Polytechnic University of Puerto Rico

Department of Electrical Engineering

Hato Rey

Module 9 Project COE 4331

Computer Networks Laboratory

January 24, 2025

Schmidt Montilla, Coral S., Student ID: 148830

Professor: Juan Tovar

Table of Contents

Introduction

This lab focuses on the practical implementation of configuring a private Domain Name System (DNS) server using BIND on a Linux system. The purpose is to establish an efficient, centralized DNS infrastructure for internal network management. By resolving private hostnames and IP addresses, this setup simplifies network configurations and enhances scalability, especially in environments with multiple servers. The step-by-step approach covered both forward and reverse DNS lookups, emphasizing precision in creating zone files, maintaining records, and troubleshooting. Optional steps included configuring a secondary DNS server, reinforcing the importance of redundancy in ensuring network reliability.

Method

TUTORIAL - How To Configure BIND as a Private Network DNS Server on Linux Ubuntu.

Prerequisites

To complete this tutorial, you will need the following:

- Some servers that are running in the same datacenter and have private networking enabled
- A new VPS to serve as the Primary DNS server, *ns1*
- Optional: A new VPS to serve as a Secondary DNS server, *ns2*
- Root access to all of the above (steps 1-4 here)

If you are unfamiliar with DNS concepts, it is recommended that you read at least the first three parts of our Introduction to Managing DNS.

Example Hosts

For example purposes, we will assume the following:

- We have two existing VPS called "host1" and "host2"
- Both VPS exist in the nyc3 datacenter
- Both VPS have private networking enabled (and are on the 10.128.0.0/16 subnet)
- Both VPS are somehow related to our web application that runs on "example.com"

With these assumptions, we decide that it makes sense to use a naming scheme that uses "nyc3.example.com" to refer to our private subnet or zone. Therefore, *host1*'s private Fully-Qualified Domain Name (FQDN) will be "host1.nyc3.example.com". Refer to the following table the relevant details:

Host	Role	Private FQDN	Private IP Address
host1	Generic Host 1	host1.nyc3.example.com	10.128.100.101

Host	Role	Private FQDN	Private IP Address
host2	Generic Host 2	host2.nyc3.example.com	10.128.200.102

Note: Your existing setup will be different, but the example names and IP addresses will be used to demonstrate how to configure a DNS server to provide a functioning internal DNS. You should be able to easily adapt this setup to your own environment by replacing the host names and private IP addresses with your own. It is not necessary to use the region name of the datacenter in your naming scheme, but we use it here to denote that these hosts belong to a particular datacenter's private network. If you utilize multiple datacenters, you can set up an internal DNS within each respective datacenter.

Our Goal

By the end of this tutorial, we will have a primary DNS server, *ns1*, and optionally a secondary DNS server, *ns2*, which will serve as a backup.

Here is a table with example names and IP addresses:

Host	Role	Private FQDN	Private IP Address
ns1	Primary DNS Server	ns1.nyc3.example.com	10.128.10.11
ns2	Secondary DNS Server	ns2.nyc3.example.com	10.128.20.12

Let's get started by installing our Primary DNS server, ns1.

Install BIND on DNS Servers

Note: Text that is highlighted in red is important! It will often be used to denote something that needs to be replaced with your own settings or that it should be modified or added to a configuration file. For example, if you see something like host1.nyc3.example.com, replace it with the FQDN of your own server. Likewise, if you see host1_private_IP, replace it with the private IP address of your own server.
On both DNS servers, *ns1* and *ns2*, update apt:

```
sudo apt-get update
```

```
•
```

Copy
Now install BIND:

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

Copy

IPv4 Mode

Before continuing, let's set BIND to IPv4 mode. On both servers, edit the `bind9` service parameters file:

```
•  sudo vi /etc/default/bind9
•
```

Copy
Add "-4" to the `OPTIONS` variable. It should look like the following:

/etc/default/bind9

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

Copy
Save and exit.

Now that BIND is installed, let's configure the primary DNS server.

Configure Primary DNS Server

BIND's configuration consists of multiple files, which are included from the main configuration file, `named.conf`. These filenames begin with "named" because that is the name of the process that BIND runs. We will start with configuring the options file.

