Chapter 10

A Security Primer

CERTIFICATION OBJECTIVES

10.01	The Layers of Linux Security	10.04	Pluggable Authentication Modules
10.02	Firewalls and Network Address Translation	10.05	Secure Files and More with GPG2
10.03	TCP Wrappers	✓	Two-Minute Drill
		Q&A	Self Test

As you start the first chapter of the RHCE section of this book, you'll start with security. Many administrators and enterprises move toward Linux because they believe it's more secure. Since most Linux software is released under open-source licenses, the source code is available to all. Some believe that provides advantages for black hat hackers who want to break into a system.

However, Linux developers are believers in collaboration. "Linus's Law," according to the open-source luminary Eric Raymond, is that "given enough eyeballs, all bugs are shallow." Some of those eyes are from the U.S. National Security Agency (NSA), which has contributed a lot of code to Linux, including the foundations of SELinux.

The NSA has also contributed a number of other concepts adapted by Red Hat that have been integrated into a layered security strategy. These include guidelines to set up firewalls, wrappers on packets, and security by service. They cover both user- and host-based security. They include access controls such as ownership, permissions, and SELinux. (A number of these layers were covered in earlier chapters.) The fundamentals of these layers of security, as they apply to the RHCE objectives, are also covered here.

In this chapter, you'll examine some of the tools provided by RHEL for managing security. You'll start with some fundamentals and continue with detailed analysis of zone-based firewalls, Pluggable Authentication Modules (PAM), TCP Wrappers, and more.

This is not the only chapter to focus on security. Strictly speaking, it covers only two of the RHCE objectives. However, this chapter covers the themes associated with security on Linux systems, and those themes can help you understand the security options associated with every service in this book.

CERTIFICATION OBJECTIVE 10.01

The Layers of Linux Security

The best computer security comes in layers. If there's a breach in one layer, such as penetration through a firewall, a compromised user account, or a buffer overflow that messes up a service, there's almost always some other security measure that prevents or at least minimizes further damage.

INSIDE THE EXAM

Inside the Exam

This chapter is the first one in this book focused on the RHCE requirements. As described in the RHCE objectives, security starts with packet filtering and NAT developed with the firewalld zone-based firewall. The related objective is

- Use firewalld and associated mechanisms, such as rich rules, zones, and custom rules, to implement packet filtering and configure Network Address Translation (NAT)

But as suggested in the introduction, security is an issue for all services covered in the RHCE objectives. This chapter provides a foundation for a discussion of security, including several methods to

- Configure host-based and user-based security for the service

Whereas host-based security can start with zone-based firewalls, host- and user-based security measures can involve TCP Wrappers and Pluggable Authentication Modules.

These options start with minimally configured bastion hosts, which minimize the functionality associated with an individual Linux system. Beyond the firewall and SELinux come security options associated with individual services. Isolation options such as chroot jails are generally configured as part of a service. A number of these options are based on recommendations from the NSA.

While the sections on bastion systems are intended to be a lead-in to the security measures used for RHCE-level services, they also incorporate those security options often associated with the RHCSA exam, which are described in earlier chapters.

Bastion Systems

Properly configured, a bastion system minimizes the risk of a security breach. It's based on a minimal installation, with less software than was installed on the systems configured in Chapters 1 and 2. A bastion system is configured with two services. One service defines the functionality of the system. It could be a web server, a file server, an authentication server, or something similar. The other service supports remote access, such as SSH, or perhaps VNC over SSH.

Before virtualization, the use of bastion systems was frequently limited. Only the wealthiest enterprises could afford to dedicate different physical systems to each service. If redundancy was required, the costs only increased further.

With virtualization, bastion systems are within reach of even smaller businesses. All that's needed is a standard minimal installation. With a few Kickstart files, you as an administrator of such a network could easily create a whole group of bastion systems. Each system could then be customized with and dedicated to a single server.

Well-constructed bastion systems follow two principles:

- If you don't need the software, uninstall it.
- If you need the software but aren't using it, make sure it's not active.

In general, black hat hackers can't take advantage of a security flaw if the associated service isn't installed. If you do have to install the service for test purposes, keep that service inactive. That can help keep risks to a minimum. Of course, firewalls configured for each bastion system should allow traffic through only for the dedicated service and the remote access method.

Best Defenses with Security Updates

Security updates are extremely important. You can review available updates with the Software Update tool. You can start that tool in a GUI with the **gpk-update-viewer** command. As discussed in Chapter 7, you can set up automatic security updates with the Software Updates Preferences tool, which you can start in a GUI with the **gpk-prefs** command.

In practice, security is often a race, between when a vulnerability is discovered and when an update is made available. Until those updates are installed, any affected services might be vulnerable.

As a Linux professional, it's your job to know these vulnerabilities. If you maintain servers such as Apache, vsFTP, and Samba, monitor the information feeds from these developers. Security news may come in various forms, from message board updates to RSS feeds. Normally, Red Hat also keeps up to speed on such issues. However, if you've subscribed to the forums maintained by the developers of a service, it's best to hear about problems and planned solutions directly from the source. To some extent, that is a province of service-specific security.

Information security vulnerabilities are tracked in a standardized format system known as Common Vulnerabilities and Exposures (CVE), which is maintained by MITRE Corporation (http://cve.mitre.org). In its "Errata" security advisories, Red Hat always references the corresponding CVE identifiers. You should familiarize yourself with the CVE format and monitor the Red Hat CVE database and Errata announcement websites for updates, available at https://access.redhat.com/security/cve and https://rhn.redhat.com/errata, respectively.

Service-Specific Security

Most major services have some level of security that can be configured within. In many cases, you can configure a service to limit access by host, by network, by user, and by group. As listed in the RHCE objectives, you need to know how to configure host- and user-based security for each listed service. SELinux options that can help secure each of these services are also available. While details are discussed in appropriate upcoming chapters, the following is a brief overview of service-specific security options.

HTTP/HTTPS Service-Specific Security

Although there are alternatives, the primary service for the HTTP and HTTPS protocols on Linux is the Apache web server. In fact, Apache is the dominant web server on the Internet. No question, Apache configuration files are complex, but they need to be, because the security challenges on the Internet are substantial. Some options for responding to these challenges are covered in Chapter 14.

Apache includes numerous optional software components. Don't install more than is absolutely necessary. If there's a security breach in a Common Gateway Interface (CGI) script and you haven't installed Apache support for CGI scripts, that security issue doesn't affect you. However, because the RHCE specifies an objective to deploy a "basic CGI application," you don't have that luxury for the exam.

Fortunately, with Apache, you can limit access in a number of ways. Limits can be created on the server or on individual virtual hosts. Different limits can be created on regular and secure websites. In addition, Apache supports the use of secure certificates.

DNS Service-Specific Security

Domain Name Service (DNS) servers are a big target for black hat hackers. With that in mind, RHEL 7 includes the bind-chroot package, which configures the necessary files, devices, and libraries in an isolated subdirectory. That subdirectory provides a limit for any user who breaks through DNS security known as a chroot jail. It's designed to limit the directories where a black hat hacker can navigate if he does break into the service. In other words, if someone breaks into a RHEL 7 DNS server, they should not be able to "escape" the subdirectory configured as a chroot jail.

Since RHCE exam candidates are not expected to create a master or a slave DNS server, the challenges and risks are somewhat limited. Nevertheless, in Chapter 13, you'll see how to limit access to the configured DNS server by host.

NFS Service-Specific Security

With the move to the Network File System version 4, it is now possible to set up Kerberos authentication to support user-based security. Although the configuration of Kerberos and LDAP servers is beyond the scope of the RHCE objectives, for the RHCE exam you need to control access to NFS shares using Kerberos. Chapter 16 will cover this topic, along with host-based security options.

SMB Service-Specific Security

The SMB listed in the RHCE objectives stands for the Server Message Block protocol. It's the networking protocol originally developed by IBM, and later modified by Microsoft, as the network protocol for its operating systems. While Microsoft now refers to it as the Common Internet File System (CIFS), the Linux implementation of this networking protocol is still known as Samba.

As implemented for RHEL 7, you can use Samba to authenticate via Microsoft Active Directory. Samba supports the mapping of such users and groups into a Linux authentication database. Samba also supports both user- and host-based security on the global and shared directory levels, as discussed in Chapter 15.

The standard version of Samba for RHEL 7 is 4.1. With the release of version 4, Samba can also act as a Domain Controller compatible with Microsoft Active Directory. However, this configuration is outside of the scope of the RHCE exam.

SMTP Service-Specific Security

RHEL supports two different services for e-mail communication through the Simple Mail Transport Protocol (SMTP): Postfix and Sendmail. Both are released under open-source licenses.

The default SMTP e-mail service for RHEL 7 is Postfix, although you can configure either service to meet the associated RHCE objective. In either case, the service normally only listens on the localhost address, which is one level of security. Other levels of security are possible based on hosts, usernames, and more. For more information, see Chapter 13.

SSH Service-Specific Security

The SSH service is installed by default even in the minimal installation of RHEL 7. That encourages its use as a remote administration tool. However, there are risks associated with the SSH server that can be minimized. For example, remote logins to the root account do not have to be allowed. Security can be further regulated by user.

Host-Based Security

Host-based security refers to access limits, not only by the system hostnames, but also by their fully qualified domain names and IP addresses. The syntax associated with host-based security can vary. For example, while every system recognizes a specific IP address such as 192.168.122.50, not all services recognize wildcards or Classless Inter-Domain Routing (CIDR) notation for a range of IP addresses. Depending on the service, you may use one or more of the following options for the noted range of network addresses:

```
192.168.122.0/255.255.255.0
192.168.122.0/24
192.168.122.*
192.168.122.
192.168.122
```

Just be careful, because some of these options may lead to syntax errors on some network services. In a similar fashion, any of the following options may or may not work to represent all of the systems on an example.com network:

```
*.example.com
.example.com
example.com
```

User-Based Security

User-based security includes users and groups. Generally, users and groups who are allowed or denied access to a service are collected in a list. That list could include a user on each line, as in a file such as /etc/cron.allow, or it could be in a list that follows a directive, such as

```
valid users = michael donna @book
```

Sometimes the syntax of a user list is unforgiving; in some cases, an extra space after a comma or at the end of a line may result in an authentication failure.

Groups are frequently included in a list of users, with a special symbol in front, such as @ or +.

Sometimes, users who are allowed access to a system are configured in a separate authentication database, such as that associated with the Samba server, configured with the **smbpasswd** command.

Console Security

As discussed in Chapter 5, console security is managed by the /etc/securetty file. It can help you regulate local console access to root and regular users.

However, console access is not just local. For a full view of console security, you need to be able to configure limits on remote console access. Two primary options are SSH, as discussed earlier, and Telnet. While the **telnet** command has its uses, as described in Chapter 2, communications to Telnet servers are inherently insecure. Usernames, passwords, and other communication to and from a Telnet server are transmitted in clear text. That means a network protocol analyzer such as Wireshark could be used to read those usernames, passwords, and any other critical information.

Even though Kerberos-based options are available for Telnet servers, most security professionals avoid Telnet for remote consoles at almost all costs—and that's consistent with the recommendations from the NSA.

Recommendations from the U.S. National Security Agency

The NSA has taken a special interest in Linux, and specifically Red Hat Enterprise Linux. Not only has the NSA taken the time to develop SELinux, but it also has created guides to help administrators like you create a more secure RHEL configuration. (Yes, the "super-secret" NSA has released SELinux code under open-source licenses for all to see.) They recognize the importance of Linux in the infrastructure of computer networks. Observers of RHEL may notice how changes between RHEL 5, RHEL 6, and RHEL 7 follow NSA recommendations.

The NSA includes five general principles for securing operating systems in general and RHEL in particular:

- *Encrypt transmitted data whenever possible.* NSA recommendations for encryption include communications over what should be private and secure networks. The use of SSH, with the security options described in Chapter 11, is an excellent step in this process.
- *Minimize software to minimize vulnerability.* As suggested by the NSA, "The simplest way to avoid vulnerabilities in software is to avoid installing that software." The NSA pays special attention to any software that can communicate over a network, including the Linux GUI. The minimal installation of RHEL 7 includes far fewer packages than the comparable installation of RHEL 5.

- *Run different network services on separate systems.* This is consistent with the concept of bastion servers described earlier in this chapter. Implementation is made easier by the flexibility afforded by virtual machine technologies such as KVM.
- *Configure security tools to improve system robustness.* The RHCSA and RHCE objectives have this well covered, with the use of zone-based firewalls, SELinux, and appropriate log collection services.
- *Use the principle of least privilege.* In principle, you should give users the minimum privileges required to accomplish their tasks. Not only does that mean minimize access to the root administrative account, but also careful use of the **sudo** command privileges. SELinux options such as the user_u role for confinement (described in Chapter 4) may also be helpful to that end.

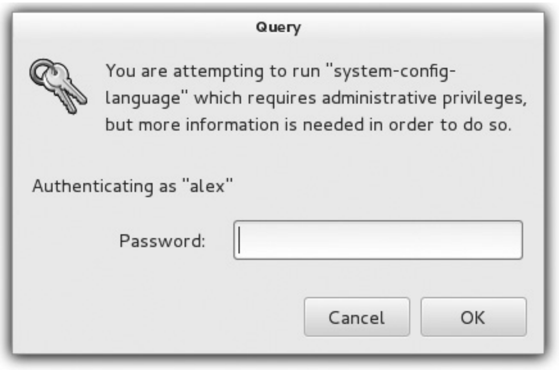
The PolicyKit

The PolicyKit is one more security mechanism designed to help protect different administrative tools. Most administrative tools started in the GUI from a regular account will prompt for the root administrative password with a window similar to the one shown in Figure 10-1.

Alternatively, you might see a slightly different window similar to that shown in Figure 10-2. Functionally, the effect is the same. As described in the window, authentication by the superuser is required. In this case, you'd still have to enter the root administrative password.

PolicyKit stores its policies in the /usr/share/polkit-1/actions directory. The corresponding file for the **system-config-date** tool is org.fedoraproject.config.date.policy.

