The course provides practical and theoretical insights into network administration using Linux as a platform. Network administrators can make use of Linux as a free and robust operating system to create and configure basic network, security and monitoring devices. The implementations shown throughout this course are derived from real case scenarios applied to corporate networks, through which students can obtain practical experience in designing and configuring network devices under Linux to answer corporate network infrastructure needs, besides understanding the underlying theoretical concepts.
## Contents

### Chapter 1: Configuring a Linux Router

Configuring a Linux Router .................................................................................................. 3
1.1 - Routing Overview ........................................................................................................ 4
1.2 - Enabling Routing ......................................................................................................... 6
1.3 - Checking the Routing Table ....................................................................................... 8
1.4 - Adding Routes ............................................................................................................. 10
1.5 - Deleting Routes ......................................................................................................... 11
1.6 - Configuring the Default Gateway ............................................................................. 12
1.7 - Troubleshooting Tools ............................................................................................... 12
1.8 - Which Router to Choose? ......................................................................................... 14

### Chapter 2: Firewall and Security

Firewall and Security .......................................................................................................... 15
2.1 - Security Concepts ..................................................................................................... 16
2.2 - What’s a Firewall? .................................................................................................... 17
2.3 - What’s Packet Filtering ............................................................................................. 18
2.4 - Firewall Types? ........................................................................................................ 20
2.5 - Protocol States .......................................................................................................... 22
2.6 - Network Address Translations (NAT) ...................................................................... 25

### Chapter 3: Configuring a DNS Server

Configuring a DNS Server ................................................................................................ 29
3.1 - Introduction to DNS .................................................................................................. 29
3.2 - Configuring Linux DNS Clients .............................................................................. 30
3.3 - DNS Under Linux (BIND) ....................................................................................... 32
3.4 - Configuring Linux DNS Server ............................................................................... 34
3.5 - Troubleshooting DNS ............................................................................................. 36

### Chapter 4: IPTABLES Real Case Scenario

IPTABLES Real Case Scenario .......................................................................................... 46
4.1 - The Case Study ......................................................................................................... 46
4.2 - Configuring Filtering ............................................................................................... 47
4.3 - Configuring NAT ..................................................................................................... 49
4.4 - Configuring User Defined Chains .......................................................................... 55
4.5 - Configuring Logging .............................................................................................. 59

### Chapter 5:...

Chapter 5: ......................................................................................................................... 60
DNS BIND (REAL CASE SCENARIO) ................................................................. 60
  5.1- Introduction ...................................................................................... 61
  5.2- Configuring Intranet Name Resolution ............................................. 61
  5.3- Configuring the Public DNS Server ................................................ 64
  5.4- Providing the Complete DNS Solution ............................................. 66

Chapter 6: ................................................................................................. 68
Monitoring with Linux (MRTG) ............................................................... 68
  6.1- Introduction ...................................................................................... 69
  6.2- The SNMP Protocol ......................................................................... 69
  6.3- Installing SNMP Utilities on Linux .................................................. 72
  6.4- Text-based SNMP Tools ................................................................... 72
  6.5- Configuring SNMP on Linux ............................................................. 74
  6.6- Graphical SNMP tools (MRTG) ....................................................... 75
Chapter 1:

Configuring a Linux Router
Aim of the Chapter

A Linux machine can act as a workstation, a server or as a router. The aim of this chapter is to show how to deploy Linux on the network to act as a router.

1.1- Routing Overview

A-What’s routing?

Routing is the act of moving information across an inter-network (i.e. many networks interconnected together) from source to destination. Routing occurs at Layer 3 (the network layer) and it involves determining optimal routing paths before transporting IP packets through an inter-network.

B-What’s a router?

A router is a layer 3 device that interconnects two or more networks by routing packets through its interfaces. A router must have at least two interfaces each of which has an IP address on the network segment it belongs to.

When we say that a network equipment functions at a given layer, it means that it cannot read the header information of any other higher layer. Examples of network equipments are hubs and repeaters functioning at layer 1, bridges and switches at layer 2. Other network equipments may function at higher layers such as firewalls and proxies.

NOTE: It’s important to note that neither a hub nor a switch can isolate broadcast traffic. This is done only by a router which implicitly prevents broadcast traffic in one connected network segment from accessing other segments connected to the same router.

C-Where to place a router:

A router sits between two different network segments. It can be used to separate different sub-networks in the same LAN, or it can be placed between the LAN and the WAN for connecting to the ISP (Internet Service Provider) or to other remote LAN.
Many layer 2 LAN technologies exist (wireless, token ring …etc.), but Ethernet 802.3 is still the most common used technology in LAN. Layer 2 WAN technologies and protocols are different from those used in LAN. As examples of WAN technologies, we have Frame Relay, ISDN, ATM, X.25, PPP or HDLC. Physical specifications at layer 1 (connectors, cables, signal forms…etc) and addressing techniques at layer 2 are generally different in WAN from those used for LAN.

Therefore, and before deploying a Linux router, we must make sure it can handle the layer 2 specifications (protocol and physical ports) for each connected network.

**D-Routing table:**

Routers maintain information in routing tables. Destination/next hop associations tell a router that a particular destination can be gained optimally by sending the packet to a particular router representing the “next hop” on the way to the final destination. When a router receives an incoming packet, it checks the destination address and attempts to associate this address with a next hop. For determining the optimal path to a destination, routers use a *metric* value which is a standard of measurement, such as path length in hops or link parameters. Routing information can be configured *statically* by the network administrator, or *dynamically* through routing protocols.

A routing table may contain three types of routes:

*Local routes:* These are routes created implicitly by the operating system and indicate networks that are directly connected to the router via its interfaces.

*Static routes:* These routes are manually configured by the network administrator.

*Dynamic routes:* These routes are dynamically configured by routing protocols.
Generally, static and dynamic routes provide paths to networks that cannot be directly reached by the router; instead, a route to another reachable gateway (next hop) may lead the packet to reach its final destination.

**NOTE:** A *default route* is used when the network destination of the packet being forwarded has no matching entry in the routing table.

**E-Routing protocols:**

In case of large networks, it would be very difficult for a network engineer to configure all routing tables manually. Routing protocols enable automatic configuration of routing tables in all routers on the network. They enable routers to exchange and update routing information via protocol specific messages. We have many types of routing protocol like **RIP** (Routing Information Protocol) and **OSPF** (Open Shortest Path First). Metrics significance differs according to the routing protocol. In RIP, it represents the number of hops (intermediate routers) to the destination, while in OSPF, it’s a function of link parameters such as delay or bandwidth. A Linux router can handle dynamic routing through *routing daemons*. In this chapter we will configure only static routing.

**NOTE:** IP is called a *routed* protocol, RIP and OSPF are called *routing* protocols.

### 1.2 - Enabling Routing

For your Linux machine to act as a router it must have the required hardware interfaces (NIC or other ports) properly configured with their IP addresses. Then, you have first to enable packet forwarding or the routing function. In simple terms *packet forwarding* enables packets to flow through the Linux machine from one network to another. Routing is disabled by default. Two ways are available to enable routing:

**Using Command:**

The special RAM memory-based `/proc` directory provides a mechanism to the kernel to communicate with processes. The system variable contained in `/proc/sys/net/ipv4/ip_forward` file will enable routing if set to “1”, and if set to “0”, routing will be disabled. To instantly enable routing, use the following command:

```
# echo 1 > /proc/sys/net/ipv4/ip_forward
```

This value will be loosed after restarting the system or the network service.
**Configuration file**

The Linux kernel configuration parameter that control routing or IPv4 forwarding is named `net.ipv4.ip_forward` and can be found in the `/etc/sysctl.conf` configuration file. For permanently enabling or disabling routing, you have first to remove the "#" from the line related to packet forwarding, and then:

To disable packet forwarding, set the parameter as follows:

```bash
net.ipv4.ip_forward=0
```

To enable packet forwarding, set the parameter as follows:

```bash
net.ipv4.ip_forward=1
```

This enables packet forwarding only after you reboot the system or the network service. The value set for net.ipv4.ip_forward parameter will be copied to the `/proc/sys/net/ipv4/ip_forward` file. Another way to activate the feature immediately instead of restarting the network service is to force Linux to read the `/etc/sysctl.conf` file with the `sysctl` command using the `-p` switch. `sysctl` is used to modify kernel parameters at runtime.

Here is how it's done:

```bash
# sysctl -p
```

This will show the following:

```bash
net.ipv4.ip_forward = 1
kernel.sysrq = 0
net.ipv4.conf.default.rp_filter = 1
kernel.sysrq = 0
kernel.core_uses_pid = 1
```

This command displays other system variables contained in `/etc/sysctl.conf`
1.3- Checking the Routing Table

The `route -n` or `netstat -nr` commands will all provide the contents of the routing table. Networks with a *gateway of 0.0.0.0* are usually *directly connected* to the interface because no gateway is needed to reach your own directly connected interface. The route with a *destination address and Genmask of 0.0.0.0* is your *default gateway*.

Here is an example of a routing table using the `route -n` command:

```
> route -n
Kernel IP routing table
Destination Gateway      Genmask Flags Metric Ref   Use Iface
10.60.10.0 192.168.1.1  255.255.255.0 UG    0     0   0 eth0
10.50.10.0 192.168.1.1  255.255.255.0 UG    0     0   0 eth0
192.168.1.0 0.0.0.0     255.255.255.0 U      0     0   0 eth0
10.90.10.0 10.20.10.100 255.255.255.0 UG    0     0   0 eth0
10.125.10.0 192.168.1.1 255.255.255.0 UG    0     0   0 eth0
10.20.10.0 0.0.0.0     255.255.255.0 U      0     0   0 eth0
169.254.0.0 0.0.0.0     255.255.255.0 U      0     0   0 eth0
0.0.0.0    192.168.1.1  0.0.0.0    UG    0     0   0 eth0
```

*Fig 1.2 The routing table as shown with the `route -n` command*

The routing table using the `netstat -nr` command:

```
> netstat -nr
Kernel IP routing table
Destination Gateway      Genmask Flags MSS Window irtt Iface
10.60.10.0 192.168.1.1  255.255.255.0 UG    0     0   0 eth0
10.50.10.0 192.168.1.1  255.255.255.0 UG    0     0   0 eth0
192.168.1.0 0.0.0.0     255.255.255.0 U      0     0   0 eth0
10.90.10.0 10.20.10.100 255.255.255.0 UG    0     0   0 eth0
10.125.10.0 192.168.1.1 255.255.255.0 UG    0     0   0 eth0
10.20.10.0 0.0.0.0     255.255.255.0 U      0     0   0 eth0
169.254.0.0 0.0.0.0     255.255.255.0 U      0     0   0 eth0
0.0.0.0    192.168.1.1  0.0.0.0    UG    0     0   0 eth0
```

*Fig 1.3 The routing table as shown with the `netstat -nr` command*
The output of the kernel routing table is organized in the following columns:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Destination</td>
<td>The destination network or destination host.</td>
</tr>
<tr>
<td>Gateway</td>
<td>The gateway address or '0.0.0.0' if none set.</td>
</tr>
<tr>
<td>Genmask</td>
<td>The netmask for the destination net; '255.255.255.255' for a host destination and '0.0.0.0' for the <strong>default</strong> route.</td>
</tr>
</tbody>
</table>
| Flags | Possible flags include:  
  U (route is **up**)  
  H (target is a **host**)  
  G (use **gateway**)  
  R (**reinstate** route for dynamic routing)  
  D (**dynamically** installed by daemon or redirect)  
  M (**modified** from routing daemon or redirect) |
| Metric | The 'distance' to the target (usually counted in hops). It is not used by recent kernels, but may be needed by routing daemons. If two routes exist for the same network, the one with less metric will be preferred. |
| Iface | Interface to which packets for this route will be sent. |
| MSS | Default maximum segment size for TCP connections over this route. |
| Window | Default window size for TCP connections over this route. |
| irtt | Initial RTT (Round Trip Time). The kernel uses this to guess about the best TCP protocol parameters without waiting on (possibly slow) answers. |

Another way to view your routing table is to use the **ip route** command:

```
$ ip route
10.90.0.0/24 via 192.168.1.1 dev eth0
10.90.10.0/24 via 192.168.1.1 dev eth0
192.168.1.0/24 dev eth0 proto kernel scope link src 192.168.1.1
10.90.10.0/24 dev eth0 proto kernel scope link src 192.168.1.1
10.20.10.0/24 via 192.168.1.1 dev eth0
10.20.10.0/24 dev eth0 proto kernel scope link src 10.20.10.1
192.168.0.0/16 dev eth0 scope link
default via 192.168.1.1 dev eth0
```

**Fig 1.4** The routing table as shown with the **ip route** command.
The output is the same but in different format. Local routes are indicated by the
“proto kernel scope link src” entry followed by the ip address of the router interface
on the local network.

