Comprehensive, hands-on training that solves real world problems

Red Hat Enterprise Linux Diagnostics and Troubleshooting

Student Workbook (ROLE)

RH342-RHEL7.2-en-1-20160321

RED HAT ENTERPRISE LINUX DIAGNOSTICS AND TROUBLESHOOTING

Red Hat Enterprise Linux 7.2 RH342
Red Hat Enterprise Linux Diagnostics and Troubleshooting
Edition 1 20160321

Authors: Wander Boessenkool, Chen Chang, George Hacker
Editor: Steven Bonneville

This instructional program, including all material provided herein, is supplied without any guarantees from Red Hat, Inc. Red Hat, Inc. assumes no liability for damages or legal action arising from the use or misuse of contents or details contained herein.

If you believe Red Hat training materials are being used, copied, or otherwise improperly distributed please e-mail training@redhat.com or phone toll-free (USA) +1 (866) 626-2994 or +1 (919) 754-3700.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, JBoss, Hibernate, Fedora, the Infinity Logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux® is the registered trademark of Linus Torvalds in the United States and other countries.

Java® is a registered trademark of Oracle and/or its affiliates.

XFS® is a registered trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

The OpenStack® Word Mark and OpenStack Logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Contributors: Andrew Blum

Document Conventions

Notes and Warnings

Note

"Notes" are tips, shortcuts or alternative approaches to the task at hand. Ignoring a note should have no negative consequences, but you might miss out on a trick that makes your life easier.

Important

"Important" boxes detail things that are easily missed: configuration changes that only apply to the current session, or services that need restarting before an update will apply. Ignoring a box labeled "Important" will not cause data loss, but may cause irritation and frustration.

Warning

"Warnings" should not be ignored. Ignoring warnings will most likely cause data loss.

References

"References" describe where to find external documentation relevant to a subject.

Introduction

Red Hat Enterprise Linux Diagnostics and Troubleshooting

Red Hat Enterprise Linux Diagnostics and Troubleshooting (RH342) provides system administrators with the tools and techniques they need to successfully diagnose, and fix, a variety of issues that can present themselves. Students will work through problems in various subsystems to diagnose and fix common issues. This approach is then used troubleshooting various types of problems, including boot issues, hardware issues, storage issues, RPM issues, network issues, third-party application issues, security issues, and kernel issues. At the end of the course students can complete various comprehensive review labs to test their skills.

Objectives

- To diagnose problems in various subsystems on RHEL7 systems, using tools provided with the distribution.
- To gather information to assist Red Hat Support in diagnosing, and fixing, issues that can occur on a Red Hat Enterprise Linux system.
- To fix common issues on a Red Hat Enterprise Linux machine, using tools provided by the distribution.
- To prepare students for the Red Hat Enterprise Linux Diagnostics and Troubleshooting Certification Exam (EX342).

Audience

- The Red Hat Enterprise Linux Diagnostics and Troubleshooting course is aimed at senior system administrators who wish to learn more about troubleshooting.

Prerequisites

- A RHCSA certification, or equivalent knowledge is required to successfully sit this course.
- A RHCE certification, or equivalent knowledge, is recommended to successfully sit this course.

Orientation to the Classroom Lab Environment

In this course, students will do most hands-on practice exercises and lab work with three computer systems, which will be referred to as **workstation**, **servera**, and **serverb**. These machines have host names **workstation.lab.example.com**, **servera.lab.example.com**, and **serverb.lab.example.com**. All machines have a standard user account, *student*, with the password *student*. The *root* password on both systems is *redhat*.

In a Red Hat Online Learning classroom, students will be assigned remote computers which will be accessed through a web application hosted at rol.redhat.com. Students should log into this machine using the user credentials they provided when registering for the class.

The systems used by each student use a private IPv4 subnet. For each student, their IPv4 network is 172.25.250.0/24.

Classroom Machines

Machine name	IP addresses	Role
workstation.lab.example.com	172.25.250.254	Student “client” computer
servera.lab.example.com	172.25.250.10	Student first “server” computer
serverb.lab.example.com	172.25.250.11	Student second “server” computer

Controlling your stations

The top of the console describes the state of your machine.

