Kathmandu University
Department of Computer Science and Engineering
Dhulikhel, Kavre

A Project Report
on
“DNS Server”

[Code No: COMP 204]

Submitted by

Nischal Baral (Roll No. 06)
Prashant Manandhar (Roll No. 30)
Roshan Sahani (Roll No. 42)
Manish Shivabhakti (Roll No. 63)

Submitted to

Dr. Prakash Poudyal
Department of Computer Science and Engineering

Submission Date:8th May, 2023

Abstract

The Domain Name System (DNS) is a critical component of the Internet infrastructure. It is responsible for translating human-readable domain names into IP addresses that can be understood by computers. In this report, we investigate the performance and security of DNS servers in the context of modern Internet usage. We focus on the analysis of various open-source DNS server implementations on BIND. Our evaluation includes both functional and performance testing. We also examine the impact of various configuration options, and entry of zones files and settings on the overall performance of the DNS server. Our results indicate that while all of the DNS servers tested provide reliable performance, there are significant differences in terms of performance and features. This report provides valuable insights for system administrators and developers looking to deploy and optimize DNS servers in our network environments.

Keywords: Dns Server, Zone File, Bind, IP address

Table of Content

Acronyms/Abbreviations

DNS	Domain Name System
IP	Internet Protocol
TLD	Top-level domain
SLD	Second-level domains
BIND	Berkeley Internet Name Domain
NS	Name Servers

Chapter 1 Introduction

1.1 Background

An essential part of the internet is the Domain Name System (DNS), which makes it easier to convert human-readable domain names into machine-readable IP addresses. It essentially functions as a hierarchical, decentralized naming system that gives every internet-connected device a distinct identity. Users must be directed to the appropriate web server, and DNS is in charge of making sure that the requested content is sent back to the user's device.

The top-level domain (TLD) is represented by the domain extension, such as .com , .org, .net, etc., in the hierarchical structure of DNS. Subdomains that describe a specific section or feature of the website come after second-level domains (SLD), which are found beneath the TLD. The DNS system makes it simple to organize and administer domain names and the IP addresses that go with them.

A DNS resolver, often offered by the user's internet service provider, receives the request when a user types a domain name into their browser. The resolver then consults the root name server to ascertain which TLD server should receive the query. The relevant authoritative name server for the domain in question receives the query from the TLD server and provides the IP address linked to the domain name. The user's device can then connect to the required server by using the IP address the resolver returns to it.

1.2 Objectives

- To translate domain names to IP addresses
- Provides redundancy and fault tolerance
- Improves internet performance
- Facilitates email delivery

- Enables load balancing and traffic management

1.3 Motivation and Significance

Humans prefer using domain names like google.com over remembering IP addresses like 8.8.8.8 because domain names are simpler to recall. However, it is more effective and accurate to use IP addresses for computers and network devices to find and connect with other devices on the internet. The computer asks a DNS server to convert a domain name into its matching IP address whenever a domain name is input into a web browser. The DNS server subsequently gives the computer the website's IP address.

Chapter 2 Project Setup and Implementation

The DNS server was built on a Debian-based Linux machine[1] (Kali Linux in Virtualbox in our case), using the Bind9 library.

2.1 Installing Bind9 and required tools

Firstly, Bind9[2] was installed in the machine with the command:

sudo apt-get install bind9

2.2 Setting up the DNS server using bind

a. **Setting up the local configuration file**

   Since we were working on the project locally, we used the configuration file called ‘named.conf.local’ located in the ‘/etc/bind’ folder. The file after configuration looks like this:

```
└─$ cat /etc/bind/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 "ku.edu.np" {
        type master;
        file "/etc/bind/db.ku.edu.np";
        allow-transfer { 127.0.0.2; };
};

zone "secondary.ku.edu.np" IN  {
        type slave;
        file "/etc/bind/db.ku.local";
        masters { 127.0.0.1; };
};
```

   Furthermore, Bind9 uses the named.conf as the main config, but we don’t need any configuration for this as the named.conf file already has the name.conf.local setting included in the configuration file.

b. Setting up the zone file

In order for the server to work, we add the zone file for the ku.edu.np in the location '/etc/bind/db.ku.edu.np'. The zone file looks like:

```
└─$ cat /etc/bind/db.ku.edu.np
$TTL 86400
$ORIGIN ku.edu.np

@ IN SOA ns1.ku.edu.np. admin.ku.edu.np. (
        2023050401 ; Serial
        3600     ; Refresh
        1800     ; Retry
        604800   ; Expire
        86400)   ; Minimum TTL

@       IN      NS      ns1.ku.edu.np.
        IN      NS      ns2.ku.edu.np.
ns1     IN      A       127.0.0.1
ns2     IN      A       127.0.0.2
www     IN      A       101.251.6.15
@       IN      A       101.251.6.15
```

c. **Configuring load balancing algorithm**

Since Bind9 is a fully-fledged DNS server provider, the options for the load balancing were already there. The option named rrset-order (called Resource Record Set Order) does all the heavy lifting, ‘round-robin’ is the option we have provided for handling the request to the multiple servers. Round Robin handles the request in multiple servers by distributing the traffic in cyclic order.

```
└─$ cat /etc/bind/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 {
        //      0.0.0.0;
        // };

        //========================================================================
        // 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; };

        recursion yes;
        allow-query { any; };
//      rrset-order { round-robin; };

};
```

d. **Checking the configuration and restarting the server**

To check the configuration for any errors, we run the command:

sudo named-checkconf /etc/bind/named.conf

To restart the server to apply the changes, we run the command:

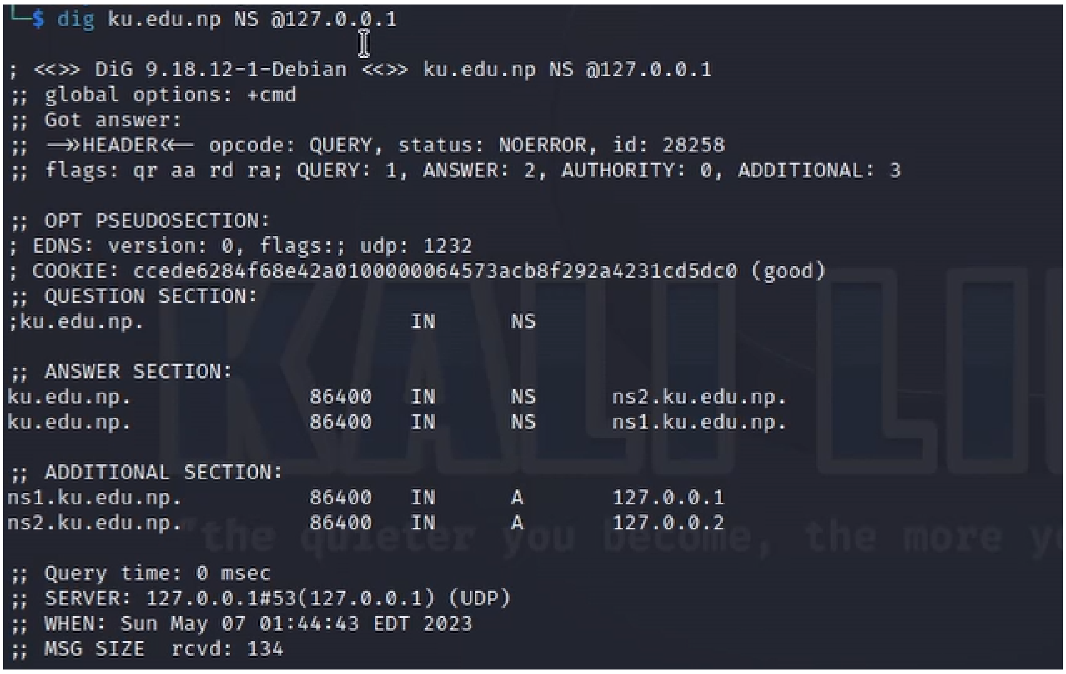
sudo systemctl restart named

where the named the alias for the bind9.

e. Final Result

After the configuration and service restart, the dns server was tested with the commands:

dig ku.edu.np NS @127.0.01 or *nslookup ku.edu.np 127.0.0.1*



```
└─$ dig ku.edu.np NS @127.0.0.1

; <<>> DiG 9.18.12-1-Debian <<>> ku.edu.np NS @127.0.0.1
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 28258
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 3

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
; COOKIE: ccede6284f68e42a0100000064573acb8f292a4231cd5dc0 (good)
;; QUESTION SECTION:
;ku.edu.np.                     IN      NS

;; ANSWER SECTION:
ku.edu.np.              86400   IN      NS      ns2.ku.edu.np.
ku.edu.np.              86400   IN      NS      ns1.ku.edu.np.

;; ADDITIONAL SECTION:
ns1.ku.edu.np.          86400   IN      A       127.0.0.1
ns2.ku.edu.np.          86400   IN      A       127.0.0.2

;; Query time: 0 msec
;; SERVER: 127.0.0.1#53(127.0.0.1) (UDP)
;; WHEN: Sun May 07 01:44:43 EDT 2023
;; MSG SIZE  rcvd: 134
```

The line *SERVER: 127.0.0.1#53* shows that the query was resolved by the nameserver running on localhost (127.0.0.1) port 53.

3.3 Software Specification

Our application is supported on Linux.

3.4 Hardware Specification

Any device which supports Linux.

3.5 Packages Used

BIND.

Chapter 3 Conclusion

Thus, we implemented a DNS server in linux with the help of Bind9utils packages. During this implementation, we learned about the structure of DNS servers, how IP addresses are stored in zone files and how to configure them, record of DNS and how DNS servers handle and respond to incoming requests.

Although we have successfully implemented a DNS server, it is limited to localhost only and works only with the primary server and does not have a secondary server. Even though it has some limitations, due to this project we were able to learn a lot about the working of DNS servers.

References

1. mailserverguru. (2021, May 23). *Install and Configure Linux DNS Server | Linux DNS Server | Part1* [Video]. YouTube. https://www.youtube.com/watch?v=FeZjRF-aVlc
2. Aitchison, R. (2011). Pro Dns and BIND 10. Apress.