1.4- Adding Routes

via Command

The route add command can be used to add new routes to your server. The reference
to the destination network has to be preceded with a -net switch and the subnet mask
and gateway values also have to be preceded by the netmask and gw switches
respectively. For example, type the following command to add a route to the
10.0.0.0/24 destination network having 192.168.1.254 as next hop or gateway on eth0.

# route add -net 10.0.0.0 netmask 255.0.0.0 gw 192.168.1.254

We can also specify the metric and interface:

# route add -net 10.0.0.0 netmask 255.0.0.0 gw 192.168.1.254 metric 1 dev eth0

It’s not necessary to specify the metric or the interface for the route. The interface
label eth0 at the end indicates from which interface the router must forward the
packet, except for point-to-point links, this may not be necessary since it can be
deduced from the gateway address from which interface to forward the packet. As for
metric, it’s only used by routing daemons, and by default, it has a value of “0”, for
static and local routes. If we added two different routes for the same network with
different metrics, the one with less metric will be preferred.

If you wanted to add a route to an individual machine, then the "-host" switch would
be used with no netmask value. (The route command automatically knows the mask
should be 255.255.255.255). Here is an example for a route to host 10.0.0.1.

# route add –host 10.0.0.1 gw 192.168.1.254

We can also use ip route command:
# ip route add 10.0.0.0/24 via 192.168.1.254

Unfortunately, routes added via commands cannot persist after a network or a computer restart.

**NOTE:** Another way of making this change after reboot would be to place the command in the file `/etc/rc.d/rc.local`, which is always run at the end of the booting process, but this will not make the route permanent.

**Configuration files**

In Fedora Linux, permanent static routes are added on a per interface basis in files located in the `/etc/sysconfig/network-scripts` directory. The filename format is `route-interface-name` so the filename for interface eth0 would be `route-eth0`.

The format of the file is quite intuitive with the target network coming in the first column followed by the word via and then the gateway's IP address. In our routing example, to set up a route to network 10.0.0.0 with a subnet mask of 255.0.0.0 (a mask with the first 8 bits set to 1) via the 192.168.1.254 gateway, we would have to configure file `/etc/sysconfig/network-scripts/route-eth0` to look like this:

```
10.0.0.0/8 via 192.168.1.254
```

For this to take effect, you must restart the network service. To verify this, you can check if the entry you added was inserted in the routing table or not.

**1.5- Deleting Routes**

Here's how to delete the routes added in the previous section.

```
# route del -net 10.0.0.0 netmask 255.0.0.0 gw 192.168.1.254
Or
# ip route del 10.0.0.0/24 via 192.168.1.254
```
If the route exists in the `/etc/sysconfig/network-scripts/route-eth0` file, the corresponding entry will also have to be deleted. For example, delete the line that reads: `10.0.0.8 via 192.168.1.254` from the file and restart the network service.

**1.6- Configuring the Default Gateway**

As indicated in *chapter 6 (paragraph 6.7)*, we can add the default route using the `route` command as follows:

```
# route add default gw 192.168.1.1
```

Or

```
# ip route add default via 192.168.1.1
```

We can also add a permanent default gateway in the `/etc/sysconfig/network`. *(See chapter 6)*

**1.7- Troubleshooting Tools**

**1.7.1- The `traceroute` command:**

`traceroute` is a command that prints the route taken by packets to the target network host. The usage is the following:

```
# traceroute [option] [hostname or IP address]
```

Some options are:

<table>
<thead>
<tr>
<th>Option</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-m</code></td>
<td>sets the max time-to-live (max number of hops) used in outgoing probe packets (default is 30 hops)</td>
</tr>
<tr>
<td><code>-I</code></td>
<td>For using ICMP ECHO instead of UDP datagrams</td>
</tr>
<tr>
<td><code>-s</code></td>
<td>For changing the IP source address in outgoing probe packets on hosts with more than one IP address.</td>
</tr>
</tbody>
</table>

`traceroute` attempts to trace the route an IP packet would follow to some internet host by launching UDP probe packets with a small `ttl` (time to live) then listening for an ICMP "time exceeded" reply from a gateway. It starts probes with a `ttl` of one and
increase by one until we get to the target host) or hit a max (which defaults to 30 hops & can be changed with the -m flag).
Example of using traceroute using ICMP is in the following:

```
[root@intranet ~]# traceroute -I 192.168.1.1
traceroute to 192.168.1.1 (192.168.1.1), 30 hops max, 38 byte packets
1 10.20.10.1 (10.20.10.1) 0.443 ms 0.419 ms 0.351 ms
2 192.168.1.1 (192.168.1.1) 0.699 ms 0.752 ms 0.529 ms
[root@intranet ~]#
```

**Fig 1.6** The output of the traceroute command.

In this example, we are tracing route to the 192.168.1.1 machine. The first line in the output indicates the target host, the maximum number of hops, and sent packet size. Then, each line of output represents 1 "hop" on the trip to the destination including the round trip times, in milliseconds, for the 3 packets that were sent. If you see times over 350ms, over 1 second, or asterisks (*), then that is the point on the network where you are being slowed down. Other possible annotations after the time are !H, !N, or !P (got a host, network or protocol unreachable, respectively. If the IP of the router or hop is replaced by an asterisk (*), it means that no ICMP "time exceeded" reply packet has been received from this router and therefore, the IP address could not be determined.

**1.7.2- The ttcp command:**

As a network administrator, it’s important to have bandwidth measurement tools while administering your Linux router. The ttcp command is useful for estimating the available bandwidth between two Linux machines, particularly, the TCP or UDP throughput. The ttcp command times the transmission and reception of data between two systems using the UDP or TCP protocols. For testing, the transmitter should be started with -t and -s after the receiver has been started with -r and -s. Tests lasting at least tens of seconds should be used to obtain accurate measurements.

```
[root@intranet ~]# ttcp -r -s
ttcp-r: buflen=8192, nbuf=2048, align=16384/0, port=5001  tcp
ttcp-r: socket
ttcp-r: accept from 10.10.100.44
ttcp-r: 16777216 bytes in 18.66 real seconds = 878.14 KB/sec +++
ttcp-r: 11528 I/O calls, msec/call = 1.66, calls/sec = 617.87
ttcp-r: 0.0user 0.1sys 0:18real 0% 0i+0d 0maxrss 0+3pf 11627+0csw
[root@intranet ~]#
```
On the receiver machine, we started the ttcp test with the `ttcp -r -s` command and at the same time, on the transmitter machine (10.10.100.44), we started the test with `ttcp -t -s`. Results, including throughput in KB/sec are shown on both machines.

**NOTE:** These are standard but not the only used Linux tools, we may found a wide variety of other multipurpose tools on the Internet. Before using any network tool, be sure to understand the protocols (ICMP, UDP...etc) on which it’s based, and check if these protocols may be blocked by any filtering device such as firewalls. Unless you do, results with such tools may be misinterpreted.

### 1.8- Which Router to Choose?

From an economic point of view, router appliances that provide basic Internet connectivity for a small office or home network are becoming more affordable every day, but when budgets are tight you might seriously want to consider modifying an existing Linux server to do the job.

Technically, a dedicated hardware router machine has a special operating system for routing (called IOS or Internetworking Operating System in case of Cisco routers for example). In case of Linux router, the routing and network services are running on top of a multipurpose operating system and will have therefore less performance and even weaker security level since any system vulnerability will impact the routing function.

**Conclusion**

In network administration, configuring routers is a building block. Linux can be configured for static and dynamic routing. For this, the machine must have proper hardware configuration. First, hardware interfaces and IP addresses must be properly configured, and then routing must be enabled. Routes can be added or deleted via command or permanently with configuration files. The `traceroute` network utility helps in determining the path to target host for troubleshooting purposes.

From an economic point of view, using Linux router may be convenient, but when it comes to performance and security, dedicated hardware routers may be more suitable especially for large networks.
Chapter 2:

Firewall and Security
Aim of the Chapter

The aim of this chapter is to introduce security and the firewall functions in today’s network.

2.1- Security Concepts

Network security is a primary consideration. Today, it’s very essential for any enterprise to computerize its workflow, to interconnect computers in a network for sharing data and resources, and most of times, to be connected to the internet. Connection to the Internet is for both allowing public access to the enterprise’s public servers (web, e-mail, application servers…), and for allowing local enterprise users to access the Internet resources. Since the Internet is considered a potential source of threats, security is becoming as important as the network service itself. Many security means of defense exist (encryption, filtering, intrusion detection…etc) for fighting against different types of threats (eavesdropping, Denial of services…etc). A standard way to categorize security threats or mechanism of defense is according to the multilayered model (Network, Host, Data, and Application). This chapter concerns Linux security at the Network level, which can be provided by configuring a Linux firewall.

2.1.1- What to Secure:

Computers hold data and operating systems. The security concept in the IT domain is based on protecting both the data and systems. When it comes to networked computers, especially those connected to the Internet, security threats become more dangerous.

2.1.2- Source of threats:

In fact, when connecting a computer to the network, a new mean of accessing this computer will be present: the network access. Accessing the computer via network will rely on network protocols, particularly, TCP and UDP. Sessions require opened ports on the target computers, on which running services will listen for user request. Therefore, any opened port on any machine on the network is considered to be a threat for the whole network.
Since corporate networks are most often connected to the Internet, we can classify network users as local or internal users (on LAN), and external users (on the Internet). However, this does not mean that external users are the only source of threat. Internal users are considered also to be an equal and even more dangerous source of threat. This is due to the fact that an internal user has more privileges than an external one, and can intentionally (due to social engineering issues), or unintentionally, (due to a network spreading virus or worm), attack hosts on the network.

2.2- What’s a Firewall?

A firewall is the device that can control and filter traffic incoming and outgoing from a given network. For this, a firewall must be placed at the point where the corporate network is connected to the Internet or to other networks. If the firewall is to be used for separating local sub-networks in the same LAN, it must be also placed at the point of their interconnection. On the other hand, and since broadcast traffic is destined to all network hosts, it’s essential for the firewall to be also able to isolate broadcast traffic between network segments.
For these reasons, and in addition to the filtering capabilities, a firewall must have first the routing functionality (for both delivering packets and separating broadcast domains). The firewall is a device that provides connectivity and security in the same time. Therefore, it can be defined as a \textit{router having filtering capabilities}.

A firewall can have more than the routing and filtering functionalities such as NAT (Network Address Translation), logging, VPN (Virtual Private Network) server. The most essential function is filtering and is explained in the following.

\section*{2.3-What’s Packet Filtering}

Packet filtering is the mechanism used by firewalls for controlling network packets Inbound and Outbound. Packet filtering could also be used to control internal network traffic between different LAN segments. Packet filtering is based on filtering rules that would allow or block the traffic. Each packet should be checked against filtering rules.

\subsection*{2.3.1- Filtering rule:}
Generally a filtering rule is a combination of:

\[
\text{<match criteria> + <action to take>}
\]

→ The \text{<match criteria>} could be based generally on one of the following:

- Source or destination IP Addresses
- Transport layer protocol (TCP, UDP)
- Source or destination port number

According to the filtering device capabilities, other criteria could be checked such as:

- Source or destination MAC Addresses
- TCP or IP header flags

Other advanced filtering devices could include some higher layer information such as \textit{user identity}, \textit{application content}. These would require special security devices (such as proxy servers). Generally, we can consider a firewall as being a \textit{layer 4 device}.

→ The \text{<action>} to take is generally \textbf{Accept} or \textbf{Reject}.

