Lab Assignment: Domain Name System (DNS) Configuration and Analysis

Objective:

- Understand how **DNS works** and its role in **network communication**.
- Set up a **local DNS server** in a simulated environment.
- Configure and test **DNS query resolution**.
- Use **Wireshark** to capture and analyze **DNS traffic**.

Lab Requirements:

- A computer with **Linux (Ubuntu/CentOS) or Windows** (WSL or Virtual Machine recommended).
- **BIND9 (for Linux)** or **Simple DNS Plus (for Windows)** installed.
- **Wireshark** for packet capture and analysis.
- **Internet access** (optional, if testing external domains).

Part 1: Setting Up a Local DNS Server

Step 1: Install a DNS Server

For Linux (Ubuntu/Debian):

1. Open the terminal and install **BIND9**:

```
bash
CopyEdit
sudo apt update
sudo apt install bind9 -y
```

2. Start and enable the DNS service:

```
bash
CopyEdit
sudo systemctl start bind9
sudo systemctl enable bind9
```

For Windows (Using Simple DNS Plus or Windows DNS Server):

1. Download and install **Simple DNS Plus** or enable **Windows DNS Server** via Server Manager.
2. Configure the primary DNS zone in the **DNS Manager**.

Step 2: Configure a Local DNS Zone (For Internal Domains)

1. Open the **BIND configuration file**:

```
bash
CopyEdit
sudo nano /etc/bind/named.conf.local
```

2. Add the following zone configuration for a **local domain (example.local)**:

```
bash
CopyEdit
zone "example.local" {
    type master;
    file "/etc/bind/db.example.local";
};
```

3. Create a **zone file** for the domain:

```
bash
CopyEdit
sudo nano /etc/bind/db.example.local
```

4. Add the following **DNS records**:

```
lua
CopyEdit
$TTL 604800
@   IN  SOA  ns1.example.local. admin.example.local. (
             2     ; Serial
        604800     ; Refresh
         86400     ; Retry
       2419200     ; Expire
        604800 )   ; Negative Cache TTL

; Name Servers
@       IN  NS   ns1.example.local.

; A Records
ns1     IN  A    192.168.1.1
www     IN  A    192.168.1.2
```

5. Restart the DNS service:

```
bash
CopyEdit
sudo systemctl restart bind9
```

6. Verify the DNS server status:

```
bash
CopyEdit
sudo systemctl status bind9
```

Part 2: DNS Query Resolution Testing

Step 1: Configure a Client to Use the Local DNS Server

1. Edit the **resolv.conf** file on the client system:

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

2. Add the following:

   ```
   nginx
   CopyEdit
   nameserver 192.168.1.1
   domain example.local
   ```

Step 2: Test DNS Resolution Using nslookup and dig

Using nslookup:

```
bash
CopyEdit
nslookup www.example.local
```

Expected Output:

```
yaml
CopyEdit
Server:  192.168.1.1
Address:  192.168.1.1#53
Name:    www.example.local
Address: 192.168.1.2
```

Using dig:

```
bash
CopyEdit
dig www.example.local
```

Expected Output:

```
less
CopyEdit
;; ANSWER SECTION:
www.example.local. 604800 IN A 192.168.1.2
```

Part 3: DNS Packet Analysis with Wireshark

Step 1: Capture DNS Traffic

1. Open **Wireshark** and select the **network interface** connected to the DNS server.
2. Start **packet capture** and apply the filter:

   ```
   nginx
   CopyEdit
   dns
   ```

3. Open a **new terminal** and run:

   ```
   bash
   CopyEdit
   nslookup www.example.local
   ```

4. Observe the **DNS query and response** packets in Wireshark.

Step 2: Analyze DNS Query Resolution

Look for:
1. DNS Request (Standard Query A) – Sent by the client to the DNS server.
2. DNS Response (Standard Query Response) – Reply from the DNS server with the IP address.
1. Query Type (A, AAAA, MX, NS, etc.) – Identifies the requested record type.
2. Query Time – Measures DNS resolution time.

Lab Report Submission Requirements:

- **Screenshots** of:
  - Local DNS configuration (`db.example.local` file).
  - **DNS query results using `nslookup` and `dig`.**
  - **Wireshark capture showing DNS request and response**.