# Chapter 13

# Network Services: DNS, SMTP, iSCSI, and NTP

## CERTIFICATION OBJECTIVES



This chapter examines four system services: the Domain Name System (DNS), the Simple Mail Transfer Protocol (SMTP), the Internet Small Computer System Interface (iSCSI), and the Network Time Protocol (NTP) service.

For DNS, the RHCE objectives require the configuration of a caching-only name server. So in this book, we do not cover the configuration of a master or secondary DNS server.

Next, we'll cover SMTP e-mail services. Linux offers a number of alternative methods for handling incoming and outgoing e-mail. RHEL 7 includes two SMTP services: sendmail and

Postfix. We focus on Postfix, the default RHEL 7 mail transfer agent. Postfix was originally developed in the late 1990s as an alternative to sendmail. It is modular and relatively easy to configure.

You will also learn how to configure a storage device and persistent mount that via the Internet Small Computer Systems Interface (iSCSI) protocol. The storage device is known as an iSCSI target, whereas clients are known as iSCSI initiators.

Finally, whereas you learned about the default NTP server in Chapter 5, you'll learn about the configuration of NTP peers in this chapter.

## INSIDE THE EXAM

### Domain Name Service

Examine the RHCE objectives associated with DNS:

- Configure a caching-only name server
- Troubleshoot DNS client issues

You'll learn how to troubleshoot DNS client issues with the help of the **dig** and **host** commands.

### The SMTP Service

The objective related to e-mail services on the RHCE exam is relatively simple:

- Configure a system to forward all email to a central mail server

The focus of this objective is to configure a null client—that is, a system that can only forward e-mails to a remote server. However, for testing purposes in our lab environment we would need a second system configured to accept inbound e-mails. Although this is not a specific RHCE requirement (it was on the RHCE objectives for RHEL 6), we will cover some of the more general configuration of mail transfer agents (MTAs).

### iSCSI Targets and Clients

This chapter also addresses the configuration of iSCSI targets and clients (initiators), as described in the following objective:

- Configure a system as either an iSCSI target or a initiator that persistently mounts an iSCSI target

### The Network Time Service

Finally, one service that was partially covered in the RHCSA objectives is based on the NTP protocol. Whereas the focus on the RHCSA was to "configure a system to use time services," the RHCE objective suggests that you need a more in-depth knowledge of the configuration of NTP servers:

- Synchronize time using other NTP peers

In addition, you need to meet the basic RHCE objectives that apply to all network services, as discussed in Chapter 11.

CERTIFICATION OBJECTIVE 13.01

# An Introduction to Domain Name Services

DNS is a service that translates human-readable hostnames such as www.mheducation.com to IP addresses such as 192.0.2.101, and vice versa. DNS is a distributed database; each server has its own delegated zone of authority for one or more domains. The DNS service associated with RHEL is the Berkeley Internet Name Domain (BIND). As no individual DNS server is large enough to keep a database for the entire Internet, each server can refer requests to other DNS servers.

RHEL 7 includes another DNS service, Unbound. The Unbound package does not include all the features of BIND, but it is simple and easy to configure if you need just a secure caching DNS resolver. In this chapter, we will cover the configuration of both BIND and Unbound. You can use either of them to meet the RHCE objective related to DNS configuration.

## The BIND Name Server

The default DNS service on RHEL 7 is based on the **named** daemon, included with the BIND 9.9 software package developed through the Internet Systems Consortium. This package includes the **rndc** command, which you can use to manage DNS operations.

### DNS Package Options

To configure a system as a BIND DNS server, start with the RPMs associated with the DNS Name Server package group, shown here:

- **bind** Includes the basic name server software and extensive documentation
- **bind-chroot** Adds directories that isolate BIND in a so-called "chroot jail," which limits access if DNS is compromised
- **bind-dyndb-ldap** Provides an LDAP back-end plug-in for BIND
- **bind-libs** Adds library files used by the bind and bind-utils RPMs
- **bind-libs-lite** Includes a lite version of the BIND libraries for client utilities
- **bind-license** Contains the license file of BIND
- **bind-utils** Includes tools such as **dig** and **host** to query DNS servers and retrieve information about hostnames and domains

By now, you should be comfortable installing these packages with commands such as **yum** from software repositories, as discussed in Chapter 7.


**RHEL 7 also supports the dnsmasq package, which can be used to set up a forwarding DNS server with an integrated DHCP service in a small network.**

## Different Types of DNS Servers

While additional options are available, there are four basic types of DNS servers:

- A master DNS server, authoritative for one or more domains, includes host records for that domain.
- A slave DNS server, which relies on a master DNS server for data, can be used in place of that master DNS server.
- A caching-only DNS server stores recent requests like a proxy server. If configured with forwarding features, it refers to other DNS servers for requests not in its current cache.
- A forwarding-only DNS server refers all requests to other DNS servers.

As described earlier, all you need to know for the RHCE exam is how to configure a caching-only DNS server.

Each of these servers can be configured with access limited to internal networks, or even just a local system. Alternatively, they can be configured as public DNS servers, accessible to the entire Internet. But such access comes with risks, as a successful attack against an authoritative corporate DNS server could easily keep their websites hidden from customers' web browsers. This attack is a form of denial of service.

**CERTIFICATION OBJECTIVE 13.02**

# Minimal DNS Server Configurations

You can configure DNS servers by directly editing the associated configuration files. In this section, we'll briefly review the configuration files installed with the BIND and Unbound software packages. You'll then learn how to configure a caching-only name server, as well as a name server that includes forwarding to specified DNS servers.

## BIND Configuration Files

DNS configuration files can help you configure a Linux system as a database of hostnames and IP addresses. That database can be cached, listed in a local database, or the request can

**TABLE 13-1** DNS Server Configuration Files

| DNS Configuration File | Description |
|---|---|
| /etc/sysconfig/named | Specifies options to be passed to the named daemon at startup. |
| /etc/named.conf | The main DNS configuration file. Includes the location of the zone files. Can incorporate data from other files, normally in the /etc/named directory, with the **include** directive. |
| /etc/named.rfc1912.zones | Adds appropriate zones for localhost names and addresses. |
| /var/named/named.empty | Includes a template zone file. |
| /var/named/named.localhost | Lists the zone file for the localhost computer. |
| /var/named/named.loopback | Lists the zone file for the loopback address. |

be forwarded to a different system. The configuration files that support the use of DNS as a server are described in Table 13-1. While the table includes references to standard /var/named database files, changes to such files are not required to configure a caching-only or forwarding DNS server.

If you've installed the bind-chroot package, a tree of directories and files will also be available in the /var/named/chroot directory to run BIND in a confined chroot jail. If you wish to run BIND in a chroot jail, you'll have to move the configuration files and DNS zones to /var/named/chroot/etc and /var/named/chroot/var/named and then enable the named-chroot service unit.

In the following sections, you'll experiment with the /etc/named.conf file. You should back it up in some fashion. Just be aware of the ownership and, yes, the SELinux contexts of the file, as shown in this output:

```
# ls -Z /etc/named.conf
-rw-r-----. root named system_u:object_r:named_conf_t:s0 /etc/named.conf
```

If backups are restored haphazardly, even by the root user, the group ownership and/or the SELinux contexts may be lost. So if there's ever a failure in starting or restarting the named service, check the ownership and SELinux contexts of the /etc/named.conf file. If necessary, apply the following commands to that file:

```
# chgrp named /etc/named.conf
# restorecon -F /etc/named.conf
```

In addition, after a DNS configuration is tested, some information may remain in a cache. That's the nature of a caching DNS server. If that cache still exists after a change to DNS

configuration files, it could affect the results. Therefore, it's wise to flush the DNS cache after each configuration change with the following command:

```
# rndc flush
```

## A BIND Caching-Only Name Server

When you request a web page such as www.mcgraw-hill.com, a request to resolve the hostname is sent to the configured DNS server. The response is the associated IP address. The request is also known as a *name query.* For requests to external DNS servers, responses can take time. That's where a caching-only name server can help, as repeated requests are stored locally.

**exam Watch**

**The default version of /etc/named.conf is set up for a caching-only name server, limited to the localhost system. Minor changes are required to open that server up to a local network.**

When you are configuring a caching-only name server, the first step is to look at the default version of the /etc/named.conf configuration file. The directives in the default version of this file are organized to set up a caching-only name server. One view of this file is shown in Figure 13-1.

- The **options** directive encompasses several basic DNS directives, including the following:
  - The **listen-on port** (and **listen-on-v6 port**) directives specify the port number to listen on (for IPv4 and IPv6).

  To extend this to a local network, you'll need to include the IP address of the local network interface. For example, if you want the server to respond to queries on the local IPv4 address of 192.168.122.50, you'd change the directive to read as follows (don't forget the semicolon followed by a space after each IP address):

  ```
  listen-on port 53 { 127.0.0.1; 192.168.122.50; };
  ```

  If IPv6 networking is active on the local network, you would need to configure similar IPv6 addresses for the **listen-on-v6** directive. If IPv6 networking is not active, the default **listen-on-v6** directive is sufficient.
  - The **directory** directive specifies where the DNS server looks for data files. Be aware, if the bind-chroot RPM is installed, these file paths are relative to the /var/named/chroot path.
  - The **dump-file** directive specifies the file on which BIND dumps the cache for the current DNS database when the **rndc dumpdb** command is issued.

**FIGURE 13-1** The /etc/named.conf configuration file for a caching-only name server

```
options {
        listen-on port 53 { 127.0.0.1; };
        listen-on-v6 port 53 { ::1; };
        directory       "/var/named";
        dump-file       "/var/named/data/cache_dump.db";
        statistics-file "/var/named/data/named_stats.txt";
        memstatistics-file "/var/named/data/named_mem_stats.txt";
        allow-query     { localhost; };

        /*
         - If you are building an AUTHORITATIVE DNS server, do NOT enable recursion.
         - If you are building a RECURSIVE (caching) DNS server, you need to enable
           recursion.
         - If your recursive DNS server has a public IP address, you MUST enable access
           control to limit queries to your legitimate users. Failing to do so will
           cause your server to become part of large scale DNS amplification
           attacks. Implementing BCP38 within your network would greatly
           reduce such attack surface
        */
        recursion yes;

        dnssec-enable yes;
        dnssec-validation yes;
        dnssec-lookaside auto;

        /* Path to ISC DLV key */
        bindkeys-file "/etc/named.iscdlv.key";

        managed-keys-directory "/var/named/dynamic";

        pid-file "/run/named/named.pid";
        session-keyfile "/run/named/session.key";
};

logging {
        channel default_debug {
                file "data/named.run";
                severity dynamic;
        };
};

zone "." IN {
        type hint;
        file "named.ca";
};

include "/etc/named.rfc1912.zones";
include "/etc/named.root.key";
```

- The **statistics-file** directive specifies the file to write statistics data when the **rndc stats** command is issued.
- The **memstatistics-file** directive specifies the location to save memory usage statistics when BIND exits.