To show the logic behind packet filtering rules, here is an example:

<table>
<thead>
<tr>
<th>Match criteria</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Source address</strong></td>
<td><strong>Destination address</strong></td>
</tr>
<tr>
<td>10.40.10.0/24</td>
<td>10.15.10.50</td>
</tr>
<tr>
<td>any</td>
<td>10.15.10.50</td>
</tr>
</tbody>
</table>

In this example, traffic destined to the \textbf{10.15.10.50} (web server) will be accepted only if destined to port 80 and if it originated from the \textbf{10.40.10.0/24} network, which is the port on which the HTTP service is listening. Any other traffic will be rejected with the second rule.

Each packet must be checked against this rule, and the set of all firewall rules will define the filtering policy.

\textbf{2.3.2- Filtering strategy:}
The strategy that must be applied is to accept the traffic we want, and then to reject everything else by default. Another less secure strategy is to reject the unwanted traffic, and to accept by default, which will increase the possibility of network attacks.

2.4- Firewall Types?

Communications are built generally on sessions (e.g. TCP or UDP sessions). During a session, source and destination parameters in the packets headers (IP addresses and port numbers) are switched from one direction to the other: the *initial traffic* is from [client \(\rightarrow\) to server], and *response traffic* is from [server \(\rightarrow\) to client]. In order to be established through the firewall, traffic for a given session must be allowed in both directions.

![Network Access Diagram](image)

*Fig 2.3 Initial and response traffic: source IP addresses and port numbers are switched.*

2.4.1- Stateless firewall:
With stateless firewalls, the initial and response traffic must be allowed explicitly. As shown in the following example, a static rule is needed for allowing the traffic in each direction.

<table>
<thead>
<tr>
<th>Source address</th>
<th>Destination address</th>
<th>Protocol</th>
<th>Destination port</th>
<th>Source port</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.40.10.0/24</td>
<td>10.15.10.50</td>
<td>TCP</td>
<td>80</td>
<td>&gt; 1023</td>
<td>Accept</td>
</tr>
<tr>
<td>10.15.10.50</td>
<td>10.40.10.0/24</td>
<td>TCP</td>
<td>&gt; 1023</td>
<td>80</td>
<td>Accept</td>
</tr>
<tr>
<td>any</td>
<td>any</td>
<td>any</td>
<td>any</td>
<td>any</td>
<td>Reject</td>
</tr>
</tbody>
</table>

With stateless firewalls, response traffic must be allowed for the session to take place. In this case, all ports numbers > 1023 must be opened for the server replies to clients, which would be considered a weakness in the firewall policy since too many ports are opened in the opposite direction. This would be more dangerous if the stateless firewall is to be connected to the Internet and HTTP connections must be allowed for any external web server.

2.4.2 Stateful firewall:

The stateful firewall examines the header information of the packet that initiated the connection. If the inspected packet matches an existing firewall rule that permits it, an entry is added to the state table. From that point forward, the packets in that particular communication session are allowed access without call for further inspection because they match an existing state table entry. State table entry may contain information such as: (transport protocol, source and destination IP addresses and port numbers, flags, sequence and acknowledgment numbers, time to live of the dynamic rule). This method decreases the number of static rules that must be handled by the firewall, and increases the overall performance and security level (since only necessary ports are dynamically allowed and removed at the end of the session).
In this case, static rules are always needed for allowing initial traffic request. One generic rule may be sufficient for allowing response traffic, for example:

<table>
<thead>
<tr>
<th>Source address</th>
<th>Destination address</th>
<th>Protocol</th>
<th>Destination port</th>
<th>Source port</th>
<th>State</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.40.10.0/24</td>
<td>10.15.10.50</td>
<td>TCP</td>
<td>80</td>
<td>&gt; 1023</td>
<td>SYN-SENT</td>
<td>Accept</td>
</tr>
<tr>
<td>10.15.10.50</td>
<td>10.40.10.0/24</td>
<td>TCP</td>
<td>-</td>
<td>-</td>
<td>ESTABLISHED</td>
<td>Accept</td>
</tr>
<tr>
<td>any</td>
<td>10.15.10.50</td>
<td>TCP</td>
<td>any</td>
<td>any</td>
<td>any</td>
<td>Reject</td>
</tr>
</tbody>
</table>

The stateful firewall can distinguish states for each packet by checking the protocol flags (SYN and ACK flags for the TCP protocol) and can then deduce the state of the protocol. Protocol states are discussed in the following.

### 2.5- Protocol States

State is the condition of being of a given communication session, which can differ according to the application or protocol that the parties are using. For connection oriented protocols, like TCP or FTP, states are well defined, while for connectionless protocols, like UDP or ICMP, states can be considered as pseudo-stateful protocols.

#### 2.5.1- TCP states:

Because TCP is a connection oriented protocol, and because the beginning and end of the communication session in TCP is well defined with flags, it’s considered a stateful protocol. TCP’s connection establishment is tracked as being in one of four states. The CLOSED state is entered when no connection exists or when connection is closed with the FIN or RST flags in the TCP header.
<table>
<thead>
<tr>
<th>TCP state</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLOSED</td>
<td>No connection exist</td>
</tr>
<tr>
<td>LISTEN</td>
<td>Host waiting for request</td>
</tr>
<tr>
<td>SYN-SENT</td>
<td>Host sent SYN and waiting for SYN-ACK</td>
</tr>
<tr>
<td>SYN-RCVD</td>
<td>Host received a SYN and will send SYN-ACK reply</td>
</tr>
<tr>
<td>ESTABLISHED</td>
<td>Initiating host enters this state after receiving SYN-ACK</td>
</tr>
<tr>
<td></td>
<td>Responding host enters this state after receiving the ACK</td>
</tr>
</tbody>
</table>

The order in which these states occur during the 3-way TCP handshake is shown in the following figure:

![TCP states: The 3 way handshake](image)

**Fig 2.5** TCP states during the 3-way handshake

### 2.5.2 FTP states:

[Image of FTP states diagram]
Ftp is TCP application that does not behave in the standard way. As shown in the following figure, the client first initializes an FTP connection on port 21 for establishing the CONTROL channel. Using the FTP PORT command, the client specify to the server the suggested port (port y) on which the client will listen for the server request to establish the DATA channel. The server then reinitializes another new connection in the opposite direction (server to client) from port 20 to the port suggested by the client.

This behavior cannot be detected by standard stateless nor stateful firewalls. In fact, this requires some application-level capabilities in the firewall. The stateful firewall with such capabilities will then pay special attention to certain types of application sessions. When the stateful firewall sees that a client is initializing an outbound FTP control session on port 21, it knows to expect the server being contacted to initiate an inbound data channel on TCP port 20 back to the client. The firewall can dynamically create a rule that allows this connection. The firewall discovers on which port the client is contacted (the suggested port y) through the use of application inspection.

**NOTE:** FTP is not the only application that requires special firewall capabilities. Other applications (such as multimedia applications) require also the opening of many parallel sessions (due to the large amount of data to be transmitted). There exists no one firewall in the world that can handle all these services behaviors. These

---

**Fig 2.6** FTP states and the PORT command
applications would require special modules to be handled by a standard stateful firewall. On the other hand, the application-level inspection capabilities with layer 4 firewalls are limited to the first exchanged packets of the session beginnings. For extended application level inspection, a proxy server will be needed. A proxy firewall sits between the client and server, and reinitializes the request to the server on behalf of the client. A proxy server can do application-level inspection without affecting the performance.

2.5.3- UDP and ICMP states:

UDP and ICMP are connectionless protocols, therefore has no states. A stateful device must track a UDP or ICMP session in a pseudo-stateful manner, keeping track of items that are specific to current connections only. Items on which state can be based in this case are source and destination IP addresses (and port numbers in case of UDP). ICMP states can also be tracked according to message types.

2.6- Network Address Translations (NAT)

NAT is generally used for both allowing internal private hosts to access public Internet servers and for allowing public Internet hosts to access internal private servers.

In general, the NAT machine should be placed on the perimeter of the network, just like any filtering machine out there. This means that the NAT and filtering machines is generally the same physical machine.

For this, both servers and hosts on the private LAN should use the NAT server as their default gateway to the Internet.

While the router only reads the destination IP address of the packet, and does not any modifications to the header (except for the TTL), The NAT server modifies the IP header of the packet. In fact, the NAT server translates the source and/or destination addresses of packets to different addresses and then recalculates the checksum of the packet. Based on this, we can classify NAT types as the following:
**SNAT (Source NAT)**

SNAT is used for allowing internal private hosts to access the Internet. It’s used when we have multiple network hosts on a private network that must access the Internet and we can’t afford a real public IP for each one of the hosts. We can use one of the private IP ranges for our local network (for example, 192.168.1.0/24), and then we turn on SNAT for our local network. SNAT will then turn all 192.168.1.0 addresses into its own public IP (for example, 194.126.23.40). In other words, traffic from all devices on the private networks will appear as if it originated from a single IP address on the Internet side of the firewall. This way, there will many clients using the same shared IP address in a **many-to-one** IP address translation. We can also specify **many-to-many** IP address translations by specifying a set of public IP addresses (virtual interfaces addresses) on the public interface of the firewall. SNAT is useful if we want to distinguish between hosts or group of hosts on the public side by assigning different public SNAT IP address for each (for example 172.16.1.0/24 is mapped to 194.126.23.50).

![SNAT Diagram](image)

**Fig 2.7** SNAT allows internal private hosts to access public Internet servers

**NOTE:** Although NAT stands for Network Address Translation, but in fact, what the NAT server does is the PAT (Port Address Translations) or overloading which consists on using different ports. Each connection to the Internet from the local private network is reinitiated with the public IP as the source IP but with different source port (one for each client or session). This will allow the NAT server to define a unique traffic flow (**source and destination IP addresses, source and destination port**
numbers) in the NAT table in order to be able to distinguish and forward outgoing and incoming packets for every different client or session.

**MASQUERADE**

Masquerading has the same functionality of SNAT with the difference that the masquerade IP address always defaults to the IP address of the firewall's main interface. The advantage of this is that you never have to specify the NAT IP address. This makes it much easier to configure NAT with DHCP on the public interface side.

**DNAT (Destination NAT)**

DNAT is used for allowing external Internet users to access our internal private servers. To make this available, the IP address configured on the firewall public interface (could be a virtual address), will be mapped to the internal server private IP address. When an incoming request is destined to the public IP on the firewall (194.126.23.100), the firewall will change the destination IP address in the IP packet header to the private IP of the internal server (172.16.1.100). The remote client will appear to be communicating with the firewall through its public IP address, while in fact it’s being communicating with the internal server. This way, your NAT server or firewall will stand in between the Internet and your private server, allowing it to be accessed via a public IP on the public interface of the firewall (**one-to-one**) while applying in the same time your security policy. With DNAT, we can also share an IP for several internal servers (**one-to-many**) that are separated into several physically different servers (using port forwarding mechanism). Here the combination of the firewall’s single IP address, the remote server’s IP address, and the source/destination port of the traffic can be used to uniquely identify a traffic flow. All traffic that matches a particular combination of these factors may then be forwarded to a single server on the private network. This can also be useful for load balancing issues.
Conclusion

Fig 2.8 DNAT allows external public hosts to access internal private servers

Corporate networks need to be connected to the Internet in order to share resources. On the other hand, data and operating systems need to be protected against network attacks whether originated from internal or external users. Today, there exist various security threats and mechanisms of defense. Firewalls can provide both connectivity and security at the network level while acting as a router that has filtering capabilities. Filtering is a mechanism that can protect a network by checking every incoming and outgoing packet against a set of rules defining the firewall policy. A rule is a combination of a match criteria and the action to be taken if the condition matched. Conditions are based generally on layer 4 and below headers, while actions are generally to accept or to reject. Filtering strategy to reject by default is the most recommended. Two types of firewalls exist: stateless and stateful firewalls. For a communication to be allowed through a firewall, packets are to be allowed in both directions. With the stateless case, static rules are to be added for reply packets to be accepted, while this can occur dynamically in the stateful firewall case. Some special applications (like FTP and multimedia applications) need special modules or configuration on the stateful firewall in order to be handled. For special application level filtering, a proxy server may be needed. Firewalls NAT function can allow both internal private hosts to access public Internet servers (through SNAT) and public Internet hosts to access internal private servers (through DNAT) while always applying the filtering policy.
Chapter 3:

Configuring a DNS Server
Aim of the Chapter

The aim of this chapter is to understand basic configurations for the DNS server (BIND) under the FEDORA Linux operating system.

