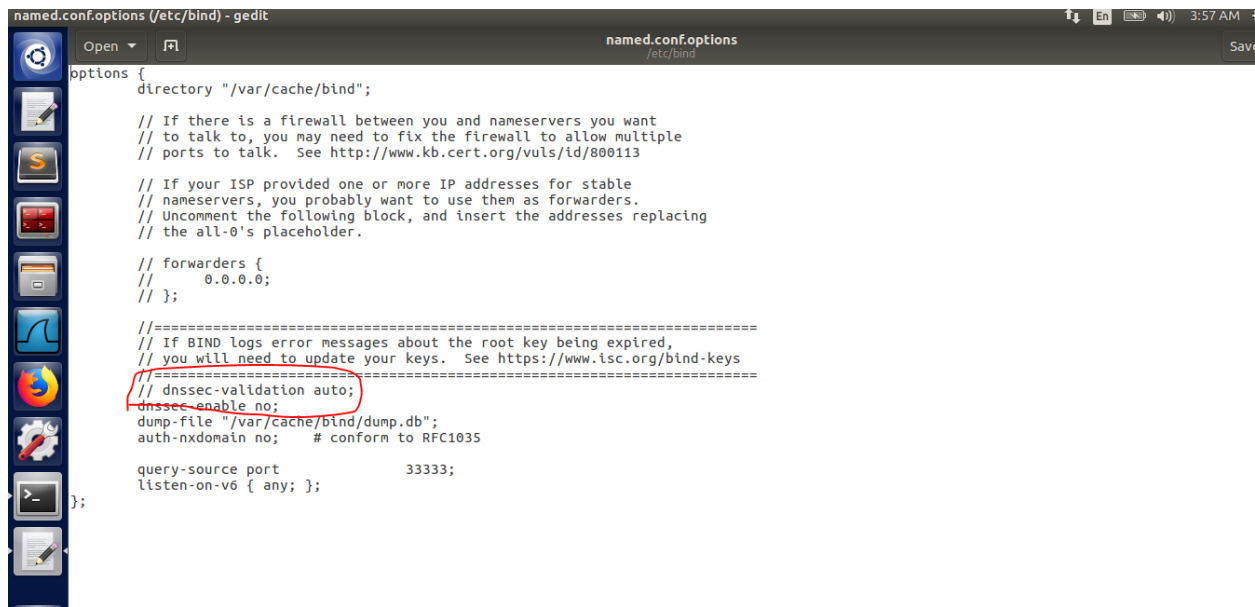
### **Step 1.1.(b): Turn off DNSSEC:**

DNSSEC is introduced to protect against spoofing attacks on DNS servers.

To show how attacks work without this protection mechanism, I turn the protection off.

This is done by modifying the ***named.conf.options*** file: comment out the ***dnssec-validation*** entry, and add a ***dnssec-enable*** entry.

```
options {  
# dnssec-validation auto;  
dnssec-enable no;  
};
```



