UNIT 14: NETWORK PROTOCOLS

UNIT STRUCTURE

14.1 Learning Objectives

14.2 Introduction

14.3 Configuring NFS

14.4 Configuring NIS

14.5 Configuring DNS

14.6 Configuring FTP

14.7 Configuring Squid Proxy

14.8 Contiguring Firewall

14.9 Configuring DHCP server

14.10 ip Tables and Firewall

14.11 Let Us Sum Up

14.11 Further Readings

14.13 Answers to Check Your Progress

14.14 Model Questions

14.1 LEARNING OBJECTIVES

After going through this unit, you will be able to:

- understand the basic configuration of NFS service
- understand NIS service and its configuration
- learn how to set up FTP and DHCP server
- learn to setup basic ip table rules to use as firewall
- browse using Squid Proxy

14.2 INTRODUCTION

In the previous unit, we have learnt how to install and configure various Linux services like proxy server, Apache web server etc. In this unit, we will discuss how to configure Network File System(NFS), Network Information Service(NIS) and Domain Name Service(DNS). We will also discuss how to set up FTP and DHCP server. In the last part of the unit, we will discuss how to use the ip table and browse using squid proxy.

14.3 CONFIGURING NFS

Sharing files between computers and servers is an essential networking task. Network File System (NFS) is a way of mounting Linux directories over a network. It was basically developed to share files and folders between two Linux systems faster and better. Sun Microsystems developed NFS in 1980 for this sole purpose. NFS uses Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) to provide a stateless connection between client and server. NFS is one of the best methods to share between to two Linux systems. The main purpose of NFS is to back up data into a central server that is accessible to all PCs in a network. To use NFS, first we need to install it.

NFS works on server/client model where one machine works as server and another one works as client. We need to install server part and client part in the respective machine separately. For installation first we need to update our systems with command **sudo apt update**.

```
saurabh@saurabh-VirtualBox:~$ sudo apt update
[sudo] password for saurabh:
Hit:1 http://in.archive.ubuntu.com/ubuntu focal InRelease
Hit:2 http://in.archive.ubuntu.com/ubuntu focal-updates InRelease
Hit:3 http://in.archive.ubuntu.com/ubuntu focal-backports InRelease
```

Then we have to install NFS server using the command **sudo apt install nfs-kernel-server.**

```
saurabh@saurabh-VirtualBox:~$ sudo apt install nfs-kernel-server
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
```

Here nfs-kernel-server is a package name.

Once the packages are finished installing, it's time to configure the exports file. To configure the export file, we can open it with any text editor. If we are not in root mode, we have to use sudo while editing. The exports file specifies which directories the server will export to the network. Once we open /etc/exports file, it displays as follows.

```
1 # /etc/exports: the access control list for filesystems which may be exported
2 #               to NFS clients.  See exports(5).
3 #
4 # Example for NFSv2 and NFSv3:
5 # /srv/homes       hostname1(rw,sync,no_subtree_check)
  hostname2(ro,sync,no_subtree_check)
6 #
7 # Example for NFSv4:
8 # /srv/nfs4        gss/krb5i(rw,sync,fsid=0,crossmnt,no_subtree_check)
9 # /srv/nfs4/homes  gss/krb5i(rw,sync,no_subtree_check)
10 #
```

The structure of the file is simple. To the left are the directories which are to be shared, and to the right are the IP addresses and subnets of the machines which are also to be shared along with any specific options. Suppose we want to share nfs _share folder to the machine having IP address 192.168.87.158, then we have to add following line to exports file at last.

/nfs_share 192.168.87.158(rw,sync,no_root_squash,no_subtree_check)

Explanation

- /nfs_share - is the share folder which server wants to share.
- 192.168.87.158 - is the IP address of the client to whom we want to share.
- rw - This means all the clients to read and write the files to the share directory.
- sync - which will confirm the shared directory once the changes are committed.
- no_subtree_check - will prevent from the scanning the shared directory, as nfs performs the scans of every share directory, Disabling the subtree check will increase the reliability, but reduces the security.
- no_root_squash - This helps all the root user to connect to the designated directory.

Alternate parameter

- subtree_check: specifies that, in the case of a directory is exported instead of an entire filesystem, the host should verify the location of files and directories on the host filesystem.
- async: ignores synchronization checks in favour of increased speed

```
1 # /etc/exports: the access control list for filesystems which may be exported
2 #               to NFS clients.  See exports(5).
3 #
4 # Example for NFSv2 and NFSv3:
5 # /srv/homes       hostname1(rw,sync,no_subtree_check)
  hostname2(ro,sync,no_subtree_check)
6 #
7 # Example for NFSv4:
8 # /srv/nfs4        gss/krb5i(rw,sync,fsid=0,crossmnt,no_subtree_check)
9 # /srv/nfs4/homes  gss/krb5i(rw,sync,no_subtree_check)
10 # /nfs_share 192.168.87.158(rw,sync,no_root_squash,no_subtree_check)
```

After we make the necessary edits in /etc/exports, use the exportfs command to export all shared directories registered in that file:

sudo exportfs -a

Once the exports file is ready to go, the server can be started and made to start on boot with Systemd. There are two parts of starting the NFS server, rpcbind and nfs-server. In Ubuntu, the server will have to start automatically, so we may need to run:

$ sudo systemctl restart rpcbind

$ sudo systemctl restart nfs-server

After that, the server will be up and running. Once again, since Debian and Ubuntu run services at boot time by default, nothing is required to make NFS run automatically at boot.

Connecting The Client

Setting up a client to connect to an NFS share requires much less effort. Again, for both Debian/Ubuntu, we need to install nfs-common package.

$ sudo apt-get install nfs-common rpcbind

Once it is started, or in the case of Debian/Ubuntu, restart using rpcbind.

$ sudo systemctl (re)start rpcbind

Once the packages are installed on the client, we need to create the directory in client side to mount point the shared folder.

 # mkdir -p /mnt/share

Mounting the Share Directory

 # mount 192.168.87.156:/nfs_share /mnt/share/

To confirm if the share is mounted or not, run the command 'df -h',

this will show the list of mounted folders. Here nfs_share folder from server 192.168.87.156 has been mounted and we can observe that the server and client are in same network (192.168.87.0).

14.4 CONFIGURING NIS

Network Information Service (NIS) is a distributed naming service based on Remote Procedure Call (RPC). It enables easy sharing of various information across the cluster including username, password, hosts, and service ports. Such a centralized user management system is also a prerequisite for setting up cluster management system including SLURM, which requires user & group synchronization across the cluster.