3.1- Introduction to DNS

DNS service

Domain Name System (DNS) service converts the name of a web site (www.example.com) or a machine (mypc.example.com) to an IP address (65.115.71.34). It can also be used to locate a mail server for a given domain (@example.com). This step is important, because the IP address of a server, not the server's name, is used in routing traffic over the Internet. For a client to resolve names to IP addresses, one way is to store these mappings in a local file on the client host. But for large scale networks, it's more accurate to store names to IP addresses mappings on a dedicated DNS server. When networks gets larger, such as in the Internet case, it becomes also more accurate to have a distributed database on many DNS servers, each delegated to hold a specific part or zone of the entire database.

The DNS service is used by both internal and external users to the corporate network. In fact, public DNS servers will help guide Internet users to your network public resources such as web and e-mail services, and private DNS servers will help your internal users to resolve and cache Intranet and Internet host names for accessing internal and external resources.

DNS Clients

A DNS client doesn’t store DNS information; it must always refer to a DNS server to get it. The only configuration for a DNS client is to define the IP address(s) of the DNS server(s) it should use. Another way to resolve names to IP addresses is to store these mappings in a local file on the client host.

DNS Domain

Everyone in the world has a first name and a last, or family, name. The same thing is true in the DNS world: A family of Internet resources can be loosely described a domain. For example, the domain example.com has a number of children (servers),
such as www.example.com and mail.example.com for the Web and mail servers, respectively. Domains may also be subdivided into sub-domains, for example, computer.com and example.com are two child domains for the com domain.

Primary DNS Servers

Primary or Authoritative servers provide the definitive information for your DNS domain, such as the names of servers and Web sites in it. They contain the editable copy of the zone information and they are the last word in information related to your domain. In fact, they help guide Internet users to your network public resources such as web and e-mail services.

Secondary DNS servers

A secondary DNS server holds a read only copy of the zone information. As specified in the zone configuration, the secondary server updates its copy from the primary server. When the primary server goes down, the secondary DNS server is used as a backup server to process DNS request for the domain, and it still try to contact the primary server until the expiration of a specified period of time called TTL (Time To live). After this period, both servers stop processing DNS queries for the domain.

DNS Caching Name Server

Most servers don’t ask authoritative servers for DNS directly, they usually ask a caching DNS server (or forwarder) to do it on their behalf. The caching DNS servers then store (or cache), the most frequently requested information to reduce the lookup overhead of subsequent queries. In fact, they help internal users to resolve and cache Intranet and Internet host names for accessing internal and external resources. After you set up your caching DNS server, you must configure each of your corporate network PCs to use it as their DNS server.

NOTE: If you want to advertise your Web site www.my-site.com to the rest of the world, then a regular DNS server not a caching one is what you require. Regular or authoritative name servers are also caching name servers by default.
How forward DNS queries work?

There are 13 root authoritative DNS servers (higher Internet domains authorities) that all DNS servers query first. These root servers know all the authoritative DNS servers for all the main domains .com, .net, and the rest. This layer of servers keeps track of all the DNS servers that Web site systems administrators have assigned for their subdomains. For example, when you register your domain my-site.com, you are actually inserting a record on the .com DNS servers that point to the authoritative DNS servers you assigned for your domain.

A DNS query may be realized through iterative and recursive queries: For example, for a DNS client to resolve www.example.com to an IP address, the DNS client forwards this query to its configured DNS cache server through a recursive query. This server in turn, acts as a DNS client that re-forwards this query (if configured) to another DNS cache server or directly to one of the 13 root authoritative DNS servers. The server will be then redirected to ask the authoritative server for the .com domain, which contains the IP address of the authoritative server of the example.com subdomain which will be finally asked for the www.example.com IP address.

How reverse DNS queries work?

Reverse DNS queries consist on getting the machine names while knowing the IP address. The forward domain query process for mysite.com, for example, scans the FQDN from right to left to get increasingly more specific information about the authoritative servers to use. The in-addr.arpa is the main domain to which all IP addresses belong. The reverse DNS entry for a given domain will be then the in-addr.arpa followed by the first 3 octets of the IP address in reverse order. For example, if 194.126.23.0/24 class C public IP address range is assigned for the mysite.com domain, then, the reverse domain name will be 23.126.194.in-addr.arpa.

3.2- Configuring Linux DNS Clients

3.2.1- Querying DNS servers (/etc/resolv.conf):

DNS clients do not need the BIND package to be installed. The /etc/resolv.conf file is used to determine the IP addresses of the DNS servers of the client. The file generally has two columns; the first contains a keyword, and the second contains the desired values separated by commas. The nameserver keyword specifies the IP address of your
DNS nameserver, and if there is more than one nameserver, you'll need to have multiple "nameserver" lines.

As an example:

```
nameserver 192.168.1.100
nameserver 192.168.1.102
```

This will indicate for the client machine to forward DNS requests to **192.168.1.100** and **192.168.1.102** DNS cache servers.

### 3.2.2- Querying local files (/etc/hosts):

The `/etc/hosts` file is just a list of IP addresses and their corresponding server names. Your server will typically check this file before referencing DNS. If the name is found with a corresponding IP address then DNS won't be queried at all.

Usually the first entry in /etc/hosts defines the IP address of the server's virtual loopback interface (**127.0.0.1**). This is usually mapped to the name `localhost.localdomain` (the universal name used when a server refers to itself) and `localhost` (the shortened alias name).

```
127.0.0.1   localhost.localdomain  localhost
```

You must add a similar line to specify the name of your machine and your domain:

```
192.168.1.254   mypc.mydomain  mypc
```

**NOTE:** You *must* always have a `localhost` and `localhost.localdomain` entry mapping to **127.0.0.1** for some Linux applications to work properly.

To add an entry for a remote machine, we use the following format:

```
10.10.15.1   pexample
```

In the example above server **pexample** has an IP address of 10.10.15.1. You can access 10.10.15.1 using the ping, telnet or any other network aware program by referring to it as **pexample**. For example:

```
# ping pexample   (or)   # telnet  pexample
```
3.3- DNS Under Linux (BIND)

BIND is an acronym for the Berkeley Internet Name Domain project (You can download BIND and many other tools from http://www.bind9.net), which is a group that maintains the DNS-related software suite that runs under Linux. The most well known program in BIND is named, the daemon that responds to DNS queries from remote machines.

3.3.1- The BIND Package:

Most Fedora Linux software products are available in the RPM format. When searching for the file, remember that the BIND RPM’s filename usually starts with the word "bind" followed by a version number, as in bind-9.2.2.P3-9.i386.rpm.

3.3.2- The BIND service

The name of the daemon running the BIND DNS service is named. You can use the chkconfig command to get BIND configured to start at boot. You can also start the BIND after booting by using the service named start command. You must restart the named service every time you make a change to the configuration file for the changes to take effect on the running process. You can also use the service named reload instead of restarting the named service. (See paragraph 2.5 to for changing BIND configuration files).

3.3.3- BIND service security (chroot)

Fedora BIND normally runs as the named process owned by the unprivileged named user. This can also limit the files the user named can see. When installed, named is fooled into thinking that the directory /var/named/chroot is actually the root or / directory. Therefore, named files normally found in the /etc directory are found in /var/named/chroot/etc directory instead, and those you'd expect to find in /var/named are actually located in /var/named/chroot/var/named.

The advantage of this chroot feature is that if a hacker enters your system via a BIND exploit, the hacker's access to the rest of your system will be isolated to the files under the chroot directory and nothing else.

You can determine whether you have the chroot add-on RPM by using the rpm -q bind-chroot command, which returns the name of the RPM.
There can be confusion with the locations: Regular BIND installs its files in the normal locations, and the chroot BIND add-on RPM installs its own versions in their chroot locations.

**NOTE:** Unfortunately, the chroot versions of some of the files are empty. Before starting Fedora BIND, copy the configuration files to their chroot locations. After this, you can only edit files in the chroot location.

### 3.3.4- BIND Configuration files

It is important to understand exactly where the files are located and for what they are used as in the following table:

<table>
<thead>
<tr>
<th>File</th>
<th>Purpose</th>
<th>BIND chroot Location</th>
<th>Regular BIND Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>named.conf</td>
<td>Tells the names of domains of the DNS server and the zone files location to be used for each domain. May contain forwarders IP addresses.</td>
<td>/var/named/chroot/etc</td>
<td>/etc</td>
</tr>
<tr>
<td>Zone files</td>
<td>Contains the domain database that links all the IP addresses in your domain to their corresponding names</td>
<td>/var/named/chroot/var/named</td>
<td>/var/named</td>
</tr>
</tbody>
</table>
3.4- Configuring Linux DNS Server

The following explains the steps needed to configure your DNS BIND server:

3.4.1- Configuring resolv.conf

You’ll have to make your DNS server refer to itself for all DNS queries by configuring the /etc/resolv.conf file to reference localhost only.

nameserver 127.0.0.1

3.4.2- Configuring named.conf

The /var/named/chroot/etc/named.conf file contains the main DNS configuration and tells BIND what domain (stated as zone) the server is responsible and where to find the for configuration files containing domain information for each zone (and forwarders if any). The configuration file for each domain or zone should be located in the default directory of /var/named/chroot/var/named.

The /etc/named.conf file usually has two zone areas:

- **Forward zone** file definitions list files to map domains to IP addresses.
- **Reverse zone** file definitions list files to map IP addresses to domains.

(1) Configuring forward lookup zones

(1.1) Primary DNS zones

In this example, you’ll set up the forward zone for www.my-site.com by placing entries at the bottom of the named.conf file. The zone file is named my-site.zone, and, although not explicitly stated, the file my-site.zone should be located in the default directory of /var/named/chroot/var/named.
Use the code:

```
zone "my-site.com" IN {
    type master;
    file "my-site.zone";
};
```

The `zone "my-site.com"` statement indicates for the server the DNS domain name which is `my-site.com`. The `IN` keyword indicates that *Internet class* is used when defining IP address mapping information for BIND. The `type master` indicates that this is a primary zone, which means also that the server is a primary DNS server for the `my-site.com` domain. The `file "my-site.zone"` indicates that zone information is `/var/named/chroot/var/named/my-site.zone`.

### (1.2) Secondary DNS zones

In order to configure BIND on a separate server to act as a secondary server for the `my-site.com`, it's sufficient to configure the zone as follows:

```
zone "my-site.com" IN {
    type slave;
    file "my-site.zone";
};
```

The only difference is in the type (`slave`) which indicates for the server that it handles a read only copy of the zone file. The zone file will be automatically created in the `/var/named/chroot/var/named` directory on the secondary server after contacting the primary server.

### NOTES:

Zone transfers that occur between primary and secondary DNS servers use **TCP** port 53. For this, we must make sure firewalls on or between the server machines are properly configured. You must also configure your firewall to allow the secondary server to access primary servers on TCP port 53.
One server could be either primary or secondary at the same time for a given zone or domain. You cannot configure master and slave zones for the same domain on the same server.

(2) Configuring Reverse lookup zones

Next, you have to format entries to handle the reverse lookups for your IP addresses. You will have to create reverse zone entries for your corporate environment using the 194.126.23.0/24 address space. This isn't important for the Windows clients on your network, but some Linux applications require valid forward and reverse entries to operate correctly. This reverse zone definition for named.conf uses a reverse zone file named myreverse.zone for the 194.126.23.0/24 network.

```plaintext
zone "23.126.194.in-addr.arpa" {
  type master;
  file "myreverse.zone";
};
```

In some cases, the forward and reverse entries in the zone files may not match. For critical applications, such as e-commerce, these entries must be the same.

(3) Configuring Forwarders

If your DNS server is not directly connected to the Internet, but you want to use it as a cache server, you must configure for it one or many IP addresses of DNS servers or forwarders. In the named.conf file, you must enter the IP address of the forwarder (in this case 194.126.23.1) in the following section:

```plaintext
options {
  forwarders {
    194.126.23.1
  }
};
```

3.4.3- Configuring the Zone Files

In all zone files, you can place a comment at the end of any line by inserting a semi-colon character and then typing in the text of your comment. By default, your zone files are located in the directory `/var/named/chroot/var/named`. Each zone file contains a variety of entries that are shown in the following.
**Time to Live Value**

The very first entry in the zone file is usually the zone's time to live (TTL) value. Caching DNS servers cache the responses to their queries from authoritative DNS servers. The authoritative servers not only provide the DNS answer but also provide the information's time to live, which is the period for which it's valid.