- The **allow-query** directive lists the IP addresses allowed to get information from this server. By default, access is limited to the local system. To extend this to another network such as 192.168.122.0/24, you'd change the directive to this:

  ```
  allow-query { 127.0.0.1; 192.168.122.0/24; };
  ```

- The **recursion** directive enables recursive queries. A recursive query will interrogate the authoritative name servers for the requested domain and always provide an answer to clients. This is the behavior that you would expect from a caching name server. As shown in the comments of the named.conf file in Figure 13-1, if the server has a public IP address, you must restrict access to legitimate clients with the **allow-query** directive.
- Since BIND version 9.5, the software has included support for DNS Security Extension (DNSSEC), with **dnssec-*** directives. DNSSEC protects a caching name server from spoofing and cache poisoning attacks by validating the integrity and authenticity of responses received from other name servers. The following directives enable DNSSEC security, validation (to check authenticity), and querying, with the noted **bindkeys-file**:

  ```
  dnssec-enable yes;
  dnssec-validation yes;
  dnssec-lookaside auto;
  bindkeys-file "/etc/named.iscdlv.key";
  managed-keys-directory "/var/named/dynamic";
  ```

- The **logging** directive specifies several more parameters; the **channel** directive specifies output methods, in this case to **default_debug**, activated in the named.run file in the /var/named/data directory, logging only **dynamic** issues.
- The **zone "."** directive specifies the root zone for the Internet, along with the root DNS servers as specified in the /var/named/named.ca file.
- Finally, the **include** directives include the localhost settings described in the /etc/named.rfc1912.zones file, along with a key for the DNSSEC security protocol stored in the /etc/named.root.key file.

No changes are required to create a caching DNS server. All you need to do is install the aforementioned **bind-*** packages and start the **named** service with the following command:

```
# systemctl start named
```

Next, run the **rndc status** command. If successful, you'll see output similar to that shown in Figure 13-2. The **rndc** command is the name server control utility.

**FIGURE 13-2**

The status of an operational DNS server

```
[root@server1 ~]# rndc status
version: 9.9.4-RedHat-9.9.4-14.el7_0.1 <id:8f9657aa>
CPUs found: 1
worker threads: 1
UDP listeners per interface: 1
number of zones: 101
debug level: 0
xfers running: 0
xfers deferred: 0
soa queries in progress: 0
query logging is OFF
recursive clients: 0/0/1000
tcp clients: 0/100
server is up and running
[root@server1 ~]#
```

## Starting named

After starting the DNS server with the **systemctl start named** command, view the systemd journal with the **journalctl -u named** command. If there are problems, you'll see error messages. The journal will usually display the file with the error. You can then stop the service with the **rndc stop** or **systemctl stop named** command and then check the applicable configuration files.

Once you are satisfied with the new configuration, make sure that DNS starts the next time you reboot Linux. As noted in other chapters, the following command makes sure that the **named** daemon starts the next time you boot Linux in the default target:

```
# systemctl enable named
```

## A Forwarding Name Server

This type of DNS server is simple. It requires a single configuration line in the /etc/named .conf configuration file. As you can see, it's straightforward; we've set it to refer to a couple of other DNS servers on our network:

```
options {
     listen-on port 53 { 127.0.0.1; };
     listen-on-v6 port 53 { ::1; };
     directory "/var/named";
     forward only;
     forwarders {
          192.168.122.1;
          192.168.0.1;
     };
};
```

With this configuration, queries to the local name server are forwarded to DNS servers on the IP addresses shown. In a home lab, these are usually the name servers of your Internet service provider.

If you want to open up this server to external queries, a couple of more changes are required. The changes are the same as made earlier to the caching-only name server configuration. As an example, if the local network card has an address of 192.168.122.50, you'd change the **listen-on** directive to

```
listen-on port 53 { 127.0.0.1; 192.168.122.50; };
```

You should also include the **allow-query** directive described earlier, with references to the localhost system and the local network address:

```
allow-query     { localhost; 192.168.122.0/24; };
```

Don't forget to enable the DNS service on the local firewall:

```
# firewall-cmd --permanent --add-service=dns
# firewall-cmd --reload
```

## Forwarding from a Caching-Only Name Server

As suggested earlier, the caching-only name server configured in the default version of the /etc/named.conf file is enabled for recursive queries. Otherwise, it would not be able to return any results from DNS requests for zones for which it is not an authoritative server.

However, you can combine aspects of the caching-only and forwarding name servers just described. Requests not in the local cache would be forwarded to the name servers specified with the **forwarders** directive. Figure 13-3 displays the relevant excerpt of a /etc/named .conf file where the forwarding directives have been included.

## BIND Commands

Two useful commands associated with the BIND service are **named-checkconf** and **rndc**. The **named-checkconf** command checks the /etc/named.conf file for syntax errors. If no errors are found, it exits with a status of 0; otherwise, it shows onscreen the problematic configuration lines.

The **rndc** command arguments are straightforward. Try **rndc** by itself. The output guides you through the available options. The options we use are straightforward: **rndc status**, **rndc flush**, **rndc reload**, and **rndc stop**. If the DNS server is running correctly, the **rndc status** command should display the results shown in Figure 13-2. The **rndc flush** command flushes the server cache. The **rndc reload** command rereads any changes made to the configuration or DNS zone files. Finally, the **rndc stop** command stops the operation of the DNS server.

**FIGURE 13-3**

A caching name server that forwards to specific DNS servers

```
//
// named.conf
//
// Provided by Red Hat bind package to configure the ISC BIND named(8) DNS
// server as a caching only nameserver (as a localhost DNS resolver only).
//
// See /usr/share/doc/bind*/sample/ for example named configuration files.
//

options {
        listen-on port 53 { 127.0.0.1; 192.168.122.50; };
        listen-on-v6 port 53 { ::1; };
        directory       "/var/named";
        dump-file       "/var/named/data/cache_dump.db";
        statistics-file "/var/named/data/named_stats.txt";
        memstatistics-file "/var/named/data/named_mem_stats.txt";
        allow-query     { localhost; };

        forwarders {
                192.168.122.1;
                192.168.0.1;
        };
```

## Unbound as a Caching-Only Name Server

If you don't need a full-featured name server such as BIND, you can opt for the Unbound DNS resolver. This is a small package that provides a caching and forwarding name server.

The Unbound project was initially funded by VeriSign. The software is currently maintained by NLnet Labs and distributed under the BSD license. It was developed with security and modularity in mind, and as such it is a viable alternative to BIND as a local DNS resolver.

To install Unbound, run the following command:

```
# yum install unbound
```

The default configuration file is /etc/unbound/unbound.conf. Although the file contains more than 500 lines, it includes lots of comments and examples. The **man unbound.conf** command provides additional information and some configuration examples.

**exam Watch**

**Red Hat exams are lab based, so results do matter, rather than the way in which you achieve them. Therefore, unless a lab question specifically asks you to install BIND or Unbound, feel free to choose either of them to set up a caching name server.**

To set up a caching/forwarding name server, you only need to enable three directives in the unbound.conf file. First, you should specify which interfaces Unbound should be listening on:

```
interface: 0.0.0.0
```

If you don't include the **interface** directive in the configuration file, Unbound listens only on the localhost interface. The **interface** directive is similar to the **listen-on port** and **listen-on-v6 port** configuration options of BIND. You can specify the IP address of a local interface or 0.0.0.0 to bind to all IPv4 interfaces. If Unbound listens on an interface other than the localhost, enable the DNS service on the local firewall.

Next, specify which clients are allowed to send queries to the server:

```
access-control: 192.168.122.0/24 allow
```

The **access-control** directive has the same function of **allow-query** in BIND. The unbound.conf file provides several commented examples of valid configuration lines:

```
# access-control: 0.0.0.0/0 refuse
# access-control: 127.0.0.0/8 allow
# access-control: ::0/0 refuse
# access-control: ::1 allow
# access-control: ::ffff:127.0.0.1 allow
```

With no **access-control** directive, Unbound allows client queries only from the localhost.

Optionally, configure forwarding to send DNS requests to another name server. Similar to the BIND configuration, you need to define a zone with **name "."** to forward all queries to a name server:

```
forward-zone:
      name: "."
      forward-addr: 192.168.0.1
```

Finally, check the syntax of the configuration with the **unbound-checkconf** command. Start and enable the unbound service with the commands listed here:

```
# systemctl start unbound
# systemctl enable unbound
```

## DNS Client Troubleshooting

After you configure a DNS resolver, examine the results with a command such as the **host mheducation.com localhost** command. The output confirms the use of the local

system as a DNS server and then provides a straightforward view of the IP address of the host and the hostname of the mail server:

```
Using domain server:
Name: localhost
Address: 127.0.0.1#53
Aliases:

mheducation.com has address 204.74.99.100
mheducation.com mail is handled by 20 ↵
mheducation-com.mail.protection.outlook.com.
```

You can use the **dig** or **host** command to examine your work. For example, with the command **dig @127.0.0.1 www.mheducation.com**, you'll see something like the output shown in Figure 13-4.