NIS architecture overview

NIS uses a client-server arrangement. By running NIS, the system administrator can provide administrative databases, called maps, among a variety of servers. Servers are further divided into master and slave servers: the master server is the true single owner of the map data. Slave NIS servers handle client requests, but they do not modify the NIS maps. The master server is responsible for all map maintenance and distribution to its slave servers. Once an NIS map is built on the master to include a change, the new map file is distributed to all slave servers.

Clients are hosts that request information from these maps. NIS clients observe these changes when they perform queries on the map file - it doesn't matter whether the clients are talking to a master or a slave server, because once the map data is distributed, all NIS servers have the same information. NIS architecture is depicted in figure 1.

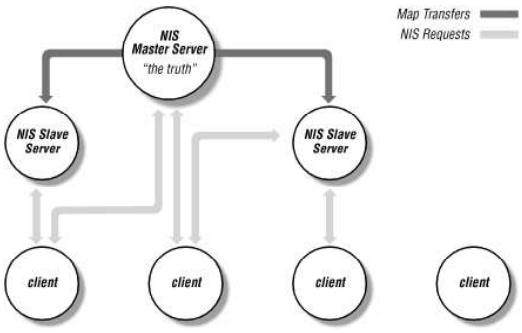


Figure 1. NIS Architecture

NIS uses domains to arrange the machines, users, and networks in its namespace. However, it does not use a domain hierarchy as the NIS namespace is flat.

Set Up NIS

NIS Master

- **Installing NIS**

First update the system then install the NIS package.

$ sudo apt install nis

System will prompt to enter preferred domain name during the installation process. Here, the domain name is test.nis.

```
Preconfiguring packages ...
```

```
# input your domain name
+-------------------------| Configuring nis |-------------------------+
| Please choose the NIS "domainname" for this system. If you want this |
| machine to just be a client, you should enter the name of the NIS domain |
| you wish to join.                                                    |
|                                                                      |
| Alternatively, if this machine is to be a NIS server, you can either |
| enter a new NIS "domainname" or the name of an existing NIS domain.  |
|                                                                      |
| NIS domain:                                                          |
|                                                                      |
| test.nis_____________________________________________________________|
|                                                                      |
|                        <Ok>                                          |
|                                                                      |
+----------------------------------------------------------------------+
```

2. **Edit /etc/default/nis**

   $ sudo vim /etc/default/nis

   Specifically, edit NISSERVER to be master

   #

   # /etc/defaults/nis Configuration settings for the NIS daemons.

   #

   # Are we a NIS server and if so what kind (values: false, slave, master)?

   NISSERVER=false # Should be master

3. **Edit /etc/hosts**

   In master's /etc/hosts, there should be at least all the slave servers.

   $ sudo vim /etc/hosts

SHELL

192.168.0.233 test # master

147.46.xx.xxx slave_test # slave

4. **Edit /etc/ypserv/securenets**

$ sudo vim /etc/ypserv.securenets # *This line gives access to everybody. PLEASE ADJUST!*
 # comment out
 # 0.0.0.0 0.0.0.0
 # add to the end: IP range you allow to access
 # Allows IP in the form of: 192.168.0. to access*
255.255.255.0 192.168.0.0
 # Allows certain internet IP to access:
255.255.255.255 147.46.xx.xxx

5. **Run ypinit -m**

By running ypinit -m, NIS will utilize the local user system as its cornerstone to build the network user information.

$ sudo /usr/lib/yp/ypinit -m

We will be prompted to enter the list of servers (master and slave). The first host added should be the hostname of the master server, followed by all the slave servers in the system. At this point, we have to construct a list of the hosts which will run NIS servers. test is in the list of NIS server hosts. It is to continue to add the names for the other hosts, one per line. When we have done with the list, type <control D>.

next host to add: test # master

next host to add: slave_test # slave

next host to add: # <- Ctrl + D

The current list of NIS servers looks like this:

test

slave_test

Is this correct? [y/n: y] y

We need a few minutes to build the databases...

Building /var/yp/test.nis/ypservers...

Running /var/yp/Makefile...

make[1]: Entering directory '/var/yp/test.nis'

Updating passwd.byname...

Updating passwd.byuid...

Updating group.byname...

Updating group.bygid...

Updating hosts.byname...

Updating hosts.byaddr...

Updating rpc.byname...

Updating rpc.bynumber...

Updating services.byname...

Updating services.byservicename...

Updating netid.byname...

Updating protocols.bynumber...

Updating protocols.byname...

Updating netgroup...

Updating netgroup.byhost...

Updating netgroup.byuser...

Updating shadow.byname...

make[1]: Leaving directory '/var/yp/test.nis'

test has been set up as a NIS master server.

Now we can run ypinit -s test on all slave server.

NIS client

1. **Install NIS**

   $ sudo apt install nis

   During the process it will also prompt to enter domain name. Use the same domain as the NIS master.

2. **Edit /etc/yp.conf**

   $ sudo vim /etc/yp.conf

   To Add the master / slave server, we want to request info from.

   ### FORMAT:

   # domain <domainname> server <IP or hostname>

   ### OR:

   # ypserver <IP or hostname>

   domain test.nis server 192.168.1.233

   If we want to use hostname instead of server IP, we make sure to specify it in /etc/hosts of the client.

3. Edit /etc/nsswitch.conf

The name service switch (named nsswitch.conf) controls how a client machine or application obtains network information. Each machine has a switch file in its /etc directory. Each line of that file identifies a particular type of network information, such as host, password, and group, followed by one or more locations of that information. A client can obtain naming information from one or more of the switch's sources. For example, an NIS client could obtain its hosts information from an NIS map and its password information from a local /etc file. In addition, the client could specify the conditions under which the switch must use each source.

The available information sources are listed in the following table:

Information Sources	Description
files	A file stored in the client's /etc directory. For example, /etc/passwd
nisplus	An NIS+ table. For example, the hosts table.
nis	An NIS map. For example, the hosts map.
compat	compat can be used for password and group information to support old-style + or - syntax in /etc/passwd, /etc/shadow, and /etc/group files.
dns	Can be used to specify that host information be obtained from DNS.
ldap	Can be used to specify entries be obtained from the LDAP directory.

Open /etc/nsswitch.conf to add nis as an information source.

$ sudo gedit /etc/nsswitch.conf

Add nis to the end of passwd, group, shadow, gshadow and hosts in the following way.

```
# ...
passwd:     files systemd nis  # add nis
group:      files systemd nis  # add nis
shadow:     files nis          # add nis
```

gshadow: files nis # add nis

hosts: files dns nis # add nis

...

If we want to look up nis first instead of local files, we have to put nis [NOTFOUND=return] in front of files. The [NOTFOUND=return] search criterion instructs the switch to stop searching the NIS tables if the switch gets a "No such entry" message. The switch searches through local files only if the NIS server is unavailable.