The purpose of a TTL is to reduce the number of DNS queries the authoritative DNS server has to answer. If the TTL is set to three days, then caching servers use the original stored response for three days before making the query again.

$TTL 3D$

BIND recognizes several suffixes for time-related values. For example, **D** signifies days, a **W** signifies weeks, and an **H** signifies hours. In the absence of a suffix, BIND assumes the value is in **seconds**.

**DNS Resource Records**

The rest of the records in a zone file are usually BIND resource records. They define the nature of the DNS information in your zone files that's presented to querying DNS clients. They all have the general format:

```
Name  Class  Type  Data
```

There are different types of records for mail (MX), forward lookups (A), reverse lookups (PTR), aliases (CNAME) and overall zone definitions, Start of Authority (SOA). The data portion is formatted according to the record type and may consist of several values separated by spaces. Similarly, the name is also subject to interpretation based on this factor.

**The SOA Record**

The first resource record is the Start of Authority (SOA) record, which contains general administrative and control information about the domain. It has the format:

```
Name Class Type Name-Server Email-Address Serial-No Refresh Retry Expiry Minimum-TTL
```

The record can be long, and will sometimes wrap around on your screen. For this, you can insert new line characters between the fields as long as you insert parenthesis at
the beginning and end of the insertion. You can also add comments to the end of each new line separated by a semicolon when you do this. Here is an example:

The following table explains what each field in the SOA record means.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The root name of the zone. The &quot;@&quot; sign is a shorthand reference to the current origin (zone) in the /etc/named.conf file for that particular database file.</td>
</tr>
<tr>
<td>Class</td>
<td>There are a number of different DNS classes. Corporate networks will be limited to the IN or Internet class used when defining IP address mapping information for BIND. Other classes exist for non Internet protocols and functions but are very rarely used.</td>
</tr>
<tr>
<td>Type</td>
<td>The type of DNS resource record. In the example, this is an SOA resource record. Other types of records exist, and will be covered later.</td>
</tr>
<tr>
<td>Name-server</td>
<td>Fully qualified name of your primary name server. Must be followed by a period.</td>
</tr>
<tr>
<td>Email-address</td>
<td>The e-mail address of the name server administrator. The regular @ in the e-mail address must be replaced with a period instead. The e-mail address must also be followed by a period.</td>
</tr>
<tr>
<td>Serial-no</td>
<td>A serial number for the current configuration. You can use the date format YYYYMMDD with an incremented single digit number tagged to the end. This will allow you to do multiple edits each day with a serial number that both increments and reflects the date on which the change was made.</td>
</tr>
<tr>
<td>Refresh</td>
<td>Tells the slave DNS server how often it should check the master DNS server.</td>
</tr>
<tr>
<td>Retry</td>
<td>The slave's retry interval to connect the master in the event of a connection failure.</td>
</tr>
<tr>
<td>Expiry</td>
<td>Total amount of time a slave should retry to contact the master before expiring the data it contains. Future references will be directed towards the root servers.</td>
</tr>
<tr>
<td>Minimum-TTL</td>
<td>There are times when remote clients will make queries for servers or subdomains that don't exist. Your DNS server will respond with a no domain or NXDOMAIN response that the remote client caches. This value defines the caching duration your DNS includes in this response.</td>
</tr>
</tbody>
</table>

In the following is an example of the SOA record:

```plaintext
my-site.com       IN     SOA      ns1.my-site.com. root.my-site.com. (2004100801; serial #
                           ; refresh
                           ; retry
                           ; expiry
                           ; minimum
```
So in the example, the primary name server is defined as ns1.my-site.com with a contact e-mail address of hostmaster@my-site.com. The serial number is 2004100801 with refresh, retry, expiry, and minimum values of 4 hours, 1 hour, 1 week, and 1 day, respectively.

**NOTE (1):** If the search key to a DNS resource record is blank it reuses the search key from the previous record which in this case of is the SOA @ sign. If you don't put a period at the end of a host name in a SOA, NS, A, or CNAME record, BIND will automatically tack on the zone file's domain name to the name of the host. So, BIND assumes an A record with www refers to www.my-site.com. This may be acceptable in most cases, but if you forget to put the period after the domain in the MX record for my-site.com, BIND attaches the my-site.com at the end, and you will find your mail server accepting mail only for the domain my-site.com.mysite.com.

**NOTE (2):** For changes to take effects locally the serial must be incremented and the named service must be restarted. It’s important to note that normally it takes about three to four days for your updated DNS information to be propagated to all 13 of the world’s root name servers. You’ll therefore have to wait about this amount of time before starting to notice people hitting your new Web site for example.

**NS, MX, A And CNAME Records**

Like the SOA record, the NS, MX, A, PTR and CNAME records each occupy a single line with a very similar general format. The following table outlines the way their formats:

<table>
<thead>
<tr>
<th>Record Type</th>
<th>Field Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name Field</td>
<td>Class Field</td>
</tr>
<tr>
<td>NS</td>
<td></td>
</tr>
<tr>
<td>Usually blank</td>
<td>IN</td>
</tr>
<tr>
<td>MX</td>
<td></td>
</tr>
<tr>
<td>Domain to be used for mail. Usually the same as the domain of the zone file itself.</td>
<td>IN</td>
</tr>
<tr>
<td>A</td>
<td></td>
</tr>
<tr>
<td>Name of a server in the domain</td>
<td>IN</td>
</tr>
<tr>
<td>Record Type</td>
<td>Field Descriptions</td>
</tr>
<tr>
<td>-------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>Name Field</td>
<td>Class Field</td>
</tr>
<tr>
<td>CNAME</td>
<td>Server name alias</td>
</tr>
<tr>
<td>PTR</td>
<td>Last octet of server's IP address</td>
</tr>
</tbody>
</table>

The following lines in the **my-site.zone** define some examples of these record types:

```
NS     ns1 ; IP Address of nameserver
my-site.com. MX 10 pcmail ; Primary Mail Exchanger

ns1     A 97.158.253.29
pcmail  A 97.158.253.27
www     CNAME pcweb
pcweb   A 97.158.253.26
```

The first line defines the name server for the domain, as being **ns1** (which means **ns1.my-site.com**), while the third line indicates that **ns1.my-site.com** has 97.158.253.29.

The second line indicates that the MX record is the **pcmail.my-site.com**, which maps to 97.158.253.27.

The www entry in the fifth line is an alias to **pcweb.my-site.com**, which is mapped to 97.158.253.26.

**NOTE:** Beginning the first line with “**my-site.com**.” or with a blank space refers to the same result since implicitly “**my-site.com**” will be added to blank.

### 3.5- Troubleshooting DNS

There are a number of commands you can use to test your DNS servers. Linux uses the **host** and **dig** commands, for example, but Windows uses **nslookup**. You can also check your DNS from the Internet by using public websites dedicated for such tests.
3.5.1- The Host Command:

The host command accepts arguments that are either the fully qualified domain name or the IP address of the server when providing results. To perform a forward lookup by querying your default DNS server specified in the `/etc/resolv.conf`, use the syntax:

```
[root@mypc tmp]# host www.example.com
www.example.com has address 65.115.71.34
```

To perform a reverse lookup

```
[root@mypc tmp]# host 65.115.71.34
34.71.115.65.in-addr.arpa domain name pointer 65-115-71-34.example.com.
```

Here is an example of querying another DNS server, ns1.my-site.com, for the IP address of www.example.com.

```
[root@mypc tmp]# host www.example.com ns1.my-site.com
Using domain server:
Name: ns1.my-site.com
Address: 192.168.1.100#53
Aliases:

www.example.com has address 65.115.71.34
```

3.5.2- The nslookup Command:

The `nslookup` command provides the same results on Windows PCs. To perform forward lookup, use.

```
C:\> nslookup www.example.com
Server: 192-168-1-200.my-site.com
Address: 192.168.1.200
Non-authoritative answer:
Name: www.example.com
Address: 65.115.71.34
```
3.5.3- The Dig command:

You can also use the dig command to determine whether known DNS servers on the Internet have received a valid update for your zone.

The following command uses the local DNS server for the query:

```
[root@mypc tmp]# dig my-site.com

... ;; AUTHORITY SECTION:
www.my-site.com. 3600 IN A 192.168.1.100

;; ADDITIONAL SECTION:
ns1.example.com. 3600 IN A 65.115.70.68
ns2.example.com. 3600 IN A 65.115.70.69
...
[root@mypc tmp]#
```

The host command does one DNS query at a time, but the dig command is much more powerful. When given the right parameters it can download the entire contents of your domain's zone file. We can specify the type of record we want to get. For example, we can use the following arguments: NS, SOA, A...etc. In this example, the AXFR argument is used to get the whole contents of the my-site.com zone file.

```
[root@pcexample tmp]# dig my-site.com AXFR

hostmaster.my-site.com. 2004110701 3600 3600 3600 3600
192-168-1-96.my-site.com. 3600 IN A 192.168.1.96
localhost.my-site.com. 3600 IN A 127.0.0.1
www.my-site.com. 3600 IN A 192.168.1.100
```

Here is a successful dig using DNS server ns1.yahoo.com for the query. As before, it returns the SOA record for the zone.

```
[root@mypc tmp]# dig ns1.yahoo.com example.com NS
;; AUTHORITY SECTION:
exmaple.com. 3600 IN NS ns2.example.com.

;; ADDITIONAL SECTION:
ns1.example.com. 3600 IN A 65.115.70.68
```
3.5.4- Websites:

After you've done with local tests for your domain name server, you can accomplish some public tests via specific websites. There exist many websites to help you check your public DNS server on the Internet, such as: **www.checkdns.net**, or **www.dnsreport.com**. For doing such tests, you just enter your domain name, and the website will generate a complete report for your DNS records by showing any configuration error or warning.

**Conclusion**

DNS is a service used by internal and external corporate network users for name to IP resolution. As a network administrator, it's essential to know how to configure the DNS service. For DNS clients, we configure name servers in the `/etc/resolv.conf` file and local DNS entries in `/etc/hosts`. For DNS servers, the `named.conf` file contains domain names and file location and forwarders IP addresses, while zone files contain name to IP addresses database classified according to many types of records such as SOA, A, CNAME…etc. After each configuration change, the named service must be restarted. Many tools (dig, host) and websites exist for testing your DNS server.
Chapter 4:

IPTABLES Real Case Scenario
Aim of the chapter

The Aim of this chapter is to study a real case scenario that we can meet in nowadays corporate networks. In this study the IPTABLES firewall is used as the central firewall and it will be configured in order to reveal most of its capabilities in secure corporate network environments.

4.1- The Case Study

4.1.1- Network Zones:

In Fig4.1 is given the architecture of a simple corporate network consisting of three logical sub networks with private ranges: 192.168.1.0/24 for the IT department, 172.16.1.0/24 for Sales department and 10.10.10.0/24 for the servers’ zone hosting a typical set of critical servers such as WEB (10.10.10.80), FTP (10.10.10.21), and DNS (10.10.10.53), servers. The corporate network is connected to the Internet with the class C public IP range 194.126.23.0/24.

4.1.2- The Firewall:

The iptables firewall is intercepting all of these network segments and it represents the gateway to the Internet. The firewall interfaces are configured in each network segment as follows: interface eth0 is configured with 192.168.1.1 in the IT department network segment (192.168.1.0/24), eth1 is configured with 172.16.1.1 in the Sales department, eth2 with 10.10.10.1 in the servers’ zone and eth3 with the public IP address 194.126.23.1 in the Internet zone.

4.1.3- The Network hosts:

All internal network hosts in each segment are configured with the firewall as their default gateway. That is, computers in the IT department are configured to have 192.168.1.1 as their default gateway, while those in the Sales department are configured with 172.16.1.1 and servers in the servers’ zone with 10.10.10.1 as the default gateway. This is essential for the firewall to handle inter-zones traffic (that is the internal traffic between zones and the external traffic between the corporate network and the Internet)
Fig. 4.1 Corporate network logical structure
4.2- Configuring Filtering

4.2.1- Configuring Access through the firewall

In the following example, the iptables firewall will be configured properly through the FORWARD chain in the filtering table in order to provide and control access between zones, particularly, the access to services hosted on the servers’ zone. We assume in the following that the “Reject by default” strategy is used in all cases.