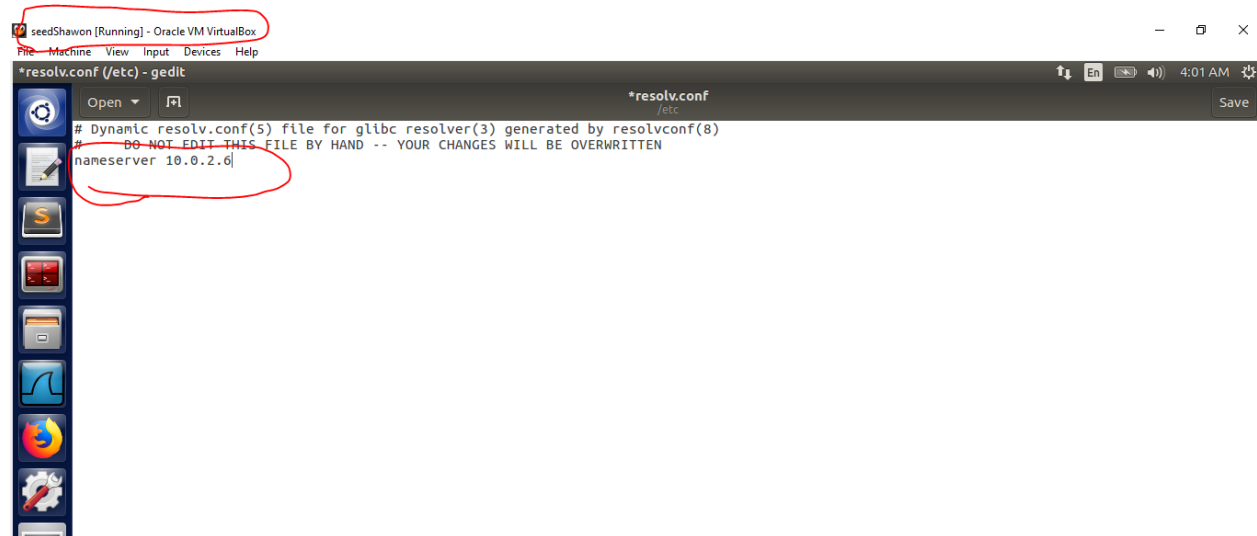
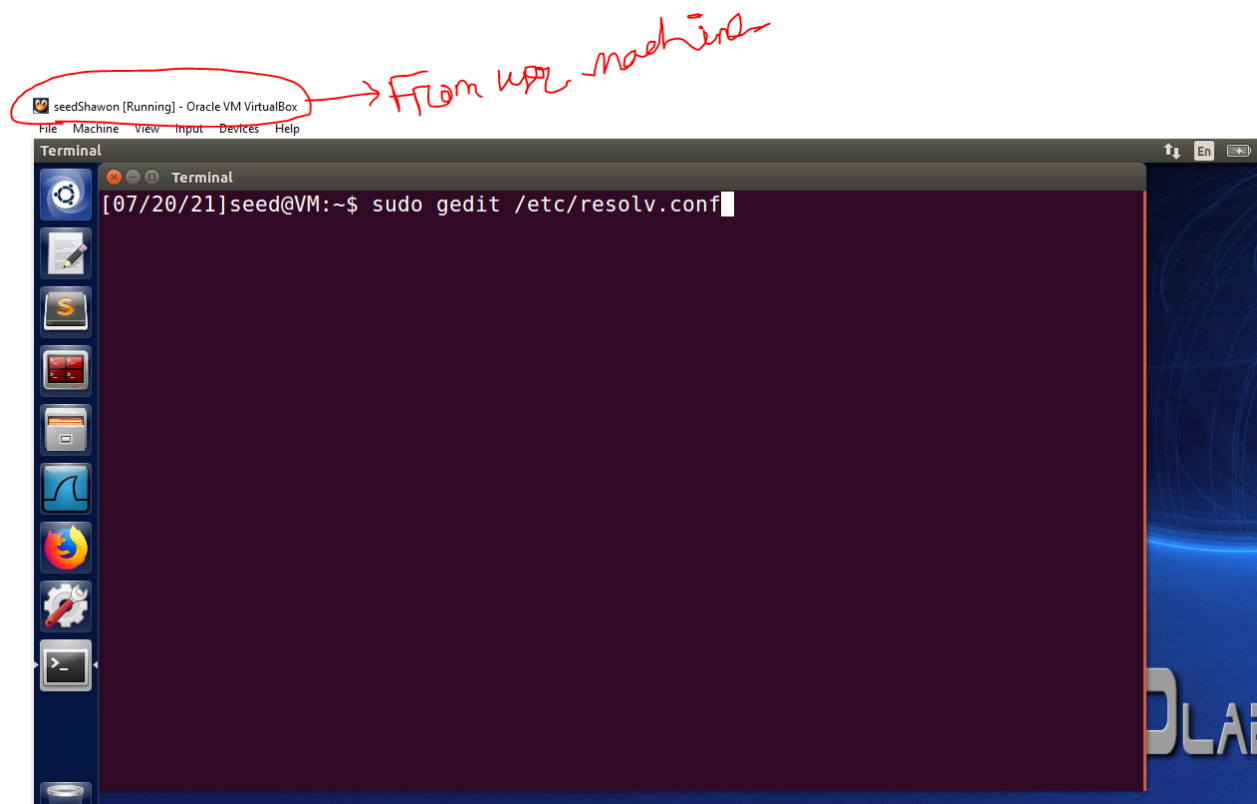
### **Step 1.1(c): Start DNS server:**

Restart the dns server by following command :

```
$ sudo service bind9 restart
```

### Step 1.2 : Configure User Machine:

On the user machine **10.0.2.5**, I use **10.0.2.6** as the local DNS server (by default, the DNS server program is already running in the SEED VM). This is achieved by changing the resolver configuration file (**/etc/resolv.conf**) of the user machine, so the server **10.0.2.6** is added as the first name server entry in the file, i.e., this server will be used as the primary DNS server.