FIGURE 10-1


Query

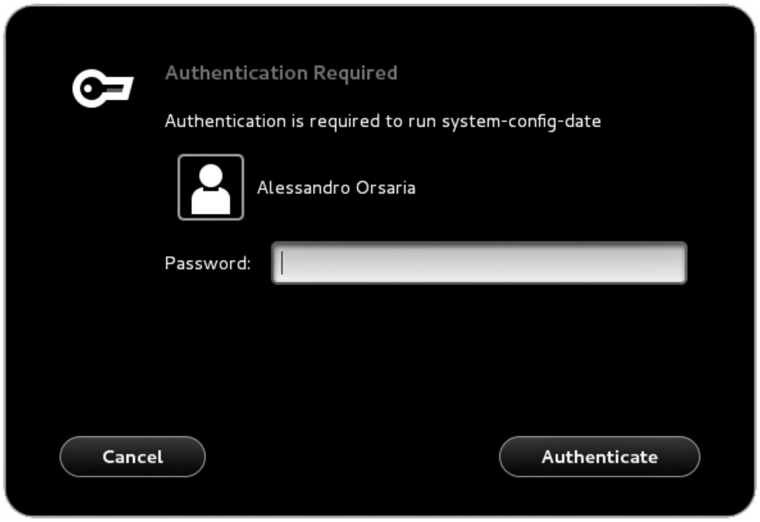
You are attempting to run "system-config-language" which requires administrative privileges, but more information is needed in order to do so.

Authenticating as "alex"

Access to administrative tools in the GUI requires the root password.

FIGURE 10-2

Access to administrative tools may be limited by the PolicyKit.


Authentication Required

Authentication is required to run system-config-date

Alessandro Orsaria

Password:

These policy files are configured in XML format and may be modified further to support fine-grained control by individual users. Although PolicyKit provides an API that can also be used by text-based programs, it is typically used to authorize the execution of GUI-based tools.

One alternative that also provides fine-grained control is the /etc/sudoers file described in Chapter 8.

CERTIFICATION OBJECTIVE 10.02

Firewalls and Network Address Translation

Typically, firewalls reside between internal LANs and outside insecure networks such as the Internet. A firewall can be configured to examine every network packet that passes into or out of your LAN. When configured with appropriate rules, it can filter out those packets that may pose a security risk to the systems on the LAN.

However, to follow the spirit of the recommendations from the NSA, you'll configure a firewall on every system.

Although network address translation (NAT) can be implemented on every system in the LAN, it is more commonly used on those systems configured as a gateway or router between a LAN and an outside network.

Definitions

Firewalls based on the firewalld service read the headers of each network packet. Based on the information contained in the headers, you can configure firewalld rules to filter each packet. To understand how *packet filtering* works, you have to understand a little bit about how information is sent across networks.

Before a message is sent over a network, that message is broken down into smaller units called *packets.* Administrative information, including the type of data, the source, and the destination addresses, as well as the source and destination ports (for TCP and UDP traffic), is added to the header of each packet. The packets are sent over the network and reach the destination Linux host. A firewall can examine the fields in each header. Based on existing rules, the firewall may then take one of the following actions with that packet:

- Allow the packet into the system.
- Forward the packet to other systems if the current system is a gateway or router between networks.
- Rate-limit the traffic.
- Reject the packet with a message sent to the originating IP address.
- Drop the packet without sending any sort of message.

Whatever the result, the decision can be logged to syslog or to the auditd subsystem. If a substantial number of packets are rejected or dropped, a log file may be useful.

RHEL 7 comes with everything you need to configure a system to be a firewall, for both IPv4 and IPv6 networks.

NAT can hide the IP address of the computers of a LAN that connect to outside networks. NAT replaces the internal source address with the IP address of the firewall interface connected to the outside network. The internal source address, along with other information that identifies the connection, is stored on the firewall connection table to keep note of which host made the request.

When the firewall receives a response, such as the content of a web page, the process is reversed. As the packets pass through the firewall, the destination host is identified in the connection table. The firewall modifies the IP header of each packet before sending the packets on their way.

This approach is useful for several reasons. Hiding internal IP addresses makes it harder for a black hat hacker to break into an internal network. NAT supports connections between systems with private IP addresses and external networks such as the Internet. It's the reason why IPv4 addressing has survived for so long. In the Linux world, this process is also known as *IP masquerading.*

While IP masquerading is usually referred to as "source NAT," there is also another form of NAT that works in the reverse direction, and is known as *port forwarding* or *destination NAT.* Port forwarding can hide the internal port and IP address of a service. As an example,

TABLE 10-1 Elements of a firewalld Zone

Zone Element	Description
Interfaces	Network interfaces associated with a zone
Sources	Source IP addresses associated with a zone
Services	Inbound services that are allowed through a zone, such as http
Ports	Destination TCP or UDP ports that are allowed through a zone, such as 8080/tcp
Masquerade	Specifies whether source network address translation (masquerading) is enabled
Forward ports	Port forwarding rules (map traffic sent to a local port onto another port on the same or another host)
ICMP blocks	Used to block ICMP messages
Rich rules	Advanced firewall rules

suppose you have an internal server with IP address 192.168.122.50, running a web service on TCP port 8080. With port forwarding, you can have clients connecting to a different IP address and port, such as a public IP on port 80, and forward traffic to the host and port on the internal network.

The Structure of firewalld

In Chapter 4 we introduced some of the basic concepts of firewalld. In this section, we will explore its advanced features, such as zone configuration and rich rules.

As you already know, firewalld is based on zones. A *zone* defines the level of trust of network connections. The basic elements that define a zone are illustrated in Table 10-1.

You can list all firewalld zones by typing the following command:

```
# firewall-cmd --get-zones
block dmz drop external home internal public trusted work
```

These correspond to the zones you have already encountered in Table 4-8 of Chapter 4. Take a few minutes to review the contents of that table.

To display the settings associated with a zone, use the **--list-all** command switch. As an example, let's show all the configuration settings of the *public* zone:

```
# firewall-cmd --list-all --zone=public
public (default, active)
  interfaces: eth0
  sources:
  services: dhcpv6-client ssh
  ports:
  masquerade: no
  forward-ports:
  icmp-blocks:
  rich-rules:
```

As you can see from the output, the *public* zone is associated with interface eth0. Incoming traffic for the DHCPv6 and SSH services is allowed through the zone, whereas masquerading is disabled. Note that on the first line of the command output, the public zone is marked as "default" and "active."

An *active zone* is a zone associated with at least one network interface or source IP address in firewalld. We introduced the concept of a *default zone* in Chapter 4: Only one zone can be marked as the "default" zone, and this special status means that any network interfaces added to the system will be automatically assigned to that zone.

Additionally, the default zone acts like a sort of "catch-all." This is related to how firewalld assigns incoming packets to a zone, based on the following rules:

- If the *source address* of the packet matches the source addresses associated with a zone, then the packet is processed according to the rules of that zone.
- If the packet comes from a *network interface* associated with a zone, then the packet is processed according to the rules of that zone.
- Otherwise, the packet is processed according to the rules of the default zone.

Once an incoming packet is matched to a zone, firewalld processes it according to the rules of that zone. As an example, based on the previous listed **firewall-cmd** output, incoming packets that reach the eth0 interface will be processed using the settings of the firewalld public zone. According to the rules of that zone, traffic will be allowed only if it belongs to the DHCPv6 or SSH protocol.

The most common **firewall-cmd** options related with zone configuration are shown in Table 10-2. Some of the listed options relate to the **--zone** command switch for non-default zones.

The following example shows how to set the default zone to the "work" zone:

```
# firewall-cmd --get-default-zone
public
# firewall-cmd --get-active-zones
public
  interfaces: virbr0 virbr1 wlp4s0
# firewall-cmd --set-default-zone=work
# firewall-cmd --get-default-zone
work
# firewall-cmd -get-active-zones
work
  interfaces: virbr0 virbr1 wlp4s0
```

TABLE 10-2 The firewall-cmd Zone Configuration Options

Command Option	Description
--get-default-zone	Lists the default zone
--set-default-zone=ZONE	Sets the default zone to ZONE
--get-zones	List all zones
--get-active-zones	List only the active zones—that is, the zones associated with at least one source interface or address in firewalld
--list-all-zones	Lists all the settings for all zones
--list-all [--zone=ZONE]	Lists all the settings for a specific ZONE or for the default zone
--add-source=NETWORK [--zone=ZONE]	Binds a source network to a ZONE or to the default zone
--change-source=NETWORK [--zone=ZONE]	Changes a source network currently assigned to a zone to a different ZONE or to the default zone
--remove-source=NETWORK [--zone=ZONE]	Removes a source network from a ZONE or from the default zone
--add-interface=INTERFACE [--zone=ZONE]	Adds an interface to a ZONE or to the default zone
--change-interface=INTERFACE [--zone=ZONE]	Changes an interface currently assigned to a zone into a different ZONE or into the default zone
--remove-interface=INTERFACE [--zone=ZONE]	Removes an interface from a ZONE or from the default zone

Observe how all interfaces that were assigned to the "public" zone have been moved to the "work" zone after the change. Also, note the **--set-default-zone** option makes a permanent change that survives after a system reboot. This is one of the few options in firewalld that does not require the **--permanent** switch, as you will see in a moment.

The next example associates the virbr1 interface with the "dmz" zone and adds the source IP range 192.168.99.0/24 to the "public" zone:

```
# firewall-cmd --change-interface=virbr1 --zone dmz
success
# firewall-cmd --add-source 192.168.99.0/24 --zone=public
success
# firewall-cmd --get-active-zones
dmz
  interfaces: virbr1
work
  interfaces: virbr0 wlp4s0
public
  sources: 192.168.99.0/24
```

exam watch

When you run firewall-cmd **to make configuration changes, don't forget the** --permanent **command option; otherwise, any changes you make won't survive a reboot.**

The configuration change made to the public zone in the previous example won't survive a reboot. As was noted in Chapter 4, to make permanent configuration changes, most **firewall-cmd** actions require the **--permanent** option. Once the new settings are saved into the permanent configuration, run **firewall-cmd --reload** to apply the settings immediately into the run-time configuration.

Services and Ports Configuration

To configure an effective firewalld zone, you need to give it the ability to allow or block traffic. You can do so by making appropriate changes to the services and port configuration. Before we dig into the details, examine Table 10-3, which provides a list of the service and port configuration options.

Two common ways to allow traffic through firewalld is by adding to a zone a pre-defined service, or a port and protocol combination, such as 8080/tcp. By default, all but one zone contains an implicit "deny all traffic" rule. The exception is the "trusted" zone, which allows all traffic by default. Hence, except for the "trusted" zone, you must explicitly allow a service or port; otherwise, the corresponding traffic will be blocked by the firewall.

TABLE 10-3 The firewall-cmd Service and Port Configuration Options

Command Option	Description
--get-services	Lists all predefined services
--list-services [--zone=ZONE]	Lists all services allowed for the specified ZONE or for the default zone
--add-service=SERVICE [--zone=ZONE]	Allows traffic for the specified SERVICE through a ZONE or the default zone
--remove-service=SERVICE [--zone=ZONE]	Removes a SERVICE from a ZONE or from the default zone
--list-ports [--zone=ZONE]	Lists TCP and UDP destination ports allowed through a ZONE or the default zone
--add-port=PORT/PROTOCOL [--zone=ZONE]	Allows traffic for the specified PORT/PROTOCOL through a ZONE or the default zone
--remove-port=PORT/PROTOCOL [--zone=ZONE]	Removes a PORT/PROTOCOL from a ZONE or from the default zone

on the job

Making configuration changes to firewalld on a production server may be dangerous and result in a loss of administrative access to the host. To avoid this problem, you can include the --timeout=*SECONDS* **option with your** firewall-cmd **command, which applies a configuration change only for the specified number of seconds.**

Take a look at the firewall services defined by default by running the following command:

```
# firewall-cmd --get-services
amanda-client bacula bacula-client dhcp dhcpv6 dhcpv6-client dns ftp
high-availability http https imaps ipp ipp-client ipsec kerberos kpasswd
ldap ldaps libvirt libvirt-tls mdns mountd ms-wbt mysql nfs ntp openvpn
pmcd pmproxy pmwebapi pmwebapis pop3s postgresql proxy-dhcp radius rpc-bind
samba samba-client smtp ssh telnet tftp tftp-client transmission-client
vnc-server wbem-https
```

These services are configured in XML files in the /usr/lib/firewalld/services/ directory. You can add services to the /etc/firewalld/services/ directory.

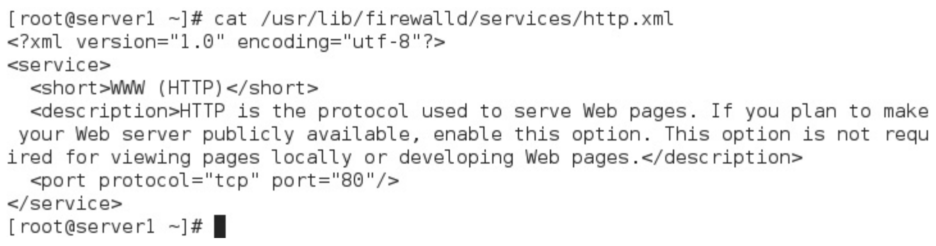
To see how this works, look at Figure 10-3. Note how the contents of the file http.xml contain an XML declaration and a <service> block, with three additional elements: the service short name, a description, and the corresponding protocol and port associated with the service (in this case, TCP/80).

Review the services associated with the default zone. If you had set the default to a different zone, revert your changes by running the command **firewall-cmd --set-default-zone=public**. Next, list the services associated with the zone with the following command:

```
# firewall-cmd --list-services
dhcpv6-client ftp http ssh
```

If you have completed the labs in Chapters 1 and 2 on your physical workstation, you should see the FTP and HTTP protocols in the list of services associated with the default

FIGURE 10-3 The firewalld configuration for the http service



```
[root@server1 ~]# cat /usr/lib/firewalld/services/http.xml
<?xml version="1.0" encoding="utf-8"?>
<service>
  <short>WWW (HTTP)</short>
  <description>HTTP is the protocol used to serve Web pages. If you plan to make
 your Web server publicly available, enable this option. This option is not requ
ired for viewing pages locally or developing Web pages.</description>
  <port protocol="tcp" port="80"/>
</service>
[root@server1 ~]# 
```

zone. If a service is missing, you can permanently add it to the zone configuration using the following commands:

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

You can also specify the traffic to be allowed through a zone using a port/protocol pair. As an example, suppose you have a web server running on a nonstandard port, such as TCP port 81. To allow connections to this port through the default firewalld zone, run the following command:

```
# firewall-cmd --permanent --add-port=81/tcp
# firewall-cmd --reload
```

The next command confirms the change:

```
# firewall-cmd --list-ports
81/tcp
```

To add a new service or port onto a different zone, the syntax of these commands is the same. You just need to specify the desired zone with the **--zone** command switch.