The **dig** command shown in the figure asks the local DNS server to look for the "A record" of www.mheducation.com. An A record maps a hostname to an IP address. Assuming the IP address information for www.mheducation.com isn't cached locally,

**FIGURE 13-4**

Test a local DNS server with the dig command.

```
; <<>> DiG 9.9.4-RedHat-9.9.4-14.el7_0.1 <<>> @127.0.0.1 www.mheducation.com
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 53296
;; flags: qr rd ra; QUERY: 1, ANSWER: 4, AUTHORITY: 4, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.mheducation.com.           IN      A

;; ANSWER SECTION:
www.mheducation.com.    600     IN      CNAME   ecom-prod-ext-460002190.us-east
1.elb.amazonaws.com.
ecom-prod-ext-460002190.us-east-1.elb.amazonaws.com. 60 IN A 52.1.15.205
ecom-prod-ext-460002190.us-east-1.elb.amazonaws.com. 60 IN A 54.175.172.124
ecom-prod-ext-460002190.us-east-1.elb.amazonaws.com. 60 IN A 52.0.232.222

;; AUTHORITY SECTION:
us-east-1.elb.amazonaws.com. 299 IN     NS      ns-1119.awsdns-11.org.
us-east-1.elb.amazonaws.com. 299 IN     NS      ns-934.awsdns-52.net.
us-east-1.elb.amazonaws.com. 299 IN     NS      ns-235.awsdns-29.com.
us-east-1.elb.amazonaws.com. 299 IN     NS      ns-1793.awsdns-32.co.uk.

;; Query time: 4901 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Mon Nov 30 00:25:17 GMT 2015
;; MSG SIZE  rcvd: 295
```

**TABLE 13-2** The Most Common DNS Resource Records

| DNS Resource Record | Description |
|---|---|
| A | Maps a hostname to an IPv4 address |
| AAAA | Maps a hostname to an IPv6 address |
| PTR | Maps an IP address to a hostname |
| CNAME | An alias; maps a hostname to another hostname |
| NS | Returns the name servers that are authoritative for a DNS zone |
| MX | Returns the mail servers for a DNS zone |
| SOA | Returns information about a DNS zone |

it then contacts one of the forward DNS systems listed in the named.conf file. If those systems are down or otherwise inaccessible, the local DNS server proceeds to forward the request to one of the name servers listed in the named.ca file. As though those are the root name servers for the Internet, the request will be passed on to another DNS server that is authoritative for the mheducation.com domain. Therefore, it may take a number of seconds before you see an answer.

In the answer section shown in Figure 13-4, it looks like www.mheducation.com is actually an alias (CNAME) that points to another hostname. The **dig** command can query all types of DNS resource records with the help of the **-t** switch. For example, to identify the mail servers for the mheducation.com domain, ask for the MX (mail exchange) record with the following command:

```
# dig -t MX mheducation.com
```

As you have noticed, there are different types of DNS resource records. The most common are summarized in Table 13-2.

## EXERCISE 13-1

### Set Up Your Own BIND DNS Server

Following the example files shown previously, set up a local caching DNS server using the BIND name server. Access will be limited to the local system.

1. Install the bind RPM package.
2. Review the contents of the /etc/named.conf file, based on the discussion so far in this chapter. Do not make any changes.

3. Start the DNS server with the following command:

```
# systemctl start named
```

4. To make sure the DNS server is running, run the **rndc status** command. The output should be similar to that shown in Figure 13-2. Compare the output with the **systemctl status named** command.
5. Flush the current cache with the **rndc flush** command.
6. Test the DNS server. Try the **dig @127.0.0.1 www.mheducation.com** command.
7. Stop the BIND service with the **systemctl stop named** command

## EXERCISE 13-2

### Set Up Your Own Unbound DNS Server

The requirements for this exercise are identical to the previous one. However, you will be using the Unbound name server, rather than BIND.

1. Install the unbound RPM package.
2. Review the /etc/unbound/unbound.conf configuration file. Do not make any changes.
3. Start the DNS server with the following command:

```
# systemctl start unbound
```

4. Test the DNS server. Try the **dig @127.0.0.1 www.mheducation.com** command.

## CERTIFICATION OBJECTIVE 13.03

# A Variety of E-Mail Agents

The Postfix configuration files may seem verbose to Linux engineers who are newer to e-mail administration. Do not let the size of the configuration files intimidate you. Just a few changes are required to meet the requirements associated with the RHCE objective. In this section, you'll explore where SMTP services fit in the hierarchy of e-mail services.

**TABLE 13-3** Mail Server Components

| Abbreviation | Meaning | Examples |
|---|---|---|
| MTA | Mail transfer agent | Postfix, sendmail, Dovecot |
| MUA | Mail user agent | mutt, Evolution, mail, Thunderbird |
| MDA | Mail delivery agent | procmail |
| MSA | Mail submission agent | Postfix, sendmail |

## Definitions and Protocols

A mail server has four major components, as described in Table 13-3. On any Linux computer, you can configure a mail transfer agent (MTA) such as Postfix or sendmail for various outbound services, such as forwarding, relaying, smart host communication with other MTAs, aliases, and spooling directories. Other MTAs, such as Dovecot, are designed to handle only incoming e-mail services, based on the protocols they serve, POP3 (Post Office Protocol, version 3) and IMAP4 (Internet Message Access Protocol, version 4).

E-mail systems are heavily dependent on name resolution. While you could handle name resolution through /etc/hosts on a small network, any mail system that is connected to the Internet needs access to a fully functional DNS server. For spam protection and more, it's important to make sure that the system that intends to send an e-mail has a valid reverse DNS record (PTR) and is actually transmitting with that IP address.

But that is only one component of how e-mail works, from transmission to delivery. E-mail messages start with a mail user agent (MUA), a client system for sending and receiving e-mail such as mutt, Evolution, or Thunderbird. With the help of a mail submission agent (MSA), such mail is normally sent to an MTA such as Postfix or sendmail. A mail delivery agent (MDA) such as procmail works locally to transfer e-mail from a server to an inbox folder. procmail can also be used to filter e-mail. Red Hat supports additional MTA services such as Dovecot to enable POP3 and/or IMAP (or the secure cousins, POP3s and IMAPs) to receive e-mail.

SMTP, the Simple Mail Transfer Protocol, has become one of the most important service protocols of the modern era. Much of the Internet-connected world lives and dies by e-mail and relies on SMTP to deliver it. Like POP3 and IMAP, SMTP is a *protocol,* a set of rules for transferring data used by various mail transfer agents.

## Relevant Mail Server Packages

The packages associated with Postfix are part of the "E-mail Server" package group. Key packages are listed in Table 13-4. You can install them with the **rpm** or **yum** command. Just remember that you do not need to install everything in this table.

**TABLE 13-4** Mail Server Packages

| RPM Package | Description |
|---|---|
| cyrus-imapd-* | Installs the Cyrus IMAP enterprise e-mail system. |
| cyrus-sasl | Adds the Cyrus implementation of the Simple Authentication and Security Layer (SASL). |
| dovecot | Supports both the IMAP and the POP incoming e-mail protocols. |
| dovecot-mysql, dovecot-pgsql, dovecot-pigeonhole | Includes database back ends and related plug-ins for Dovecot. |
| mailman | Supports e-mail discussion lists. |
| postfix | The default mail server on RHEL 7. It is an alternative to sendmail. |
| sendmail | Installs the most popular open-source mail server of the same name. |
| sendmail-cf | Adds a number of templates that you can use to generate your sendmail configuration file; required to process several sendmail configuration files. |
| spamassassin | Includes the spam filter package of the same name. |

When installed, the default E-mail Server package group includes software packages for the Postfix and Dovecot servers, along with the SpamAssassin filter. For the purpose of the RHCE exam, you do not need all of these packages, just Postfix. Install Postfix with the **rpm** or **yum** command, if it is not installed by default.

## Use the alternatives Command to Select an E-Mail System

The **alternatives** command, with the ***--config*** switch, supports choices between different services such as Postfix and sendmail:

```
# alternatives --config mta
```

The command leads to the following output, which allows you to choose from installed SMTP e-mail servers. Other SMTP services, if installed, would be included in the list that follows:

```
There are 2 programs which provide 'mta'.

  Selection    Command
-----------------------------------------------
*+ 1           /usr/sbin/sendmail.postfix
   2           /usr/sbin/sendmail.sendmail

Enter to keep the current selection[+], or type selection number:
```

The preceding output assumes that both Postfix and sendmail are installed in the system.

The **alternatives** command does not by itself stop or start a service. If you do not stop the original mail service, the daemon will still be running. It's important to have only one SMTP service running on a system. Interactions between sendmail and Postfix would lead to errors.

In this chapter, we assume that you are running the Postfix mail transfer agent. You can confirm that Postfix is the default MTA with the following command:

```
# alternatives --list | grep mta
mta     auto     /usr/sbin/sendmail.postfix
```

## General User Security

By default, all users are allowed to use locally configured SMTP services, without passwords. You'll see how this can be changed for Postfix later in this chapter.

In some cases, you may want to set up local users just so they have access to such services. If you don't want such users to log in to the server with regular accounts, one option is to make sure that such users don't have a login shell. For example, the following command can set up a user named tempworker on a local system without a login shell:

```
# useradd tempworker -s /sbin/nologin
```

That tempworker user can then set up his own e-mail manager, such as Evolution, Thunderbird, or even Outlook Express, to connect to networked Postfix or sendmail SMTP services. Any attempts by that user to open an SSH session to the server are rejected.

Of course, access is limited to configured users, whether or not their accounts are configured with a login shell. That's configured courtesy of the Simple Authentication and Security Layer (SASL). As implemented in RHEL 7, it's based on the cyrus-sasl package, configured in the /etc/sasl2 directory. The configuration file for Postfix (smtpd.conf) refers back to the same authentication scheme with the following directive:

```
pwcheck_method:saslauthd
```

The /etc/sysconfig/saslauthd configuration file confirms the standard mechanism for password checks with the following directive:

```
MECH=pam
```

That's a reference to the Pluggable Authentication Modules (PAM) described in Chapter 10. In other words, users who are configured on the local system are controlled by an associated file in the /etc/pam.d directory—namely, smtp.postfix and smtp.sendmail for Postfix and sendmail, respectively. However, you'll need to make a few changes to Postfix to actually make it read the authentication database.

## Mail Logging

Most log messages associated with SMTP services can be found in the /var/log/maillog file. Messages that you might expect to see in this file relate to

- Restarts of Postfix
- Successful and failed user connections
- Sent and rejected e-mail messages

## Common Security Issues

By default, the SMTP service uses port 25. If you open port 25 on the firewall, outside users may have access to that server. You can open that port on the default zone with the following commands:

```
# firewall-cmd --permanent --add-service=smtp
# firewall-cmd --reload
```

To create a custom rule that supports access only from systems on the 192.168.122.0/24 network, you can add a rich rule with the following command:

```
# firewall-cmd --permanent --add-rich-rule='rule family=ipv4↵
source address=192.168.122.0/24 service name=smtp accept'
```

In general, SELinux is not an issue for SMTP services. Only one SELinux boolean applies to the Postfix service, allow_postfix_local_write_mail_spool. It's active by default. As suggested by the name, it allows the Postfix service to write e-mail files to local spools in the /var/spool/postfix directory.