Machine States

State	Description
none	Your machine has not yet been started. When started, your machine will boot into a newly initialized state (the disk will have been reset).
starting	Your machine is in the process of booting.
running	Your machine is running and available (or, when booting, soon will be.)
stopping	Your machine is in the process of shutting down.
stopped	Your machine is completely shut down. Upon starting, your machine will boot into the same state as when it was shut down (the disk will have been preserved).
impaired	A network connection to your machine cannot be made. Typically this state is reached when a student has corrupted networking or firewall rules. If the condition persists after a machine reset, or is intermittent, please open a support case.

Depending on the state of your machine, a selection of the following actions will be available to you.

Machine Actions

Action	Description
Start Station	Start ("power on") the machine.

Action	Description
Stop Station	Stop ("power off") the machine, preserving the contents of its disk.
Reset Station	Stop ("power off") the machine, resetting the disk to its initial state. **Caution: Any work generated on the disk will be lost**.
Refresh	Refresh the page will re-probe the machine state.
Increase Timer	Adds 15 minutes to the timer for each click.

The station timer

Your Red Hat Online Learning enrollment entitles you to a certain amount of computer time. In order to help you conserve your time, the machines have an associated timer, which is initialized to 60 minutes when your machine is started.

The timer operates as a "dead man's switch," which decrements as your machine is running. If the timer is winding down to 0, you may choose to increase the timer.

Internationalization

Language support

Red Hat Enterprise Linux 7 officially supports 22 languages: English, Assamese, Bengali, Chinese (Simplified), Chinese (Traditional), French, German, Gujarati, Hindi, Italian, Japanese, Kannada, Korean, Malayalam, Marathi, Odia, Portuguese (Brazilian), Punjabi, Russian, Spanish, Tamil, and Telugu.

Per-user language selection

Users may prefer to use a different language for their desktop environment than the system-wide default. They may also want to set their account to use a different keyboard layout or input method.

Language settings

In the GNOME desktop environment, the user may be prompted to set their preferred language and input method on first login. If not, then the easiest way for an individual user to adjust their preferred language and input method settings is to use the **Region & Language** application. Run the command **`gnome-control-center region`**, or from the top bar, select **(User)** > **Settings**. In the window that opens, select **Region & Language**. The user can click the **Language** box and select their preferred language from the list that appears. This will also update the **Formats** setting to the default for that language. The next time the user logs in, these changes will take full effect.

These settings affect the GNOME desktop environment and any applications, including **`gnome-terminal`**, started inside it. However, they do not apply to that account if accessed through an **`ssh`** login from a remote system or a local text console (such as **`tty2`**).

Note

A user can make their shell environment use the same **`LANG`** setting as their graphical environment, even when they log in through a text console or over **`ssh`**. One way to do this is to place code similar to the following in the user's **`~/.bashrc`** file. This example code will set the language used on a text login to match the one currently set for the user's GNOME desktop environment:

```
i=$(grep 'Language=' /var/lib/AccountService/users/${USER} \
  | sed 's/Language=//')
if [ "$i" != "" ]; then
    export LANG=$i
fi
```

Japanese, Korean, Chinese, or other languages with a non-Latin character set may not display properly on local text consoles.

Individual commands can be made to use another language by setting the **`LANG`** variable on the command line:

```
[user@host ~]$ LANG=fr_FR.utf8 date
```

```
jeu. avril 24 17:55:01 CDT 2014
```

Subsequent commands will revert to using the system's default language for output. The **locale** command can be used to check the current value of **LANG** and other related environment variables.

Input method settings

GNOME 3 in Red Hat Enterprise Linux 7 automatically uses the **IBus** input method selection system, which makes it easy to change keyboard layouts and input methods quickly.

The **Region & Language** application can also be used to enable alternative input methods. In the **Region & Language** application's window, the **Input Sources** box shows what input methods are currently available. By default, **English (US)** may be the only available method. Highlight **English (US)** and click the **keyboard** icon to see the current keyboard layout.

To add another input method, click the **+** button at the bottom left of the **Input Sources** window. An **Add an Input Source** window will open. Select your language, and then your preferred input method or keyboard layout.

Once more than one input method is configured, the user can switch between them quickly by typing **Super**+**Space** (sometimes called **Windows**+**Space**). A *status indicator* will also appear in the GNOME top bar, which has two functions: It indicates which input method is active, and acts as a menu that can be used to switch between input methods or select advanced features of more complex input methods.

Some of the methods are marked with gears, which indicate that those methods have advanced configuration options and capabilities. For example, the Japanese **Japanese (Kana Kanji)** input method allows the user to pre-edit text in Latin and use **Down Arrow** and **Up Arrow** keys to select the correct characters to use.