...

passwd: nis [NOTFOUND=return] files systemd # add nis

...

4. **Edit /etc/pam.d/common-session for creating home directory automatically.**

   $ gedit /etc/pam.d/common-session

   # add to the end

   session optional pam_mkhomedir.so =/etc/skel umask=000

5. **Restart NIS to apply changes**

   $ sudo systemctl restart rpcbind

   $ sudo systemctl restart nis

NIS Slave

The set-up process of slave server is an approximate combination of client set up and master set up.

1. **Go through NIS client set-up steps**

   Specifically through Step 1 to 4. This is for establishing RPC connection with master.

2. **Edit /etc/default/nis**

   $ sudo gedit /etc/default/nis

   Specifically, edit NISSERVER to be slave.

   #

   # /etc/defaults/nis Configuration settings for the NIS daemons.

   #

   # Are we a NIS server and if so what kind (values: false, slave, master)?

   NISSERVER=false # Should be slave

3. **Edit /etc/ypserv.securenets**

   $ gedit /etc/ypserv.securenets

   It is important to allow the master server to access.

   # This line gives access to everybody. PLEASE ADJUST!

   # comment out

   # 0.0.0.0 0.0.0.0

   # add to the end: IP range you allow to access

   # Allows IP in the form of: 192.168.100.* to access

   255.255.255.0 192.168.100.0

   # Allows certain internet IP to access:

   255.255.255.255 147.46.xx.xxx

4. **Edit /etc/hosts**

   $ sudo gedit /etc/hosts

   In slave's /etc/hosts, there should be at least the IP of master server.

   147.47.xxx.xx test # master

5. **Run ypinit -s <master>**

   $ sudo /usr/lib/yp/ypinit -s tes

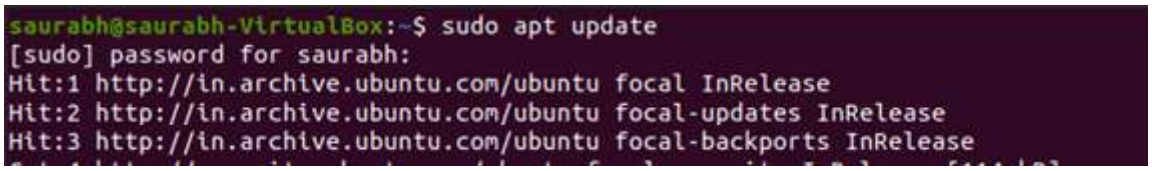
   This operation will pull information from the master to the slave.

14.5 CONFIGURING DNS

Domain Name Service (DNS) is an Internet service that maps IP addresses and fully qualified domain names (FQDN) to one another. In this way, DNS alleviates the need to remember IP addresses. Computers that run DNS are called name servers. Ubuntu ships with BIND (Berkley Internet Naming Daemon), the most common program used for maintaining a name server on Linux.

To use domain name server, first we need to install it. To install DNS, we must update our system as:

$sudo apt update



Now we will install the DNS server by using the command bind9:

$sudo apt install bind9

```
saurabh@saurabh-VirtualBox:~$ sudo apt install bind9
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  bind9-dnsutils bind9-libs bind9-utils python3-ply
Suggested packages:
```

Next step is to install the utilities of the DNS by using the command of "dnsutils":

$sudo apt install dnsutils

```
saurabh@saurabh-VirtualBox:~$ sudo apt install dnsutils
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  dnsutils
```

There are many ways to configure BIND9. Some of the most common configurations are a caching nameserver, primary server, and secondary server.

- When configured as a caching nameserver, BIND9 will find the answer to name queries and remember the answer when the domain is queried again.
- As a primary server, BIND9 reads the data for a zone from a file on its host and is authoritative for that zone.
- As a secondary server, BIND9 gets the zone data from another nameserver that is authoritative for the zone.

The DNS configuration files are stored in the /etc/bind directory. The primary configuration file is /etc/bind/named.conf, which in the layout provided by the package just includes these files.

- /etc/bind/named.conf.options: global DNS options
- /etc/bind/named.conf.local: for your zones
- /etc/bind/named.conf.default-zones: default zones such as localhost, its reverse, and the root hints

The root nameservers is used to be described in the file /etc/bind/db.root. This is now provided instead by the /usr/share/dns/root.hints file shipped with the dns-root-data package and is referenced in the

named.conf.default-zones configuration file above.

Now to configure the DNS, we will first go to the address /etc/bind/ named.conf.options and add the Google DNS for just understanding. We can edit it by gedit

sudo gedit /etc/bind/named.conf.options

Replace the forwarders text in config file as, 8.8.8.8 i.e. is Google's DNS:

```
12
13          // forwarders {
14          //      0.0.0.0;
15          // };
```

Restart the DNS service using systemctl

$ sudo systemctl restart bind9

Now check the status of bind using the command

$ sudo systemctl status bind9

```
named.service - BIND Domain Name Server
     Loaded: loaded (/lib/systemd/system/named.service; enabled; vendor preset>
     Active: active (running) since Sat 2022-07-23 22:57:31 IST; 34min ago
       Docs: man:named(8)
   Main PID: 12421 (named)
      Tasks: 5 (limit: 1105)
     Memory: 12.7M
     CGroup: /system.slice/named.service
             └─12421 /usr/sbin/named -f -u bind

Jul 23 22:58:36 saurabh-VirtualBox named[12421]: network unreachable resolving>
Jul 23 22:58:36 saurabh-VirtualBox named[12421]: network unreachable resolving>
Jul 23 22:58:41 saurabh-VirtualBox named[12421]: network unreachable resolving>
Jul 23 22:58:41 saurabh-VirtualBox named[12421]: network unreachable resolving>
Jul 23 22:58:43 saurabh-VirtualBox named[12421]: network unreachable resolving>
Jul 23 22:58:43 saurabh-VirtualBox named[12421]: network unreachable resolving>
Jul 23 22:58:43 saurabh-VirtualBox named[12421]: network unreachable resolving>
Jul 23 22:58:43 saurabh-VirtualBox named[12421]: network unreachable resolving>
Jul 23 22:58:44 saurabh-VirtualBox named[12421]: resolver priming query comple>
Jul 23 22:58:44 saurabh-VirtualBox named[12421]: managed-keys-zone: Unable to >
```

As the bind9 is running now, we will test the domain which we edit in the configuration file as:

$ dig google.com

```
saurabh@saurabh-VirtualBox:~$ dig google.com

; <<>> DiG 9.16.1-Ubuntu <<>> google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: SERVFAIL, id: 55443
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;google.com.                    IN      A

;; Query time: 0 msec
;; SERVER: 127.0.0.53#53(127.0.0.53)
;; WHEN: Sat Jul 23 23:41:58 IST 2022
;; MSG SIZE  rcvd: 39
```