## Testing an E-Mail Server

Besides the **telnet** command described later in this chapter, the appropriate way to test an e-mail server is with an e-mail client. Of course, it would be convenient to have a GUI e-mail client available; however, as discussed in Chapter 2, only text clients such as **mutt** might be available.

### EXERCISE 13-3

### Create Users Just for E-Mail

In this exercise, you will create three users on the local system, just so they can access the local SMTP server. It is understood that additional configuration is required to set up access or limits for these users on the Postfix SMTP server. The users are mailer1, mailer2, and mailer3.

1. Review the **useradd** command. Identify the switch associated with the default login shell.
2. Review the contents of the /etc/passwd file. Find a shell that does not allow logins:

   ```
   /sbin/nologin
   ```

3. Run commands such as **useradd mailer1 -s /sbin/nologin** to add a new user. Make sure to assign that user a password.
4. Review the result in /etc/shadow.
5. Repeat Step 3 for the mailer2 and mailer3 users.
6. Try logging in to one of the new accounts as a regular user. It should fail. Review associated messages in the /var/log/secure file.
7. Keep the new users.

## CERTIFICATION OBJECTIVE 13.04

# The Configuration of Postfix

The Postfix mail server is one way to manage the flow of e-mail on a system and for a network. Standard configuration files are stored in the /etc/postfix directory. The **postconf** command can be used to test the configuration. As installed, Postfix accepts e-mail from only the local system. The configuration changes required to set up Postfix to accept incoming e-mail and to forward e-mail through a smart host are relatively simple.

For the purpose of this chapter, Postfix was installed on the physical host system. Another Postfix server was installed on server1.example.com and configured to forward e-mails to the central mail server running on the physical host. Access tests were performed from the VMs configured in Chapters 1 and 2, representing different external networks.

The details of Postfix configuration files include options for user- and host-based security. If you already know how to configure Postfix for basic operation and just want to know what's required to meet the SMTP objectives for Postfix, jump ahead to the section associated with configuring Postfix as a null client.

## Configuration Files

The configuration files are stored in the /etc/postfix directory. The main configuration file, main.cf, is somewhat simpler than the sendmail alternative, sendmail.cf. It's still complex, as it includes nearly 700 lines.

Except for the .cf files, any changes must first be processed into a database with the **postmap** command. For example, if you've added limits to the access file, it can be processed into a binary access.db file with the following command:

```
# postmap /etc/postfix/access
```

In many cases, the content of files in the /etc/postfix directory is a commented version of the associated man page. The following sections do not cover the main.cf and master.cf files, as those are explained later. They also do not cover the header_checks file, as that's more of a message filter.

After any changes are made to Postfix configuration files, it's normally best to reload them into the daemon with the following command:

```
# systemctl reload postfix
```

## The Postfix access File

The access file may be configured with limits on users, hosts, and more. It includes a commented copy of the associated man page, which can also be called with the **man 5 access** command. When limits are included in that file, they're configured in the following form:

```
pattern action
```

Patterns can be set up in a number of ways. As suggested by the **man 5 access** man page, you can limit users with patterns such as

```
username@example.com
```

Patterns can be configured with individual IP addresses, network addresses, and domains, such as in the following examples. Pay attention to the syntax, specifically the lack of a dot at the end of the 192.168.100 and at the beginning of the example.org expressions. These expressions are inclusive of all systems on the 192.168.100.0/24 network and the *.example.org domain.

```
192.168.122.50
server1.example.com
192.168.100
example.org
```

Of course, such patterns have no meaning without an action. Typical actions include **REJECT** and **OK**. The following examples of lines in the /etc/postfix/access file follow the pattern action format:

```
192.168.122.50 OK
server1.example.com OK
192.168.100 REJECT
example.org REJECT
```

**exam Watch**

**One way to configure host- and user-based security for Postfix is through the access file in the /etc/postfix directory. Another way to configure host-based security is with rich firewall rules, as described in Chapter 10. While there are more complex methods to configure user-based security, the RHCE objectives suggest that you "configure the service for basic operation."**

## The Postfix canonical and generic Files

The files named canonical and generic in the /etc/postfix directory work like an alias file. In other words, when users move from place to place, or if a company moves from one domain to another, the canonical file can ease that transition. Whereas the canonical file applies to incoming e-mail from other systems, the generic file applies to e-mail being sent to other systems.

Similar to the access file, options in these files follow a pattern:

```
pattern result
```

The simplest iteration is the following, which forwards e-mail sent to a local user to a regular e-mail address:

```
michael michael@example.com
```

For companies that use different domains, the following line would forward e-mail directed to michael@example.org to michael@example.com. It would forward other example.org e-mail addresses in a similar fashion.

```
@example.org @example.com
```

Don't forget to process the resulting files into databases with the **postmap canonical** and **postmap generic** commands. If you modify the relocated, transport, or virtual files in the /etc/postfix directory, apply the **postmap** command to those files as well.

## The Postfix relocated File

The /etc/postfix/relocated file is designed to contain information for users who are now on external networks, such as users who have left a current organization. The format is similar to the aforementioned canonical and generic files in the same directory. For example, the following entry might reflect forwarding from a local corporate network to a personal e-mail address:

```
john.doe@example.com  john.doe@example.net
```

### The Postfix transport File

The /etc/postfix/transport file may be useful in some situations where mail is forwarded, such as from a smart host. For example, the following entry forwards e-mail directed to the example.com domain to an SMTP server such as Postfix on the server1.example.com system:

```
example.com  smtp:server1.example.com
```

### The Postfix virtual File

The /etc/postfix/virtual file can forward e-mail addressed in a normal fashion, such as to elizabeth@example.com, to the user account on a local system. For example, if user elizabeth is actually the administrator on a system, the following entry forwards mail sent to the noted e-mail address to the root administrative user:

```
elizabeth@example.com root
```

## The main.cf Configuration File

Back up the main.cf configuration file and open it in a text editor. There are several things that you should configure in this file to get it working. When the service is properly configured, the changes should limit access to the local system and network. This section also describes the function of other active directives, based on the default version of the file.

First, Postfix queues include either e-mail that has yet to be sent or e-mail that has been received. They can be found in the queue_directory:

```
queue_directory = /var/spool/postfix
```

The following directory is a standard. It describes the location of most Postfix commands.

```
command_directory = /usr/sbin
```

Postfix includes a substantial number of executable files for configuration in the master.cf file. The daemon_directory directive specifies their location:

```
daemon_directory = /usr/libexec/postfix
```

Postfix includes writable data files in the following directory; it normally includes a master.lock file with the PID of the Postfix daemon:

```
data_directory = /var/lib/postfix
```

As defined in the comments of the main.cf file, some files and directories should be owned by the root administrative user; others should be owned by the specified mail_owner. In the /etc/groups file, you can confirm a dedicated group named postfix, as well as a group named mail that contains the postfix user:

```
mail_owner = postfix
```

While Postfix works for the local system "out of the box," you need to do more to get it running on a network. To that end, you may need to activate and modify the following **myhostname** directive to point to the fully qualified domain name of the local system, as returned by the **hostname** command. Unless this differs from the Internet hostname of the system, there's no need to change the entry

```
#myhostname = host.domain.tld
```

to the fully qualified domain name, such as

```
myhostname = server1.example.com
```


**An MX record can be configured on the authoritative DNS server for a domain to specify the hostname of the SMTP server that accepts e-mails for that domain.**

You need to configure an SMTP server for an entire domain name with the **mydomain** directive. To that end, change the comment

```
#mydomain = domain.tld
```

to reflect the domain name of the local network:

```
mydomain = example.com
```

Normally, you'd just uncomment the following **myorigin** directive to label e-mail addresses coming from this Postfix server with an origination domain. In this case, the origination domain is example.com:

```
myorigin = $mydomain
```

By default, the following active directive limits the scope of the Postfix service to the local system:

```
#inet_interfaces = all
inet_interfaces = localhost
```

For an e-mail server that handles incoming e-mails for an entire domain, you'd normally change the active directive so that Postfix listens on all active network interfaces:

```
inet_interfaces = all
#inet_interfaces = localhost
```

Normally, Postfix listens on both IPv4 and IPv6 networks, based on the following **inet_protocols** directive:

```
inet_protocols = all
```

The **mydestination** directive specifies the systems served by this Postfix server. Based on the previous settings, the following default directive means that accepted mail may be sent to the local system's FQDN (server1.example.com), the localhost address on the example .com network, and the localhost system:

```
mydestination = $myhostname, localhost.$mydomain, localhost
```

For a Postfix server configured for the local network, you should add the name of the local domain, already assigned to the **mydomain** directive:

```
mydestination = $mydomain, $myhostname, localhost.$mydomain,
localhost
```

The RHCE objectives require you to configure a null client—that is, a system that forwards all e-mails to a central mail server. In that case, you should leave the **mydestination** directive empty to indicate that the local Postfix system is not the final destination for any e-mail domains:

```
mydestination =
```

In addition, you'll want to set up the **mynetworks** directive to point to the client IP address to be trusted by this Postfix server. The default commented directive does not point to the example.com network defined for this book:

```
#mynetworks = 168.100.189.0/28, 127.0.0.0/8
```

So for systems like server1.example.com, this directive may be changed to

```
mynetworks = 192.168.122.0/24, 127.0.0.0/8
```

If you are configuring a null client, this directive should instead be set to the localhost IP address:

```
mynetworks = 127.0.0.0/8
```

Once the changes made to the main.cf file (and any other files in the /etc/postfix directory) are complete and saved, you may want to review the current Postfix parameters. To do so, run the following command:

```
# postconf
```

Of course, most of these parameters are defaults. To review the parameters defined by the main.cf file, run the following command:

```
# postconf -n
```

**exam Watch**

**You can configure host-based security in Postfix through the** mynetworks **directive in the /etc/postfix/main.cf file.**

**FIGURE 13-5** Custom Postfix settings, based on /etc/posfix/main.cf

```
[root@Maui postfix]# postconf -n
alias_database = hash:/etc/aliases
alias_maps = hash:/etc/aliases
command_directory = /usr/sbin
config_directory = /etc/postfix
daemon_directory = /usr/libexec/postfix
data_directory = /var/lib/postfix
debug_peer_level = 2
debugger_command = PATH=/bin:/usr/bin:/usr/local/bin:/usr/X11R6/bin ddd $daemon_
directory/$process_name $process_id & sleep 5
html_directory = no
inet_interfaces = localhost
inet_protocols = all
mail_owner = postfix
mailq_path = /usr/bin/mailq.postfix
manpage_directory = /usr/share/man
mydestination = $myhostname, localhost.$mydomain, localhost
mydomain = example.com
myhostname = maui.example.com
mynetworks = 192.168.122.0/24, 127.0.0.0/8
newaliases_path = /usr/bin/newaliases.postfix
queue_directory = /var/spool/postfix
readme_directory = /usr/share/doc/postfix-2.10.1/README_FILES
sample_directory = /usr/share/doc/postfix-2.10.1/samples
sendmail_path = /usr/sbin/sendmail.postfix
setgid_group = postdrop
unknown_local_recipient_reject_code = 550
[root@Maui postfix]#
```

The output is shown in Figure 13-5.