US English speakers may find also this useful. For example, under **English (United States)** is the keyboard layout **English (international AltGr dead keys)**, which treats **AltGr** (or the right **Alt**) on a PC 104/105-key keyboard as a "secondary-shift" modifier key and dead key activation key for typing additional characters. There are also Dvorak and other alternative layouts available.

Note

Any Unicode character can be entered in the GNOME desktop environment if the user knows the character's Unicode code point, by typing **Ctrl**+**Shift**+**U**, followed by the code point. After **Ctrl**+**Shift**+**U** has been typed, an underlined **u** will be displayed to indicate that the system is waiting for Unicode code point entry.

For example, the lowercase Greek letter lambda has the code point U+03BB, and can be entered by typing **Ctrl**+**Shift**+**U**, then **03bb**, then **Enter**.

System-wide default language settings

The system's default language is set to US English, using the UTF-8 encoding of Unicode as its character set (**en_US.utf8**), but this can be changed during or after installation.

From the command line, *root* can change the system-wide locale settings with the **localectl** command. If **localectl** is run with no arguments, it will display the current system-wide locale settings.

To set the system-wide language, run the command **localectl set-locale LANG=*locale***, where *locale* is the appropriate **$LANG** from the "Language Codes Reference" table in this chapter. The change will take effect for users on their next login, and is stored in **/etc/locale.conf**.

```
[root@host ~]# localectl set-locale LANG=fr_FR.utf8
```

In GNOME, an administrative user can change this setting from **Region & Language** and clicking the **Login Screen** button at the upper-right corner of the window. Changing the **Language** of the login screen will also adjust the system-wide default language setting stored in the **/etc/locale.conf** configuration file.

Important

Local text consoles such as **tty2** are more limited in the fonts that they can display than **gnome-terminal** and **ssh** sessions. For example, Japanese, Korean, and Chinese characters may not display as expected on a local text console. For this reason, it may make sense to use English or another language with a Latin character set for the system's text console.

Likewise, local text consoles are more limited in the input methods they support, and this is managed separately from the graphical desktop environment. The available global input settings can be configured through **localectl** for both local text virtual consoles and the X11 graphical environment. See the **localectl**(1), **kbd**(4), and **vconsole.conf**(5) man pages for more information.

Language packs

When using non-English languages, you may want to install additional "language packs" to provide additional translations, dictionaries, and so forth. To view the list of available langpacks, run **yum langavailable**. To view the list of langpacks currently installed on the system, run **yum langlist**. To add an additional langpack to the system, run **yum langinstall *code***, where *code* is the code in square brackets after the language name in the output of **yum langavailable**.

References

locale(7), **localectl**(1), **kbd**(4), **locale.conf**(5), **vconsole.conf**(5), **unicode**(7), **utf-8**(7), and **yum-langpacks**(8) man pages

Conversions between the names of the graphical desktop environment's X11 layouts and their names in **localectl** can be found in the file **/usr/share/X11/xkb/rules/base.lst**.

Language Codes Reference

Language Codes

Language	$LANG value
English (US)	en_US.utf8
Assamese	as_IN.utf8
Bengali	bn_IN.utf8
Chinese (Simplified)	zh_CN.utf8
Chinese (Traditional)	zh_TW.utf8
French	fr_FR.utf8
German	de_DE.utf8
Gujarati	gu_IN.utf8
Hindi	hi_IN.utf8
Italian	it_IT.utf8
Japanese	ja_JP.utf8
Kannada	kn_IN.utf8
Korean	ko_KR.utf8
Malayalam	ml_IN.utf8
Marathi	mr_IN.utf8
Odia	or_IN.utf8
Portuguese (Brazilian)	pt_BR.utf8
Punjabi	pa_IN.utf8
Russian	ru_RU.utf8
Spanish	es_ES.utf8
Tamil	ta_IN.utf8
Telugu	te_IN.utf8

CHAPTER 1

WHAT IS TROUBLESHOOTING

Overview	
Goal	Describe a generalized strategy for troubleshooting.
Objectives	• Identify a systematic approach to troubleshooting using the scientific method. • Collect various pieces of information to aid in troubleshooting. • Use Red Hat resources to aid in troubleshooting.
Sections	• Using the Scientific Method (and Guided Exercise) • Collecting Information (and Guided Exercise) • Using Red Hat Resources (and Guided Exercise)
Lab	• What Is Troubleshooting?