Now we will do primary zone configuration, but before configuration let's have an idea what is going on. At bind directory, we have two zone files: one is global DNS conf file named.conf & second is local DNS config file named.conf.local. We need to create forward and reverse zones. We can store some files at some host and in this configuration, the DNS gets its data from that specific file for its zone. Now for such primary master configuration, we will run the editor and will confirm that the following three commands are there and not commented on.

include "/etc/bind/named/.conf.option";

include "/etc/bind/named/.conf.local";

include "/etc/bind/named/.conf.default-zones";

To verify this, we will open named.conf file with the editor:

$ sudo gedit /etc/bind/named.conf

```
1 // This is the primary configuration file for the BIND DNS server named.
2 //
3 // Please read /usr/share/doc/bind9/README.Debian.gz for information on the
4 // structure of BIND configuration files in Debian, *BEFORE* you customize
5 // this configuration file.
6 //
7 // If you are just adding zones, please do that in /etc/bind/named.conf.local
8
9 include "/etc/bind/named.conf.options";
10 include "/etc/bind/named.conf.local";
11 include "/etc/bind/named.conf.default-zones";
```

All the three lines are present in the output. Now for example, we have a domain "example.com" for which we are going to configure.

Primary Server

In this section, BIND9 will be configured as the Primary server for the domain example.com.

Forward Zone File

To add a DNS zone to BIND9, turning BIND9 into a Primary server, first we need to edit /etc/bind/named.conf.local: so we will edit the file named.conf.local by gedit and add following lines :

```
zone "gamer.com" {
type master;
file "/etc/bind/db.example.com";
};
```

$ sudo gedit /etc/bind/named.conf.local

```
1 //
2 // Do any local configuration here
3 //
4
5 // Consider adding the 1918 zones here, if they are not used in your
6 // organization
7 //include "/etc/bind/zones.rfc1918";
8 zone "example.com" {
9 type master;
10 file "/etc/bind/db.example.com";
11 };
```

We will copy the contents from the db.local to the db.example.com:

```
saurabh@saurabh-VirtualBox:~$ sudo cp /etc/bind/db.local /etc/bind/db.example.c
om
[sudo] password for saurabh:
saurabh@saurabh-VirtualBox:~$
```

Now we are opening the newly created file example.com

```
saurabh@saurabh-VirtualBox:~$ sudo gedit /etc/bind/db.example.com
```

```
1 ;
2 ; BIND data file for local loopback interface
3 ;
4 $TTL    604800
5 @       IN      SOA     localhost. root.localhost. (
6                               2                 ; Serial
7                          604800                 ; Refresh
8                           86400                 ; Retry
9                         2419200                 ; Expire
10                         604800 )       ; Negative Cache TTL
11 ;
12 @       IN      NS      localhost.
13 @       IN      A       127.0.0.1
14 @       IN      AAAA    ::1
```

Edit the new zone file /etc/bind/db.example.com and change localhost. to the name of your server, leaving the additional . at the end. Change 127.0.0.1 to the nameserver's IP Address and root.localhost to a valid email address, but with a . instead of the usual @ symbol, again leaving the . at the end. Change the comment to indicate the domain that this file is for.

```
;
; BIND data file for example.com
;
$TTL    604800
@       IN      SOA     example.com. root.example.com. (
                              2                 ; Serial
                         604800                 ; Refresh
                          86400                 ; Retry
                        2419200                 ; Expire
                         604800 )       ; Negative Cache TTL
;
@       IN      NS      ns.example.com.
@       IN      A       10.0.2.15
@       IN      AAAA    ::1
ns      IN      A       10.0.2.15
```

Create an A record for the base domain, example.com. Also, create an A record for ns.example.com, the name server in this example:

Here 10.0.2.15 is my nameserver's IP address. If DNS server and nameserver are same then no need to change anything, localhost bind data file is enough. Now we need to restart the bind services to make effect of all changes

$ sudo systemctl restart bind9

Reverse Zone File

Now that the zone is setup and resolving names to IP Addresses, a Reverse zone needs to be added to allow DNS to resolve an address to a name.

Edit /etc/bind/named.conf.local and add the following:

```
zone "10.0.2.in-addr.arpa" {
    type master;
    file "/etc/bind/db.10";
};
```

```
//
// 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 "example.com" {
type master;
file "/etc/bind/db.example.com";
};
zone "10.0.2.in-addr.arpa" {
type master;
file "/etc/bind/db.10";
};
```

10.0.2 this one will be the first three octet of IP address in the used network. Now, create the /etc/bind/db.10 file:

sudo cp /etc/bind/db.127 /etc/bind/db.10

Next edit /etc/bind/db.10 for changing the same options as /etc/bind/db.example.com:

```
;
; BIND reverse data file for local 10.0.2.xxx net
;
$TTL    604800
@       IN      SOA     ns.example.com. root.example.com. (
                              1         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                         604800 )       ; Negative Cache TTL
;
@       IN      NS      ns.
1.0.0   IN      PTR     ns.example.com.
```

In this last part, we will confirm the configurations first by executing all these commands and check whether they generate errors or not:

$ named-checkzone gamer.com /etc/bind/db.gamer.com

$ named-checkzone your_IP /32 /etc/bind/db.10

$ named-checkconf /etc/bind/named.conf.local

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

The results will be somewhat like this if all is correct.

```
saurabh@saurabh-VirtualBox:~$ named-checkzone example.com /etc/bind/db.example.
com
zone example.com/IN: loaded serial 2
OK
saurabh@saurabh-VirtualBox:~$
```

14.6 CONFIGURING FTP

FTP stands for "File Transfer Protocol" and is a great protocol for downloading files from a remote or local server, or uploading files onto the server. It works by having a server that is listening for connections (on port 21 by default) from clients. The clients can access a remote directory with their user account, and then download or upload files there, depending on the permissions that have been granted to them. It is also possible to configure anonymous authorization, which means that users will not need their own account in order to connect to the FTP server.

On Ubuntu Linux, there are a multitude of different FTP server and client software packages available. We can even use default GUI and command line tools as an FTP client. A very popular and highly configurable FTP server package is vsftpd, available for many Linux systems, including Ubuntu. We can use this sever.

Install and configure vsftpd on Ubuntu

First, we need to update our system as earlier than we can install using the command:

$ sudo apt install vsftpd

where vsftpd is the package name.

```
saurabh@saurabh-VirtualBox:~$ sudo apt install vsftpd
[sudo] password for saurabh:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  vsftpd
```

Once the installation is completed, the service will be disabled initially. Therefore, we need to start it manually and also enable it to start automatically from the next system boot.