4.2.1.1- Allowing ICMP traffic

(case 1) ICMP is a connectionless protocol and it is essential for troubleshooting your network and servers. Sometimes you need to test your network by using tools based on this protocol, such as ping or traceroute commands. For this, you must configure your filtering table to accept traffic related to such tests.

➔ **Policy**: Let’s assume that the stated filtering policy is the following:

“*Allow Any ICMP traffic*”

➔ **Rules**: For this, you must configure the following command rule in your FORWARD chain:

```
iptables -A FORWARD -p icmp -j ACCEPT
```

➔ **Interpretation**: This rule will allow ICMP traffic between any two hosts and in both directions.

(case 2) Now Since allowing ICMP can be source of threat especially if your firewall is directly exposed to the Internet, it’s better to restrict your rules in order to decrease ICMP attacks possibilities, by protecting your critical server zone as in the following example:

➔ **Policy**: Let’s assume that the stated filtering policy in the corporate network is the following:

“*Allow any ICMP traffic except for the servers’ zones*”
→ **Rules:**

```bash
iptables -A FORWARD -p icmp -d 10.10.10.0/24  -j REJECT
iptables -A FORWARD -p icmp   -j ACCEPT
```

→ **Interpretation:** The first rule will reject any ICMP traffic destined for the servers’ zone (we could also specify source destination address as 10.10.10.0/24 since blocking traffic in one way will block the whole ICMP session. We could also specify the input or output interface as eth2). The second rule will allow any other ICMP traffic.

**IMPORTANT NOTE!** Changing the order of rules will change the policy itself because “each packet is checked against each rule successively through the policy”. For example, if the first and second rules were switched, all ICMP traffic will be accepted even if it was destined to the servers’ zone.

### 4.2.1.2- Configuring HTTP

In corporate networks, web servers may need to be accessed by internal and external users. This can be shown in the following example:

→ **Policy:** Let’s assume that the stated filtering policy is the following:

“Allow any user in the corporate network or the Internet to access the web server”

→ **Rules:** For this, you must successively configure the following command rules in your FORWARD chain:

```bash
iptables -A FORWARD -d 10.10.10.80 -p tcp --multiport --sport 1024:65535 --dport 80 --state NEW -j ACCEPT
iptables -A FORWARD --state ESTABLISHED --state NEW -j ACCEPT
```

→ **Interpretation:** The first rule will allow any source IP address to access the web service hosted on the web server having the IP address 10.10.10.80 on destination port 80 (http service) and with source port >1023 (client source ports). Since it’s not specified with the -s option, the source IP address is considered to be “any”, we could also specify the “any” source IP address with 0/0 option. This rule will allow the 3-way-TCP handshake since it specified the NEW state. The established traffic is allowed
in both directions (client-to-server and server-to-client) through the second rule. *(Note that first rule will not allow the established traffic to pass from client to server)*

**NOTE:** Just allowing access to the web server will make it accessible for internal users but not from the outside Internet. For this, we need to also configure DNAT *(see paragraph 3.3)*

### 4.2.1.3- Configuring DNS

**Policy:** In this example, DNS is given in its simplest configuration: one DNS server is used for serving requests for both internal and external users. Therefore, the filtering policy that must be applied is to:

*“Allow any user in the corporate network or the Internet to query the DNS server.”*

**Rules:** On your iptables firewall, the following rules must be added:

```bash
iptables -A FORWARD -d 10.10.10.53 -p udp -m multiport --sport 1024:65535 --dport 53 -j ACCEPT
iptables -A FORWARD -m state --state ESTABLISHED -j ACCEPT
```

**Interpretation:** In the first rule, the initial traffic is allowed from any source IP with client source port >1023 to the DNS server 10.10.10.53 on UDP destination port 53. Since DNS is UDP based, the keyword NEW has less significance, but the ESTABLISHED keyword will allow the established traffic to pass in both directions in the second rule. *(Note that the established traffic in the client to server direction is also allowed by the first rule).*

### 4.2.1.4- Configuring FTP

Here, the same principle is used as in the HTTP case, but with the difference that RELATED traffic must be also allowed.

**Policy:** Let’s assume that the stated filtering policy in the corporate network is the following:

*“Allow only users in the Sales department to access the FTP server”*
**Rules:** On your iptables firewall, the following rules must be added:

```bash
iptables -A FORWARD -s 172.16.1.0/24 -d 10.10.10.21 -p tcp -m multiport --sport 1024:65535 --dport 21 -m state --state NEW -j ACCEPT
iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
```

**Interpretation:** The first rule allows the establishment of the control channel (destination port 21) from client to server (10.10.10.21) only if client is in the 172.16.1.0/24 IP range with client source port>1023.

In the second rule, the RELATED keyword represents traffic belonging to the data connection from server to client which will be allowed. The ESTABLISHED keyword represents the established traffic belonging to both data and control channels which will be also allowed in order to establish a complete FTP session.

**NOTE:** The `ip_nat_ftp` and `ip_conntrack_ftp` modules need to be loaded with the `modprobe` command for handling the FTP service through the firewall. (See chapter IX for more information)

### 4.2.1.5- Configuring Filtering strategy

The filtering strategy is to reject by default. Therefore, the *last appended rule* must be as follows:

For the FORWARD chain:

```bash
iptables -A FORWARD -j REJECT
```

### 4.2.1.6- The overall policy (etc/sysconfig/iptables):

All of the previous command rules can be applied successively and then saved to the `/etc/sysconfig/iptables` file. It's possible to have one common rule that accepts any ESTABLISHED or RELATED traffic for all previous connections allowed by previous rules. The last rule rejects any traffic by default in order to apply the filtering strategy. The format of the file will show the order and body of the `iptables` commands that were applied before save (command: `service iptables save`). Such a file will look like this:
NOTE: that the order in which `iptables` commands with the `-A` option are applied defines the order of rules in the chain since this option appends the rule to the end of the chain. For using these rules in different order, use the `-I` option. (See chapter IX for more details)

### 4.2.2- Configuring Access to the firewall

In the following, we will give examples of basic iptables configuration for allowing the access to the iptables firewall itself. For this, you must configure the filtering table, particularly, the INPUT and OUTPUT chains.

#### 4.2.2.1- Allowing ICMP traffic

**Policy:** Since allowing ICMP can be source of threat especially if your firewall is directly exposed to the Internet, it’s better to restrict your rules in order to decrease icmp attacks possibilities, by specifying for example, the type of ICMP packets. Therefore, the filtering policy that must be applied is to:

*Allow all users to access the firewall via ICMP with only specific ICMP types*

**Rules:** On your iptables firewall, the following rules must be added:

```bash
iptables -A OUTPUT -p icmp --icmp-type echo-reply -j ACCEPT
iptables -A INPUT -p icmp --icmp-type echo-request -j ACCEPT
```
For allowing incoming ICMP ping tests, the firewall must accept ICMP echo-requests (INPUT chain) and in turn, let pass the expected ICMP echo-replies (OUTPUT chain) generated by the firewall.

4.2.2.2- Allowing incoming ssh traffic

Policy: It’s useful to configure your firewall to be accessed via ssh or telnet sessions through trusted networks. For example, the stated policy could be as follows:

“Allow only users in the IT department to access the firewall via ssh.”

Rules: On your iptables firewall, the following rules must be added:

```
iptables -A INPUT -s 192.168.1.0/24 -p tcp --dport 22 -j ACCEPT
iptables -A OUTPUT -m state --state ESTABLISHED -j ACCEPT
```

Interpretation: The first iptables rule allows the firewall to accept TCP packets destined to the firewall (since this rule is in the INPUT chain, we do not need to specify the destination IP address. However, we can specify on which firewall IP address with the –d if we have multiple IP addresses) from any host having IP address in the 192.168.1.0/24 range. Since we did not specify any state for this direction, this rule will allow in the same incoming direction, both the 3-way-TCP related handshake traffic and the following incoming packets for the established traffic. As for the outgoing established traffic, it’s permitted with the ESTABLISHED keyword in the OUTPUT chain with the second rule.

NOTE: The following rules could give equivalent result:

```
iptables -A INPUT -s 192.168.1.0/24 -i eth0 -d 192.168.1.1 -p tcp --dport 22 -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A OUTPUT -m state --state ESTABLISHED -j ACCEPT
```

In this example, we allowed in the first rule NEW or ESTABLISHED ssh traffic for the INPUT chain. This accepts traffic from client-to-firewall (TCP handshake and established traffic). The second rule in the OUTPUT chain allows the established traffic from firewall-to-client.
4.2.2.3- Configuring Filtering strategy

The filtering strategy is to reject by default. Therefore, the last appended rule must be as follows for the INPUT chain:

```
iptables -A INPUT -j REJECT
```

4.3- Configuring NAT

Configuring NAT table in the iptables firewall consists on the POSTROUTING (SNAT and MASQUERADE), PREROUTING (DNAT) chains.

**IMPORTANT NOTE:** Once SNAT, DNAT or MASQUERADE have been configured in the `nat` table, you will have to configure iptables to allow the corresponding packet flow in the `filter` table to pass through the firewall. In fact, nat policy may "enable" the access or not while Filtering policy may "accept" the connection or not. For example, if in the filter table web browsing (http 80) was prohibited, internet users will not be able to browse the internal web server even if DNAT was properly configured in the nat table.
### 4.3.1- Configuring SNAT

SNAT could be configured to enable the access to the Internet for internal users according to the following IP scheme:

![SNAT IP scheme](image-url)

**Fig 4.2** SNAT IP scheme

**Policy:** Let’s assume that the stated nat policy in the corporate network is the following:

"Enable users in the IT and Sales departments to access the Internet"
Rules: On your iptables firewall, the following rules must be added:

```bash
iptables -t nat -A POSTROUTING -s 192.168.1.0/24 -j SNAT --to-source 194.126.23.40
iptables -t nat -A POSTROUTING -s 172.16.1.0/24 -j SNAT --to-source 194.126.23.50
```

Interpretation: All clients on the private network 192.168.1.0/24 network will be SNATed to the 194.126.23.40 public external IP address. The same is for clients on 172.16.1.0/24 and the 194.126.23.50 IP address. This will enable the access to public Internet from clients with private IP ranges.

NOTE: We can also specify a one-to-one relation by determining a single IP address on the private network.

```bash
iptables -t nat -A POSTROUTING -s 192.168.1.100 -j SNAT --to-source 194.126.23.100
```

In this example, only the client with the 192.168.1.100 IP address will be SNATed to the 194.126.23.100 public interface. Note also that the order in which rules are placed is obviously an important issue. In this example, it will not make sense to place the (one-to-one) rule after the (one-to-many) rule because it will be masked: The 192.168.1.100 client IP address will be SNATed to 194.126.23.40 instead of 194.126.23.100 since the rule that matches the condition first will be applied first.

4.3.2- Configuring Masquerading:

To enable IT users to be MASQUERADEd on eth3, use the following command:

```bash
iptables -A POSTROUTING -t nat -o eth3 -s 192.168.1.0/24 -j MASQUERADE
```

Note that here no need to specify the IP address to do MASQUERADE (since the default is taken) but we need to specify the interface from which to take the default IP address. Interface eth3 is the internet public interface. All client on the inside network 192.168.1.0/24 will be masqueraded to the default IP address on the public interface (eth3).
4.3.3- Configuring DNAT

DNAT could be configured to enable access from the Internet to the WEB and DNS servers according to the following IP scheme example:

Fig 4.3 DNAT IP scheme

⇒ Policy: Let's assume that the stated nat policy in the corporate network is the following:

“Enable Internet users to access the WEB and DNS servers”
→ **Rules:** On your iptables firewall, the following rules must be added:

```bash
iptables -t nat -A PREROUTING -d 194.126.23.80 -p tcp --dport 80
    -j DNAT --to-destination 10.10.10.80

iptables -t nat -A PREROUTING -d 194.126.23.53 -p udp --dport 80
    -j DNAT --to-destination 10.10.10.53
```

→ **Interpretation:** In this example, all requests on the public interface **194.126.23.80** on port 80 tcp will be redirected to 10.10.10.80, and requests on **194.126.23.53** on port 80 udp will be redirected to 10.10.10.53. Note that we may not specify the destination ports in these rules, but we will be obliged to in case we have one public IP address that maps to many different private servers.