Using the Scientific Method

Objectives

After completing this section, students should be able to use a systematic approach to troubleshooting using the scientific method.

What is troubleshooting?

Troubleshooting is the art of taking a problem, gathering information about it, analyzing it, and finally solving it. While some problems are inherently "harder" than others, the same basic approach can be taken for every problem.

Using a fixed approach ensures that critical steps are not left out, and that troubleshooting becomes a repeatable process.

Not just fixing

While fixing a problem is one of the major parts of troubleshooting, there are other parts that cannot be neglected: documenting the problem (and fix), and performing a *root cause analysis* (RCA).

Documenting the problem (and the fix) can help in the future when another (or possibly the same) administrator is faced with the same, or a similar, problem. Performing a root cause analysis can help in preventing similar problems in the future.

Using the scientific method

A good schema to follow when troubleshooting is the scientific method:

1. Clearly define the issue

   Take a step and view the larger picture, then clearly define the actual problem. Most problems that get reported are symptoms of another problem, not the actual problem.

   For example, a user might call about a problem signing into a machine. While this is a problem for the user, the actual problem can be a forgotten password, an incorrectly configured machine, a networking issue, or something else entirely. Further investigation is needed to determine what the cause of this symptom is.

   A typical action during this process is attempting to recreate the issue, or observing the issue as it happens.

2. Collect information

   The next step is collecting as much (relevant) information as possible. This information can come from a wide variety of sources: reading log files, information displayed on screen or in a GUI, follow-up questions for the original reporter, etc.

   During this step, "relevant information" is still a fairly broad term; information from seemingly unconnected subsystems can turn out to be useful.

   On the other hand, gathering *too much* information is also unnecessary and might even be counterproductive, as all information will need to be viewed, assessed, and interpreted.

3. Form a hypothesis

   After looking at all gathered information, and the symptoms observed/reported, it is time to form a hypothesis about the cause of the problem.

   Sometimes this can be easy; for example, when a user has forgotten his password. Other times, it can be harder; for example, when a single service in a high-availability cluster fails to start on Mondays during months with an "e" in their name. The key to remember during this step is that the hypothesis is just that, a hypothesis: a best guess as to what can be the cause of the issue. During the following steps, this hypothesis will be tested. If it turns out the hypothesis was wrong, a new one can be formed.

4. Test the hypothesis

   With an initial hypothesis formed, it can be tested for validity. How this testing happens depends on the problem and the hypothesis.

   For example, when the hypothesis for a login problem states, "The network connection between the workstation and the KDC is being interrupted by a firewall," the testing will be different from a hypothesis for a spontaneously rebooting server including a faulty UPS.

   If, after testing, the hypothesis still appears valid, an attempt at fixing the problem can be made. If the hypothesis is found to be invalid, a new hypothesis will need to be formed, possibly using extra information found during the testing of the previous hypothesis.

5. Fixing the problem

   If a hypothesis was not found to be invalid, an attempt can be made to fix the problem. During this stage, it is vital to only change one variable at a time, documenting all changes made, and testing every change individually.

   Keeping backups of any changed configuration files, and reverting to those backups if a change was found to be ineffective, is also crucial. Modifying multiple configurations at once typically only leads to further issues, not fixes.

   After any change, the entire configuration will need to be tested to see if the issue has been resolved, reverting any changes, and possibly forming a new hypothesis, if a change has not resolved the issue.

6. Rinse & repeat

   If the proposed fixes did not actually resolve the issue, the process will need to be restarted from the top. This time, any new information discovered during this cycle can be added to the mix to form a new hypothesis.

Guided Exercise: Using the Scientific Method

In this lab, you will solve a login issue using the scientific method.

Resources	
Machines	• **workstation** • **servera**

Outcome(s)

You should be able to troubleshoot a login issue using the scientific method.

Before you begin

On your **workstation** system, execute the command **lab scientificmethod setup** to prepare your systems for this exercise.

```
[student@workstation ~]$ lab scientificmethod setup
```

Over the weekend, some of your colleagues have been running user maintenance. This morning, one of your users, **student**, calls the help desk to complain that his login is no longer working on **servera**.

The user normally logs in with the account name **student**, and, against all security protocols, the user has informed you that his password is also **student**.