One setting from the **postconf -n** output is important to authentication. Specifically, when the following directive is added to the main.cf file, Postfix requires authorized usernames and passwords for access:

```
smtpd_sender_restrictions = permit_sasl_authenticated, reject
```

In addition, Postfix includes a syntax checker in the basic daemon. Run the following command to see if there are any fatal errors in the main.cf file:

```
# postfix check
```

## The /etc/aliases Configuration File

Another directive from the /etc/postfix/main.cf file includes the database hash from the /etc/aliases file, which is processed into the /etc/aliases.db file when the Postfix system is restarted:

```
alias_maps = hash:/etc/aliases
```

The /etc/aliases file is normally configured to redirect e-mail sent to system accounts, such as to the root administrative user. As you might see at the end of that file, e-mail messages sent to root can be redirected to a regular user account:

```
# root   marc
```

While there are a number of additional directives available in this file, they're beyond the basic configuration associated with the RHCE objectives. When changes are complete, you can and should process this file into an appropriate database with the **newaliases** command.

## Test the Current Postfix Configuration

As noted in previous chapters, the **telnet** command is an excellent way to review the current status of a service on a local system. Based on the default configuration of Postfix, an active version of this service should be listening on port 25. In that case, a **telnet localhost 25** command should return messages similar to the following:

```
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
220 server1.example.com ESMTP Postfix
```

If IPv6 networking is enabled on the local system, the IPv4 loopback address (127.0.0.1) would be replaced by the regular IPv6 loopback address (::1). The **quit** command can be used to exit from this connection. But don't quit yet. Type the **EHLO localhost** command and press ENTER; the **EHLO** is the enhanced **HELO** command, which returns the basic parameters of an SMTP server.

```
EHLO localhost
250-maui.example.com
250-PIPELINING
250-SIZE 10240000
250-VRFY
250-ETRN
250-ENHANCEDSTATUSCODES
250-8BITMIME
250 DSN
```

For our purposes, the most important information is what's missing. No authentication is required on this server. When authentication is properly configured on Postfix, you'll also see the following line in the output:

```
250-AUTH GSSAPI
```

**FIGURE 13-6**

Directions to set up Postfix authentication

```
Quick Start to Authenticate with SASL and PAM:
----------------------------------------------

If you don't need the details and are an experienced system
administrator you can just do this, otherwise read on.

1) Edit /etc/postfix/main.cf and set this:

smtpd_sasl_auth_enable = yes
smtpd_sasl_security_options = noanonymous
broken_sasl_auth_clients = yes

smtpd_recipient_restrictions =
  permit_sasl_authenticated,
  permit_mynetworks,
  reject_unauth_destination

2) Turn on saslauthd:

   /sbin/chkconfig --level 345 saslauthd on
   /sbin/service saslauthd start

3) Edit /etc/sysconfig/saslauthd and set this:

   MECH=pam

4) Restart Postfix:

   /sbin/service postfix restart
```

## Configure Postfix Authentication

When authentication is configured in Postfix, user limits can apply. However, as there are no hints in the standard main.cf configuration file, you'll have to refer to Postfix documentation for clues. As suggested in Chapter 3, most packages include some level of documentation in the /usr/share/doc directory. Fortunately, Postfix documentation in that directory is rather extensive. In RHEL 7, you'll be able to find that documentation in the postfix-2.10.1/ subdirectory.

The directives that you need to add to the main.cf file to set up authentication are shown in the README-Postfix-SASL-RedHat.txt file in that directory. The key excerpt is shown in Figure 13-6.

For the first step listed, it's sufficient to copy the four directives indicated to the end of the main.cf file. The first enables SASL authentication for Postfix connections:

```
smtpd_sasl_auth_enable = yes
```

Next, the following disables anonymous authentication:

```
smtpd_sasl_security_options = noanonymous
```

The directive that follows allows authentication from nonstandard and deprecated clients such as Microsoft Outlook Express:

```
broken_sasl_auth_clients = yes
```

The next line allows authenticated users, grants access from networks configured with the **mynetworks** directive, and rejects destinations other than the Postfix server:

```
smtpd_recipient_restrictions = permit_sasl_authenticated,
  permit_mynetworks, reject_unauth_destination
```

## Configure Postfix as an SMTP Server for a Domain

The directives required to set up Postfix to accept incoming e-mail from other systems have been previously shown in the description of the main.cf file. However, that discussion was a more comprehensive description of that file. This section just summarizes the minimum requirements to configure Postfix to accept inbound e-mails from other systems. Given a Postfix server configured on the maui.example.com system, on the 192.168.122.0/24 network, you'd make the changes shown in Table 13-5 to the main.cf file in the /etc/postfix directory.

Each of these options replaces either a comment or an active directive in the default /etc/postfix/main.cf file. For example, you should at least comment out the following directive:

```
#inet_interfaces = localhost
```

## Configure Postfix as a Null Client

This section covers the minimum requirements to configure Postfix, in the words of the RHCE objectives, "to forward all email to a central mail server." A smart host provides this

**TABLE 13-5** Postfix Configuration as an SMTP Server for example.com

| Postfix Parameter | Description |
|---|---|
| myhostname = maui.example.com | Specifies the hostname of the system |
| mydomain = example.com | Sets the local domain name |
| myorigin = $mydomain | Specifies the domain that local e-mails will appear to be sent from |
| mydestination = $myhostname, localhost .$mydomain, localhost, $mydomain | Lists the domain that this machine is a destination for |
| inet_interfaces = all | Tells Postfix to listen on all interfaces |
| mynetworks = 192.168.122.0/24, 127.0.0.0/8 | Lists the IP range of trusted SMTP clients |

**TABLE 13-6** Postfix Configuration as a Null Client

| Postfix Parameter | Description |
|---|---|
| myhostname = server1.example.com | Specifies the hostname of the system |
| mydomain = example.com | Sets the local Internet domain name |
| myorigin = server1.example.com | Tells Postfix that e-mails must appear to be sent from the server1.example.com domain |
| mydestination = | Configures the system as a null client (in other words, as a machine that is not a destination for any domain) |
| local_transport = error: local mail delivery is disabled | Disables e-mail delivery to the local system |
| inet_interfaces = localhost | Directs Postfix to listen only to the localhost interface |
| relayhost = maui.example.com | Forwards all e-mails to the host maui.example.com |
| mynetworks = 127.0.0.0/8 | Lists the IP range of the trusted SMTP clients |

functionality and works as a regular SMTP server, except for the forwarding of all e-mail through a second SMTP server. The location of the smart host can be specified with the **relayhost** directive. For example, if the remote smart host is outsider1.example.org, you'd add the following directive to the /etc/postfix/main.cf file:

```
relayhost = outsider1.example.org
```

A null client configuration is even more restrictive than a smart host. Similarly to a smart host configuration, all e-mails are forwarded to a central mail server. In addition, no e-mail message is accepted for local delivery. The corresponding configuration settings are shown in Table 13-6.

## CERTIFICATION OBJECTIVE 13.05

# iSCSI Targets and Initiators

The relevant RHCE objective of this section is to "configure a system as either an iSCSI target or initiator that persistently mounts an iSCSI target." The iSCSI initiator is a client. The iSCSI target is the shared storage on the server, which communicates with the client over TCP port 3260.

The iSCSI protocol encapsulates and delivers SCSI commands over an IP network. Once the server and the client are configured, you'll have access to the storage LUN on the iSCSI target; that LUN will look like just another SCSI hard drive on the client.

## Set Up an iSCSI Target

Today, many storage arrays support the iSCSI protocol. However, for the purpose of the RHCE exam, you need to learn how to configure a Linux server as an iSCSI target (that is, an iSCSI storage server). Of course, the latency and response time will probably be slower than on an enterprise-class iSCSI storage array, but that depends on many factors, including the type of disks and throughput of the network.

on the job

**In a production iSCSI deployment, you may consider enabling "jumbo frames" on all targets, initiators, and Ethernet switches in the iSCSI fabric. Jumbo frames are Ethernet frames with a larger MTU size (typically 9,000 bytes), and as such they usually provide better throughput than the default 1,500 bytes MTU. To enable Jumbo frames on a network card on RHEL 7, add the** MTU=9000 **directive to the corresponding ifcfg-* configuration file in the /etc/sysconfig/network-scripts directory.**

One way you can set up an iSCSI target is with the targetcli package. Install it as shown:

```
# yum install targetcli
```

The package includes the **targetcli** command, which starts a user-friendly configuration shell that guides you through all the steps to deploy an iSCSI target. After starting the **targetcli** shell, type the **ls** command. You will see the output displayed in Figure 13-7.

From the **targetcli** shell, you can navigate to different configuration sections using the **cd** command, like in a filesystem. The **ls** command displays the content of the current section,

**FIGURE 13-7**

The targetcli administrative shell

```
[root@server1 ~]# targetcli
targetcli shell version 2.1.fb34
Copyright 2011-2013 by Datera, Inc and others.
For help on commands, type 'help'.

/> ls
o- / ..................................................................... [...]
  o- backstores ........................................................... [...]
  | o- block ..................................... [Storage Objects: 0]
  | o- fileio .................................... [Storage Objects: 0]
  | o- pscsi ..................................... [Storage Objects: 0]
  | o- ramdisk ................................... [Storage Objects: 0]
  o- iscsi ..................................................... [Targets: 0]
  o- loopback .................................................. [Targets: 0]
/>
```

while **help** provides a useful contextual help screen. Like in a normal shell, you can use TAB completion to fill partially typed commands or arguments.