**NOTE:** As stated earlier, in order for your web and DNS servers to be accessible from the outside Internet, your filtering table must also be configured to allow this access. *(See paragraphs 2.1.2 and 2.1.3)*

### 4.4- Configuring User Defined Chains

As an example of user defined chains, refer to the following:

```bash
iptables -N checkicmp
iptables -A checkicmp -p icmp ! -s 192.168.0.1/24 -d 10.10.10.0/24
    -j REJECT
iptables -A FORWARD -p icmp -j checkicmp
```

In this example we have created one user defined chain in the filter table named **checkicmp** with the first rule. In the second rule, we have appended one rule to this chain that rejects ICMP traffic if it did not originate from 192.168.0.1 and was destined to 10.10.10.0/24. The third rule makes use of the **checkicmp** chain by jumping to it as a target when the protocol is ICMP in the FORWARD chain.

### 4.5- Configuring Logging

For example, if your FTP server 10.10.10.21 is holding confidential data, you may want to log all traffic destined to it as in the following command example:

```bash
iptables -A FORWARD -d 10.10.10.21 -j LOG
```
Chapter 5:

DNS BIND (REAL CASE SCENARIO)
Aim of the Chapter

The Aim of this chapter is to study a DNS real case scenario that we can meet in nowadays corporate networks.

5.1- Introduction

The DNS service is generally used in corporate networks by both internal and external users. In fact, public DNS server will help guide Internet users to your network public resources such as web and e-mail services, while private DNS servers are used to help your internal users to resolve Intranet and Internet host names for accessing internal and external resources. Because of this, hostnames will be registered with their public IP addresses, on the public DNS server, while registered with their private IP addresses on the private DNS server. This chapter will show public and private DNS implementations in order to understand the most common used DNS solution: the SPLIT DNS.

5.2- Configuring Intranet Name Resolution

In the following figure is given the corporate network topology:
5.2.1- Configuring DNS clients:

→ Recommendations

All corporate users must be able to resolve names by querying the private DNS server.

→ Configuration

Each DNS client must be configured as follows:

/etc/resolv.conf

nameserver 10.10.10.200

**NOTE:** the firewall must be configured to allow client DNS request on UDP port 53.

5.2.2- Configuring the private DNS server

→ Recommendations

The private DNS server must be configured to be the authoritative DNS server for the "iul.edu.lb" domain in order to help internal users to resolve only internal corporate servers’ names.

→ Configuration

In this case, the private DNS server will be used to help only internal users to resolve Intranet host names for accessing internal resources. Therefore, servers’ names must be configured with their private IP addresses according to the following:

<table>
<thead>
<tr>
<th>Server role</th>
<th>Servers Name(s)</th>
<th>IP address</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Web server</strong></td>
<td><a href="http://www.iul.edu.lb">www.iul.edu.lb</a></td>
<td>10.10.10.80</td>
</tr>
<tr>
<td></td>
<td>web.iul.edu.lb</td>
<td></td>
</tr>
<tr>
<td><strong>Mail server</strong></td>
<td>mail.iul.edu.lb</td>
<td>10.10.10.25</td>
</tr>
<tr>
<td><strong>FTP server</strong></td>
<td>ftp.iul.edu.lb</td>
<td>10.10.10.21</td>
</tr>
<tr>
<td><strong>Private DNS server</strong></td>
<td>privns.iul.edu.lb</td>
<td>10.10.10.200</td>
</tr>
</tbody>
</table>
(1) **Forward Zone configuration:**

On the private DNS server, we must configure the `/var/named/chroot/etc/named.conf` with the following zone:

```plaintext
zone "iul.edu.lb" IN {
  type master;
  file "iul.zone";
};
```

(2) **Zone file:**

The “`iul.edu.lb`” zone file (`/var/named/chroot/var/named/iul.zone`) must be configured on the private DNS server as follows:

```plaintext
$TTL 3D
iul.edu.lb. IN SOA privns.iul.edu.lb. root.privns..iul.edu.lb. {
  42; serial
  3H; refresh
  15M; retry
  1W; expiry
  1D ; minimum
}
privns NS privns ; Nameserver
iul.edu.lb. MX mail ; Mail server
mail A 10.10.10.200 ; IP Address of Nameserver
web A 10.10.10.80 ; IP address of web server
www CNAME web ; alias to web server
ftp A 10.10.10.21 ; IP address of ftp server
```
5.3- Configuring the Public DNS Server

We assume that the nat and filter tables are configured on the firewall in order to allow public access to your internal servers including the DNS server itself.

-> Recommendations

The public DNS server must be configured to be the authoritative DNS server for the "iul.edu.lb" domain in order to help internet users to resolve internal corporate servers’ names.
Configuration

In this case, an additional public DNS server will be configured to help guiding Internet users to your network public resources such as web and e-mail services. Therefore, servers' names must be configured with their public IP addresses according to the following:

<table>
<thead>
<tr>
<th>Server role</th>
<th>Servers Name(s)</th>
<th>IP address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web server</td>
<td><a href="http://www.iul.edu.lb">www.iul.edu.lb</a>, web.iul.edu.lb</td>
<td>194.126.23.80</td>
</tr>
<tr>
<td>Mail server</td>
<td>mail.iul.edu.lb</td>
<td>194.126.23.25</td>
</tr>
<tr>
<td>Public DNS server</td>
<td>pubns.iul.edu.lb</td>
<td>194.126.23.53</td>
</tr>
</tbody>
</table>

NOTE: Here, we don’t need to access the ftp server from the outside Internet. The public DNS server needs to be published (DNAT) to the Internet for iterative DNS client queries to work.

(1) Forward Zone configuration:

On the public DNS server, we can configure the /var/named/chroot/etc/named.conf with the same zone configurations as with the private DNS server:

```
zone "iul.edu.lb" IN {
    type master ;
    file "iul.zone" ;
};
```
(2) Zone file:

The “iul.edu.lb” zone file (/var/named/chroot/var/named/iul.zone) must be configured as follows:

```
$TTL 3D
iul.edu.lb. IN SOA pubns.iul.edu.lb. root.pubns.iul.edu.lb.  ( 42 ; serial 3H ; refresh 15M ; retry 1W ; expiry 1D ) ; minimum

pubns            NS     pubns ; Nameserver
iul.edu.lb.      A       194.126.23.53 ; IP Address of nameserver
mail             MX      mail ; Mail server
A                194.126.23.25 ; IP address of Mail server
web             A       194.126.23.80 ; web server
www            CNAME     web ; alias to web server
```

5.4- Providing the Complete DNS Solution

⇒ Recommendations

Corporate users need to resolve both Internet and intranet domain names.

⇒ Configuration

By configuring the private and public DNS servers as mentioned previously (paragraph 5.2 and 5.3), Internet users are able to resolve your corporate network servers’ names into their public IP addresses while corporate users are only able to resolve corporate network servers’ names into their private IP addresses.

Therefore, we need to configure the private DNS server to forward DNS queries to the public DNS server. In this case, both servers will act as authoritative and cache servers in the same time. This DNS solution is referred to as the **SPLIT DNS**.
On the private DNS server, we must configure the `var/named/chroot/etc/named.conf`, as follows:

```
options {
    forwarders {
        10.10.10.53;
        10.10.10.53;
    }
};
```

→ Another more advanced solution would be to use "different views" feature in the BIND `named.conf` file. This feature allows us to configure on the same physical DNS server and for the same domain, different zone databases which can be queried according to client IP addresses. By using the "different views", we could merge the private and public DNS servers on one physical machine.
Chapter 6:

Monitoring with Linux (MRTG)
Aim of the Chapter

The aim of this chapter is to configure a Linux monitoring server. You'll learn how to use graphical monitoring tools such as MRTG, which is based on the SNMP protocol for monitoring network traffic and systems' parameters.

6.1- Introduction

Monitoring is an essential issue in troubleshooting your network. As a network or administrator, you'll need to monitor your servers and network devices' performance in order to predict or locate technical problems. It would be important to provide your network with monitoring tools for examining real time network traffic flow and some critical systems' parameters. For example, you may need to monitor the amount of traffic passing through your firewall or router, or accessing your web server. You may need also to monitor the amount of memory usage on your web or application server, or the amount of free hard disk space on your file server. The SNMP (Simple Network Management Protocol) is the protocol made for this purpose.

6.2- The SNMP Protocol

The SNMP (Simple Network Management Protocol) is the protocol made for the purpose of network management or monitoring. Most servers, routers and firewalls keep their operational statistics in object identifiers (OIDs) that you can remotely retrieve via the Simple Network Management Protocol (SNMP). For ease of use, equipment vendors provide Management Information Base (MIB) files for their devices that define the functions of the OIDs they contain. The network node to be monitored could be any device running any type of operating system such as a switch, a Cisco router, an Access point antenna, a Linux machine, a Microsoft Windows server as long as it respects the SNMP protocol standard.

6.2.1- OIDs and MIBs:

OIDs are arranged in a structure of management information (SMI) tree defined by the SNMP standard. The tree starts from a root node, which then descends through
branches and leaves that each adds their own reference value to the path separated by a period. Fig 6.1 shows an OID structure in which the path to the enterprises OID branch passes through the org, dod, internet, and private branches first. The OID path for enterprises is, therefore, 1.3.6.1.4.1.

Management Information Bases (MIBs) are text definitions of each of the OID branches. The following table shows how some commonly used OIDs map to their MIB definitions. For example, the SMI org MIB defines all the topmost OIDs found at the next layer, which is named dod; the internet MIB under dod defines the function of the topmost OIDs in the directory, mgmt, experimental, and private branches. This MIB information is very useful for SNMP management programs, enabling you to see for each OID, its value, type, and description.
You can refer to an OID by substituting the values in a branch with one of these more readable MIB aliases. For example, you can reference the OID 1.3.6.1.4.1.9.9.109.1.1.1.1.5 as `enterprises.9.9.109.1.1.1.5.1` by substituting the branch name (enterprises) for its OID numbers (1.3.6.1.4.1).

Remember, only the OID value at the very tip of a branch, the **leaf**, actually has a readable value. Think of OIDs like the directory structure on a hard disk. Each branch is equivalent to a subdirectory, and the very last value at the tip (the leaf) correlates to a file containing data.

Equipment manufacturers are usually assigned their own dedicated OID branch under the enterprises MIB, and they must also provide information in universally accepted OIDs for ease of manageability. For example, NIC interface data throughput values must always be placed in a predefined location in the general tree, but a memory use value on a customized processor card may be defined in a MIB under the manufacturers’ own OID branch.

### 6.2.2- SNMP Community Strings:

As a security measure, you need to know the SNMP password, or **community string**, to query OIDs. It would be simpler if you define the same community string *(e.g. mycommunity)* on all your corporate network devices. There are a number of types of community strings, the most commonly used ones are the **Read-Only** or "get" community string that only provides access for viewing statistics and system parameters. In many cases the Read Only community string or password is set to the word "public;" you should change it from this easy-to-guess value whenever possible. The **Read-Write** or "set" community string is for not only viewing statistics and system parameters but also for updating the parameters.

### 6.2.3- SNMP Versions:

There are currently three versions of SNMP: SNMP Version 1, SNMP Version 2, and SNMP Version 3. The third version was designed to provide device statistics and error reporting with greater security and remote configuration capabilities than its predecessors.
6.3- Installing SNMP Utilities on Linux

If you intend to use your Linux machine to query your network devices, other servers or even itself using text based or graphical or tools, such as MRTG or any other tool, you need to have the SNMP utility tools package **net-snmp-utils** installed. When searching for the file, remember that the SNMP utility tools RPM's filename usually starts with **net-snmp-utils**, which is followed by a version number, as in net-snmp-utils-5.1.1-2.i386.rpm.

6.4- Text-based SNMP Tools

The SNMP utility tools package installs a number of new commands on your system for doing SNMP queries, most notably **snmpget** for individual OIDs and **snmpwalk** for obtaining the contents of an entire MIB. These commands can be used for reading SNMP information on local or remote machines. The Linux snmpget command outputs the value of a single leaf, and the snmpwalk command provides the values of all leaves under a branch. The command output frequently doesn't list the entire OID, just the MIB file in which it was found and the alias within the MIB. For example

```
SNMPv2-MIB::sysUpTime.0
```

Here the OID value was found in the SNMPv2-MIB file and occupies position zero in the sysUpTime alias.