Furthermore, the user has also informed you that passwordless login over **ssh** normally works from the **student** account on **workstation**.

Looking up the default settings for users of this class, you note that the home directory for this user should be set to **/home/student**, the UID and GID should both be **1000**, and the user prefers the **bash** shell.

1. Begin by reproducing the problem, noting as much detail and information as possible about the issue.

    1.1. From a command prompt on **workstation**, attempt to use **ssh** to log into the **student** account on **servera**. What do you notice, and what does this tell you?

    ```
    [student@workstation ~]$ ssh student@servera
    Last login: Wed Dec  9 14:02:23 from workstation.lab.example.com
    Connection to server closed.
    ```

    The connection is closed immediately, but authentication did go through.

    1.2. Open a console window to **servera** and attempt to log in as **student** with the password **student**. What happens?

    The login appears to go through fine without authentication errors, but the session is closed immediately.

2. Now that we know that **student** can authenticate successfully, but the resulting session is closed immediately, it is time to gather some more information on the **servera** system.

2.1. Attempt to **ssh** into the **root** account on the **servera** system from **workstation**. If needed, the password should be **redhat**.

```
[student@workstation ~]$ ssh root@servera
```

2.2. As **root** on **servera**, attempt to use **su -** to switch to the **student** account.

```
[root@servera ~]# su - student
Last login: Wed Dec  9 14:32:08 CET 2015 from workstation.lab...
```

2.3. Since the **root** account still works, you can use that to gather some extra information on the **student** account. Verify when **student** last successfully logged in.

```
[root@servera ~]# lastlog -u student
Username         Port     From             Latest
student          pts/0    workstation.lab. Wed Dec  9 14:32:08 +0100 2015
```

According to this output, the user **student** did successfully log in during our previous tests, confirming the suspicion that something is ending the session prematurely.

2.4. As **root** on **servera**, use the **getent** tool to inspect the settings for the **student** user.

```
[root@servera ~]# getent passwd student
student:x:1000:1000:Student User:/home/student:/bin/false
```

3. Using the information gathered in the previous step, form a hypothesis as to what could be wrong with the **student** account.

3.1. Looking at the **getent** output, note that the user's shell is set to **/bin/false**. This is not a valid shell, and also not the shell the user prefers.

4. Reset the shell for **student** to **/bin/bash**, then verify if the problem has been solved.

4.1. Reset the shell for **student** on **servera** to **/bin/bash**.

```
[root@servera ~]# chsh -s /bin/bash student
Changing shell for student.
Shell changed.
```

4.2. Attempt to **ssh** from **workstation** to **student** on **servera**.

```
[student@workstation ~]$ ssh student@servera
```

5. Use the command **lab scientificmethod grade** on **workstation** to verify your work.

```
[student@workstation ~]$ lab scientificmethod grade
```

If you did not successfully complete this exercise, try again, or use the command **lab scientificmethod solve** to solve the problem for you.

```
[student@workstation ~]$ lab scientificmethod solve
```

Collecting Information

Objectives

After completing this section, students should be able to collect various pieces of system information to aid in troubleshooting.

Various forms of information

When approaching a troubleshooting problem, there are various pieces of information that can and/or need to be collected before a hypothesis can be formed. Some of this information may need to be collected from the report of the problem, such as "What were you doing when the problem happened?" and "Did you see any error messages?".

Other information can be collected firsthand; for example, when recreating the issue, or by querying the RPM database, reading log files, etc. Not all of this information may be readily available; sometimes a service will need to be configured to increase the amount of logging it performs, or the system will need to be configured to store information that would normally be discarded upon a reboot.

Using the system journal

By default, a Red Hat Enterprise Linux 7 system uses two logging services for the system logs: **systemd-journald**, which is configured to only keep logs in memory, and **rsyslogd**, which gets messages sent to it by **systemd-journald** (and others) and stores them on disk.

To view messages in the system journal, a tool called **`journalctl`** can be used. If used without any parameters it will show the full contents of the system journal, presented in a pager (by default **`less`** is used).

The output of **`journalctl`** can be modified by using both options and filters. Options can be used to change the number of lines displayed, to turn on follow mode, change the displayed field, specify a time range, etc.

Filters can be used to modify for what services and units information is displayed, which executables to display information for, etc.

journalctl examples

`journalctl -ef`

Jump to the end of the journal (**`-e`**, and enable follow mode (**`-f`**). This will keep the journal open on screen, displaying new messages as they come in.