When you run a service on a nonstandard port, you may need to change the default SELinux port label configuration. This is not required for a web server running on TCP port 81, but it may be required in other situations, as you will see in Chapter 11.

Rich Rules

Adding services and ports to a zone is the most common way to allow traffic through a firewall. However, in some situations you may need the flexibility to create more complex rules. As an example, you may want to allow connections from all the IP addresses in a subnet, except for one specific host. With rich rules, you can satisfy this type of requirement and set up firewall rules that match a more complex logic. But there's more. You can also rate-limit incoming connections and log any connection attempts to syslog or to the audit service.

The common **firewall-cmd** options associated with rich rules are listed in Table 10-4. A rich rule does two things: it specifies the conditions a packet must meet to match the rule, and it specifies the action to execute if the packet matches.

A rich rule uses the following basic format:

```
rule [family=<rule_family>]
[source address=<address> [invert=true]]
[destination address=<address> [invert=true]]
service|port|protocol|icmp-block|masquerade|forward-port
[log] [audit] [accept|reject|drop]
```

TABLE 10-4 The firewall-cmd Rich Rule Configuration Options

Command Option	Description
--list-rich-rules [--zone=ZONE]	Lists all rich rules for the specified ZONE or for the default zone
--add-rich-rule='RULE' [--zone=ZONE]	Adds a rich rule for the specified ZONE or for the default zone
--remove-rich-rule='RULE' [--zone=ZONE]	Removes a rich rule for the specified ZONE or for the default zone
--query-rich-rule='RULE' [--zone=ZONE]	Checks whether a rich rule has been added for the specified ZONE or for the default zone

Now let's analyze this command, item by item. The first item is the **rule** keyword, followed by an optional family type. There are two family options for a rule:

- **family="ipv4"** Limits the action of the rule to IPv4 packets
- **family="ipv6"** Limits the action of the rule to IPv6 packets

Without the **family** keyword, the rule applies to both IPv4 and IPv6 packets.

The next two optional items are the source and destination addresses. You may specify them using the following format:

- **source address=*address[/mask]* [invert=true]** Matches all source IP addresses within the address/mask range. If you add **invert=true**, the rich rule applies to all but the specified address(es).
- **destination address=*address[/mask]* [invert=true]** Matches all destination IP addresses within the address/mask range. If you add **invert=true**, the rich rule applies to all but the specified address(es).

Packet patterns can be more complex. In TCP/IP, most packets are sent using the Transport Control Protocol (TCP), the User Datagram Protocol (UDP), or the Internet Control Message Protocol (ICMP) protocols. The associated packet patterns are listed here:

- **service name=*service_name*** All packets are checked for a specific service.
- **port=*port_number* protocol=tcp|udp** All packets are checked for a specific port number and protocol.
- **icmp-block name=*icmptype_name*** All packets are checked for a specific ICMP type. To display a list of supported ICMP types, run the command **firewall-cmd --get-icmptypes**.

The **masquerade** and **forward-port** options will be covered in the next sections of the chapter.

Once a rich rule finds a packet pattern match, it needs to know what to do with that packet. The last part of the rich rule options determines what happens to matched packets. There are five basic options:

- **drop** The packet is dropped. No message is sent to the source host.
- **reject** The packet is dropped. An ICMP error message is sent to the source host.
- **accept** The packet is allowed through the firewall.
- **log** The packet is logged to syslog.
- **audit** The packet is logged to the audit system.

The **limit value=*rate/duration*** directive is used to limit the amount of connections and packets logged in a time interval. For example, **limit value=5/m** specifies that a maximum of five log messages per minute will be logged or accepted, **limit value=10/h** sets the limit to 10 per hour, and so on. For a syntax reference of the firewalld rich language, refer to the **firewalld.richlanguage** man page.

EXERCISE 10-1

Configure Rich Rules

In this exercise, you will create a rich rule to allow all web traffic from all hosts in the network, except the tester1.example.com host. The rule you create will deny access to that host with an ICMP error message. In addition, the host oustider1.example.org should be allowed to connect to the web server, with all its connection attempts logged to syslog, at a rate limited to two messages per minute. Use the default (public zone) for all traffic. Assign the 192.168.100.0/24 network segment to the dmz zone.

This exercise assumes that you have installed the virtual machines and configured a default Apache server on your physical workstation, as explained in the labs of Chapters 1 and 2.

1. On your physical host, make sure that Apache is running and that you can access the Apache home page by navigating to the following URL: http://127.0.0.1.

   ```
   # systemctl status httpd
   # elinks --dump http://127.0.0.1
   ```

2. Check that the default zone is set to the public zone.

   ```
   # firewall-cmd --get-default-zone
   # firewall-cmd --set-default-zone=public
   ```

3. List the settings associated with the default zone. You should see the two virbr0 and virbr1 virtual bridges in the list of interfaces.

```
# firewall-cmd --list-all
```

4. Confirm that virbr1 is associated with the 192.168.100.0/24 network and move this interface to the dmz zone.

```
# ip addr show virbr1
# firewall-cmd --permanent --change-interface=virbr1 --zone=dmz
```

5. If the HTTP service is not allowed through the default zone, add it to the firewalld configuration.

```
# firewall-cmd --permanent --add-service=http
```

6. Create a rich rule to reject web connections from server1.example.com (192.168.122.50):

```
# firewall-cmd --permanent--add-rich-rule='rule family=ipv4 source ↵
address=192.168.122.50 service name=http reject'
```

7. Create a rich rule to log all connection attempts from outsider1.example.org (192.168.100.100) and rate-limit the logs to two messages per minute.

```
# firewall-cmd --permanent --zone=dmz --add-rich-rule='rule ↵
family=ipv4 source address=192.168.100.100 service name=http log limit ↵
value=2/m'
```

8. Reload the firewall configuration to apply the permanent changes to the run-time configuration.

```
# firewall-cmd --reload
```

9. Test from server1 and point the ELinks browser to 192.168.122.1. Is the host allowed to connect?

```
# elinks --dump http://192.168.122.1
```

10. Test from outsider1 and point the ELinks browser to 192.168.100.1. Is the host allowed to connect? Do you see any connection attempts logged to /var/log/messages?

```
# elinks --dump http://192.168.100.1
```

11. Revert the firewalld configuration to the initial settings.

Further Recommendations from the NSA

Simple firewalls are frequently the most secure. On an exam, it's best to keep everything, firewalls included, as simple as possible. But the NSA would go further. It has recommendations for the default rules, for limitations on the **ping** command, and for blocking suspicious groups of IP addresses. To those recommendations, we add a couple more suggestions to reduce risks to a system. Although these recommendations go beyond what's suggested by the RHCE objectives, read this section. If you're less than comfortable with the **firewall-cmd** command, this section can help. While you could implement these changes with the Rich Rules option in the Firewall Configuration tool, that is less efficient than with **firewall-cmd**.

exam watch

The suggested changes to firewalld are just recommendations. However, because the requirement to "implement packet filtering" is generic, it's useful to examine a variety of examples.

You can test any of these suggestions on a system like the server1.example.com VM created in Chapter 2.

Regulate the ping Command

One earlier attack on various Internet systems involved the **ping** command. From Linux, it's possible to flood another system with the **-f** switch. If the attacker uses multiple systems, it may transmit thousands or millions of packets per second. It's important to be able to defend a system from such attacks or to limit their impact because they can prevent others from accessing your websites and more.

on the job

The -f **switch to the** ping **command was described solely to point out one of the major risks on a network. On most Linux distributions, only root is allowed to specify the** -f **switch. In many cases, it is illegal to run such a command on or against someone else's system. For example, one article suggests that such an attack could be a violation of the Police and Justice Act in the United Kingdom with a penalty of up to 10 years in prison. Similar laws exist in other countries.**

One potentially troublesome rule in the default firewall is that all ICMP traffic is allowed by default. ICMP messages go both ways. If you run the **ping** command on a remote system, the remote system responds with an ICMP Echo Reply packet. So if you want to limit ICMP messages, use the following rules to filter ICMP Echo Requests:

```
# firewall-cmd --add-icmp-block=echo-request
```

Add this rule to the server1.example.com VM, and measure the amount of packets sent and received from your physical host to server1 with the **ping -f** command. Then, do the same after removing the ICMP block and compare your results.

Block Suspicious IP Addresses

Black hat hackers who want to break into a system may hide their source IP address. As nobody is supposed to use a private or experimental IPv4 address on the public Internet, such addresses are one way to hide. The following additions to firewalld would drop packets sourced from the specified IPv4 network address blocks:

```
# firewall-cmd --add-rich-rule='rule family=ipv4 source↵
address=10.0.0.0/8 drop'
# firewall-cmd --add-rich-rule='rule family=ipv4 source↵
address=172.16.0.0/12 drop'
# firewall-cmd --add-rich-rule='rule family=ipv4 source↵
address=192.168.0.0/16 drop'
# firewall-cmd --add-rich-rule='rule family=ipv4 source↵
address=169.254.0.0/16 drop'
```

Regulate Access to SSH

Since SSH is important for the administration of remote systems, additional measures to protect this service are appropriate. It's certainly possible to set up a nonstandard port for SSH communication. Such a measure can be a part of a layered security strategy. However, tools such as **nmap** can detect the use of SSH on such nonstandard ports. So it's generally better to set up the configuration of the SSH server as discussed in Chapter 11 along with firewall rules such as the following. The following rich rule allows all SSH traffic, but limits incoming connections to three per minute:

```
# firewall-cmd --add-rich-rule='rule service name=ssh accept↵
limit value=3/m'
```

Make Sure That firewalld Is Running

Once desired changes are saved with the **--permanent** option, make sure that firewalld is in operation with the new rules. Don't forget to reload the configuration with the following command:

```
# firewall-cmd --reload
```

To avoid starting the old iptables firewall (which was the default in RHEL 6), it is a good idea to mask the corresponding service units, as shown here:

```
# systemctl mask iptables
# systemctl mask ip6tables
```

These commands link the **iptables** and **ip6tables** service units to /dev/null, preventing a system administrator from accidentally starting those services.

exam watch

It's critical to understand how to secure a Red Hat Enterprise Linux system against unauthorized access.

IP Masquerading

Red Hat Enterprise Linux supports a variation of NAT called *IP masquerading.* IP masquerading is often used to allow Internet access from multiple internal hosts, while sharing only a single public IP address. IP masquerading maps multiple internal IP addresses to that single valid external IP address. That helps, because all public IPv4 address blocks have now been allocated. IPv4 addresses are often still available from third parties, but at a cost. That cost is one more reason for IP masquerading. On the other hand, you may think that systems on IPv6 networks do not need masquerading, as it's relatively easy for many requesting users to get their own subnet of public IPv6 addresses. Nevertheless, even on IPv6 networks, masquerading can help keep that system secure.

exam watch

The RHCE objectives specify the use of firewalld to configure network address translation.

IP masquerading is a fairly straightforward process. It's implemented on a gateway or router, which, by definition, has two or more network interfaces. One network interface is typically connected to an outside network such as the Internet, and the second network interface is connected to a LAN. In a small office, the interface connected to the outside network may connect through an external device such as a cable "modem" or Digital Subscriber Line (DSL) adapter. The following assumptions are made for the configuration:

- The public IP address is assigned to the network interface that is directly connected to the outside network.
- Network interfaces on the LAN get IP addresses associated with a single private network.
- One network interface on the gateway or router system gets an IP address on that same private network.
- IP forwarding is enabled on the router or gateway system, as discussed later in this chapter.
- Each system on the LAN is configured with the private IP address of the router or gateway system as the default gateway address.

When a computer on a LAN wants a web page on the Internet, packets are routed to the firewall. The firewall replaces the source IP address on each packet with the firewall's public IP address. It then assigns a new port number to the packet. The firewall caches the original source IP address and port number.

When a packet comes back from the Internet to the firewall, it should include a port number. If the firewall can match an associated rule with the port number assigned to a previous outgoing packet, the process is reversed. The firewall replaces the destination IP address and port number with the internal computer's private IP address and then forwards the packet back to the original client on the LAN.

In practice, the following command enables masquerading. The noted command assumes that the zone that is directly connected to the Internet is the "dmz":

```
# firewall-cmd --permanent --zone=dmz --add-masquerade
```

You can also use a rich rule to enable masquerading. This offers the option to control which source IP addresses should be masqueraded:

```
# firewall-cmd --permanent --zone=dmz --add-rich-rule='rule family-ipv4↵
source address=192.168.0.0/24 masquerade'
```

In most cases, the private IP network address is not required because most LANs protected by a masquerade are configured on a single private IP network.

On the "external" zone, masquerading is enabled by default. Assigning the Internet-facing network interface to this zone would automatically masquerade all internal clients that establish a connection to the Internet.

IP Forwarding

IP forwarding is more commonly referred to as *routing.* Routing is critical to the operation of the Internet or any IP network. Routers connect and facilitate communication between multiple networks. When you set up a computer to find a site on an outside network, it needs a gateway address. This corresponds to the IP address of a router on the LAN.

A router looks at the destination IP address of each packet. If the IP address is on one of the router's LANs, it routes the packet directly to the proper computer. Otherwise, it sends the packet to another router closer to its final destination. To use a Red Hat Enterprise Linux system as a router, you should enable IP forwarding in the /etc/sysctl.conf configuration file by adding this line:

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

These settings take effect on the next reboot. Until then, the new settings in sysctl.conf can be enabled with the following command:

```
# sysctl -p
```

On a physical host running the KVM hypervisor, IP forwarding is usually enabled by default, as you can verify with the following command:

```
# sysctl net.ipv4.ip_forward
```

The Red Hat Firewall Configuration Tool

Chapter 4 introduced the RHCSA elements of the Red Hat Firewall Configuration tool. In this section, we will explain how you can implement the RHCE objectives related to firewalld using the Firewall Configuration tool.