Configure Options File

On *ns1*, open the `named.conf.options` file for editing:

```
•  sudo vi /etc/bind/named.conf.options
•
```

Copy

Above the existing `options` block, create a new ACL block called "trusted". This is where we will define list of clients that we will allow recursive DNS queries from (i.e. your servers that are in the same datacenter as ns1). Using our example private IP addresses, we will add *ns1*, *ns2*, *host1*, and *host2* to our list of trusted clients:

/etc/bind/named.conf.options — 1 of 3

```
acl "trusted" {
        10.128.10.11;    # ns1 - can be set to localhost
        10.128.20.12;    # ns2
        10.128.100.101;  # host1
        10.128.200.102;  # host2
};
```

Copy

Now that we have our list of trusted DNS clients, we will want to edit the `options` block. Currently, the start of the block looks like the following:

/etc/bind/named.conf.options — 2 of 3

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

Copy

Below the `directory` directive, add the highlighted configuration lines (and substitute in the proper *ns1* IP address) so it looks something like this:

/etc/bind/named.conf.options — 3 of 3

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

        recursion yes;                 # enables resursive queries
        allow-recursion { trusted; };  # allows recursive queries from "trusted"
clients
        listen-on { 10.128.10.11; };   # ns1 private IP address - listen on private
network only
        allow-transfer { none; };      # disable zone transfers by default

        forwarders {
                8.8.8.8;
                8.8.4.4;
```

```
        };
...
};
```
Copy
Now save and exit `named.conf.options`. The above configuration specifies that only your own servers (the "trusted" ones) will be able to query your DNS server.

Next, we will configure the local file, to specify our DNS zones.

Configure Local File

On *ns1*, open the `named.conf.local` file for editing:

- `sudo vi /etc/bind/named.conf.local`

Copy
Aside from a few comments, the file should be empty. Here, we will specify our forward and reverse zones.

Add the forward zone with the following lines (substitute the zone name with your own):

/etc/bind/named.conf.local — 1 of 2
```
zone "nyc3.example.com" {
    type master;
    file "/etc/bind/zones/db.nyc3.example.com"; # zone file path
    allow-transfer { 10.128.20.12; };         # ns2 private IP address - secondary
};
```
Copy
Assuming that our private subnet is *10.128.0.0/16*, add the reverse zone by with the following lines (note that our reverse zone name starts with "128.10" which is the octet reversal of "10.128"):

/etc/bind/named.conf.local — 2 of 2
```
zone "128.10.in-addr.arpa" {
    type master;
    file "/etc/bind/zones/db.10.128";  # 10.128.0.0/16 subnet
    allow-transfer { 10.128.20.12; };  # ns2 private IP address - secondary
};
```
Copy

If your servers span multiple private subnets but are in the same datacenter, be sure to specify an additional zone and zone file for each distinct subnet. When you are finished adding all of your desired zones, save and exit the `named.conf.local` file.

Now that our zones are specified in BIND, we need to create the corresponding forward and reverse zone files.

Create Forward Zone File

The forward zone file is where we define DNS records for forward DNS lookups. That is, when the DNS receives a name query, "host1.nyc3.example.com" for example, it will look in the forward zone file to resolve *host1*'s corresponding private IP address.

Let's create the directory where our zone files will reside. According to our *named.conf.local* configuration, that location should be `/etc/bind/zones`:

```
sudo mkdir /etc/bind/zones
```

Copy

We will base our forward zone file on the sample `db.local` zone file. Copy it to the proper location with the following commands:

```
cd /etc/bind/zones

sudo cp ../db.local ./db.nyc3.example.com
```

Copy

Now let's edit our forward zone file:

```
sudo vi /etc/bind/zones/db.nyc3.example.com
```

Copy

Initially, it will look something like the following:

/etc/bind/zones/db.nyc3.example.com — original

```
$TTL    604800
@       IN      SOA     localhost. root.localhost. (
                              2         ; Serial
                         604800         ; Refresh
```