## Configure a Backstore

As suggested by Figure 13-7, the first step consists of configuring a backstore—that is, a backing store device that will be later exported to iSCSI clients. If you have set up the virtual machines as suggested in Chapter 1, you should have enough free space on the local drive to create a new logical volume to be dedicated to an iSCSI backstore area.

As an example, log in to server1.example.com and create a new logical volume that's 1GB in size on the default rhel_server1 volume group that was created during the operating system installation (substitute accordingly if the name of your volume group is different):

```
# lvcreate -L 1G -n backstore rhel_server1
  Logical volume "backstore" created
```

For the backstore device, you can use any block device, such as a logical volume, a disk partition, or even an entire disk drive. But there's more. As shown in Figure 13-7, **targetcli** not only supports block devices as backing storage, but also image files (**fileio**), physical SCSI disks in pass-through mode (**pscsi**), and temporary in-memory filesystems (**ramdisk**). For the purpose of this section, we'll be using a block device.

Once you have a block device ready to be configured, go back to the **targetcli** shell and create a block storage object:

```
/> cd backstores/block
/backstores/block> create disk1 /dev/rhel_server1/backstore
Created block storage object device1 using /dev/rhel_server1/backstore
```

This **create** command tells **targetcli** to use the /dev/rhel_server1/backstore volume as a block storage object, with a name of disk1.

## Set Up an iSCSI Qualified Name

From the **targetcli** shell, navigate to the /iscsi path:

```
/backstores/block> cd /iscsi
/iscsi>
```

Type the **help** command to display a list of available options. The next step consists of creating an iSCSI Qualified Name (IQN). This is a unique string that identifies an iSCSI initiator or target, for example:

```
iqn.2015-01.com.example:server1-disk1
```

The IQN must follow a specific format. It must start with the "iqn." string, followed by the year and month (YYYY-MM) in which an organization registered its public domain.

Next, there is the domain of the organization in reverse order, followed by a column-separated optional string.

Back to the **targetcli** shell, let's create the IQN:

```
/iscsi> create iqn.2015-01.com.example:server1-disk1
Created target iqn.2015-01.com.example:server1-disk1.
Created TPG 1.
/iscsi>
```

As indicated by the output of the preceding command, **targetcli** created the IQN for the target and a new entity, TPG 1. A TPG is a target portal group. Its purpose is to link together several parts of the configuration, as you will see in the next section.

## Configure a Target Portal Group

If you have completed the configuration steps illustrated so far, type the **ls** command from the **targetcli** shell. You will see the output illustrated in Figure 13-8.

As you can see in the figure, the **targetcli** shell includes new menu entries underneath the IQN and TPG lines. Following the order shown in Figure 13-8, we will perform the following steps:

1. We will configure an access control list (ACL) to allow access to the target only from a specific client.
2. We will create a logical unit number (LUN) for the current backstore device.
3. We will define a portal—that is, an IP address and optionally a custom port that the iSCSI target will listen on for connections.

To configure an ACL and restrict access to the IQN of a specific iSCSI initiator, type the following commands:

```
/iscsi> cd iqn.2015-01.com.example:server1-disk1/tpg1/acls
/iscsi/iqn.20...sk1/tpg1/acls> create iqn.2015-01.com.example:tester1
Created Node ACL for iqn.2015-01.com.example:tester1
/iscsi/iqn.20...sk1/tpg1/acls>
```

**FIGURE 13-8**

Configuring a TPG from the targetcli shell

```
/iscsi> ls
o- iscsi ..................................................... [Targets: 1]
  o- iqn.2015-01.com.example:server1-disk1 ..................... [TPGs: 1]
    o- tpg1 ..................................... [no-gen-acls, no-auth]
      o- acls ..................................................... [ACLs: 0]
      o- luns ..................................................... [LUNs: 0]
      o- portals ................................................. [Portals: 0]
/iscsi>
```

Next, navigate to the LUNs section and associate a LUN number to the backstore device previously created:

```
/iscsi/iqn.20...sk1/tpg1/acls> cd ../luns
/iscsi/iqn.20...sk1/tpg1/luns> create /backstores/block/disk1 0
Created LUN 0.
Created LUN 0->0 mapping in node ACL iqn.2015-01.com.example:tester1
/iscsi/iqn.20...sk1/tpg1/luns>
```

Finally, navigate to the portal section and create a new iSCSI portal to listen to the local IP address (192.168.122.50 in this example). If you don't specify a TCP port, by default **targetcli** will be using TCP port 3260:

```
/iscsi/iqn.20...sk1/tpg1/luns> cd ../portals
/iscsi/iqn.20...tpg1/portals> create 192.168.122.50
Using default IP port 3260
Created network portal 192.168.122.50:3260
/iscsi/iqn.20...tpg1/portals>
```

This completes the configuration of the TPG. Type **ls /** to show the full configuration of the iSCSI target, as illustrated in Figure 13-9. Then, type **exit** to close the **targetcli** shell. The configuration is saved automatically.

All system services must be configured to start at boot. There's no exception for the iSCSI target service, so you must ensure that it is configured to start the next time that the machine is powered on. This can be done by running the following commands:

```
# systemctl enable target
# systemctl start target
```

**FIGURE 13-9**

The configuration of an iSCSI target

```
/iscsi/iqn.20.../tpg1/portals> ls /
o- / ..................................................................... [...]
  o- backstores ........................................................... [...]
  | o- block ............................................ [Storage Objects: 1]
  | | o- disk1 ..... [/dev/rhel_server1/backstore (1.0GiB) write-thru activated]
  | o- fileio ........................................... [Storage Objects: 0]
  | o- pscsi ............................................ [Storage Objects: 0]
  | o- ramdisk .......................................... [Storage Objects: 0]
  o- iscsi ......................................................... [Targets: 1]
  | o- iqn.2015-01.com.example:server1-disk1 ......................... [TPGs: 1]
  |   o- tpg1 ........................................... [no-gen-acls, no-auth]
  |     o- acls ..................................................... [ACLs: 1]
  |     | o- iqn.2015-01.com.example:tester1 ................... [Mapped LUNs: 1]
  |     |   o- mapped_lun0 ............................ [lun0 block/disk1 (rw)]
  |     o- luns ..................................................... [LUNs: 1]
  |     | o- lun0 ................... [block/disk1 (/dev/rhel_server1/backstore)]
  |     o- portals ................................................. [Portals: 1]
  |       o- 192.168.122.50:3260 ...................................... [OK]
  o- loopback ...................................................... [Targets: 0]
/iscsi/iqn.20.../tpg1/portals>
```

Don't forget to allow connections through the local firewall. By default, the iSCSI target service uses TCP port 3260:

```
# firewall-cmd --permanent --add-port=3260/tcp
# firewall-cmd --reload
```

## Connect to Remote iSCSI Storage

In this section we will configure the tester1.example.com virtual machine to mount the LUN exported by the iSCSI target defined in the previous section. To set up an iSCSI client, you'll need the iscsi-initiator-utils packages, along with any dependencies:

```
# yum install iscsi-initiator-utils
```


**The iscsi-initiator-utils package implements a software iSCSI initiator. However, today most network adapters provide iSCSI initiator functionalities in hardware. The configuration of a hardware iSCSI initiator depends on the manufacturer of the card, and as such it varies between different card vendors and models.**

Next, configure an IQN for the initiator. This is defined in the /etc/iscsi/initiatorname .iscsi file. Edit the content and type a custom IQN:

```
InitiatorName=iqn.2015-01.com.example:tester1
```

If you have configured an ACL on the iSCSI target, the IQN of the client must match the IQN defined on the ACL; otherwise, the client will not be granted access to the target.

Next, enable the iscsi service on the client and ensure that it starts at the next boot:

```
# systemctl start iscsi
# systemctl enable iscsi
```

You'd use the **iscsiadm** utility to discover available iSCSI targets. One method is with the following command:

```
# iscsiadm -m discoverydb -t st -p 192.168.122.50 -D
```

To interpret, this **iscsiadm** command queries iSCSI targets. It works in discovery database (**discoverydb**) mode (**-m**), where the discovery type (**-t**) requests that the **sendtargets** (or **st**) command is sent to the iSCSI target, defined on a portal (**-p**) listening on the noted IP address, to discover (**-D**) shared storage LUNs.

If successful, you'll see output similar to the following:

```
192.168.122.50:3260,1 iqn.2015-01.com.example:server1-disk1
```

To use the target you have just discovered, you need to run the following command:

```
# iscsiadm -m node -T iqn.2015-01.com.example:server1-disk1 -l
```

**exam Watch**

**To set up an iSCSI client, you'll need access to an iSCSI target acting as a server. Normally, TCP port 3260 should be open on that server, making it accessible with the** iscsiadm **command described in this chapter.**

This command works in node mode (**-m**) to log in to (**-l**) the target IQN (**-T**) iqn.2015-01 .com.example:server1-disk1. If successful, you'll be able to see an additional disk storage device by running the **fdisk -l** command

You should then be able to manage the shared storage as if it were a new hard drive on the local system. The hard drive device file will show up in the /var/log/messages file with information such as the following, which points to device file /dev/sdc:

```
Sep 25 20:22:15 tester1 kernel sd 6:0:0:0: [sdc] Attached SCSI disk
```

Then, you can create partitions and more on the new /dev/sdc drive, just as if it were a local drive, based on the techniques discussed in Chapter 6. Of course, a "persistent mount" as described in the relevant RHCE objective requires that you make sure the iSCSI service starts the next time the system is rebooted.

To make sure there's an actual mount, you may also need to set up a partition that's actually mounted in the /etc/fstab file. In practice, the actual device file for the iSCSI drive may vary on each reboot. Therefore, such mounts should be configured with the Universally Unique Identifier (UUID) numbers described in Chapter 6.

## CERTIFICATION OBJECTIVE 13.06

# The Network Time Service

The configuration of NTP as a client and a default server is covered in Chapter 5. In contrast, the configuration of NTP as a server that synchronizes time using NTP peers is covered here. Nevertheless, you need to know how to secure NTP just as you secure other network services such as Samba and NFS.

To allow NTP to work as a server, you need to allow access through UDP port 123. This can be achieved by adding the ntp service to the relevant firewalld zone.