`journalctl _SYSTEMD_UNIT=sshd.service`

This will display all messages generated by the **`sshd.service`** systemd unit.

`journalctl -u sshd.service`

This will display all messages generated by, *and about*, the **`sshd.service`** systemd unit.

`journalctl -p emerg..err`

Display all messages in the journal with a priority in the range **`emerg`** up to and including **`err`**.

If a single priority is specified, for example, **`-p err`**, all messages up to and including that level are displayed.

`journalctl -b -1`

Only show messages from the last system boot. This is useful for searching for information about a system crash.

This requires a persistent journal to be configured.

`journalctl --since "2015-02-02 20:30:00" --until "2015-03-31 12:00:00"`

Displays all messages between February 2, half past eight in the evening, and noon on March 31st.

This requires a persistent journal to be configured.

`journalctl -o verbose`

Use verbose output mode (**`-o verbose`**). This will show all fields stored in the journal with their field name and contents.

All field names can be used as filters on the **`journalctl`** command line.

For a complete list of options and filters, refer to the **`journalctl`**(1) man page.

Persisting the journal

By default, Red Hat Enterprise Linux 7 stores the system journal in **`/run/log/journal`**, which is stored on a **`tmpfs`**. This implies that on a reboot all stored information will be lost. If the directory **`/var/log/journal`** is present the journal will be stored there, thus enabling a persistent journal across reboots.

Enabling a persistent journal can be done by using the following steps:

1. Create the directory **`/var/log/journal`**.

   ```
   [root@demo ~]# mkdir /var/log/journal
   ```

2. Set the group ownership of the new directory to **`systemd-journal`**, and the permissions to **`2755`**.

   ```
   [root@demo ~]# chown root:systemd-journal /var/log/journal
   [root@demo ~]# chmod 2755 /var/log/journal
   ```

3. Inform **`systemd-journald`** that the new location should be used by sending a **`USR1`** signal to it. A reboot will also suffice.

   ```
   [root@demo ~]# killall -USR1 systemd-journald
   ```

Enabling verbose information

Many tools and services can increase the amount of logging they perform, as well as the amount of information they display when run from the command line, by using various configuration options or command-line flags.

Command-line options typically include **-v**, which can sometimes be specified multiple times, to increase verbosity, or include a **--debug** option that can be used.

Services will typically have configuration options, either in their main configuration file or in **/etc/sysconfig/*SERVICENAME***, that can be used to increase their logging level and/or verbosity as well.

Refer to the documentation for these individual services to increase their verbosity and logging levels.

Warning

When using the debug option for a service in **/etc/sysconfig/*SERVICENAME***, that option will sometimes also stop the daemon from disconnecting from the terminal. When such a service is started using **systemctl**, and the service type is set to **forking**, the **systemctl** command will not return until the service is killed by pressing **Ctrl**+**C**. In these cases, running a service manually from the command line can be an option too.

References

journalctl(1), **systemd.journal-fields**(7), and **systemd-journald.service**(8) man pages.

Guided Exercise: Collecting Information

In this lab, you will use log files to troubleshoot an issue with a web server.

Resources	
Files	**`/var/www/html/test.html`**
Machines	• **`workstation`** • **`servera`**

Outcome(s)

You should be able to use log files to troubleshoot a web server issue.

Before you begin

Prepare your systems for this exercise by running the command **`lab logread setup`** on your **`workstation`** machine.

```
[student@workstation ~]$ lab logread setup
```

Your **`servera`** machine is running a web server, serving the file **`http://servera.lab.example.com/test.html`**. A ticket just came in from your testing manager that this file is not accessible from a web browser. No further information has been given in the ticket.

Investigate this issue using the log files on **`servera`**, then fix the issue. For testing from the command line on **`workstation`**, if you do not want to open a graphical browser, you can use the command **`elinks -dump http://servera.lab.example.com/test.html`**.

1. Begin by trying to reproduce the problem.

   1.1. As **`student`** on **`workstation`**, attempt to access **`http://servera.lab.example.com/test.html`**. You can do this using a **Firefox** browser, or by executing the following command:

   ```
   [student@workstation ~]$ elinks -dump http://servera.lab.example.com/test.html
   ```

   1.2. Think about the possible causes for the HTTP 403 forbidden error you just encountered. This can have a number of reasons: file permissions, SELinux types, internal **httpd** configurations, etc.