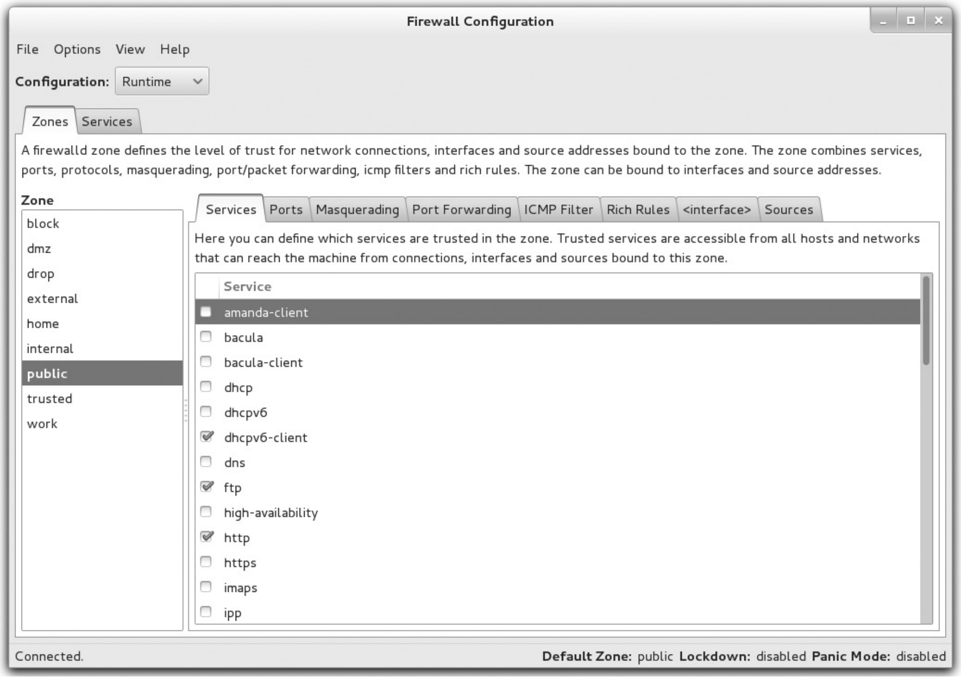
Start the Firewall Configuration tool with the **firewall-config** command or by clicking Applications | Sundry | Firewall. The Firewall Configuration tool has a number of capabilities, as shown in Figure 10-4. In general, if you have selected the run-time configuration mode, it implements changes immediately, but those won't persist a system reboot. In most cases, you will want to select Permanent mode, which writes changes that survive system reboot.

After you make a change, you can apply the saved configuration immediately by clicking Options | Reload Firewalld.

Default Zone and Interfaces

In the Firewall Configuration tool, the left pane lists zones; the default zone is marked in bold. You can do everything in the Firewall Configuration tool that you can do with the

FIGURE 10-4 The Firewall Configuration tool



Configuration: Runtime

A firewalld zone defines the level of trust for network connections, interfaces and source addresses bound to the zone. The zone combines services, ports, protocols, masquerading, port/packet forwarding, icmp filters and rich rules. The zone can be bound to interfaces and source addresses.

Zone: block, dmz, drop, external, home, internal, **public**, trusted, work

Here you can define which services are trusted in the zone. Trusted services are accessible from all hosts and networks that can reach the machine from connections, interfaces and sources bound to this zone.

- ☐ amanda-client
- ☐ bacula
- ☐ bacula-client
- ☐ dhcp
- ☐ dhcpv6
- ✓ dhcpv6-client
- ☐ dns
- ✓ ftp
- ☐ high-availability
- ✓ http
- ☐ https
- ☐ imaps
- ☐ ipp

Connected. **Default Zone:** public **Lockdown:** disabled **Panic Mode:** disabled

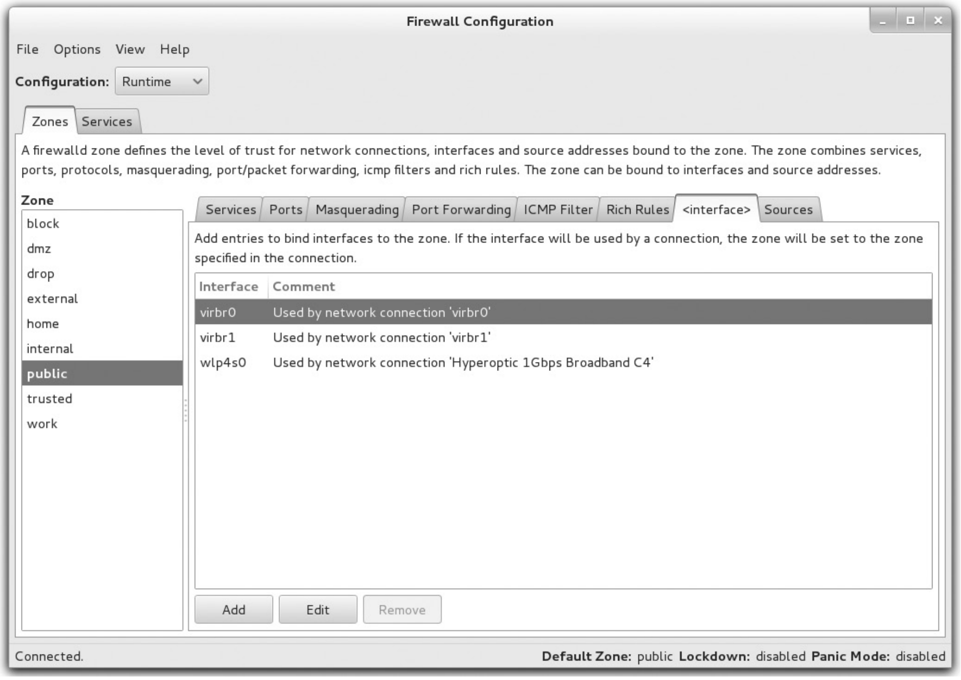
firewall-cmd command. You can assign interfaces to zones, select which services and ports are allowed through a zone, enable masquerading, and so on.

To bind interfaces to a zone, click the Interface tab to reveal the window shown in Figure 10-5. Routers have two or more network interfaces. Administrators who trust the systems on the internal network may click Add to assign internal interfaces to the "trusted" zone. However, this can be a risky practice. Threats can come from within as well as from outside a network.

In most cases, the gateway or router system has two or more Ethernet devices. Assume you are configuring a system with three devices: eth0, eth1, and eth2, where eth0 is connected to an external network, eth1 to a DMZ, and eth2 to the internal network. If you trust all systems on a local network, you might assign device eth2 as part of the trusted zone.

Sometimes a device may be listed using a different naming convention. As an example, a wireless adapter can be named wlan0 or even ath0. In other cases, it may be named wlp4s0, where p4 and s0 indicate the PCI bus and slot number, respectively. So it's important to know the device files associated with each network device on a system.

FIGURE 10-5 Zone interfaces



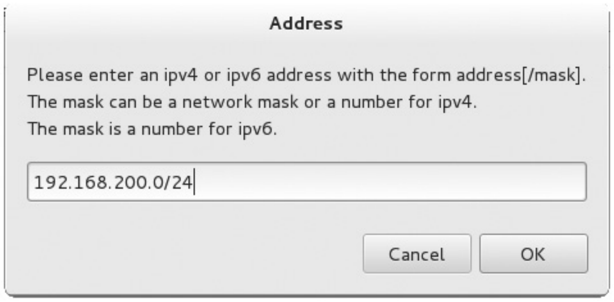
Configuration: Runtime

A firewalld zone defines the level of trust for network connections, interfaces and source addresses bound to the zone. The zone combines services, ports, protocols, masquerading, port/packet forwarding, icmp filters and rich rules. The zone can be bound to interfaces and source addresses.

Add entries to bind interfaces to the zone. If the interface will be used by a connection, the zone will be set to the zone specified in the connection.

Interface	Comment
virbr0	Used by network connection 'virbr0'
virbr1	Used by network connection 'virbr1'
wlp4s0	Used by network connection 'Hyperoptic 1Gbps Broadband C4'

Connected. **Default Zone:** public **Lockdown:** disabled **Panic Mode:** disabled



Address

Please enter an ipv4 or ipv6 address with the form address[/mask].
The mask can be a network mask or a number for ipv4.
The mask is a number for ipv6.

192.168.200.0/24

FIGURE 10-6

A source IP address range assigned to a zone

In the Firewall Configuration tool, you can bind source IP addresses to a zone. To do so, select the Source tab and click the Add button. This opens the Address window shown in Figure 10-6. Then, enter the source IP address range, such as 192.168.200.0/16.

Similarly, you can enable services and ports from the corresponding tabs. Once the configuration is applied, traffic matching the selected services and ports will be trusted through the interfaces and source IP addresses bound to the zone.

Masquerading

In the Firewall Configuration tool, select a zone and then click the Masquerading tab to reveal the window shown in Figure 10-7. In most cases, you should set up masquerading for the traffic that is going out to the Internet. That has three advantages:

- It hides the IP address identity of the internal systems from external networks.
- It requires only one public IP address.
- It sets up IP forwarding across the configured network devices.

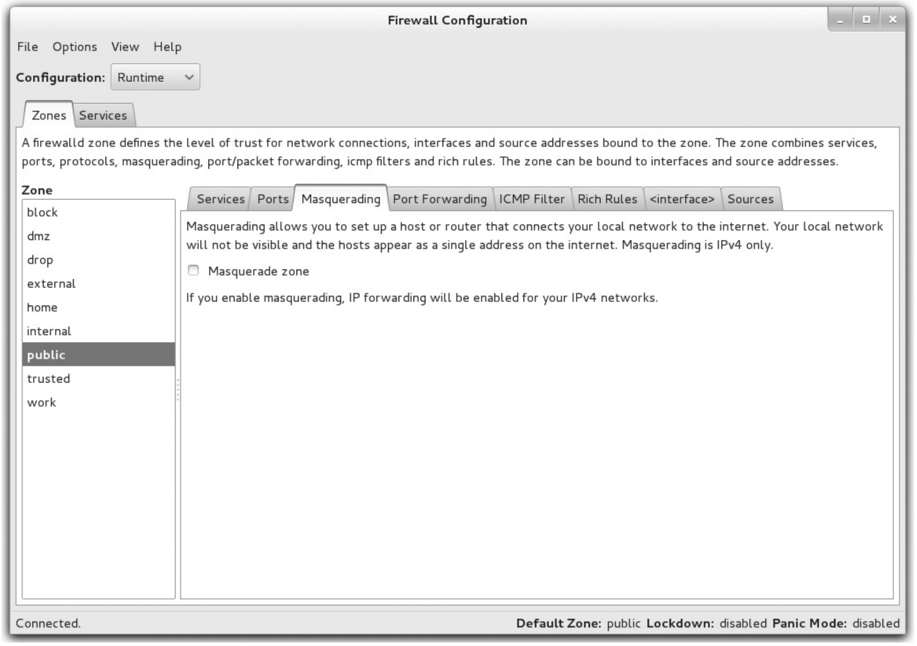
Administrators can choose to set up masquerading on the zone of their choice. The selected zone should be the one connected to an external network such as the Internet.

Port Forwarding

In the Firewall Configuration tool, select a zone and then select the Port Forwarding tab. Typically, forwarding in this fashion works only in combination with masquerading. With such rules, port forwarding can be used to set up communications by mapping one port and protocol to a port on the local or a remote system, as defined by its IP address. One example is shown in Figure 10-8.

The options shown in the figure would redirect traffic destined for TCP port 80 on the public zone to a remote destination, with an IP address of 192.168.122.150. The port on that remote system is 8008. Port forwarding is also known as “destination NAT” because it is used to translate the destination IP address and service port of a packet. Port forwarding is typically used to make an internal service visible to other machines on the Internet, while hiding all the other services running on the internal host.

FIGURE 10-7 Masquerading with the Firewall Configuration tool



Firewall Configuration

Configuration: Runtime

A firewalld zone defines the level of trust for network connections, interfaces and source addresses bound to the zone. The zone combines services, ports, protocols, masquerading, port/packet forwarding, icmp filters and rich rules. The zone can be bound to interfaces and source addresses.

Zone
block
dmz
drop
external
home
internal
public
trusted
work
Services Ports Masquerading Port Forwarding ICMP Filter Rich Rules <interface> Sources

Masquerading allows you to set up a host or router that connects your local network to the internet. Your local network will not be visible and the hosts appear as a single address on the internet. Masquerading is IPv4 only.

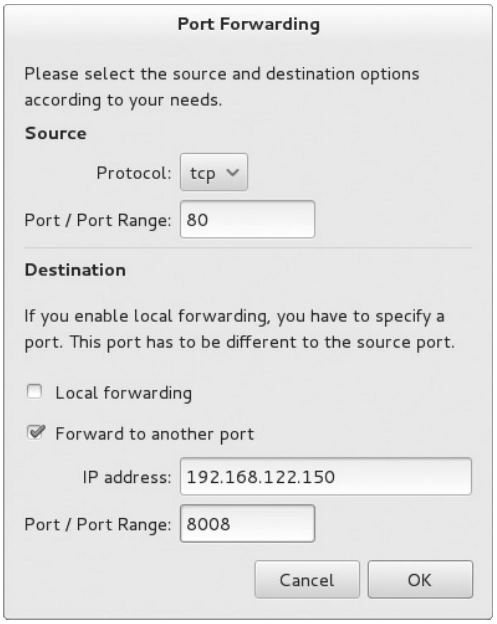
☐ Masquerade zone

If you enable masquerading, IP forwarding will be enabled for your IPv4 networks.

Connected. Default Zone: public Lockdown: disabled Panic Mode: disabled

FIGURE 10-8

Port forwarding with the Firewall Configuration tool



Port Forwarding

Please select the source and destination options according to your needs.

Source

Protocol: tcp

Port / Port Range: 80

Destination

If you enable local forwarding, you have to specify a port. This port has to be different to the source port.