---------- On SysVInit ----------

 # service vsftpd start

 # chkconfig --level 35 vsftpd on

Next, if we have UFW firewall enabled on the server, we have to open ports 21 and 20 where the FTP daemons are listening, in order to allow access to FTP services from remote machines, then add the new

firewall rules as follows:

$ sudo ufw allow 20/tcp

$ sudo ufw allow 21/tcp

$ sudo ufw status

 # chkconfig --level 35 vsftpd on

Configure vsftpd server

The configuration file for vsftpd server exists in /etc directory. It is always a best practice to keep a copy of original configuration file.

$ sudo mv /etc/vsftpd.conf /etc/vsftpd.conf_origNe

Next open the config file using any editor. Here we have opened using gedit.

$ sudo gedit /etc/vsftpd.conf

Then we can add/modify the following options with these values in config file.

anonymous_enable=NO	# disable anonymous login
local_enable=YES	# permit local logins
write_enable=YES	# enable FTP commands which change the filesystem
local_umask=022	# value of unmask for file creation for local users
dirmessage_enable=YES	# enable showing of messages when users first enter a new directory
xferlog_enable=YES	# a log file will be maintained detailing uploads and downloads
connect_from_port_20=YES	# use port 20 on the server machine for PORT connect
xferlog_std_format=YES	# keep standard log file format
listen=NO	# prevent vsftpd from running in standalone mode
listen_ipv6=YES	# vsftpd will listen on an IPv6 socket instead of an IPv4 one
pam_service_name=vsftpd	# name of the PAM service vsftpd

```
                                    will use
userlist_enable=YES                 # enable vsftpd to load a list of
                                    usernames
tcp_wrappers=YES                    # turn on tcp wrappers
```

Now, we have to configure VSFTPD to allow/deny FTP access to users based on the user list file /etc/vsftpd.userlist. Note that by default, users listed in userlist_file=/etc/vsftpd.userlist are denied login access with userlist_deny=YES option if userlist_enable=YES. But, the option userlist_deny=NO twists the meaning of the default setting, so only users whose username is explicitly listed in userlist_file=/etc/vsftpd.userlist will be allowed to login to the FTP server.

```
userlist_enable=YES             # vsftpd will load a list of usernames,
                                from userlist_file
userlist_file=/etc/vsftpd.userlist  # stores usernames.
userlist_deny=NO
```

With the configuration file saved and the firewall rules updated, restart vsftpd to apply the new changes.

$ sudo systemctl restart vsftpd

Create an FTP user

Our FTP server is ready to receive incoming connections. So, now it's time to create a new user account that we'll use to connect to the FTP service. Use this first command to create a new account called ftpuser1, and the second command to set a password for the account:

$ sudo useradd -m ftpuser

$ sudo passwd ftpuser

New password:

Retype new password:

passwd: password updated successfully

In order to verify that everything is working properly, we should store one file in ftpuser home directory. This file should be visible when we login to FTP in the next steps.

$ sudo bash -c "echo FTP TESTING > /home/ftpuser/FTP-TEST"

Connect to FTP server via command line

We should now be able to connect to your FTP server either by IP address or hostname. First, ensure that the default FTP client utility is installed on the system by running the following command. It will either install the software or tell you that it is already available on the system.

$ sudo apt install ftp

To connect from command line and verify that everything is working properly, open a terminal and use Ubuntu's ftp command to connect to your loopback address (127.0.0.1).

$ ftp 127.0.0.1

Now, we can login to the FTP server by specifying the username and password that we configured earlier.

14.7 CONFIGURING SQID PROXY

Squid is a full-featured caching proxy supporting popular network protocols like HTTP, HTTPS, FTP, and more. It can be used to improve the web server's performance by caching repeated requests, filter web traffic, and access geo-restricted content.

Installing Squid on Ubuntu

The squid package is included in the standard Ubuntu 20.04 repositories. To install it, run the following commands as sudo user

sudo apt update

sudo apt install squid

Once the installation is completed, the Squid service will start automatically. To verify it, check the service status using command.

sudo systemctl status squid

```
● squid.service - Squid Web Proxy Server
     Loaded: loaded (/lib/systemd/system/squid.service; enabled; vendor preset: enabled)
     Active: active (running) since Fri 2020-10-23 19:02:43 UTC; 14s ago
       Docs: man:squid(8)
```

The output will look similar to this.

Configuring Squid

The squid service can be configured by editing the /etc/squid/squid.conf file. The configuration file contains comments that describe what each configuration option does. Before making any changes, it is recommended to back up the original configuration file

sudo cp /etc/squid/squid.conf /etc/squid/squid.conf.original

To start configuring the squid instance, open the file in any text editor, here we use gedit

sudo gedit /etc/squid/squid.conf

By default, squid is set to listen on port 3128 on all network interfaces on the server. If we want to change the port and set a listening interface, locate the line starting with http_port and specify the interface IP address and the new port. If no interface is specified, Squid will listen on all interfaces. Running Squid on all interfaces and on the default port should be fine for most users.

```
# Squid normally listens to port 3128
http_port IP_ADDR:PORT
```

Squid allows us to control how the clients can access the web resources using Access Control Lists (ACLs). By default, access is permitted only from the localhost. If all clients who use the proxy have a static IP address. The easiest option to restrict access to the proxy server is to create an ACL that will include the allowed IPs. Otherwise, we can set squid to use authentication. Instead of adding the IP addresses in the main configuration file, create a new text that will store the allowed IPs:

```
etc/squid/allowed_ips.txt
192.168.33.1
192.168.33.2
# All other allowed IPs
```

Now open the main configuration file and create a new ACL named allowed_ips and allow access to that ACL using the http_access directive. The order of the http_access rules is important. Make sure we have added

the line before http_access deny all.

```
/etc/squid/squid.conf
# ...
acl allowed_ips src "/etc/squid/allowed_ips.txt"
# ...
#http_access allow localnet
http_access allow localhost
http_access allow allowed_ips
# And finally deny all other access to this proxy
http_access deny all
```

After changes to the configuration file, we need to restart the Squid service for the changes to take effect:

sudo systemctl restart squid

Squid Authentication

If restricting access based on IP doesn't work for our use case, we can configure squid to use a back-end to authenticate users. Squid supports Samba, LDAP, and HTTP basic auth. Here, we will use basic auth. It is a simple authentication method built into the HTTP protocol.