   You do know that the web server itself is running, you got an answer, and that the firewall is open.

2. Collect information from the web server logs on **`servera`**. The main logs for **httpd** are **`/var/log/httpd/access_log`** for all access attempts, and **`/var/log/httpd/error_log`** for all errors.

   2.1. Check **`/var/log/httpd/access_log`** for any message related to this failure.

   ```
   [root@servera ~]# grep test.html /var/log/httpd/access_log
   ...
   ```

```
172.25.250.254 - - [10/Dec/2015:11:28:34 +0100] "GET /test.html HTTP/1.1" 403
 211 "-" "Elinks/0.12pre6 (textmode; Linux; -)"
...
```

The **403** in this output is the HTTP status code. Other than that, you can see the requested URL, date and time of the request, and the user-agent used, but nothing that can help you further with this problem.

2.2. Check **/var/log/httpd/error_log** for any message related to this failure.

```
[root@servera ~]# tail /var/log/httpd/error_log
...
[Thu Dec 10 11:28:34.378778 2015] [core:error] [pid 2028] (13)Permission Denied:
 [client 172.25.250.254:57245] AH00132: file permissions deny server access: /
var/www/html/test.html
...
```

This message tells you that **httpd** is blocked by file permissions from reading the **test.html** file. This rules out an internal configuration error for **httpd**, but leaves file permissions and SELinux as possible culprits.

3. Inspect the file permissions on **/var/www/html/test.html**, and fix if necessary.

3.1. Inspect the file permissions on **/var/www/html/test.html**.

```
[root@servera ~]# ls -l /var/www/html/test.html
-rw-------. 1 root root 5 Dec 10 11:27 /var/www/html/test.html
```

3.2. Those permissions do not look correct. Make the file world-readable.

```
[root@servera ~]# chmod 644 /var/www/html/test.html
```

3.3. Test access to the file again, either with **Firefox** or **elinks**.

```
[student@workstation ~]$ elinks -dump http://servera.lab.example.com/test.html
```

3.4. File permissions were an issue, but the problem is still not solved. This leaves one likely culprit: SELinux.

4. Check the SELinux log for any denials that happened today, and fix any issues you might spot.

4.1. Check the SELinux log for any denials today.

```
[root@servera ~]# ausearch -i -m avc -ts today
...
type=AVC msg=audit(12/10/2015 10:41:19.890:3045) : avc: denied {open} for
 pid=32712 comm=httpd path=/var/www/html/test.html dev="vda1" ino=25245202
 scontext=system_u:system_r:httpd_t:s0 tcontext=unconfined_u:object_r:tmp_t:s0
 tclass=file
...
```

This shows that the **test.html** file has an SELinux type of **tmp_t**, which **httpd** is not allowed to open.

4.2. Fix this issue by running a recursive **restorecon** on **/var/www**.

```
[root@servera ~]# restorecon -Rv /var/www
```

4.3. Test if you can now access **http://servera.lab.example.com/test.html** from **workstation**.

```
[student@workstation ~]$ elinks -dump http://servera.lab.example.com/test.html
```

5. Grade your work, then clean up your systems for the next exercise.

5.1. Grade your work.

```
[student@workstation ~]$ lab logread grade
```

5.2. Clean up your systems for the next exercise.

```
[student@workstation ~]$ lab logread reset
```

Using Red Hat Resources

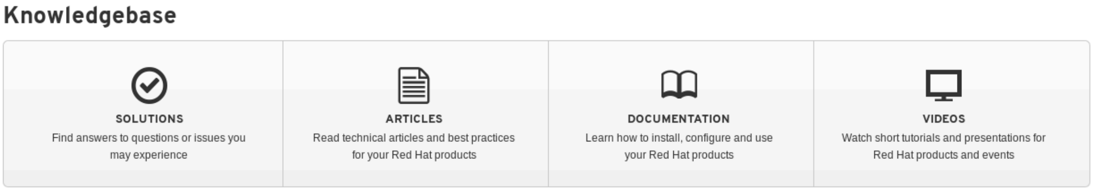
Objectives

After completing this section, students should be able to use Red Hat resources to aid in troubleshooting.