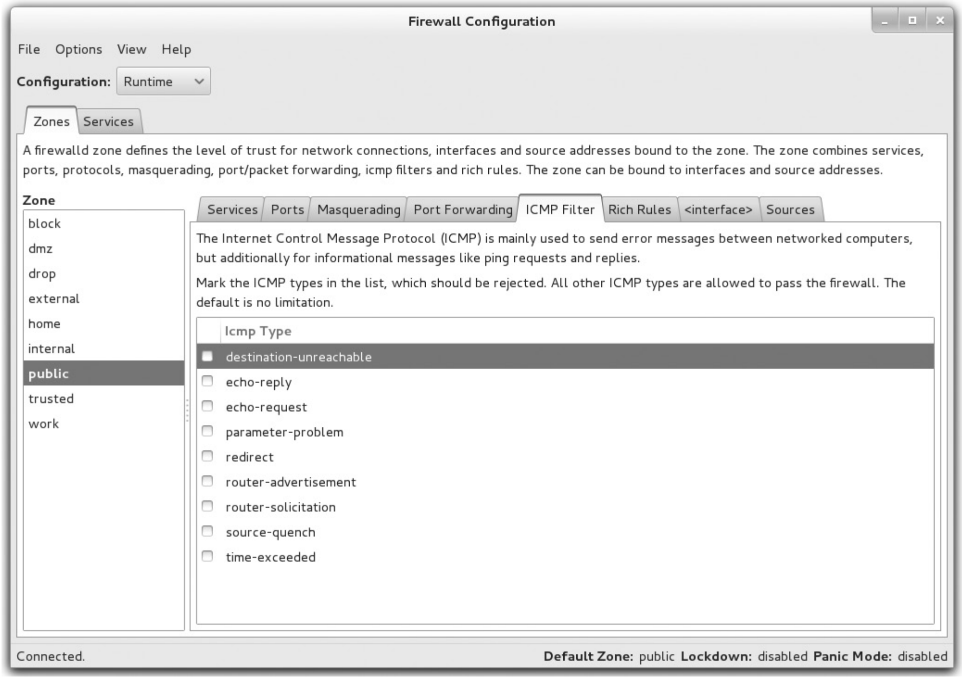
☐ Local forwarding

☑ Forward to another port

IP address: 192.168.122.150

Port / Port Range: 8008

FIGURE 10-9 ICMP Filter with the Firewall Configuration tool



A firewalld zone defines the level of trust for network connections, interfaces and source addresses bound to the zone. The zone combines services, ports, protocols, masquerading, port/packet forwarding, icmp filters and rich rules. The zone can be bound to interfaces and source addresses.

The Internet Control Message Protocol (ICMP) is mainly used to send error messages between networked computers, but additionally for informational messages like ping requests and replies.

Mark the ICMP types in the list, which should be rejected. All other ICMP types are allowed to pass the firewall. The default is no limitation.

Connected. **Default Zone:** public **Lockdown:** disabled **Panic Mode:** disabled

ICMP Filter

In the Firewall Configuration tool, select a zone and click the ICMP Filter tab to open the screen shown in Figure 10-9. As suggested in the description, the options relate to different messages associated with the ICMP protocol, including but not limited to, the **ping**-related packets. The options shown are further described in Table 10-5. If you activate a filter in that table, the filter blocks that category of packets.

Rich Rules

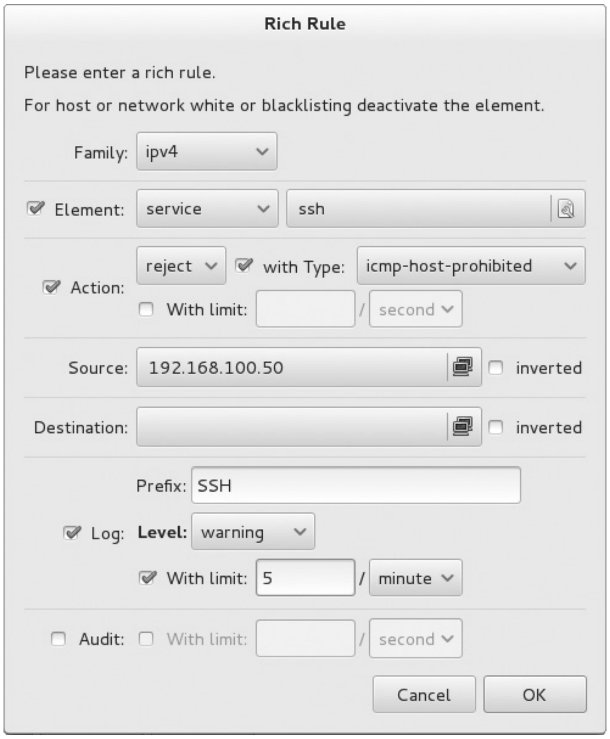
Select a zone, click the Rich Rules tab, and then click Add to open the Rich Rule window shown in Figure 10-10.

For the purpose of this section, we have created a new rule, as shown in Figure 10-10, to block all SSH connections from the host 192.168.100.50. Connection attempts are rejected with a ICMP Host Prohibited message and are logged to /var/log/messages with an “SSH” prefix. Log messages are limited to a maximum of five per minute.

TABLE 10-5 ICMP Filter Options

Filter	Description
Destination Unreachable	A message generated by a router to inform that a destination address is unreachable
Echo Reply	Regular response messages to Echo Requests
Echo Request	A message that can be generated with the **ping** command
Parameter Problem	Error messages not otherwise defined by other ICMP messages
Redirect	A message to notify the source host to send packets via an alternative route
Router Advertisement	Periodic message multicasted to other routers to announce the IP addresses assigned to an interface
Router Solicitation	A request for a router advertisement
Source Quench	Response to a host to slow down packet transfers
Time Exceeded	Error message if a "Time To Live" field in a packet is exceeded

FIGURE 10-10



Please enter a rich rule.

For host or network white or blacklisting deactivate the element.

Family: ipv4

[✓] Element: service ssh

[✓] Action: reject [✓] with Type: icmp-host-prohibited

[ ] With limit: / second

Source: 192.168.100.50 [ ] inverted

Destination: [ ] inverted

Prefix: SSH

[✓] Log: **Level:** warning

[✓] With limit: 5 / minute

[ ] Audit: [ ] With limit: / second

Take advantage of rich rules.

CERTIFICATION OBJECTIVE 10.03

TCP Wrappers

As suggested by its name, TCP Wrappers protects those services that communicate using the TCP protocol. It was originally designed to help protect services configured through the Extended Internet Super-Server daemon (xinted). However, TCP Wrappers' protection is no longer limited to such services; the protection can apply to all services statically and dynamically linked to the associated library wrapper file, libwrap.so.0.

The way TCP Wrappers protects a service is defined in the /etc/hosts.allow and /etc/hosts.deny configuration files.

Is a Service Protected by TCP Wrappers?

The **strings** command can be used to identify those daemons protected by TCP Wrappers. It does so by listing the printable character sequences included in a binary. The string associated with TCP Wrappers is hosts_access. Daemons can be found in the /usr/sbin directory. Thus, the quickest way to scan the daemons in these directories for the hosts_access string is with the following command:

```
# strings -f /usr/sbin/* | grep hosts_access
```

The output depends on installed packages. One example is the SSH daemon, /usr/sbin/sshd.

You can also use the shared library dependencies command, **ldd**, to confirm a link to the TCP Wrappers library, libwrap.0.so. To identify those dependencies for the **sshd** daemon, run the following command:

```
# ldd /usr/sbin/sshd
```

However, that's not convenient because it returns the files for more than a couple of dozen library files. As an expert at the Linux command line, you should know how to pipe that output to the **grep** command to see if it's associated with the TCP Wrappers library file, libwrap.so.0:

```
# ldd /usr/sbin/sshd | grep libwrap.so.0
```

And from the output, it's confirmed:

```
libwrap.so.0 => /lib64/libwrap.so.0 (0x00007f13b94380000)
```

The TCP Wrappers configuration files can help you protect the SSH service. That protection comes over and above any settings included in the standard zone-based firewalld, the SSH server configuration file, SELinux, and so on. However, such redundant protection is important in a layered security strategy.

TCP Wrappers Configuration Files

When a system receives a network request for a service linked to the libwrap.so.0 library, it passes the request on to TCP Wrappers. This system logs the request and then checks its access rules. If there are no limits on the particular host or IP address, TCP Wrappers passes control back to the service.

The key files are hosts.allow and hosts.deny in the /etc directory. The philosophy is fairly straightforward: users and clients listed in hosts.allow are allowed access; users and clients listed in hosts.deny are denied access. As users and/or clients may be listed in both files, the TCP Wrappers system takes the following steps:

1. It searches /etc/hosts.allow. If TCP Wrappers finds a match, it grants access. No additional searches are required.
2. It searches /etc/hosts.deny. If TCP Wrappers finds a match, it denies access.
3. If the host isn't found in either file, access is automatically granted to the client.

You use the same access control language in both /etc/hosts.allow and /etc/hosts.deny files. The basic format for commands in each file is as follows:

```
daemon_list : client_list
```

The simplest version of this format is

```
ALL : ALL
```

This specifies all services and makes the rule applicable to all hosts on all IP addresses. If you set this line in /etc/hosts.deny, access is prohibited to all services. Of course, since that is read after /etc/hosts.allow, services in that file are allowed.

You can create finer-grained filters than just prohibiting access to *all* daemons from *all* systems. For example, the following line in /etc/hosts.allow allows the client with an IP address of 192.168.122.50 to connect to the local system through the Secure Shell:

```
sshd : 192.168.122.50
```

The same line in /etc/hosts.deny would prevent the computer with that IP address from using SSH to connect. If the same line exists in both files, /etc/hosts.allow takes precedence, and users from the noted IP address will be able to connect through SSH, assuming other

TABLE 10-6 Sample Client Lists in /etc/hosts.allow and /etc/hosts.deny

Client	Description
.example.com	Domain name. Since this domain name begins with a dot, it specifies all clients on the example.com domain.
172.16.	IP address. Since this address ends with a dot, it specifies all clients with an IP address of 172.16.*x*.*y*.
172.16.72.0/255.255.254.0	IP network address with subnet mask.
172.16.72.0/23	IP network address with subnet mask, but using CIDR notation.
ALL	Any client, any daemon.

security settings such as zone-based firewalls allow it. You can specify clients a number of different ways, as shown in Table 10-6.

As you can see in Table 10-6, there are two different types of wildcards. **ALL** can be used to represent any client or service, and the dot (**.**) specifies all hosts with the specified domain name or IP network address.

You can set up multiple services and addresses with commas. Exceptions are easy to make with the **EXCEPT** operator. Review the following excerpt from the /etc/hosts.allow file:

```
ALL : .example.com
sshd : 192.168.122.0/24 EXCEPT 192.168.122.150
vsftpd : 192.168.100.100
```

The first line in this file opens **ALL** services to all computers in the example.com domain. The following line opens the SSH service to any computer on the 192.168.122.0/24 network, except the one with an IP address of 192.168.122.150. Then the vsFTP service is opened to the computer with an IP address of 192.168.100.100. You may want to add the localhost IP address network to the noted daemons in the /etc/hosts.allow file, as follows:

```
sshd : 127. 192.168.122.0/24 EXCEPT 192.168.122.150
vsftpd : 127. 192.168.100.100
```

Otherwise, attempts to connect from the local system may be denied based on other directives in the /etc/hosts.deny file.

The configuration that follows contains a hosts.deny file to see how lists can be built to control access:

```
ALL EXCEPT vsftpd : .example.org
sshd : ALL EXCEPT 192.168.122.150
ALL : ALL
```

The first line in the hosts.deny file denies all services except vsFTP to computers in the example.org domain. The second line states that the only computer allowed to access the local SSH server has an IP address of 192.168.122.100. Finally, the last line is a blanket denial; all other computers are denied access to all services controlled by TCP Wrappers.

EXERCISE 10-2

Configure TCP Wrappers

In this exercise, you will use TCP Wrappers to control access to network resources. Since such controls are enabled by default, you shouldn't have to make any modifications to installed services.

1. Try to connect to the local vsFTP server using the address localhost. You may need to do several things first:
   A. Install the Very Secure FTP daemon from the vsftpd RPM.
   B. Install the **lftp** client from the lftp RPM.
   C. Activate the vsFTP service with the **systemctl start vsftpd** command.
   D. Configure the service to start at boot with the **systemctl enable vsftpd** command.
   E. Allow the FTP protocol through firewalld.
2. Edit /etc/hosts.deny and add the following line (don't forget to write the file):

   ```
   ALL : ALL
   ```

3. What happens when you try to run **lftp 127.0.0.1**? And what if you run a command such as **ls** after logging into the FTP server?
4. Edit /etc/hosts.allow and add the following line:

   ```
   vsftpd : 127.0.0.1
   ```

5. Now what happens when you try to run **lftp 127.0.0.1**? Does the **ls** command return any output from the FTP server?
6. Undo any changes made when you are finished.

CERTIFICATION OBJECTIVE 10.04

Pluggable Authentication Modules

RHEL uses the Pluggable Authentication Modules (PAM) system as another layer of security primarily for administrative tools and related commands. PAM includes a group of dynamically loadable library modules that govern how individual applications verify their users. You can modify PAM configuration files to customize security requirements for different administrative utilities. Most PAM configuration files are stored in the /etc/pam.d directory.

PAM modules also standardize the user authentication process. For example, the login program uses PAM to require usernames and passwords at login. Open the /etc/pam.d/login file. Take a look at the first line:

```
auth [user_unknown=ignore success=ok ignore=ignore default=bad] ↵
pam_securetty.so
```

To interpret, this line means that the root user can log in only from secure terminals as defined in the /etc/securetty file, and unknown users are ignored.

The configuration files shown in the /etc/pam.d directory often have the same name as the command that starts the administrative utility. These utilities are "PAM aware." In other words, you can change the way users are verified for applications such as the console login program. Just modify the appropriate configuration file in the /etc/pam.d directory.

Configuration Files

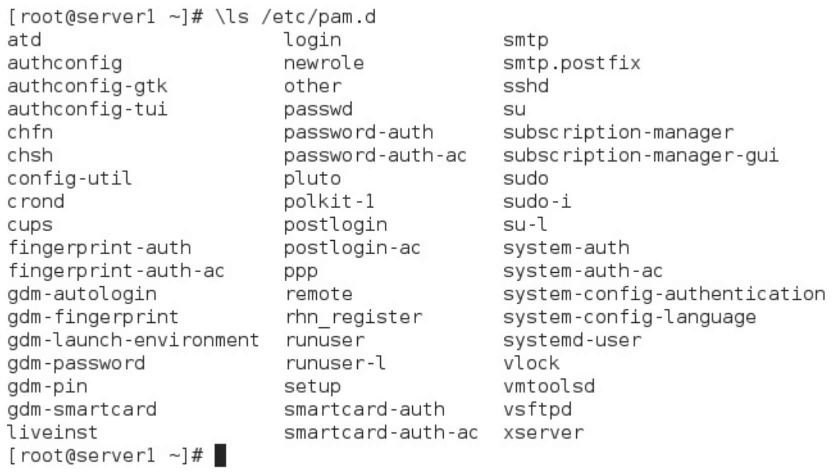
Take a look at the configuration files in a typical /etc/pam.d directory, as shown in Figure 10-11. Depending on what's installed, you may see a somewhat different list of files.