To generate a crypted password, use the openssl tool. The following command appends the

USERNAME:PASSWORD pair to the /etc/squid/htpasswd file.

```
printf "USERNAME:$(openssl passwd -crypt PASSWORD)\n"
| sudo tee -a /etc/squid/htpasswd
```

```
printf "user1:$(openssl passwd -crypt 'Password1')\n" | sudo
tee -a /etc/squid/htpasswd
```

For example, to create a user "user1" with password "Password1", we would run:

Output of above command will look like this:

user1:QMxVjdyPchJl6

The next step is to enable the HTTP basic authentication and include the file containing the user credentials to the squid configuration file. Open the main configuration using any editor

sudo gedit /etc/squid/squid.conf

And add the following:

```
/etc/squid/squid.conf
# ...
auth_param basic program /usr/lib/squid3/basic_ncsa_auth
/etc/squid/htpasswd
auth_param basic realm proxy
acl authenticated proxy_auth REQUIRED
# ...
#http_access allow localnet
http_access allow localhost
http_access allow authenticated
# And finally deny all other access to this proxy
http_access deny all
```

The first three highlighted lines are creating a new ACL named authenticated, and the last highlighted line is allowing access to authenticated users.

Restart the Squid service:

sudo systemctl restart squid

14.8 CONFIGURING FIREWALL

To open the Squid ports, enable the UFW 'Squid' profile:

sudo ufw allow 'Squid'

If Squid is running on another, non-default port, for example, 8888, we can allow traffic on that port with: sudo ufw allow 8888/tcp.

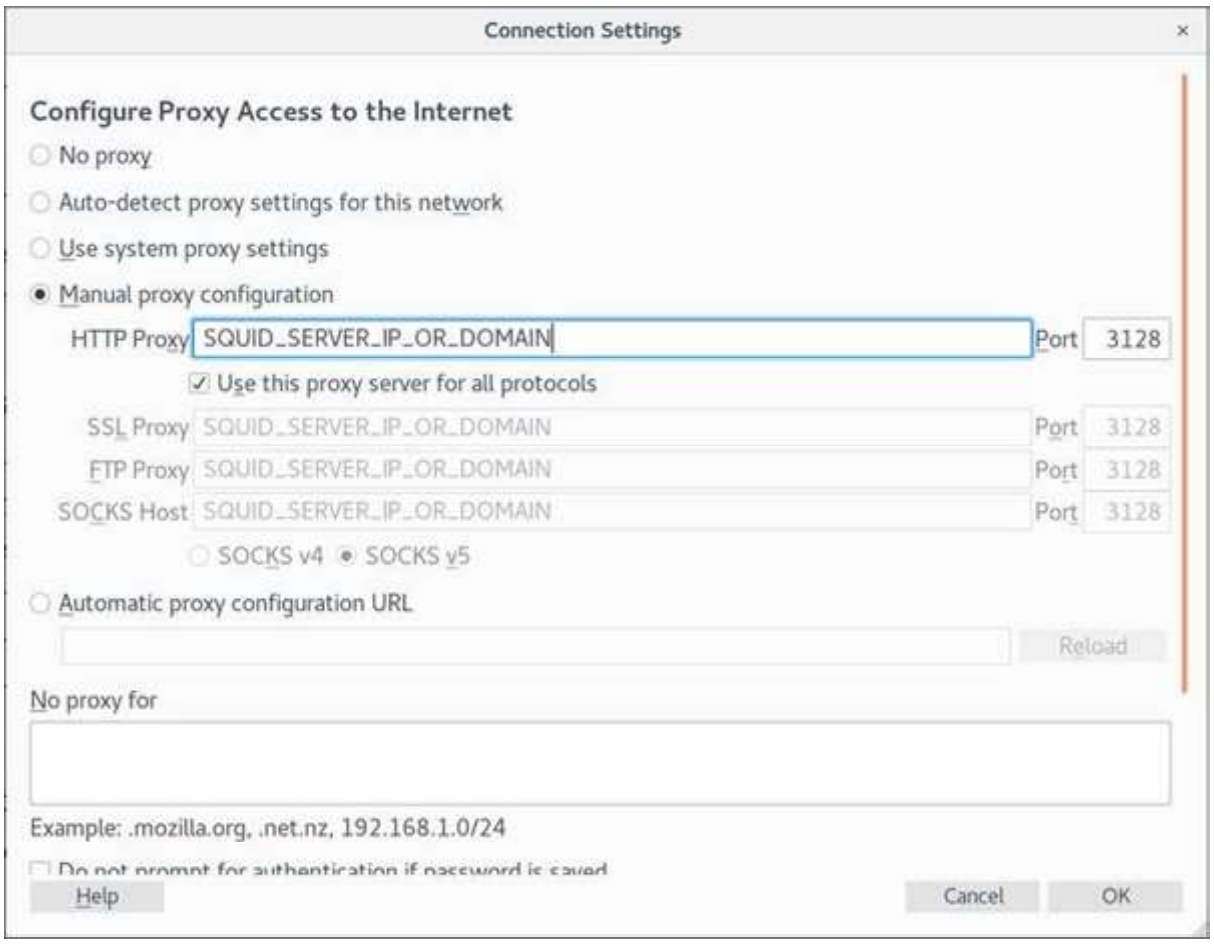
Configuring Your Browser to Use Proxy

Now that we have Squid set up, the last step is to configure a browser to use it. As ubunttu uses Firefox as default browser we will try to configure that.

Firefox

- In the upper right-hand corner, click on the hamburger icon ≡ to open Firefox's menu:

- Click on the ? Preferences link.
- Scroll down to the Network Settings section and click on the Settings... button.
- A new window will open.
- Select the Manual proxy configuration radio button.
- Enter the Squid server IP address in the HTTP Host field and 3128 in the Port field.
- Select the Use this proxy server for all protocols checkboxes.
- Click on the OK button to save the settings.



At this point, our Firefox has been configured, and we can browse the Internet through the Squid proxy. To verify it, open google.com, type "what is my ip" and you should see your Squid server IP address. To revert back to the default settings, go to Network Settings, select the Use system proxy settings radio button and save the settings.

14.9 CONFIGURING DHCP SERVER

When the computer boots up, it doesn't have an IP Address. It sends a broadcast message called a DHCP Discover. DHCP Servers are designed to respond to such broadcasts. They then send unicast traffic known as the DHCP Offer back to the requesting client. This DHCP Offer typically contains the Assigned IP Address, the Default Gateway's IP Address, and the DNS Server's IP Address. These are dynamic IP allocation.

The client on receiving the Offer sends a DHCP Request to the DHCP Server acknowledging that it has accepted the information given to it by the server. DHCP Servers keep a record of the assigned IP Addresses to prevent duplicate IP Address Collisions. DHCP Servers present on the local network and there shouldn't be more than 1 DHCP Server on a local network.