## The NTP Server Configuration File

As discussed in Chapter 5, the time configuration depends on the time zone that the /etc/localtime symbolic link points to, as well as the NTP servers configured in the /etc/ntp.conf file (or /etc/chrony.conf file, if chronyd is in use). Now it's time to configure one of those NTP servers. We'll focus on ntpd, as this is the standard NTP daemon for systems that are always connected to the network.

The default /etc/ntp.conf configuration file starts with the **driftfile** directive, which monitors the error drift in the local system clock:

```
driftfile /var/lib/ntp/drift
```

There are also **restrict** directives that can help protect an NTP server. This directive works with IPv4 and with IPv6 networking, as shown here:

```
restrict default nomodify notrap nopeer noquery
restrict 127.0.0.1
restrict ::1
```

The options to the **restrict** directive can be described as follows:

- **default** Refers to default connections from other systems; may be further limited by other **restrict** directives.
- **nomodify** Denies queries that attempt to change the local NTP server configuration.
- **notrap** Denies the control message trap service; you might remove this option to enable remote logging.
- **nopeer** Stops access from potential peer NTP servers.
- **noquery** Ignores queries.

However, these restrictions, when combined, are good only for an NTP client. To set up an NTP server, specifically one that "synchronizes time using other NTP peers," you should remove at least the **nopeer** directive from the **restrict** list. Some NTP servers may need to synchronize with yours, which is possible if you remove the **noquery** from the list as well.

The next two restrict directives limit access to the local NTP server to the local system. You should recognize the default IPv4 and IPv6 loopback addresses here:

```
restrict 127.0.0.1
restrict ::1
```

Of course, when setting up an NTP server for other clients, you'll want to loosen that restriction. The comment that follows includes a network address in the required format. So to set up an NTP server for the 192.168.122.0/24 network, you'd change the **restrict** directive to

```
restrict 192.168.122.0 mask 255.255.255.0 notrap nomodify
```

For a basic "default configuration NTP server," no additional changes should be required. Of course, the local NTP server should also be configured as a client to master NTP servers. Substitute the hostname for the actual NTP servers on your network for the following:

```
server 0.rhel.pool.ntp.org iburst
server 1.rhel.pool.ntp.org iburst
server 2.rhel.pool.ntp.org iburst
server 3.rhel.pool.ntp.org iburst
```

And to repeat the reference from the RHCE objectives, the other reference is to peers. The relevant directive is **peer**.

To test the directive on one NTP server, you could set this machine to **peer** with another host, as shown here:

```
peer server1.example.com
```

Alternatively, you could be given the hostname to an NTP peer server, perhaps on a corporate network, perhaps on a network that has been configured during the exam.

## Security Limits on NTP

As just described, the **restrict** directive from the /etc/ntp.conf configuration file can be used to limit access to a local NTP server, but that assumes an open port 123. Security limits can also refer to configured firewalls. Just be aware that an appropriate firewall rule for NTP opens up UDP (not TCP) port 123. This can be configured by adding the ntp service to the appropriate firewall zone, like so:

```
# firewall-cmd --permanent --add-service=ntp
# firewall-cmd --reload
```

To test a connection to an NTP server, the **ntpq -p *hostname*** command can help. That command looks for the peers listed in the /etc/ntp.conf file. If the server is operational, you'll see something similar to the following output to the **ntpq -p localhost** command shown in Figure 13-10.

The * sign in front of a hostname or IP address indicates the current NTP peer or server is used as a primary reference, whereas the + sign identifies additional peers designated as acceptable for synchronization. Of course, if the **ntpq** command works from a remote system, using the local hostname or IP address, you've verified that the remote NTP server is operational.

**exam Watch**

**Since NTP is a UDP-based service, the** telnet **command won't verify the operation of that service. You can check the local status of the NTP service with the** ntpq -p **and** nmap -sU -p123 **commands.**

**FIGURE 13-10**

NTP server status, verified with the ntpq -p command

```
[root@server1 ~]# ntpq -p
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
+time2.mediainve 131.188.3.220    2 u   21   64    3   26.719    1.814  12.969
+stz-bg.com      192.53.103.104   2 u   19   64    3   54.418    6.646  28.389
*ntp2.litnet.lt  .GPS.            1 u   18   64    3   48.583    2.647  22.977
+betelgeuse.retr 193.62.22.90     2 u   17   64    3    2.299    0.548  25.659
[root@server1 ~]#
```

| SCENARIO & SOLUTION | |
|---|---|
| You need to configure a caching-only DNS server for the local network. | Use the default named.conf file; modify the **listen-on** and **allow-query** directives. |
| You need to configure a caching-only DNS server to forward requests elsewhere. | Use the named.conf file, configured as a caching-only name server; add a **forwarders** directive to point to a desired DNS server. |
| You're told to configure an SMTP server for the 192.168.0.0/24 network. | Use the default Postfix server; modify the **myhostname**, **mydomain**, **myorigin**, **mydestination**, **inet_interfaces**, and **mynetworks** directives in /etc/postfix/main.cf. |
| You're asked to configure Postfix as a null client. | Set **relayhost** to the remote server to forward e-mails to; modify **mydestination** and **local_transport**; and limit access to the server with the **inet_interfaces** and **mynetworks** directives. |
| You're told to allow access just to the SMTP server for user1, user2, and user3. | Create the noted users with a /sbin/nologin default shell. |
| You need to set up an NTP server as a peer. | Modify the ntp.conf file to specify the host or IP address of a peer NTP server with the **peer** directive. Add a **restrict** directive without the **nopeer** option to allow access to the desired host. |

## CERTIFICATION SUMMARY

DNS provides a database of domain names and IP addresses that help hosts translate hostnames to IP addresses on various networks, including the Internet. It's a distributed database where each administrator is responsible for his or her own zone of authority, such as mheducation.com. The default DNS server uses the **named** daemon, based on the Berkeley Internet Name Domain (BIND). Other alternatives, such as the Unbound resolver, are also available.

There are four basic types of DNS servers: master, slave (or secondary), caching-only, and forwarding-only. The RHCE objectives specifically exclude master and slave name servers. The default /etc/named.conf file is built for a caching-only DNS server configuration. A forwarding-only name server uses the **forward only** and **forwarders** directives in the named.conf file. In either case, you should configure the **listen-on** and **allow-query** directives to support access from the local system and desired network(s). To test a DNS server, use commands such as **rndc status**, **dig**, and **host**.

Red Hat includes two servers associated with the SMTP protocol: Postfix and sendmail. Postfix is the default and is somewhat easier to configure than sendmail. Different Postfix

configuration files can be found in the /etc/postfix directory. User and host limits can be configured in the access file. Several other files relate to redirected or renamed e-mail accounts or domains. You'll need to modify Postfix configuration directives in the /etc/postfix/main.cf file, including **myhostname**, **mydomain**, **myorigin**, **mydestination**, **inet_interfaces**, and **mynetworks**. The **relayhost** directive can help configure forwarding to a smart host. If you need to configure a null client, you also need to set the **local_transport** directive to avoid e-mails being delivered to the local system.

The iSCSI protocol emulates a SCSI bus over an IP network. With the **targetcli** command shell, you can configure interactively a Linux host as an iSCSI target to export local storage to remote iSCSI initiators. The iSCSI initiators can discover and log in to remote targets using the **iscsiadm** command.

Finally, to configure an NTP server for a network, you need to modify the /etc/ntp.conf file. The **restrict** directive should be changed to specify the network address. To support the peers suggested in the RHCE objectives, you also need a **restrict** directive without the **noquery** and (most important) **nopeer** options. Then to set up other systems as peers, you'd use the **peer *hostname*** format.

# TWO-MINUTE DRILL

Here are some of the key points from the certification objectives in Chapter 13.

## An Introduction to Domain Name Services

- ❑ DNS is based on the Berkeley Internet Name Domain (BIND), using the **named** daemon.
- ❑ Key packages include bind-chroot, which adds security by supporting DNS in a chroot jail, and bind-utils, which includes command utilities such as **dig** and **host**.
- ❑ Four basic types of DNS servers are master, slave (secondary), caching-only, and forwarding-only. The RHCE objectives specifically require coverage of just caching-only DNS services.

## Minimal DNS Server Configurations

- ❑ Critical BIND configuration files include /etc/named.conf and the files in the /var/named directory.
- ❑ The default /etc/named.conf file is set up for a caching-only name server, limited to the local system. Changes to the **listen-on** and **allow-query** directives can enable access from DNS clients on a network.
- ❑ A forwarding name server requires a **forward only** and **forwarders** directive that specifies the IP addresses of the remote DNS servers.
- ❑ Unbound provides an alternative to BIND for setting up a secure caching-only and forwarding name server.

## A Variety of E-Mail Agents

- ❑ Postfix is the default mail transfer agent in RHEL 7.
- ❑ Mail server information is logged in the /var/log/maillog file.

## The Configuration of Postfix

- ❑ The Postfix server can be configured through configuration files in the /etc/postfix directory. The main configuration file is main.cf file.
- ❑ You can configure e-mail aliases in /etc/aliases.
- ❑ You can set up various kinds of e-mail forwarding in files such as canonical, generic, and relocated, all in the /etc/postfix directory.
- ❑ The **relayhost** directive can be used to set up a connection to a smart host.
- ❑ To set up Postfix as a null client, you need to prevent e-mails from being delivered to the local system with the **local_transport** directive.
- ❑ You can test a standard Postfix configuration from the local system with the **telnet localhost 25** command.

## iSCSI Targets and Initiators

- ❑ You can configure an iSCSI target by activating the target service and running the **targetcli** administrative shell.
- ❑ To set up an iSCSI target, you need to define a backstore device, set an IQN, define a LUN, create a portal, and (optionally) define ACLs.
- ❑ To configure an iSCSI client, you need the iscsi-initiator-utils package, which can be used to discover and log in to iSCSI targets with the **iscsiadm** command.
- ❑ To make sure the iSCSI connection survives a reboot, you'll need to activate the iscsi service.

## The Network Time Service

- ❑ The default NTP configuration file, /etc/ntp.conf, sets up a client with access limited to the local system.
- ❑ A standard **restrict** directive in the default ntp.conf file is available to open access to systems on a specified network. You'll also need to allow NTP traffic through the local firewall.
- ❑ The RHCE objectives suggest connections to peers; such connections can be configured with the **peer** directive.