As suggested earlier, most of the filenames in the /etc/pam.d directory are descriptive. Take a look at some of these files. In most cases, they refer to PAM modules. These modules can be found in the /usr/lib64/security directory. Excellent descriptions of each module can be found in the /usr/share/doc/pam-*versionnumber* directory, in the txts/ and html/ subdirectories. For example, the functionality of the pam_securetty.so module is described in the README.pam_securetty file.

You can also refer to the HTML version of the Linux-PAM System Administrators' Guide available in the /usr/share/doc/pam-*versionnumber*/html directory, starting with the Linux-PAM_SAG.html file.

FIGURE 10-11

PAM configuration files in the /etc/pam.d directory



```
[root@server1 ~]# \ls /etc/pam.d
atd                      login              smtp
authconfig               newrole            smtp.postfix
authconfig-gtk           other              sshd
authconfig-tui           passwd             su
chfn                     password-auth      subscription-manager
chsh                     password-auth-ac   subscription-manager-gui
config-util              pluto              sudo
crond                    polkit-1           sudo-i
cups                     postlogin          su-l
fingerprint-auth         postlogin-ac       system-auth
fingerprint-auth-ac      ppp                system-auth-ac
gdm-autologin            remote             system-config-authentication
gdm-fingerprint          rhn_register       system-config-language
gdm-launch-environment   runuser            systemd-user
gdm-password             runuser-l          vlock
gdm-pin                  setup              vmtoolsd
gdm-smartcard            smartcard-auth     vsftpd
liveinst                 smartcard-auth-ac  xserver
[root@server1 ~]#
```

Control Flags

The PAM system provides four different types of services. These are associated with four different types of PAM rules:

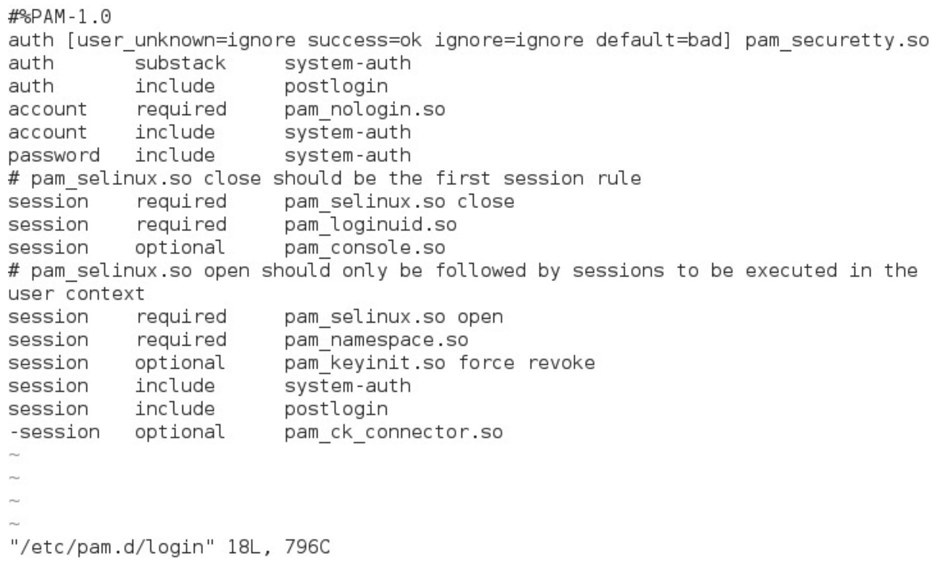
- **Authentication management (auth)** Validates the identity of a user. For example, a PAM **auth** rule verifies whether a user has provided valid username and password credentials.
- **Account management (account)** Allows or denies access according to the account policies. For example, a PAM **account** rule may deny access according to time, password expiration, or a specific list of restricted users.
- **Password management (password)** Manages password change policies. For example, a PAM **password** rule may enforce a minimum password length when a user tries to change her password.
- **Session management (session)** Applies settings for an application session. For example, a PAM **session** rule may set default settings for a login console.

The configuration shown in Figure 10-12 is from a sample PAM configuration file, /etc/pam.d/login. Every line in all PAM configuration files is written in the following format:

```
type  control_flag  module_name  [arguments]
```

The **type**, as described previously, can be **auth**, **account**, **password**, or **session**. The **control_flag** determines what PAM does if the module succeeds or fails.

FIGURE 10-12 The PAM /etc/pam.d/login configuration file



```
#%PAM-1.0
auth [user_unknown=ignore success=ok ignore=ignore default=bad] pam_securetty.so
auth       substack     system-auth
auth       include      postlogin
account    required     pam_nologin.so
account    include      system-auth
password   include      system-auth
# pam_selinux.so close should be the first session rule
session    required     pam_selinux.so close
session    required     pam_loginuid.so
session    optional     pam_console.so
# pam_selinux.so open should only be followed by sessions to be executed in the
user context
session    required     pam_selinux.so open
session    required     pam_namespace.so
session    optional     pam_keyinit.so force revoke
session    include      system-auth
session    include      postlogin
-session   optional     pam_ck_connector.so
~
~
~
~
"/etc/pam.d/login" 18L, 796C
```

The **module_name** specifies the name of the actual PAM module file. Finally, you can specify options for each module.

The **control_flag** field requires additional explanation. It determines how PAM reacts when a module returns success or failure. The five most common control flags are described in Table 10-7.

TABLE 10-7 PAM Control Flags

control_flag	Description
required	If this module returns success, PAM proceeds to the next rule of this type. If it fails, PAM proceeds to the next rule in the configuration file—but the final result is failure.
requisite	If this module fails, PAM does not check any additional rules and returns a failure.
sufficient	If this module passes, no other rules of this type need to be processed, and the result is success. Otherwise, if the check fails, PAM continues processing the remaining rules.
optional	PAM ignores the success or failure of this rule.
include	Includes all directives of the same **type** from the noted configuration file; for example, if the directive is **password include system-auth**, this includes all password directives from the PAM system-auth file.

To see how control flags work, take a look at the rules from the /etc/pam.d/runuser configuration file:

```
auth   sufficient    pam_rootok.so
```

The first **auth** command checks the pam_rootok.so module. In other words, if the root user tries to run the **runuser** command, the rule will return a pass and the **runuser** command will be executed. As the **control_flag** is **sufficient**, if there were other **auth** commands in this file, they would be ignored.

```
session   optional    pam_keyinit.so ignore
```

The purpose of the second line is to clear the session key ring of the **runuser** process when this exits. In this case, the **control_flag** is **optional**, meaning that the outcome of this rule does not have any effect on the other **session** rules.

```
session   required    pam_limits.so
```

The third line sets the resource limits defined in /etc/security/limits.conf when invoking an instance of the **runuser** application. The **control_flag** is **required**, which will cause the command session to fail if the limits cannot be set.

```
session   required    pam_unix.so
```

The module associated with the last **session** type (pam_unix.so) logs the username and service type at the beginning and end of each command session.

The Format of a PAM File

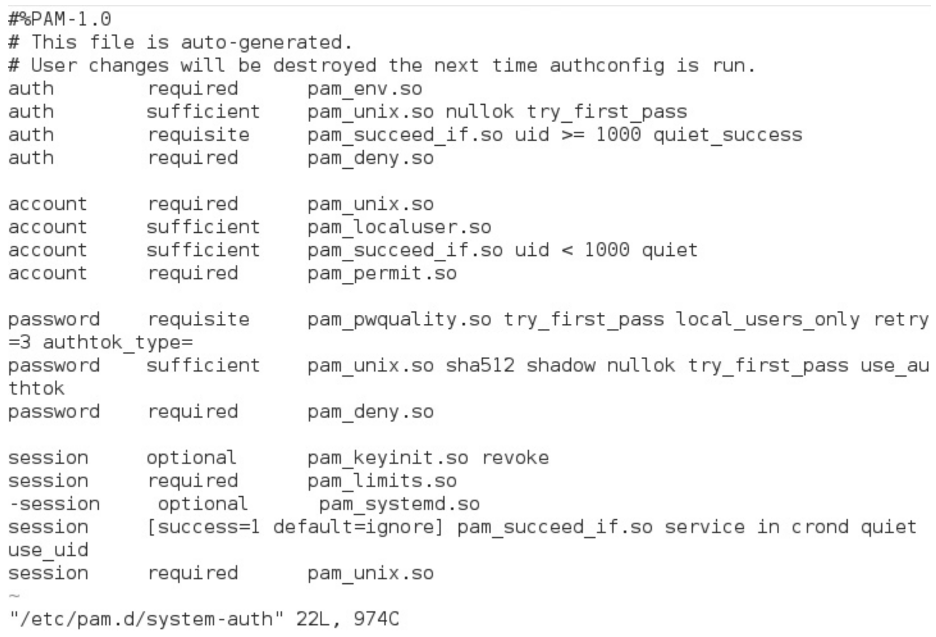
This section is a little complex. It starts with the /etc/pam.d/login configuration file shown in Figure 10-12. In addition, as the file includes references to the /etc/pam.d/system-auth configuration file, shown in Figure 10-13, you'll need to go back and forth between files to follow along with this section.

You don't have to memorize the content in this section. Instead, you should use it to become more familiar with PAM configuration files. While you read this section, familiarize yourself with each PAM module by reading the corresponding man page. The **rpm -qd pam** command gives a full list of the installed PAM man pages in your system.

When a user opens a text console and logs in, Linux goes through the /etc/pam.d/login configuration file line by line. As previously noted, the first line in /etc/pam.d/login limits root user access to secure terminals as defined in the /etc/securetty file:

```
auth [user_unknown=ignore success=ok ignore=ignore default=bad] ↵
pam_securetty.so
```

FIGURE 10-13 The PAM /etc/pam.d/system-auth configuration file



```
#%PAM-1.0
# This file is auto-generated.
# User changes will be destroyed the next time authconfig is run.
auth        required      pam_env.so
auth        sufficient    pam_unix.so nullok try_first_pass
auth        requisite     pam_succeed_if.so uid >= 1000 quiet_success
auth        required      pam_deny.so

account     required      pam_unix.so
account     sufficient    pam_localuser.so
account     sufficient    pam_succeed_if.so uid < 1000 quiet
account     required      pam_permit.so

password    requisite     pam_pwquality.so try_first_pass local_users_only retry
=3 authtok_type=
password    sufficient    pam_unix.so sha512 shadow nullok try_first_pass use_au
thtok
password    required      pam_deny.so

session     optional      pam_keyinit.so revoke
session     required      pam_limits.so
-session     optional      pam_systemd.so
session     [success=1 default=ignore] pam_succeed_if.so service in crond quiet
use_uid
session     required      pam_unix.so
~
"/etc/pam.d/system-auth" 22L, 974C
```

The next line includes the **auth** commands from the system-auth PAM configuration file:

```
auth  substack     system-auth
```

For the purpose of this example, you can assume that the **substack** control flag is equivalent to an **include** directive. The system-auth configuration file shown in Figure 10-13 includes four **auth** directives. On your system, you may see additional lines—for example, if you configured the machine as an LDAP or Kerberos client. In that case, your configuration will reference additional PAM modules, such as pam_ldap or pam_krb5.

```
auth         required      pam_env.so
auth         sufficient    pam_unix.so nullok try_first_pass
auth         requisite     pam_succeed_if.so uid >= 1000 quiet_success
auth         required      pam_deny.so
```

In order, the preceding lines set up environment variables and check password authentication against the local /etc/passwd and /etc/shadow databases (pam_unix.so). The **sufficient** flag associated with the second module means that authentication succeeds if a valid password has been entered, and no further rules from the **auth** section are processed.

exam watch

If the /etc/nologin file exists, regular users are not allowed to log in to the local console. Any regular user who tries to log in gets to read the contents of /etc/nologin as a message. This behavior is controlled by the pam_nologin.so module.

If the pam_unix.so module returns a fail, PAM will process the next rule, which disables logging if the user ID of the account is 1000 and above. Then, if PAM gets to the last rule, the user is denied access (pam_deny.so).

Now return to the /etc/pam.d/login file. The next line includes the directive in the postlogin file, which does not contain any **auth** rules. Moving to the subsequent line, this invokes the **account** type of the pam_login.so module. This module disables user logins if the file /etc/nologin exists:

```
account   required    pam_nologin.so
```

The following rule includes the **account** rules from the /etc/pam.d/system-auth configuration file:

```
account    include     system-auth
```

These are the **account** type rules lines from the default /etc/pam.d/system-auth:

```
account   required    pam_unix.so
account   sufficient  pam_localuser.so
account   sufficient  pam_succeed_if.so uid < 1000 quiet
account   required    pam_permit.so
```

The first line refers to the pam_unix.so module in the /usr/lib64/security directory, which checks in /etc/shadow whether the account is valid and has not expired. Based on the pam_localuser.so module and on the **sufficient** control type, if the username is listed in /etc/passwd, no further directives are processed. The pam_succeed_if.so module disables logging for service users (with user IDs less than 1000). Then, the pam_permit.so module always returns success.

Now return to the /etc/pam.d/login file. The next line is a **password** directive, which includes other password rules from the /etc/pam.d/system-auth file:

```
password  include     system-auth
```

These are the **password** type rules from the default /etc/pam.d/system-auth:

```
password requisite   pam_pwquality.so try_first_pass ↵
local_users_only retry=3 authok_type=
password sufficient  pam_unix.so sha512 shadow nullok try_first_pass ↵
use_authok
password required    pam_deny.so
```

The first rule from this output performs a password strength check. It allows the use of the password collected by the application that called PAM (**try_first_pass**) and applies its checks to local users only, with a maximum of three password change attempts.

The next rule updates the user's password using the SHA512 encryption hash, supports the Shadow Password Suite described in Chapter 8, allows the use of an existing null (zero-length) password, and forces the module to set the new password to the value provided by the previous module (**use_authok**).

The **password required pam_deny.so** directive is trivial; as noted in README.pam_deny in the /usr/share/doc/pam-*versionlevel*/txt directory, this module always fails.

Finally, there are six **session** rules in the default /etc/pam.d/login file. Let's take them three at a time:

```
session   required    pam_selinux.so open
session   required    pam_namespace.so
session   optional    pam_keyinit.so force revoke
```

The first line (**pam_selinux.so open**) sets up a few SELinux security contexts. The second line (**pam_namespace.so**) creates separate namespaces for users at logon. The third line initializes the key ring of the login session (**pam_keyring.so**).

```
session   include     system-auth
session   include     postlogin
-session   optional   pam_ck_connector.so
```

Jumping ahead, the last of this group of rules registers a login session with the ConsoleKit daemon. Note the minus character in front of the line: this tells PAM not to send any error message to syslog if the module is missing. The preceding rules include the following **session** type lines from the system-auth and postlogin files:

```
session     optional      pam_keyinit.so revoke
session     required      pam_limits.so
-session    optional      pam_systemd.so
session     [success=1 default=ignore] pam_succeed_if.so service in ↵
crond quiet use_uid
session     required      pam_unix.so
```

The first of these rules is identical to the line in the main /etc/pam.d/login file that revokes the session key ring of the invoking process. The next rule sets limits (pam_limits.so) on individual user resources through /etc/security/limits.conf. The next rule registers the user session with the systemd login manager. The fourth rule will skip the next rule (**success=1**) for cron jobs. The final rule logs the result when the user logs in.