### Step 1.3: Checking Dns server working properly before attack

Now from the client machine I visit [www.codeforces.com](https://www.codeforces.com) & open wireshark to track the request. Here I see the user machine (ip:10.0.2.5) uses our local DNS server(ip:10.0.2.6) for ip translation. We can properly visit our requested website.

The screenshot displays a virtual machine environment with two windows open. The left window is Wireshark, showing a network capture of DNS traffic. The right window is Mozilla Firefox, displaying the Codeforces website.

**Wireshark Capture Data:**

| No.  | Time                           | Source   | Destination | Protocol |
|------|--------------------------------|----------|-------------|----------|
| 5336 | 2021-07-20 04:30:57.0896249... | 10.0.2.5 | 10.0.2.6    | DNS      |
| 5337 | 2021-07-20 04:30:57.0896654... | 10.0.2.5 | 10.0.2.6    | DNS      |
| 5339 | 2021-07-20 04:30:57.0900664... | 10.0.2.6 | 10.0.2.5    | DNS      |
| 5341 | 2021-07-20 04:30:57.0900958... | 10.0.2.6 | 10.0.2.5    | DNS      |
| 5343 | 2021-07-20 04:30:57.0940575... | 10.0.2.5 | 10.0.2.6    | DNS      |
| 5344 | 2021-07-20 04:30:57.0940948... | 10.0.2.5 | 10.0.2.6    | DNS      |
| 5346 | 2021-07-20 04:30:57.0946410... | 10.0.2.6 | 10.0.2.5    | DNS      |
| 5347 | 2021-07-20 04:30:57.0946656... | 10.0.2.6 | 10.0.2.5    | DNS      |
| 5368 | 2021-07-20 04:30:57.4077434... | 10.0.2.5 | 10.0.2.6    | DNS      |
| 5369 | 2021-07-20 04:30:57.4077434... | 10.0.2.5 | 10.0.2.6    | DNS      |
| 5370 | 2021-07-20 04:30:57.4091320... | 10.0.2.6 | 10.0.2.5    | DNS      |
| 5371 | 2021-07-20 04:30:57.4091862... | 10.0.2.6 | 10.0.2.5    | DNS      |
| 5399 | 2021-07-20 04:30:58.1297337... | 10.0.2.5 | 10.0.2.6    | DNS      |
| 5400 | 2021-07-20 04:30:58.1298062... | 10.0.2.5 | 10.0.2.6    | DNS      |
| 5401 | 2021-07-20 04:30:58.1302886... | 10.0.2.6 | 10.0.2.5    | DNS      |
| 5402 | 2021-07-20 04:30:58.1306865... | 10.0.2.6 | 10.0.2.5    | DNS      |
| 5403 | 2021-07-20 04:30:58.1476645... | 10.0.2.5 | 10.0.2.6    | DNS      |
| 5404 | 2021-07-20 04:30:58.1477470... | 10.0.2.5 | 10.0.2.6    | DNS      |

**Firefox Browser Content:**

Codeforces - Mozilla Firefox

Codeforces

Most Visited SEED Labs Sites for Labs

**CODEFORCES**  
Sponsored by Telegram

HOME TOP CONTESTS GYM PROBLEMSET GROUPS RATING

Codeforces Round #733 (Div. 1 + Div. 2)

By tourist, 4 days ago, translation,

Hello, Codeforces!

Welcome to the Codeforces Round #733 (Div. 1 + Div. 2, based on VK Cup 2021 Elimination (Engine)) that will start on Saturday, July 17, 2021 combined rated round for both divisions.

This round is a mirror of VK Cup 2021 Elimination. VK Cup is a Russian-speaking competitors organized by VK — a social network in Saint Petersburg and the most popular website in Russia. VK Cup has grown to be a four-track competition in competitive programming development.

All the problems were authored and prepared by me. Big thanks to this round could not be possible: [DmitryKrasovskiy](#), [VAM](#), [Igor](#)

## **Step 2:Execute Attacking Code from Attacker Machine & Attack Dns server:**

Now from the attacker machine I write a code named **attack.c**. This code takes a website address and dns server ip address from the command line . Then make a Dns request and send this request within an infinite loop. While sending dns requests it also spoofs it's ip address randomly.

I also write a bash file named **attack.sh** . Which have following line of code for execute our code :

**echo "compile code"**

**gcc -o dnsdosAttack attack.c** // compile our code and save as dnsdosAttack

**echo "compiled"**

**echo "send request to 10.0.2.6 dns server"**

**sudo ./dnsdosAttack www.buet.ac.bd 10.0.2.6** //perform attack

**echo "sending request"**

Now the attacker machine runs the attacker.sh file from the terminal .