```
                        86400         ; Retry
                      2419200         ; Expire
                       604800 )       ; Negative Cache TTL
;
@       IN      NS      localhost.      ; delete this line
@       IN      A       127.0.0.1       ; delete this line
@       IN      AAAA    ::1             ; delete this line
```

Copy

First, you will want to edit the SOA record. Replace the first "localhost" with *ns1*'s FQDN, then replace "root.localhost" with "admin.nyc3.example.com". Also, every time you edit a zone file, you should increment the *serial* value before you restart the `named` process–we will increment it to "3". It should look something like this:

/etc/bind/zones/db.nyc3.example.com — updated 1 of 3

```
@       IN      SOA     ns1.nyc3.example.com. admin.nyc3.example.com. (
                              3         ; Serial
```

Copy

Now delete the three records at the end of the file (after the SOA record). If you're not sure which lines to delete, they are marked with a "delete this line" comment above.

At the end of the file, add your nameserver records with the following lines (replace the names with your own). Note that the second column specifies that these are "NS" records:

/etc/bind/zones/db.nyc3.example.com — updated 2 of 3

```
; name servers - NS records
    IN      NS      ns1.nyc3.example.com.
    IN      NS      ns2.nyc3.example.com.
```

Copy

Then add the A records for your hosts that belong in this zone. This includes any server whose name we want to end with ".nyc3.example.com" (substitute the names and private IP addresses). Using our example names and private IP addresses, we will add A records for *ns1*, *ns2*, *host1*, and *host2* like so:

/etc/bind/zones/db.nyc3.example.com — updated 3 of 3

```
; name servers - A records
ns1.nyc3.example.com.          IN      A       10.128.10.11
ns2.nyc3.example.com.          IN      A       10.128.20.12
```

```
; 10.128.0.0/16 - A records
host1.nyc3.example.com.        IN      A      10.128.100.101
host2.nyc3.example.com.        IN      A      10.128.200.102
```

Copy

Save and exit the `db.nyc3.example.com` file.

Our final example forward zone file looks like the following:

/etc/bind/zones/db.nyc3.example.com — updated

```
$TTL    604800
@       IN      SOA     ns1.nyc3.example.com. admin.nyc3.example.com. (
                  3       ; Serial
             604800     ; Refresh
              86400     ; Retry
            2419200     ; Expire
             604800 )   ; Negative Cache TTL
;
; name servers - NS records
     IN      NS      ns1.nyc3.example.com.
     IN      NS      ns2.nyc3.example.com.

; name servers - A records
ns1.nyc3.example.com.          IN      A       10.128.10.11
ns2.nyc3.example.com.          IN      A       10.128.20.12

; 10.128.0.0/16 - A records
host1.nyc3.example.com.        IN      A      10.128.100.101
host2.nyc3.example.com.        IN      A      10.128.200.102
```

Copy

Now let's move onto the reverse zone file(s).

Create Reverse Zone File(s)

Reverse zone file are where we define DNS PTR records for reverse DNS lookups. That is, when the DNS receives a query by IP address, "10.128.100.101" for example, it will look in the reverse zone file(s) to resolve the corresponding FQDN, "host1.nyc3.example.com" in this case.

On *ns1*, for each reverse zone specified in the `named.conf.local` file, create a reverse zone file. We will base our reverse zone file(s) on the sample `db.127` zone file. Copy it to the proper location with the following commands (substituting the destination filename so it matches your reverse zone definition):

```
cd /etc/bind/zones

sudo cp ../db.127 ./db.10.128

```

Copy

Edit the reverse zone file that corresponds to the reverse zone(s) defined in `named.conf.local`:

```
sudo vi /etc/bind/zones/db.10.128

```

Copy

Initially, it will look something like the following:

/etc/bind/zones/db.10.128 — original

```
$TTL    604800
@       IN      SOA     localhost. root.localhost. (
                              1         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                         604800 )       ; Negative Cache TTL
;
@       IN      NS      localhost.      ; delete this line
1.0.0   IN      PTR     localhost.      ; delete this line
```

Copy

In the same manner as the forward zone file, you will want to edit the SOA record and increment the *serial* value. It should look something like this:

/etc/bind/zones/db.10.128 — updated 1 of 3