Finally, the next three rules included from the postlogin file invoke the pam_lastlogin.so module with different options, depending on whether the requesting service is the graphical login, the **su** command, or another process. The pam_lastlogin.so module shows the amount of previous failed login attempts to the user, and is also commonly used to record the date of the last login in /var/log/lastlog.

```
session     [success=1 default=ignore] pam_succeed_if.so ↵
service !~ gdm* service !~ su* quiet
session     [default=1]   pam_lastlog.so nowtmp showfailed
session     optional      pam_lastlog.so silent noupdate showfailed
```

EXERCISE 10-3

Configure PAM to Limit root Access

In this exercise, you can experiment with some of the PAM security features of Red Hat Enterprise Linux 7.

1. Make a backup copy of /etc/securetty with the following command:

   ```
   # cp /etc/securetty /etc/securetty.bak
   ```

2. Edit /etc/securetty and remove the lines for tty3 through tty11. Save the changes and exit.
3. Use ALT-F3 (CTRL-ALT-F3 if you're running X Window) to switch to virtual console number 3. Try to log in as root. What happens?
4. Repeat Step 3 as a regular user. What happens? Do you know why?
5. Use ALT-F2 to switch to virtual console number 2 and try to log in as root.
6. Review the messages in /var/log/secure. Do you see where you tried to log in as root in virtual console number 3?
7. Restore the original /etc/securetty file with the following command:

   ```
   # mv /etc/securetty.bak /etc/securetty
   ```

One thing to remember is that the /etc/securetty file governs the consoles from which you can log in as the root user. Therefore, the changes that were made do not affect regular (non-root) users.

PAM and User-Based Security

In this section, you'll learn how to configure PAM to limit access to specific users. The key to this security feature is the pam_listfile.so module, which is located in the /usr/lib64/security directory. If you've installed the vsFTP server, the /etc/pam.d/vsftpd file includes an example of this module.

First, the following line in the vsftpd file initializes and clears out any existing key rings when the session is closed:.

```
session optional pam_keyinit.so  force revoke
```

The way PAM can limit user access is shown in the next rule:

```
auth required pam_listfile.so item=user sense=deny ↵
file=/etc/vsftpd/ftpusers onerr=succeed
```

To understand how this works, let's break this rule into its component parts. You already know the first three parts of the rule from the previous section. The options that are shown are associated with the pam_listfile.so module, as described in the pam_listfile man page and in Table 10-8.

Thus, for the specified rule (**onerr=succeed**), an error, strangely enough, returns success (**item=user**). If the user is in the specified list (**file=/etc/vsftpd/ftpusers**), the rule allows that user (**sense=allow**) to access the specified tool.

exam watch

Make sure you understand how Red Hat Enterprise Linux handles user authorization through the /etc/pam.d configuration files. When you test these files, make sure you create a backup of everything in PAM before making any changes, because any errors that you make on a PAM configuration file can disable access to your system completely (PAM is that secure).

TABLE 10-8 Options for the pam_listfile.so Module

pam_listfile Option	Description
item	This option can be used to limit access to a terminal (**tty**), user (**user**), group (**group**), or more.
sense	If the item is found in the specified **file**, take the noted action. For example, if the user is listed in /etc/special and **sense=allow**, then this command grants the user permission for the specified tool.
file	The path of the file with a list of users, groups, and so on, such as **file = /etc/special**.
onerr	If there is a problem, tell the module what to do. The options are **onerr=succeed** or **onerr=fail**.

EXERCISE 10-4

Use PAM to Limit User Access

You can also use the PAM system to limit access to all non-root users. In this exercise, you'll limit access using the pam_nologin.so module. It should work hand in hand with the default /etc/pam.d/login security configuration file, specifically the following line:

```
account  required  pam_nologin.so
```

1. Look for an /etc/nologin file. If it doesn't already exist, create one with a message such as the following:

   ```
   I'm sorry, access is limited to the root user
   ```

2. Access another terminal with a command such as CTRL-ALT-F2. Try logging in as a regular user. What do you see?
3. Log in as the root user. You'll see the same message; but as the root user, you're allowed access.
4. Inspect the /var/log/secure file. Did your system reject the attempted login from the regular user? What were the associated messages for the root user?

SCENARIO & SOLUTION

You have only one public IP address, but you need to provide Internet access to all of the systems on your LAN. Each computer on the LAN has its own private IP address.	Use firewalld to implement IP masquerading. Make sure IP forwarding is active.
You have installed an SSH server on a corporate network and want to restrict access to certain departments. Each department has its own subnet.	Use the /etc/hosts.deny file in the tcp_wrappers package to block SSH access to the unwanted subnets. A better alternative would be to use /etc/hosts.allow to support access to desired departments, and then use /etc/hosts.deny to deny access to everyone else. Similar options are possible using firewalld rich rules.
You want to restrict access to a service, such as SSH, only to certain users.	Add a line in the appropriate Pluggable Authentication Module configuration files in /etc/pam.d to use the **pam_listfile.so** module.
You want to modify the local firewall to defend against ICMP attacks such as **ping** command floods.	Modify the firewall to reject or deny certain types of ICMP packets.

CERTIFICATION OBJECTIVE 10.05

Secure Files and More with GPG2

With the importance of network security, you should know how to encrypt files for secure transmission. The computer standard for file encryption and signature services is known as Pretty Good Privacy (PGP). The open-source implementation of PGP is known as the GNU Privacy Guard (GPG). The version released for RHEL 7 is more advanced and capable, documented as GPG version 2 (GPG2) You've likely already used GPG2 to verify the authenticity of RPM packages, as discussed in Chapter 7. This section takes such checks one step further; you'll generate private and public keys and then use those keys to encrypt and decrypt selected files.

Although GPG is not listed in the RHCE objectives, it's a security topic consistent with other security objectives discussed in this book. We believe it's an excellent topic that might be included in future versions of the RHCE exam.

GPG2 Commands

The GPG version 2 included in RHEL 7 has a more modular approach to encryption and authentication. There's even a related package used for smartcard authentication. But that's not the point of the new **gpg2** commands. Available GPG commands are briefly described in Table 10-9.

Table 10-9 is just intended to describe the range of capabilities associated with the RHEL 7 GPG2 packages. The focus of this section is on the encryption and decryption of files.

TABLE 10-9 GPG2 Commands

Command	Description
gpg	Symbolic link to the **gpg2** command
gpg2	The GPG2 encryption and signing tool
gpg-agent	GPG2 key management daemon
gpgconf	Provides access and modifies configuration files in ~/.gnupg
gpg-connect-agent	Utility to communicate with an active GPG2 agent
gpg-error	Command to interpret a GPG2 error number
gpgsplit	Command to split a GPG2 message into packets
gpgv	Symbolic link to the **gpg2v** command
gpgv2	Command to verify GPG signatures; requires a signature file
gpg-zip	Command to encrypt or sign files into an archive

Current GPG2 Configuration

While the man page for the **gpgconf** command suggests that it's just used to modify the directory with associated configuration files, that command does more. By itself, it defaults to the **--list-components** switch, which specifies the full path to related executable files. With the **--check-programs** switch, it makes sure all related programs can be executed. **gpgconf** can also be used to check the syntax of a GPG2 configuration file. One typical option is in the current user's home directory, in the .gnupg/ subdirectory. Another typical option is in the /etc/gnupg directory.

GPG2 Encryption Options

The generation of a GPG2 key includes a choice of three different cryptographic algorithms, as listed next. Each of these algorithms includes a public and a private key. The public key can be distributed to others for use in encrypting files and messages. The private key is used by the owner, and is the only way to decrypt the file or message.

- **RSA** Named for its developers, Rivest, Shamir, and Adleman. Although typical RSA keys are 1024 or 2048 bits in length, they can be larger. Shorter keys of 512 bits have been cracked. RSA is in the public domain.
- **DSA** The Digital Signature Algorithm. Proposed by the U.S. National Institute of Science and Technology, DSA has been made available for worldwide use, royalty free. This is a U.S. government standard that uses Secure Hash Algorithm (SHA) versions SHA-1 and SHA-2 as message digest hash functions. SHA-1 is being phased out; SHA-2 includes six hash functions with message digests of up to 512 bits, also known as SHA-512, the same hash as is now used for the RHEL 7 shadow password suite.
- **ElGamal** Developed by Taher Elgamal, this probabilistic encryption algorithm is used in GPG in combination with DSA, with an ElGamal key pair used for encryption and another DSA key pair for making signatures. ElGamal is the first encryption scheme based on the Diffie-Hellman key exchange method.

Generate a GPG2 Key

The **gpg --gen-key** command can be used to set up key pairs with different types of encryption schemes. Before running the command, be prepared with answers to the following questions:

- The number of bits for the encryption keys. Normally, the maximum number of bits is 4096, but an encryption key that complex may take a number of minutes to develop.
- The desired lifetime of the keys. Especially if you set up keys with a smaller number of bits, you should assume that a determined black hat hacker would be able to decrypt the key within some number of months or weeks.

- A name, an e-mail address, and a comment. Although the name and e-mail address do not have to be real, they will be seen by others as part of the public key.
- A passphrase. Good passphrases should include spaces, lower- and uppercase letters, numbers, and punctuation.

As given, the **gpg --gen-key** command prompts for one of four different encryption schemes. As suggested by the (sign only) label associated with choices 3 and 4, those options work just as digital signatures, not for encryption.

```
Please select what kind of key you want:
   (1) RSA and RSA (default)
   (2) DSA and Elgamal
   (3) DSA (sign only)
   (4) RSA (sign only)
Your selection?
```

All four options follow a similar sequence of steps. For example, if you select option 2, the following output appears:

```
DSA keys may be between 1024 and 3072 bits long.
What keysize do you want? (2048)
```

The default is 2048 bits, which is selected if you just press ENTER. The command then prompts for a key lifetime:

```
Requested keysize is 2048 bits
Please specify how long the key should be valid.
         0 = key does not expire
      <n>  = key expires in n days
      <n>w = key expires in n weeks
      <n>m = key expires in n months
      <n>y = key expires in n years
Key is valid for? (0) 2m
```

In this case, we've selected two months. The command responds with a date and time two months into the future and prompts for confirmation.

```
Key expires at Sat 27 Apr 2015 11:14:17 AM BST
Is this correct? (y/N) y
```

At this point, the **gpg** command prompts for identifying information for the key. The "User ID" requested here is not related to the UID in the standard Linux authentication database. In this example, the responses are shown in bold:

```
Real name: Michael Jang
Email address: michael@example.com
Comment: DSA and Elgamal key
```

```
You selected this USER-ID:
    "Michael Jang (DSA and Elgamal key) <michael@example.com>"

Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? o
```

The system should now prompt for a passphrase. Then, the **gpg** command goes to work. Especially with larger keys, it may seem to pause for a few minutes with a message about creating random bytes. You might need to run some other programs to stimulate the process. When complete, it displays a message similar to the following:

```
gpg: key D385AFDD marked as ultimately trusted
public and secret key created and signed.
```

To make sure the public and private keys were actually written, run the following command:

```
$ gpg --list-key
```

The output should include the latest key, along with any others created from the user's home directory, in the .gnupg/ subdirectory. For the given options, if this is the only key pair on the local account, you'll see something similar to the following output:

```
/home/michael/.gnupg/pubring.gpg
--------------------------------
pub   2048D/9F688440 2015-04-28 [expires: 2015-06-27]
uid   Michael Jang (DSA and ElGamal) <michael@example.com>
sub   2048g/306A91C0 2015-04-28 [expires: 2015-06-27]
```

Use a GPG2 Key to Encrypt a File

Now you can send the public key to a remote system. To start the process, you'll need to export the public key. For the key pair just created, you could do so with the following command (substitute your name for "Michael Jang"):

```
$ gpg --export Michael Jang > gpg.pub
```

Now copy that key to a remote system. The delivery vehicle, such as e-mail, a USB stick, or the **scp** command shown here, is not important. This particular command from user michael's account would copy the gpg.pub key to user michael's home directory on the tester1.example.com system. If you prefer, substitute the IP address, like so:

```
$ scp gpg.pub tester1.example.com:
```

Now go to the remote system (in this case, tester1.example.com). Log in to user michael's account (or the account home directory to which you copied the gpg.pub key). Once connected to that system, first check for existing GPG keys with the following command:

```
$ gpg --list-key
```

If this is a system without previous GPG keys, the list should be empty, and nothing will appear in the output to this command. Now import the gpg.pub file into the list of local GPG keys with the following command:

```
$ gpg --import gpg.pub
```

Confirm the import by running the **gpg --list-key** command again.

Now on the remote system, you can encrypt a file with the **gpg** command. The following example encrypts the local keepthis.secret file:

```
$ gpg --out underthe.radar --recipient 'Michael Jang' ↵
--encrypt keepthis.secret
```

The username in this case is Michael Jang. If you've just imported a private key, the username as shown in the output to the **gpg --list-key** command may be different. Substitute as appropriate.

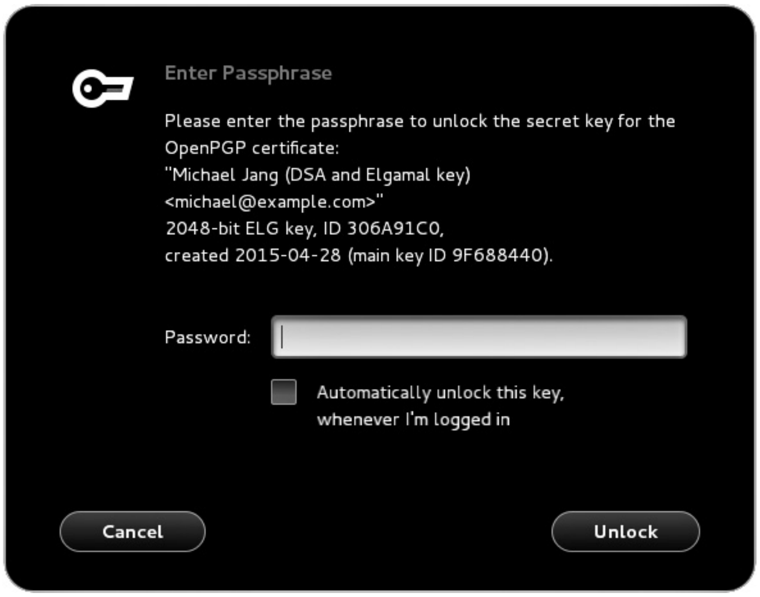
Now when the underthe.radar file is copied to the original system, server1.example.com, you can start the decryption process with the private key with the following command:

```
$ gpg --out keepthis.secret --decrypt underthe.radar
```

In a console, you'd be prompted for the passphrase created earlier, with a screen similar to that shown in Figure 10-14.