The image shows a screenshot of an Oracle VM VirtualBox window titled "seedAtacker [Running] - Oracle VM VirtualBox". The window contains a terminal window titled "attack.sh (~/Desktop/Attacking-code) - gedit". The terminal window has a dark background and a light-colored text area. The text area contains the following code:

```
echo "compile code"
gcc -o dnsdosAttack attack.c
echo "compiled"
echo "send rrequest to 10.0.2.6 dns server"
sudo ./dnsdosAttack www.buet.ac.bd 10.0.2.6
echo "sending request"
```

The terminal window also shows the output of the script execution:

```
[07/20/21]seed@VM:~/.../Attacking-code$ bash attack.sh
compile code
attack.c: In function 'main':
attack.c:95:25: warning: implicit declaration of function 'time' [-Wimplicit-function-declaration]
    random((unsigned long)time(NULL));
                        ^
attack.c:122:3: warning: implicit declaration of function 'inet_pton' [-Wimplicit-function-declaration]
    inet_pton(AF_INET, argv[optind], &sin_dst.sin_addr);
    ^
compiled
send request to 10.0.2.6 dns server
```

The terminal window also shows the status bar at the bottom, which includes the text "sh Tab Width: 8 Ln 6, Col 23 INS" and a "Right Ctrl" button.

Fig:Execute Attacking code

## **Part(b) Observed output in attacker PC, victim PC, and other related PC:**

At this point we observed our output of the attack. Initially our user machine can visit any website using our local dns server .

Now we run our attacking code from the attacker machine . This continuously sends dns queries and all queries Ip is spoofed. For observing this result we run wireshark in the dns server and see the packet of the dns request .Here we see continuously coming dns requests each time having a different ip address for each.