# SELF TEST

The following questions will help measure your understanding of the material presented in this chapter. As no multiple choice questions appear on the Red Hat exams, no multiple choice questions appear in this book. These questions exclusively test your understanding of the chapter. It is okay if you have another way of performing a task. Getting results, not memorizing trivia, is what counts on the Red Hat exams. There may be more than one answer to many of the questions.

### An Introduction to Domain Name Services

1. Name two packages that provide DNS services on RHEL 7.

   _____

## Minimal DNS Server Configurations

2. To configure DNS communication on port 53, what changes would you make to a firewall to support access by other clients to the local DNS server?

   _____

3. What file includes a basic template for a BIND DNS caching-only name server?

   _____

4. What command makes sure that the BIND DNS service starts the next time you boot Linux in the default target?

   _____

### A Variety of E-Mail Agents

5. List two examples of an MTA available on RHEL 7.

   _____

6. What command can be used to switch between the installed Postfix and sendmail services?

   _____

### The Configuration of Postfix

7. How would you change the following directive in /etc/postfix/main.cf to open Postfix to all systems?

```
inet_interfaces = localhost
```

_____

8. If you use /etc/aliases for forwarding e-mail, what command processes these files into an appropriate database file for Postfix?

_____

9. What directive in the main.cf file is used to specify the domain served by the Postfix server?

_____

### iSCSI Targets and Initiators

10. What service should be running on reboot on a properly configured iSCSI target?

_____

11. What command utility can be used to configure an iSCSI target?

_____

### The Network Time Service

12. Enter a directive, suitable for /etc/ntp.conf, that limits access to the 192.168.0.0/24 network.

_____

## LAB QUESTIONS

Several of these labs involve configuration exercises. You should do these exercises on test machines only. It's assumed that you're running these exercises on KVM-based virtual machines. For this chapter, it's also assumed that you may be changing the configuration of a physical host system for such virtual machines.

Red Hat presents its exams electronically. For that reason, the labs in this and future chapters are available from the media that accompanies the book. Labs for this chapter are in the Chapter13/ subdirectory. In case you haven't yet set up RHEL 7 on a system, refer to Chapter 1 for installation instructions.

The answers for each lab follow the Self Test answers for the fill-in-the-blank questions.

# A SELF TEST ANSWERS

## An Introduction to Domain Name Services

1. The BIND and Unbound software packages provide DNS services on RHEL 7.

## Minimal DNS Server Configurations

2. To support access by other clients to the local DNS server, make sure TCP and UDP traffic is supported through the firewall on port 53 by enabling the dns service on the required firewalld zone.
3. The default /etc/named.conf file includes a basic template for a DNS caching name server.
4. The command that makes sure that the BIND DNS service starts the next time you boot Linux is

```
# systemctl enable named
```

## A Variety of E-Mail Agents

5. Two examples of MTAs supported on RHEL 7 are Postfix and sendmail.
6. The command that can be used to help switch between the Postfix and sendmail MTAs is **alternatives --config mta**.

## The Configuration of Postfix

7. The simplest solution is to change the directive to

```
inet_interfaces = all
```

8. Forwarding e-mail addresses are normally stored in /etc/aliases. Make sure to process these files into appropriate databases; for /etc/aliases, the database is updated with the **newaliases** command.
9. The directive in the main.cf file that is used to specify the domain served by the Postfix server is **mydestination**.

### iSCSI Targets and Initiators

10. The target service should be running on reboot on a properly configured iSCSI target.
11. The **targetcli** command shell can be used to configure an iSCSI target.

### The Network Time Service

12. One directive in the /etc/ntp.conf file that limits access based on the noted conditions is

```
restrict 192.168.122.0 mask 255.255.255.0
```

# LAB ANSWERS

## Lab 1

In this lab, you can use the existing configuration in the /etc/named.conf configuration. All you need to do is follow these steps:

1. Install the bind RPM package.
2. Modify the **listen-on port 53** directive to include the local IP address; for example, if the local IP address is 192.168.122.150, the directive will look like

```
listen-on port 53 { 127.0.0.1; 192.168.122.150; };
```

3. Modify the **allow-query** directive to include the local IP network address:

```
allow-query { localhost; 192.168.122.0/24; };
```

4. Save your changes to /etc/named.conf.
5. Start the **named** service:

```
# systemctl start named
```

6. Change the local client to point to the local DNS caching name server; replace any **nameserver** directives in /etc/resolv.conf with the IP address of the local system. For example, if the local computer is on 192.168.122.150, the directive is

```
nameserver 192.168.122.50
```

7. Test out the new local DNS server. Try commands such as **dig www.mheducation.com**.
8. Point client systems to the DNS server. Add the aforementioned **nameserver** directive to /etc/resolv.conf on those remote client systems:

```
nameserver 192.168.122.50
```

9. To make sure the DNS service starts the next time Linux is booted, run the following command:

```
# systemctl enable named
```

10. Open up TCP and UDP ports 53 in the firewall on the local system. The simplest method is with the firewall-cmd utility:

```
# firewall-cmd --permanent --add-service=dns
# firewall-cmd --reload
```

## Lab 2

As with Lab 1, the focus in this lab is on the configuration of the /etc/named.conf file. Nominally, the default caching-only DNS server already includes forwarding features. However, to set up a specific forwarding server, you should add a **forwarders** entry such as the following in the **options** stanza:

```
forwarders { 192.168.122.1; };
```

As for the other requirements of the lab, you can clear the current cache with the **rndc flush** command and reload the configuration file with the **rndc reload** command.

## Lab 3

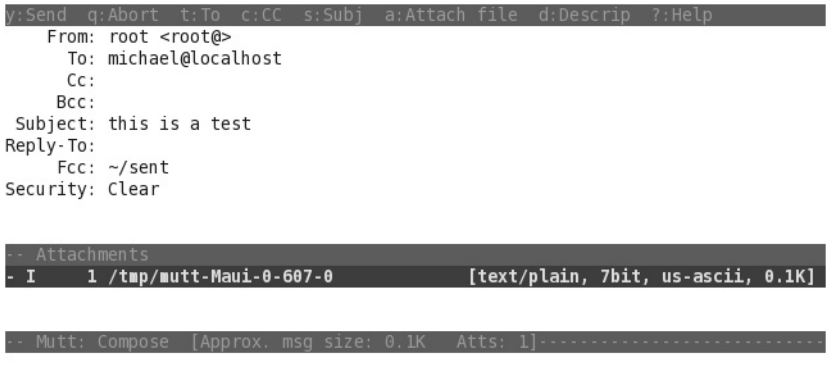
For Labs 3, 4, and 5, you may be using an e-mail client such as **mutt**. To send an e-mail to user michael@localhost, take the following steps:

1. Run the **mutt michael@localhost** command. The **To: michael@localhost** message should appear.
2. Press ENTER. At the **Subject:** prompt, enter an appropriate test subject name and press ENTER.
3. You're taken to a blank screen in the vi editor. Use commands appropriate to that editor in a screen similar to that shown in Figure 13-11.
4. From the screen shown in Figure 13-11, press Y to send the noted message.

In addition, you may be verifying e-mail receipt in a username file in the /var/spool/mail directory. Normally, such e-mail can be reviewed from within a user account with the **mail** or **mutt** command.

**FIGURE 13-11**

The mutt e-mail client



In Postfix, to disable local-only access in the /etc/postfix/main.cf file, change the **inet_interfaces** directive to accept all connections:

```
inet_interfaces = all
```

However, to meet the requirements of the lab, you'll want to retain the default value of that directive:

```
inet_interfaces = localhost
```

In general, to verify authentication on an SMTP server, connect from the local system with the **telnet localhost 25** command. When you see a message similar to

```
220 maui.example.com ESMTP Postfix
```

type in the following command:

```
EHLO localhost
```

To verify receipt of e-mail in a user account, log in to that account, or at least verify the timestamp associated with the username in the /var/mail directory. To make sure e-mail directed to the root user is redirected to a regular user account, you'd add a line like the following to the /etc/aliases file:

```
root:  michael
```

Given the wording for the question, any standard user account would be acceptable. Of course, to implement this change, you'll have to run the **newaliases** command, which processes this file into the /etc/aliases.db file.

## Lab 4

To enable access from more than just the localhost, you'll need to modify the inet_interfaces directive in /etc/postfix/main.cf to

```
inet_interfaces = all
```

The next job is to limit access to a specific network (in this case, example.com). While there are options in /etc/postfix files, perhaps the most efficient way to limit access to a specific network is with an appropriate firewalld rich rule. For example, the following custom rule would limit access to TCP port 25 to systems on the given IP address network. The network shown is based on the originally defined configuration for example.com, the 192.168.122.0/24 network:

```
# firewall-cmd --permanent --add-rich-rule='rule family=ipv4↵
 source address=192.168.122.0/24 service name=smtp accept'
```

In addition, you can set up this network in the /etc/postfix/access file with a rule like the following:

```
192.168.122 OK
```

Once Postfix is running, you should be able to confirm the result with an appropriate **telnet** command from a remote system. For example, if Postfix is configured on a system with a 192.168.122.50 IP address, the command would be

```
# telnet 192.168.122.50 25
```

The configuration of a smart host in Postfix is based on the **relayhost** directive. For the parameters given in the lab, if the physical host is located on the system maui.example.com, the directive in the main.cf file would be

```
relayhost = maui.example.com
```

If Postfix on the server1.example.com system is properly configured as a smart host, e-mails to the forwarded host should be reliably delivered and logged in to the appropriate /var/log/maillog file.

## Lab 5

The configuration of Postfix as a null client is straightforward and is summarized in Table 13-6.

At a minimum, you should configure the **myorigin**, **mydestination**, **local_transport**, and **relayhost** directives. Other directives such as **myhostname**, **mydomain**, and **mynetworks** should already have appropriate default values.

Confirm your settings with the **postconf -n** command and start the service. Don't forget to configure Postfix to start automatically the next time the machine is powered on.

## Lab 6

This is a long lab, but it is very similar to the configuration example described in the "iSCSI Targets and Initiators" section of this chapter. Please refer to that section for an in-depth discussion.

## Lab 7

In this lab, you'll set up one NTP server as a peer to another. That's possible with the **peer** directive, configured in the /etc/ntp.conf configuration file. For example, if a regular NTP server is configured on IP address 192.168.122.50, you can set up a peer on the 192.168.122.150 server with the following directive:

```
peer 192.168.122.50
```

Just remember, an NTP peer doesn't work unless the **nopeer** option has been removed from the **restrict** directive in the ntp.conf file.