FIGURE 10-14

Prompting for a passphrase for decryption



Enter Passphrase

Please enter the passphrase to unlock the secret key for the OpenPGP certificate:
"Michael Jang (DSA and Elgamal key)
<michael@example.com>"
2048-bit ELG key, ID 306A91C0,
created 2015-04-28 (main key ID 9F688440).

Password:

Automatically unlock this key, whenever I'm logged in

CERTIFICATION SUMMARY

To help defend the data, the services, and the systems on a network, Linux provides layers of security. If a service is not installed, a black hat hacker can't use it to break into a system. Those systems that are installed should be kept up to date. Such services can be protected by firewalls, along with host- and user-based security options. Many services include their own layers of security. RHEL 7 incorporates several recommendations from the NSA, including SELinux.

Zone-based firewalls can regulate and protect gateways as well as individual systems. That same firewalld daemon can be used to set up packet forwarding, as well as masquerading of private networks. Such options can be configured directly via the CLI using **firewall-cmd**, or set up with the help of the Firewall Configuration tool.

Those daemons linked to the TCP Wrappers library can be protected by appropriate settings in the /etc/hosts.allow and /etc/hosts.deny files. If there is a conflict, /etc/hosts.allow is read first. Regulation through TCP Wrappers is possible by user or host.

PAM supports user-based security for a number of administrative tools. They're configured individually through files in the /etc/pam.d directory. These files refer to modules in the /usr/lib64/security directory.

Linux supports encryption with the help of GPG. RHEL 7 includes GPG2 for this purpose as well as commands such as **gpg** to set up private/public key pairs using the RSA, DSA, or ElGamal scheme.

✓ TWO-MINUTE DRILL

The following are some of the key points from the certification objectives in Chapter 10.

The Layers of Linux Security

- ❑ Bastion systems are more secure because they're configured with a single service. With virtualization, bastion systems are now a practical option even for smaller organizations.
- ❑ You may choose to automate at least security updates with the Software Updates Preference tool.
- ❑ Many services include their own security options in their configuration files.
- ❑ Host-based security can be configured by domain name or IP address.
- ❑ User-based security includes specified users and groups.
- ❑ The PolicyKit can regulate security of administrative tools run from the GNOME desktop environment.

Firewalls and Network Address Translation

- ❑ The firewalld configuration command is **firewall-cmd**.
- ❑ With firewalld, you can assign interfaces and source IP ranges to different zones as well as control which traffic is allowed into a zone.
- ❑ With firewalld, you can also masquerade the IP addresses from one network on an outside network such as the Internet.
- ❑ firewalld can also be configured with the help of the Firewall Configuration tool, which you can start with the **firewall-config** command.

TCP Wrappers

- ❑ The **strings -f /usr/sbin/*** command can identify services that can be regulated by TCP Wrappers.
- ❑ Clients and users listed in /etc/hosts.allow are allowed access; clients and users listed in /etc/hosts.deny are denied access.
- ❑ Remember to use the actual executable name of the daemon in /etc/hosts.allow and /etc/hosts.deny (normally in /usr/sbin), such as **vsftpd**.

Pluggable Authentication Modules

- ❑ RHEL 7 uses the Pluggable Authentication Modules (PAM) system to provide a common application programming interface for authentication services.
- ❑ PAM modules are called by configuration files in the /etc/pam.d directory. These configuration files are usually named after the service or command they control.
- ❑ There are four main types of PAM rules: authentication, account, password, and session management.
- ❑ PAM configuration files include lines that list the type, the control_flag, and the name of the actual module, followed by optional arguments.
- ❑ PAM modules are well documented in the /usr/share/doc/pam-*versionnumber*/txts directory and in the man pages.

Secure Files and More with GPG2

- ❑ GPG is the open-source implementation of PGP.
- ❑ RHEL 7 includes version 2 of GPG, known as GPG2.
- ❑ GPG2 encryption and signature can use the DSA, RSA, and ElGamal keys.
- ❑ GPG2 keys can be created with the **gpg --gen-key** command and listed with the **gpg --list-key** command.

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 these questions.

The Layers of Linux Security

1. What security option is best for a service that isn't currently required on a system?

___

Firewalls and Network Address Translation

2. Consider a system with the default firewalld settings, where the following commands have been entered:

```
# firewall-cmd --zone=dmz --add-source=192.168.77.77
# firewall-cmd --zone=dmz --remove-service=ssh
```

Once these are entered, and before a reboot, what effect will they have when a client with an IP address of 192.168.77.77 tries to connect to this system?

___

3. What directories include the configuration files that define the firewalld services?

___

4. You are setting up a small office and would like to provide Internet access to a small number of users but don't have enough dedicated public IPv4 addresses for each system on the network. What can you do?

___

5. What **firewall-cmd** command switch sets up masquerading?

___

6. What **firewall-cmd** command adds a rich rule to the default zone to allow HTTP connections only from the 192.168.122.50 host?

   ____

7. What **firewall-cmd** command option is used to make permanent configuration changes?

   ____

TCP Wrappers

8. With TCP Wrappers configuration files, how could you limit FTP access to clients on the 192.168.170.0/24 network? Hint: the vsFTP daemon, when installed, is in /usr/sbin/vsftpd.

   ____

9. What happens to a service if you allow the service in /etc/hosts.allow and prohibit it in /etc/hosts.deny?

   ____

Pluggable Authentication Modules

10. What are the four basic PAM rule types?

    ____

    ____

    ____

    ____

11. You are editing a PAM configuration file by adding a module. Which control flag immediately terminates the authentication process if the module succeeds?

    ____

Secure Files and More with GPG2

12. What command lists the GPG public keys loaded on the current local account?

    ____

LAB QUESTIONS

Several of these labs involve exercises that can seriously affect a system. You should do these exercises on test machines only. The second lab of Chapter 1 sets up KVM for this purpose.

Red Hat presents its exams electronically. For that reason, the labs for this chapter are available from the media that accompanies the book, in the Chapter10/ subdirectory. They're available in .doc, .html, and .txt formats. In case you haven't yet set up RHEL 7 on a system, refer to the first lab of Chapter 2 for installation instructions. The answers for each lab follow the Self Test answers for the fill-in-the-blank questions.

SELF TEST ANSWERS

The Layers of Linux Security

1. The security option that is best for a service that isn't currently required on a system is to not install that service.

Firewalls and Network Address Translation

2. Based on the given commands, any connection attempt to the SSH service from the 192.168.77.77 system is rejected.
3. The configuration of firewalld services is stored in the /usr/lib/firewalld/services/ and /etc/firewalld/services/ directories.
4. To set up a small office while providing Internet access to a small number of users, all you need is one dedicated IP address. The other addresses can be on a private network. Masquerading makes this possible.
5. The **firewall-cmd** command switch that sets up masquerading is **--add-masquerade**.
6. The following rich rule allows HTTP traffic from 192.168.122.50 for the default zone:

```
# firewall-cmd --add-rich-rule='rule family=ipv4 source ↵
address=192.168.122.50 service name=http accept'
```

7. The **--permanent** option of **firewall-cmd** is used to make permanent configuration changes. Don't forget to load the saved configuration into the run-time configuration using the **firewall-cmd --reload** command.

TCP Wrappers

8. To limit FTP access to clients on the 192.168.170.0 network, you'd allow access to the network in /etc/hosts.allow and deny it to all others in /etc/hosts.deny. As /usr/sbin is in the root user path, you can cite the **vsftpd** daemon directly and add the following directive to /etc/hosts.allow:

   ```
   vsftpd : 192.168.170.0/255.255.255.0
   ```

   Then you would add the following to /etc/hosts.deny:

   ```
   vsftpd : ALL
   ```

9. If you allow a service in /etc/hosts.allow and prohibit it in /etc/hosts.deny, the service is allowed.

Pluggable Authentication Modules

10. The four basic PAM rule types are **auth**, **account**, **password**, and **session**. The **include** type refers to one or more of the other PAM types in a different file.
11. The **sufficient** control flag immediately terminates the authentication process if the module succeeds.

Secure Files and More with GPG2

12. The command that lists currently loaded public keys is **gpg --list-key**. The **gpg2 --list-key** command is also acceptable.

LAB ANSWERS

Lab 1

Verifying this lab should be straightforward. If it works, you should be able to confirm with the following command on the tester1.example.com system:

```
$ gpg --list-keys
```

It should include the GPG2 public key just imported to that system. Of course, if the encryption, file transfer, and decryption worked, you should also be able to read the decrypted text file in a local text editor.

Lab 2

This lab is somewhat self-explanatory, in that it can help you think about how to make a system more secure. As discussed in the chapter, you can start with a minimal installation. The minimal installation of RHEL 7 happens to include the SSH server for remote administrative access.

While RHEL 7 has greatly reduced the number of standard services installed, most users will find some services that are not required. For example, how many administrators actually need Bluetooth services for a RHEL 7 system installed on a virtual machine?

Lab 3

If you want to set up a RHEL computer as a secure web server, it's a straightforward process that's described in Chapter 14. However, firewall configuration is part of the process covered in this chapter. To that end, you'll want to set up a firewall to block all but the most essential ports. This should include TCP/IP ports 80 and 443, which allow outside computers to access local regular and secure web services. Open ports should also include port 22 for SSH communication.

The easiest way to set this up is with the Red Hat Firewall Configuration tool, which you can start with the **firewall-config** command. Once in the Firewall Configuration tool, take the following steps, which vary slightly between the GUI- and console-based versions of the tool:

1. Set the Configuration mode to Permanent.
2. Select the "public" zone. Click the <interface> tab and ensure that the system network interfaces are listed under this zone.
3. Select the Services tab and enable the http service. This allows access from outside the local computer to the local regular website. Activate the https option as well. Make sure the ssh option remains active.
4. Click Options | Reload Firewalld to apply the changes to the run-time configuration.
5. Enter the following command to check the resulting firewall:

   ```
   # firewall-cmd --list-all
   ```

6. Once you've configured a web service as described in Chapter 14, users will be able to access both the regular and secure web servers from remote systems.

Lab 4

The following steps demonstrate two different methods to limit access to the noted system on IP address 192.168.122.150. Any of the two methods would be acceptable. These methods secure vsFTP in two ways: through TCP Wrappers and with the appropriate firewall commands. In a "real-world" scenario, you might use all methods in a layered security strategy. These steps assume you're performing this lab on the server1.example.com system.

1. Make sure that the vsftpd RPM is installed.
2. Start the FTP service. Use the **systemctl start vsftpd** command.
3. Back up the current version of the /etc/hosts.deny file. Open that file in a text editor. Add the **vsftpd : ALL EXCEPT 192.168.122.150** line.
4. Try accessing the FTP service from the computer with the IP address of 192.168.122.50. What happens? Try again from a different computer on your LAN.
5. Restore the previous /etc/hosts.deny file.
6. Block the FTP service for all IP addresses except 192.168.122.150 with the following commands:

```
# firewall-cmd --permanent --add-rich-rule='rule family=ipv4 source ↵
address=192.168.122.150 service name=ftp drop'
# firewall-cmd --permanent --add-service=ftp
```

7. Promote the permanent configuration into the run-time configuration with the **firewall-cmd --reload** command.
8. Try accessing the FTP server from the computer with the IP address of 192.168.122.150. What happens? Try again from a different computer on the LAN.
9. Remove the firewall rules you have added.
10. Bonus: Repeat these commands for the SSH service on port 22.

Lab 5

To confirm that TCP Wrappers can be used to help protect the SSH service, run the following command:

```
# ldd /usr/sbin/sshd | grep libwrap
```

Output, which includes a reference to the libwrap.so.0 library, confirms a link to the TCP Wrappers library. In general, it's safest to deny access to all services by including the following entry in the /etc/hosts.deny file:

```
ALL : ALL
```

You can then set up access to the SSH service with a line like the following in the /etc/hosts.allow file:

```
sshd : 192.168.122.50
```

While in most cases the use of the fully qualified domain name for the noted IP address (server1 .example.com) should work too, the use of the IP address is often appropriate. Limits by IP address don't depend on connections to DNS servers or reverse DNS being accurate.

Of course, this is not the only way to limit access to the SSH service to one system. It's possible within the /etc/hosts.deny file with a directive such as the following:

```
sshd : ALL EXCEPT 192.168.122.50
```

It's possible to set this up with other security options such as the firewalld zone–based service.

Lab 6

Before this lab can work, you'll need to activate one SELinux boolean: **ftp_home_dir**. It's listed in the SELinux Administration tool as "Determine whether ftpd can read and write files in user home directories." Therefore, with the key boolean identified, you should be able to set up vsFTP as described.

The description in this lab should point you to the /etc/pam.d/vsftpd configuration file. The model command line in this file is

```
auth required  pam_listfile.so item=user sense=deny ↵
file=/etc/vsftpd/ftpusers onerr=succeed
```

which points to the /etc/vsftpd/ftpusers file, a list of users to "deny" access to. As the conditions in the lab suggest that you need a list of (one) user to which access is to be allowed, a second line of a similar type in this file is appropriate. For example,

```
auth required  pam_listfile.so item=user sense=allow ↵
file=/etc/vsftpd/testusers onerr=succeed
```

allows all users listed in the /etc/vsftpd/testusers file. The **onerr=succeed** directive means that the vsFTP server still works if there's an error elsewhere. For example, if there is no testusers file in the /etc/vsftpd directory, the directives in this line are forgiving, allowing the conditions for the **auth** module type to succeed.

As an experiment, try this lab with the boolean **ftp_home_dir** variable set and unset. That should demonstrate the power of SELinux and serve as an appropriate preview of Chapter 11.