Allocation Methods for DHCP

Following are the two allocation methods for a DHCP Server:

Manual: In this method, the IP Address is given based on the MAC Address of the client system. It ensures that a particular machine always gets a fixed IP Address as its IP Address has been tied to its MAC Address. The DHCP Server sends a constant configuration to the client depending on its MAC Address in this type of allocation.

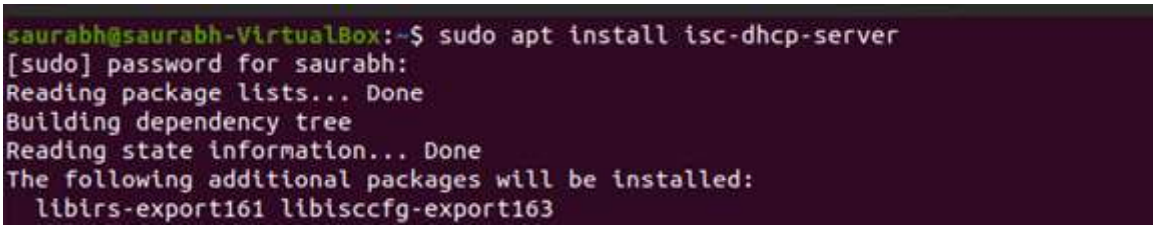
Automatic: In this method, the IP Addresses are assigned automatically by the DHCP Server on FCFS basis from a pool of addresses. It can be further divided into two categories based on the Lease Time - The time for which an IP Address is assigned to a client.

- **Fixed Lease Time:** When a DHCP client is no longer on the network for a specified period, the configuration is expired and released back to the address pool for use by other DHCP Clients. The client has to renegotiate to keep the previous IP Address.
- **Infinite Lease Time:** This has the effect of permanently assigning an IP Address to a client.

Installing DHCP Server

We can install the DHCP Server using the apt command as follows:

sudo apt install isc-dhcp-server



The main configuration file of ISC DHCP server is /etc/dhcp/ dhcpd.conf. It is always a good practice to backup original configuration files. In case if something goes wrong, the original configuration can easily be restored. We can use the cp command or mv command to create a backup.

$sudo mv /etc/dhcp/dhcpd.conf /etc/dhcp/dhcpd.conf.backup

Now we can use the editor of our choice to edit dhcpd.conf file.

$sudo gedit /etc/dhcp/dhcpd.conf

Assigning Random IP Addresses from a pool

The most common thing done by a DHCP server is to assign addresses randomly. The settings for the same would be as follows:

```
# a simple /etc/dhcp/dhcpd.conf
default-lease-time 600;
max-lease-time 7200;
authoritative;
subnet 192.168.1.0 netmask 255.255.255.0 {
 range 192.168.1.100 192.168.1.200;
 option routers 192.168.1.254;
 option domain-name-servers 192.168.1.1, 192.168.1.2;
#option domain-name "mydomain.example";
}
```

According to this configuration:

- The default lease time for a client is 10 mins (600 seconds) and the maximum lease time is 2 hrs (7200 seconds).
- This DHCP Server is the official server for the local network. (indicated by authoritative).
- The Server will hand over the IP Address from the range 192.168.1.100

to 192.168.1.200.

- The server will also "advise" the client to use 192.168.1.254 as the default-gateway and 192.168.1.1 and 192.168.1.2 as its DNS servers.

Assigning Static IP Address to a client

The DHCP Server will need the MAC Address of that client to guarantee that a particular client will always get the same IP Address, this would be the Manual Allocation mode. To get the MAC Address of a client, we can use the ip command on the client machine:

$ ip a

```
saurabh@saurabh-VirtualBox:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defau
lt qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP g
roup default qlen 1000
    link/ether 08:00:27:a8:d5:fc brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic noprefixroute enp0s3
       valid_lft 84310sec preferred_lft 84310sec
    inet6 fe80::ff62:6fe1:63bf:606e/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

Underline address is the MAC address of the ethernet interface. We need to select the MAC Address of the interface from which we are planning to connect to the network. Here we have only one interface and we will choose that one. Now, we have to put MAC Address in the configuration file:

```
host archmachine {
hardware ethernet 08:00:27:a8:d5:fc;
fixed-address 10.0.2.15;
}
```

This will reserve the IP Address 10.0.2.15 for the client with the MAC Address. 08:00:27:a8:d5:fc

Bind the DHCP Server to an interface

The DHCP Server can have many interfaces. We need to define the interface it should listen to. We can view the interfaces on the server using the command ip a just like the client. The interface to bind with is

defined in the /etc/default/isc-dhcp-server file. Open it using an editor. The interface for our server is eth0. It may be different for another server. Edit the file as:

INTERFACESv4="eth0"

Now that changes to the configuration are made, we need to restart the service to enable those changes. To do that, we will use the systemctl command:

sudo systemctl restart isc-dhcp-server.service

Check the status of DHCP server using the systemctl command. An active status indicates that the DHCP Server has successfully picked up the configuration and is ready to hand out IP Addresses.

sudo systemctl status isc-dhcp-server.service

14.10 IP TABLES AND FIREWALL

Simply put, iptables is a firewall program for Linux. It will monitor traffic from and to our server using tables. These tables contain sets of rules, called chains that will filter incoming and outgoing data packets.

Iptables filters packets based on:

- **Tables:** Tables are files that join similar actions. A table consists of several chains.
- **Chains:** A chain is a string of rules. When a packet is received, iptables finds the appropriate table, then runs it through the chain of rules until it finds a match.
- **Rules:** A rule is a statement that tells the system what to do with a packet. Rules can block one type of packet, or forward another type of packet. The outcome, where a packet is sent, is called a target.
- **Targets:** A target is a decision of what to do with a packet. Typically, this is to accept it, drop it, or reject it (which sends an error back to the sender).

Tables and Chains

Linux firewall iptables has four default tables. We will list all four along with the chains each table contains.

- **Filter**

The Filter table is the most frequently used one. It acts as a bouncer, deciding who gets in and out of your network. It has the following default chains:

- Input - the rules in this chain control the packets received by the server.
- Output - this chain controls the packets for outbound traffic.
- Forward - this set of rules controls the packets that are routed through the server.

- **Network Address Translation (NAT)**

This table contains NAT (Network Address Translation) rules for routing packets to networks that cannot be accessed directly. When the destination or source of the packet has to be altered, the NAT table is used. It includes the following chains:

- Prerouting - this chain assigns packets as soon as the server receives them.
- Output - works the same as the output chain we described in the filter table.
- Postrouting - the rules in this chain allow making changes to packets after they leave the output chain.

- **Mangle**

The Mangle table adjusts the IP header properties of packets. The table has all the following chains we described above: Prerouting , Postrouting ,Output ,Input ,Forward.

- **Raw**

The Raw table is used to exempt packets from connection tracking. The raw table has two of the chains we previously mentioned: Prerouting and Output.

- **Security (Optional)**

Some versions of Linux also use a Security table to manage special access rules. This table includes input, output, and forward chains, much like the filter table.

Targets

A target is what happens after a packet matches a rule criterion.

Non-terminating targets keep matching the packets against rules in a chain even when the packet matches a rule. With terminating targets, a packet is evaluated immediately and is not matched against another chain. The terminating targets in Linux iptables are:

- Accept - this rule accepts the packets to come through the iptables firewall.
- Drop - the dropped package is not matched against any further chain. When Linux iptables drop an incoming connection to the server, the person trying to connect does not receive an error. It appears as if they are trying to connect to a non-existing machine.
- Return - this rule sends the packet back to the originating chain so you can match it against other rules.
- Reject - the iptables firewall rejects a packet and sends an error to the connecting device.

Installing Iptables

Iptables comes pre-installed in most Linux distributions. However, if we don't have it in Ubuntu/Debian system by default, use the following commands :

sudo apt-get update

sudo apt-get install iptables

Check the status of your current iptables configuration by running:

sudo iptables -L -v