Both commands require you to specify the community string with a **-c** operator. They also require you to specify the version of the SNMP query to be used with a **-v 1** or **-v 2c**, or **-v 3** operator for versions 1, 2, and 3, respectively. The first argument is the name or IP address of the target device and all other arguments list the MIBs to be queried.

**6.4.1- snmpwalk:**

The most common options used at the end of this command are **system** (for querying system parameters such as CPU or Memory usage) and **interface** (for querying interface parameters such as input or output traffic rates).

→ The following example gets all the values in the interface MIB of the local server using SNMP version 1 and the community string of mycommunity.
 NETWORK ADMINISTRATION UNDER LINUX  |  2016

[root@mypc tmp]# snmpwalk -v 1 -c mycommunity localhost interface
...
...
IF-MIB::ifDescr.1 = STRING: lo
IF-MIB::ifDescr.2 = STRING: eth0
IF-MIB::ifDescr.3 = STRING: eth1
...
...
IF-MIB::ifPhysAddress.1 = STRING:
IF-MIB::ifPhysAddress.2 = STRING: 0:9:5b:2f:9e:d5
IF-MIB::ifPhysAddress.3 = STRING: 0:b0:d0:46:32:71
...
...
[root@mypc tmp]#

Another example for querying system parameters on the localhost is in the following:

[root@mypc snmp]# snmpwalk -v 1 -c mycommunity localhost system

SNMPv2-MIB::sysDescr.0 = STRING: Linux mypc 2.4.18-14 #1 Wed Sep 4 11:57:57 EDT 2002 i586
SNMPv2-MIB::sysObjectID.0 = OID: NET-SNMP-MIB::netSnmpAgentOIDs.10
SNMPv2-MIB::sysUpTime.0 = Timeticks: (425) 0:00:04.25
SNMPv2-MIB::sysContact.0 = STRING: root@localhost
SNMPv2-MIB::sysName.0 = STRING: mypc

➔ In the previous example, you were polling localhost SNMP information. You can
poll any SNMP-aware network device that has SNMP enabled. All you need is the IP
address and SNMP Read Only string and you'll be able to get similar results. All we
need on the remote device (192.168.1.1) is to define the mycommunity
string (see paragraph 6.5). For querying interface parameters on a remote
network machine (192.168.1.254) belonging to the mycommunity string, use the
following:

[root@mypc tmp]# snmpwalk -v 1 -c mycommunity 192.168.1.254 interface

6.4.2- snmpget:

➔ Upon inspecting the output of the snmpwalk command, you can see that the
second interface seems to have the name eth0 and the MAC address 0:9:5b:2f:9e:d5.
You can now retrieve the individual MAC address using the snmpget command.

[root@mypc tmp]# snmpget -v 1 -c mycommunity localhost ifPhysAddress.2

IF-MIB::ifPhysAddress.2 = STRING: 0:9:5b:2f:9e:d5
[root@mypc tmp]#
You can confirm this information using the `ifconfig` command for interface eth0; the very first line shows a matching MAC address.

```
[root@mypc tmp]# ifconfig -a eth0
eth0      Link encap:Ethernet  HWaddr 00:09:5B:2F:9E:D5
    inet addr:216.10.119.244  Bcast:216.10.119.255  Mask:255.255.255.240
```

For getting the interface on a remote network machine (192.168.1.254) belonging to the `mycommunity` string, use the following:

```
[root@mypc tmp]# snmpget -v 1 -c mycommunity 192.168.1.254 ifPhysAddress.2
```

You'll now see how you can configure SNMP on your Linux server to achieve these results. Configuring SNMP on a server isn't hard, but it does require a number of detailed steps.

### 6.5- Configuring SNMP on Linux

By default Fedora, installs the `net-snmp` package as its SNMP server product. This package uses a configuration file named `/etc/snmp/snmpd.conf` in which the community strings and other parameters may be set. The version of the configuration file that comes with net-snmp is quite complicated. It would be better to archive it and use a much simpler version with only a single line containing the keyword `rocommunity` followed by the community string. Here is an example.

Configuring SNMP on a server isn't hard, but it does require a number of detailed steps.

1) Save the old configuration file

```
[root@mypc tmp]# cd /etc/snmp/
[root@mypc snmp]# mv snmpd.conf snmpd.conf.old
[root@mypc snmp]# vi snmpd.conf
```

2) Enter the following line in the new configuration file to set the Read Only community string to `mycommunity`.

```
rocommunity mycommunity
```
rocommunity mycommunity

3) Restart SNMP to load the current configuration file.

[root@mypc root]# service snmpd restart

You must restart your snmpd service so the configuration settings become active. Remember, the snmpd.conf file is only read by the snmpd daemon when it starts up.

4) Test whether SNMP can locally read the system and interface MIBs using the snmpwalk command.

[root@mypc snmp]# snmpwalk -v 1 -c mycommunity localhost system
[root@mypc snmp]# snmpwalk -v 1 -c mycommunity localhost interface

**NOTE:** As mentioned previously, the network node to be monitored can be any network device as long as it respects the SNMP standard. Each type of device may require a different method for defining the community string according to the running operating system.

→ Now that you know SNMP is working correctly on your local Linux machine, you can configure SNMP statistics gathering software, such as MRTG, to create online graphs of your traffic flows.

### 6.6- Graphical SNMP tools (MRTG)

#### 6.6.1- What's MRTG?

MRTG (Multi-Router Traffic Grapher) is a public domain package for producing graphs of various router statistics via a Web page. MRTG relies on gathering the information provided by the SNMP protocol to produce instant daily, weekly, monthly and per year graphs. You can easily create graphs of traffic flow statistics through your network devices using MRTG. MRTG could be configured to show graphs for different network devices belonging to different community strings in the same time. The product is available from the MRTG Web site (www.mrtg.org). The following figure, Fig 6.2, shows a sample MRTG graph.
You can now access your MRTG graphs by pointing your browser to the URL:

http://server-ip-address/mrtg

The previous MRTG example shows multiple advanced SNMP parameters (CPU, TCP connections) that were configured to produce graphs for different network devices. MRTG will generate HTML pages with daily, weekly, monthly, and yearly statistics for your interfaces. By default, MRTG provides only network interface statistics. The MRTG Web site, www.mrtg.org, also has links to other sites that show you how to monitor many other subsystems on a variety of devices and operating systems.

6.6.2- MRTG Download and Installation:

You need to install MRTG before proceeding. Most RedHat and Fedora Linux software products are available in the RPM format. When searching for the file, remember that the MRTG RPM’s filename usually starts with mrtg and a version number, as in mrtg-2.10.5-3.i386.rpm.
In addition to MRTG, you need to install the **SNMP utility** tools as explained earlier and you need to have a **Web server** package installed for MRTG to work. RedHat Linux usually comes with the Apache Web server software preinstalled. The easiest way to tell if Apache is installed is to run the `rpm -q httpd` command. By default Apache expects the HTML files for your Web site to be located in `/var/www/html`. MRTG places its HTML files in `/var/www/mrtg`.

### 6.6.3- Configuring MRTG:

By default, MRTG maps the inbound and outbound data throughput rates on the device it is polling (i.e. traffic parameters). Other advanced OIDs (system parameters), such as CPU and memory usage, are out of the scope of this chapter.

When the MRTG RPM is installed, it creates a directory called `/etc/mrtg` in which all future configuration files are stored. MRTG depends on the following services: The `snmpd` service (for generating snmp statistics), the `httpd` service (for accessing MRTG graphs via web) and the `crond` service, (for scheduling mrtg commands to be executes every 5 minutes).

→ To configure MRTG to collect SNMP data to create graphs, follow these steps:

1) On the machine to be monitored, make sure that: `/etc/snmp/snmpd.conf` is configured properly with the community string (e.g. `rocommunity mycommunity`) and that the snmpd service is running (command: `service snmpd start`).

2) Use MRTG’s `cfgmaker` command to create a configuration file named `/etc/mrtg/mrtg.cfg` for the IP address of **192.168.1.254** (using a Read Only community string of `mycommunity`). This will configure MRTG to show inbound and outbound traffic for the network interface configured with **192.168.1.254** only.

```
[root@mypc]# cfgmaker mycommunity@192.168.1.254 > /etc/mrtg/mrtg.cfg
```

You must open the mrtg.cfg file (command: `vi /etc/mrtg/mrtg.cfg`) to add the **workdir** (in order to specify where to place files for the http web site) and **options[...]** (`growright` to specify the left to right option and **bits** to specify the scale in bits). The following html lines specify the format of the graph to be shown in your browser.
NOTE: The *cfgmaker* utility detects all of your network interface cards and creates a corresponding part (Except for the *workdir* and options lines) for each interface in the */etc/mrtg/mrtg.cfg*, but if the network interface is down during the execution of the command, the corresponding entries will be commented out with the # sign.

→ Here, we can make many files for monitoring many network devices by specifying the IP address. On these devices, the proper community strings must have been already configured.

3) Run MRTG using */etc/mrtg/mrtg.cfg* as your argument **three times**. You'll get an error the two times as MRTG tries to move old data files, and naturally, the first time it is run, MRTG has no data files to move.

```
[root@mypc tmp]# env LANG=C /usr/bin/mrtg /etc/mrtg/mrtg.cfg
```

Rateup WARNING: /usr/bin/rateup could not read the primary log file for localhost_192.168.1.254
Rateup WARNING: /usr/bin/rateup The backup log file for localhost_192.168.1.254 was invalid as well
Rateup WARNING: /usr/bin/rateup Can't remove localhost_192.168.1.254.old updating log file
Rateup WARNING: /usr/bin/rateup Can't rename localhost_192.168.1.254.log to localhost_192.168.1.254.old updating log file
[root@mypc tmp]# env LANG=C /usr/bin/mrtg /etc/mrtg/mrtg.cfg
Rateup WARNING: /usr/bin/rateup Can't remove localhost_192.168.1.254.old updating log file
[root@mypc tmp]# env LANG=C /usr/bin/mrtg /etc/mrtg/mrtg.cfg
This command will create automatically the graph files and put them in the 
/var/www/mrtg as indicated by the /etc/mrtg/mrtg.cfg file in the workdir entry.

**NOTE:** The env LANG=C must be placed before using the MRTG command for changing the language character set. The explanation for this is that MRTG usually works only if your system uses an ASCII-based (Western European) character set. If it isn’t set, then you’ll get errors such as this every time you run MRTG from the command line.

4) MRTG is run **every five minutes** by default for refreshing files in the 
/var/www/mrtg, and the file that governs this is /etc/cron.d/mrtg. For MRTG to work correctly, edit this file, replacing all occurrences of /usr/bin/mrtg with env LANG=C /usr/bin/mrtg. Make sure also that the crond service is started (service crond start).

5) Use MRTG’s **indexmaker** command to create a Web index page using your new mrtg.cfg file as a guide. The MRTG Web GUI expects to find the index file in the default MRTG Web directory of /var/www/mrtg/, so the format of the command would be.

```bash
# indexmaker --output=/var/www/mrtg/index.html /etc/mrtg/mrtg.cfg
```

6) With Fedora Core, MRTG creates an add-on configuration file named 
/etc/httpd/conf.d/mrtg.conf that includes all the necessary Apache commands for MRTG to work. This file specifies which network can access your MRTG website (not a replacement). You can allow universal access by commenting out that line along with the Deny from line. This example adds access from any network including the Internet.

```xml
<Location /mrtg>
  Allow from all
</Location>
```

Remember to restart Apache once you have made these modifications in order for these changes to take effect. (service httpd restart).
Conclusion

Server monitoring is always a good practice, because it can help you predict when things are going to go wrong or long term trends in your network traffic. It would be a good indicator for troubleshooting or dimensioning your network. SNMP is the protocol used for gathering network and systems information parameters. In order to be monitored, each network node must be configured with the SNMP protocol to belong to a community string. After being configured with SNMP, information about a network device could be retrieved via text based tools (locally or remotely) such as `snmpwalk` and `snmpget`, or via graphical tools, such as MRTG based on the Apache Linux web server.