```
@       IN      SOA     ns1.nyc3.example.com. admin.nyc3.example.com. (
                              3         ; Serial
```

Copy

Now delete the two records at the end of the file (after the SOA record). If you're not sure which lines to delete, they are marked with a "delete this line" comment above.

At the end of the file, add your nameserver records with the following lines (replace the names with your own). Note that the second column specifies that these are “NS” records:

/etc/bind/zones/db.10.128 — updated 2 of 3

```
; name servers - NS records
      IN      NS      ns1.nyc3.example.com.
      IN      NS      ns2.nyc3.example.com.
```

Copy

Then add `PTR` records for all of your servers whose IP addresses are on the subnet of the zone file that you are editing. In our example, this includes all of our hosts because they are all on the 10.128.0.0/16 subnet. Note that the first column consists of the last two octets of your servers’ private IP addresses in reversed order. Be sure to substitute names and private IP addresses to match your servers:

/etc/bind/zones/db.10.128 — updated 3 of 3

```
; PTR Records
11.10   IN      PTR     ns1.nyc3.example.com.    ; 10.128.10.11
12.20   IN      PTR     ns2.nyc3.example.com.    ; 10.128.20.12
101.100 IN      PTR     host1.nyc3.example.com.  ; 10.128.100.101
102.200 IN      PTR     host2.nyc3.example.com.  ; 10.128.200.102
```

Copy

Save and exit the reverse zone file (repeat this section if you need to add more reverse zone files).

Our final example reverse zone file looks like the following:

/etc/bind/zones/db.10.128 — updated

```
$TTL    604800
@       IN      SOA     nyc3.example.com. admin.nyc3.example.com. (
                              3         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                         604800 )       ; Negative Cache TTL
; name servers
      IN      NS      ns1.nyc3.example.com.
      IN      NS      ns2.nyc3.example.com.
```

```
; PTR Records
11.10   IN      PTR     ns1.nyc3.example.com.    ; 10.128.10.11
12.20   IN      PTR     ns2.nyc3.example.com.    ; 10.128.20.12
101.100 IN      PTR     host1.nyc3.example.com.  ; 10.128.100.101
102.200 IN      PTR     host2.nyc3.example.com.  ; 10.128.200.102
```

Copy

Check BIND Configuration Syntax

Run the following command to check the syntax of the `named.conf*` files:

```
sudo named-checkconf
```

Copy

If your named configuration files have no syntax errors, you will return to your shell prompt and see no error messages. If there are problems with your configuration files, review the error message and the Configure Primary DNS Server section, then try `named-checkconf` again.

The `named-checkzone` command can be used to check the correctness of your zone files. Its first argument specifies a zone name, and the second argument specifies the corresponding zone file, which are both defined in `named.conf.local`.

For example, to check the "nyc3.example.com" forward zone configuration, run the following command (change the names to match your forward zone and file):

```
sudo named-checkzone nyc3.example.com db.nyc3.example.com
```

Copy

And to check the "128.10.in-addr.arpa" reverse zone configuration, run the following command (change the numbers to match your reverse zone and file):

```
sudo named-checkzone 128.10.in-addr.arpa /etc/bind/zones/db.10.128
```

Copy

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

Restart BIND

Restart BIND:

```
sudo service bind9 restart
```

Copy

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 Secondary DNS Server

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. Luckily, the secondary DNS server is much easier to configure.

On *ns2*, edit the `named.conf.options` file:

```
sudo vi /etc/bind/named.conf.options
```

Copy

At the top of the file, add the ACL with the private IP addresses of all of your trusted servers:

/etc/bind/named.conf.options — updated 1 of 2 (secondary)

```
acl "trusted" {
        10.128.10.11;   # ns1
        10.128.20.12;   # ns2 - can be set to localhost
        10.128.100.101;  # host1
        10.128.200.102;  # host2
};
```

Copy

Below the `directory` directive, add the following lines:

/etc/bind/named.conf.options — updated 2 of 2 (secondary)

```
recursion yes;
allow-recursion { trusted; };
listen-on { 10.128.20.12; };      # ns2 private IP address
allow-transfer { none; };          # disable zone transfers by default

forwarders {
        8.8.8.8;
        8.8.4.4;
```

```
};
```
Copy
Save and exit `named.conf.options`. This file should look exactly like *ns1*'s `named.conf.options` file except it should be configured to listen on *ns2*'s private IP address.

Now edit the `named.conf.local` file:

```
sudo vi /etc/bind/named.conf.local
```
Copy
Define slave zones that correspond to the master zones on the primary DNS server. Note that the type is "slave", the file does not contain a path, and there is a `masters` directive which should be set to the primary DNS server's private IP. If you defined multiple reverse zones in the primary DNS server, make sure to add them all here:

/etc/bind/named.conf.local — updated (secondary)
```
zone "nyc3.example.com" {
    type slave;
    file "slaves/db.nyc3.example.com";
    masters { 10.128.10.11; };  # ns1 private IP
};

zone "128.10.in-addr.arpa" {
    type slave;
    file "slaves/db.10.128";
    masters { 10.128.10.11; };  # ns1 private IP
};
```
Copy
Now save and exit `named.conf.local`.

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

```
sudo named-checkconf
```
Copy
Once that checks out, restart bind

```
sudo service bind9 restart
```

Copy

Now you have primary and secondary DNS servers for private network name and IP address resolution. Now you must configure your servers to use your private DNS servers.

Configure DNS Clients

Before all of your servers in the "trusted" ACL can query your DNS servers, you must configure each of them to use *ns1* and *ns2* as nameservers. This process varies depending on OS, but for most Linux distributions it involves adding your name servers to the `/etc/resolv.conf` file.

Ubuntu Clients

On Ubuntu and Debian Linux VPS, you can edit the `head` file, which is prepended to `resolv.conf` on boot:

```
sudo vi /etc/resolvconf/resolv.conf.d/head
```

Copy

Add the following lines to the file (substitute your private domain, and *ns1* and *ns2* private IP addresses):

/etc/resolvconf/resolv.conf.d/head

```
search nyc3.example.com  # your private domain
nameserver 10.128.10.11  # ns1 private IP address
nameserver 10.128.20.12  # ns2 private IP address
```

Copy

Now run `resolvconf` to generate a new `resolv.conf` file:

```
sudo resolvconf -u
```

Copy

Your client is now configured to use your DNS servers.

CentOS Clients

On CentOS, RedHat, and Fedora Linux VPS, simply edit the `resolv.conf` file:

```
sudo vi /etc/resolv.conf
```

Copy

Then add the following lines to the TOP of the file (substitute your private domain, and *ns1* and *ns2* private IP addresses):

/etc/resolv.conf

```
search nyc3.example.com  # your private domain
nameserver 10.128.10.11  # ns1 private IP address
nameserver 10.128.20.12  # ns2 private IP address
```

Copy

Now save and exit. Your client is now configured to use your DNS servers.

Test Clients

Use `nslookup` to test if your clients can query your name servers. You should be able to do this on all of the clients that you have configured and are in the "trusted" ACL.

Forward Lookup

For example, we can perform a forward lookup to retrieve the IP address of *host1.nyc3.example.com* by running the following command:

```
nslookup host1
```

Copy

Querying "host1" expands to "host1.nyc3.example.com because of the `search` option is set to your private subdomain, and DNS queries will attempt to look on that subdomain before looking for the host elsewhere. The output of the command above would look like the following:

```
Output:
Server:     10.128.10.11
Address:    10.128.10.11#53

Name:   host1.nyc3.example.com
Address: 10.128.100.101
```

Reverse Lookup

To test the reverse lookup, query the DNS server with *host1*'s private IP address:

```
nslookup 10.128.100.101
```

Copy
You should see output that looks like the following:

```
Output:
Server:     10.128.10.11
Address:    10.128.10.11#53

11.10.128.10.in-addr.arpa   name = host1.nyc3.example.com.
```

If all of the names and IP addresses resolve to the correct values, that means that your zone files are configured properly. If you receive unexpected values, be sure to review the zone files on your primary DNS server (e.g. `db.nyc3.example.com` and `db.10.128`).

Congratulations! Your internal DNS servers are now set up properly! Now we will cover maintaining your zone records.

Maintaining DNS Records

Now that you have a working internal DNS, you need to maintain your DNS records so they accurately reflect your server environment.

Adding Host to DNS

Whenever you add a host to your environment (in the same datacenter), you will want to add it to DNS. Here is a list of steps that you need to take:

Primary Nameserver

- Forward zone file: Add an "A" record for the new host, increment the value of "Serial"
- Reverse zone file: Add a "PTR" record for the new host, increment the value of "Serial"
- Add your new host's private IP address to the "trusted" ACL (`named.conf.options`)

Then reload BIND:

```
sudo service bind9 reload
```

Copy

Secondary Nameserver

- Add your new host's private IP address to the "trusted" ACL (`named.conf.options`)

Then reload BIND:

```
sudo service bind9 reload
```

Copy

Configure New Host to Use Your DNS

- Configure resolv.conf to use your DNS servers
- Test using `nslookup`

Removing Host from DNS

If you remove a host from your environment or want to just take it out of DNS, just remove all the things that were added when you added the server to DNS (i.e. the reverse of the steps above).

Related questions:

1) What is the purpose of the /etc/hosts file in Linux systems, and how does it relate to DNS resolution?

In Linux systems, the /etc/hosts file plays a key role as a local tool for resolving hostnames to IP addresses. It takes priority over DNS servers when resolving names, making it a valuable resource for local network setups or testing services. By manually editing this file, administrators can quickly override name resolution, which is especially useful when DNS servers are unavailable or when a temporary solution is needed. Though simple, its function is indispensable in network management.

2) Explain the concept of Fully Qualified Domain Name (FQDN) and provide an example.

A Fully Qualified Domain Name (FQDN) is the complete name that uniquely identifies a host within the DNS hierarchy. It includes both the hostname and the domain name, ending with a dot that signifies the root domain. For instance, in www.example.com., www represents the hostname, while example.com is the domain name. This specificity ensures that systems are precisely located in global networks, especially in complex configurations.

3) Describe the different categories of Top Level Domain Names (TLDs) and provide examples for each.

Top-Level Domain Names (TLDs) fall into three main categories. Generic TLDs (gTLDs), such as .com, .org, and .net, are widely used across various industries. Country Code TLDs (ccTLDs), like .us for the United States and .uk for the United Kingdom, represent specific geographical regions. Lastly, Sponsored TLDs (sTLDs), such as .gov for government entities and .edu for educational institutions, are intended for specific groups. Each category serves a distinct purpose in the DNS ecosystem.

4) Identify and describe the three types of DNS servers mentioned in the lecture.

In DNS infrastructure, there are three main types of servers: recursive, authoritative, and caching servers. Recursive servers act as intermediaries by querying other servers to resolve domain names into IP addresses on behalf of clients. Authoritative servers store DNS records for specific domains and provide definitive answers for queries within their zones. Caching servers temporarily store query responses to enhance performance and reduce latency for repeated requests. These types of servers work together to ensure efficient and reliable DNS functionality.

5) What is the purpose of a caching zone in DNS configuration, and how is it defined in BIND?

A caching zone in DNS configuration is essential for improving efficiency by temporarily storing responses to DNS queries. This reduces the load on external servers and speeds up response times for frequently accessed domains. In BIND, caching is achieved by defining a zone type of hint or configuring forward zones. This functionality is particularly beneficial in networks with high query volumes, optimizing overall server performance.

6) Explain the functionality of the nslookup, rndc, host, and whois commands in managing and troubleshooting DNS configurations.

Several tools are essential for managing and troubleshooting DNS configurations. The nslookup command is used to query DNS servers for domain names or IP address information, making it useful for diagnosing issues. The rndc command provides control over the BIND DNS server, enabling tasks like reloading zones and making configuration changes without restarting the server. The host command is a versatile tool for DNS lookups and gathering information about domains and IPs. Lastly, the whois command retrieves domain registration details, including ownership and expiration information, making it valuable for verifying domain data.

Conclusion

The lab provided an engaging opportunity to gain practical knowledge about DNS configuration using BIND. Through hands-on tasks, we learned to set up a primary DNS server and configure forward and reverse zones to manage hostname resolution efficiently. Optional steps highlighted the process of adding redundancy with a secondary DNS server, which enhances fault tolerance. Using tools like nslookup, rndc, and host, we explored troubleshooting techniques, deepening our understanding of DNS operations. This experience underscored the importance of meticulous configurations, robust documentation, and effective testing to achieve a reliable network environment.