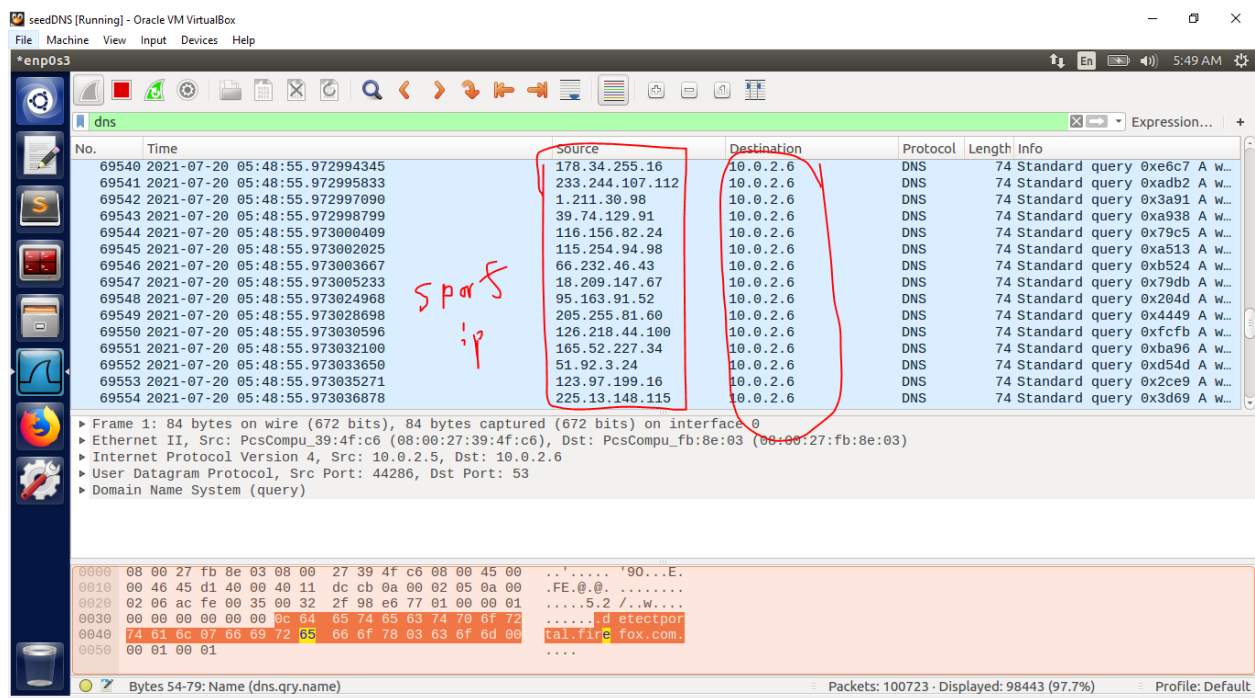
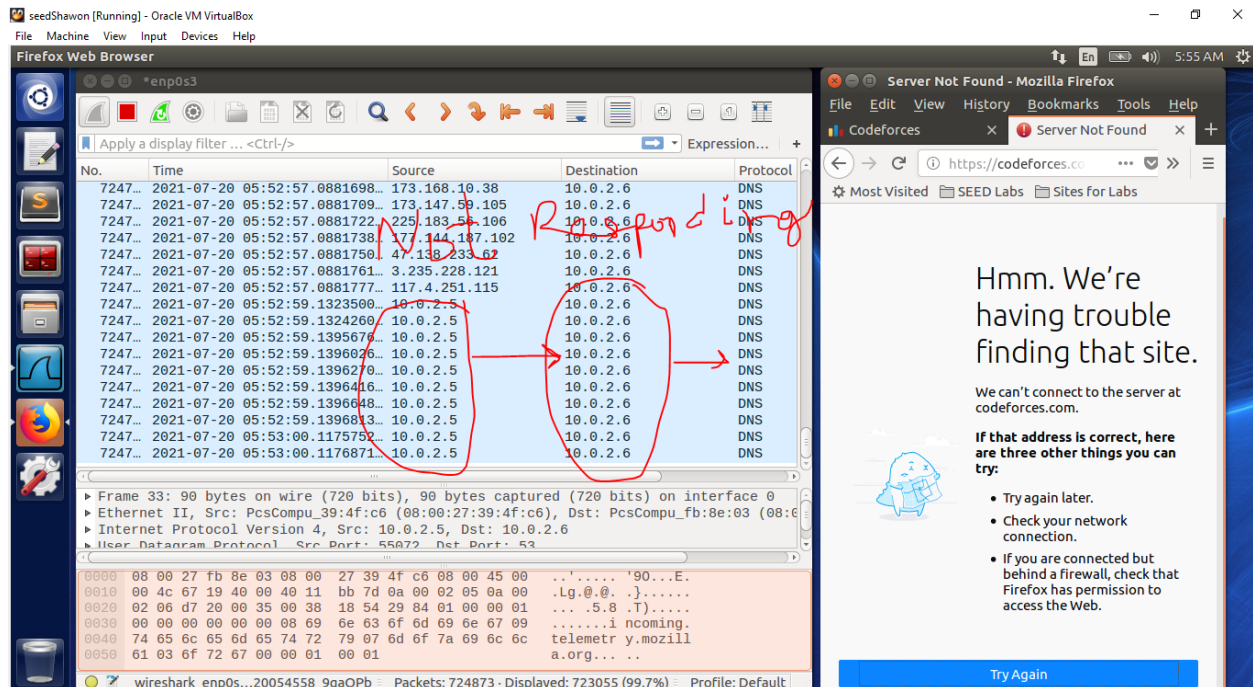


Fig : wireshark dns packet in dns server

For each 30micro second later a packet comes from the attacking machine. Each Packet source ip is different from the attacking machine ip address and they are different from each other. So I can also spoof the source ip address of the dns query.

Now from the user machine we try to visit [www.codeforces.com](https://www.codeforces.com) . And this time we can't access the website . If we run wireshark on this user machine . Then we see a dns request sent to our local DNS server . But the dns server did not respond. So our attack was expectedly successful. Our Dns server did not respond to other requests because it was affected by dos attack.



The attack effect lasted about 25-40 minutes after the code was stopped in the attacker machine. We can't visit any websites during this time. After half an hour we can again visit [www.codeforces.com](https://www.codeforces.com) .

**Part(c). Is your attack successful? Why do you think it was successful? Why not?**

Yes, my attack is successful. I successfully implemented a dos attack on my local dns server using ip spoofing. I send dns query from an infinite loop changing source ip address . So for each 20/30 micro second later the server gets a dns query request whose ip is different each time. I observed this from the wireshark of the dns server machine. So I can spoof the source ip . Because of so many query requests my server can't accept the query from another machine and if I run my code for about 5-10 mins my server becomes down . That's why i think my attack was successful.

**Part(d). Did you design any countermeasures for such an attack? How?**

No, I didn't.

***Go to this link to see my demonstration :***

**[My demonstration link](#)**