```
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target prot opt in out source destination
Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target prot opt in out source destination
Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target prot opt in out source destination
```

Here, the -L option is used to list all the rules, and -v is for showing the info in a more detailed format. Below is the example output:

Now we have the Linux firewall installed. At this point, we can notice that all chains are set to ACCEPT and have no rules. This is not secure since any packet can come through without filtering.

Defining Chain Rules

Defining a rule means appending it to the chain. To do this, we need to insert the -A

option (Append) right after the iptables command, like:

sudo iptables -A

It will alert iptables that we are adding new rules to a chain. Then, we can combine the command with other options, such as:

- **-i (interface) -** the network interface whose traffic we want to filter, such as eth0, lo, ppp0, etc.
- **-p (protocol) -** the network protocol where our filtering process takes place. It can be either tcp, udp, udplite, icmp, sctp, icmpv6, and so on. Alternatively, we can type all to choose every protocol.
- **-s (source) -** the address from which traffic comes from. We can add a hostname or IP address.
- **-dport (destination port) -** the destination port number of a protocol, such as 22 (SSH), 443 (https), etc.
- **-j (target) -** the target name (ACCEPT, DROP, RETURN). We need to insert this every time we make a new rule.

To use all of them, we must write the command in this order:

sudo iptables -A <chain> -i <interface> -p <protocol (tcp/udp) > -s <source> --dport <port no.> -j <target>

.

Enabling Traffic on Localhost

To allow traffic on localhost, type this command:

sudo iptables -A INPUT -i lo -j ACCEPT

For this iptables, we use lo or loopback interface. It is utilized for all communications on the localhost. The command above will make sure that the connections between a database and a web application on the same machine are working properly.

Enabling Connections on HTTP, SSH, and SSL Port

Next, we want http (port 80), https (port 443), and ssh (port 22) connections to work as usual. To do this, we need to specify the protocol (-p) and the corresponding port

(-dport). We can execute these commands one by one:

sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT

sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT

sudo iptables -A INPUT -p tcp --dport 443 -j ACCEPT

Filtering Packets Based on Source

Iptables allows to filter packets based on an IP address or a range of IP addresses. We need to specify it after the -s option. For example, to accept packets from 192.168.1.3, the command would be:

sudo iptables -A INPUT -s 192.168.1.3 -j ACCEPT

We can also reject packets from a specific IP address by replacing the ACCEPT target with DROP.

sudo iptables -A INPUT -s 192.168.1.3 -j DROP

If we want to drop packets from a range of IP addresses, we have to use the -m option and iprange module. Then, specify the IP address range with -src-range, a hyphen should separate the range of ip addresses without space, like this:

sudo iptables -A INPUT -m iprange --src-range 192.168.1.100-192.168.1.200 -j DROP

Dropping all Other Traffic

It is crucial to use the DROP target for all other traffic after defining -dport rules. This will prevent an unauthorized connection from accessing the server via other open ports.:

sudo iptables -A INPUT -j DROP

Now, the connection outside the specified port will be dropped.

Deleting Rules

If we want to remove all rules and start with a clean slate, we can use the -F option (flush):

sudo iptables -F

This command removes all current rules. However, to delete a specific rule, we must use the -D option.

Persisting Changes

The iptables rules that we have created are saved in memory. That means we have to redefine them on reboot. To make these changes

persistent after restarting the server, we can use this command:

sudo /sbin/iptables-save

It will save the current rules on the system configuration file, which will be used to reconfigure the tables every time the server reboots.



CHECK YOUR PROGRESS

Q 1: Which protocol is used by Network File System(NFS) to provide connection between client and server?

Q 2: What is NIS? Mention its uses.

Q 3: Fill in the blanks:

i) Computers that run DNS are called _____servers.

ii) The most common program used for maintaining a name server on Linux is _____

iii) _____ is a protocol for downloading files from a remote or local server, or uploading files onto the server.

iv) To improve the web server's performance by caching repeated requests, filter web traffic, and access geo-restricted content, _____ server can be used.

v) The squid service can be configured by editing the _____file.



14.11 LET US SUM UP

- Network File System (NFS) is a way of mounting Linux directories over a network. It was basically developed to share files and folders between two Linux systems faster and better.
- FTP stands for "File Transfer Protocol" and is a great protocol for downloading files from a remote or local server, or uploading files onto the server.
- iptables is a firewall program for Linux.
- NAT table contains NAT (Network Address Translation) rules for routing packets to networks that cannot be accessed directly.



14.12 FURTHER READINGS

1) Linux Administration Cookbook By Adam K. Dean
2) Linux Administration By Jason Cannon



14.13 ANSWERS TO CHECK YOUR PROGRESS

Ans to Q No 1: NFS uses Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) to provide a stateless connection between client and server.

Ans to Q No 2: Network Information Service (NIS) is a distributed naming service based on Remote Procedure Call (RPC). It enables easy sharing of various information across the cluster including username, password, hosts, and service ports.

Ans to Q No 3:

i. Name
ii. Berkley Internet Naming Daemon (BIND)
iii. File Transfer Protocol (FTP)
iv. Squid Proxy
v. /etc/squid/squid.conf

14.14 MODEL QUESTIONS

Q 1: What is NFS service? Why We use them?

Q 2: Explain the basic configurations for FTP server?

Q 3: Explain how MAC address are bind to a specific IP address.

Q 4: Construct a rule that disallow any traffic from 192.168.53.10 ip address at port 198.

Q 5: Give a brief overview of NIS architecture.

Q 6: How to change default FTP Ports and directory ?

*** ***** ***