Red Hat Customer Portal

Red Hat Customer Portal (**`https://access.redhat.com`**) provides customers with access to everything provided with their subscription through one convenient location. Customers can search for solutions, FAQs, and articles through Knowledgebase. Access to official product documentation is provided. Support tickets can be submitted and managed. Subscriptions to Red Hat products can be attached to and detached from registered systems, and software downloads, updates, and evaluations can be obtained. Parts of the site are accessible to everyone, while others are exclusive to customers with active subscriptions. Help with getting access to Customer Portal is available at **`https://access.redhat.com/help/`**.

Customers can work with Red Hat Customer Portal through a web browser. This section will introduce a command-line tool that can also be used to access Red Hat Customer Portal services, **`redhat-support-tool`**.



Knowledgebase

SOLUTIONS
Find answers to questions or issues you may experience

ARTICLES
Read technical articles and best practices for your Red Hat products

DOCUMENTATION
Learn how to install, configure and use your Red Hat products

VIDEOS
Watch short tutorials and presentations for Red Hat products and events

Figure 1.1: Knowledgebase at the Red Hat Customer Portal

Collecting information with sosreport

Red Hat ships a tool called **`sosreport`** in the package *sos*. This tool can collect various bits of system information, including log files and configuration files, and package them up in a tarball ready to ship to Red Hat Support.

System administrators can also use this tool to quickly gather information on their own systems.

When run without options, **`sosreport`** will prompt the user for his or her initials, a case number, and then proceed to collect a default set of files and settings in a tarball. Command-line options can be used to enable or disable plug-ins, configure plug-in options, and to make the process noninteractive.

A full overview of all plug-ins, and whether they are currently enabled or disabled, along with any configurable plug-in options, can be requested with **`sosreport -l`**. When running a report, the **`-e`** ***`PLUGINS`*** option can be used to enable additional plug-ins, and the **`-k`** ***`PLUGOPTS`*** option can be used to set plug-in options. The **`-n`** ***`NOPLUGINS`*** option can be used to disable unwanted plug-ins.

Using redhat-support-tool to search the Knowledgebase

The Red Hat Support Tool utility **redhat-support-tool** provides a text-console interface to the subscription-based Red Hat Access services. Internet access is required to reach the Red Hat Customer Portal. The **redhat-support-tool** is text-based for use from any terminal or SSH connection; no graphical interface is provided.

The **redhat-support-tool** command may be used as an interactive shell or invoked as individually executed commands with options and arguments. The tool's available syntax is identical for both methods. By default, the program launches in shell mode. Use the provided **help** subcommand to see all available commands. Shell mode supports tab completion and the ability to call programs in the parent shell.

```
[student@demo ~]$ redhat-support-tool
Welcome to the Red Hat Support Tool.
Command (? for help):
```

When first invoked, **redhat-support-tool** prompts for required Red Hat Access subscriber login information. To avoid repetitively supplying this information, the tool asks to store account information in the user's home directory (**~/.redhat-support-tool/redhat-support-tool.conf**). If a Red Hat Access account is shared by many users, the **--global** option can save account information to **/etc/redhat-support-tool.conf**, along with other systemwide configuration. The tool's **config** command modifies tool configuration settings.

The **redhat-support-tool** allows subscribers to search and display the same Knowledgebase content seen when on the Red Hat Customer Portal. Knowledgebase permits keyword searches, similar to the **man** command. Users can enter error codes, syntax from log files, or any mix of keywords to produce a list of relevant solution documents.

The following is an initial configuration and basic search demonstration:

```
[student@demo ~]$ redhat-support-tool
Welcome to the Red Hat Support Tool.
Command (? for help): search How to manage system entitlements with subscription-manager
Please enter your RHN user ID: subscriber
Save the user ID in /home/student/.redhat-support-tool/redhat-support-tool.conf (y/n): y
Please enter the password for subscriber: password
Save the password for subscriber in /home/student/.redhat-support-tool/redhat-support-
tool.conf (y/n): y
```

After prompting the user for the required user configuration, the tool continues with the original search request:

```
Type the number of the solution to view or 'e' to return to the previous menu.
  1 [ 253273:VER] How to register and subscribe a system to Red Hat Network
    (RHN) using Red Hat Subscription Manager (RHSM)?
  2 [  17397:VER] What are Flex Guest Entitlements in Red Hat Network?
  3 [ 232863:VER] How to register machines and manage subscriptions using Red
    Hat Subscription Manager through an invisible HTTP proxy / Firewall?
3 of 43 solutions displayed. Type 'm' to see more, 'r' to start from the beginning
 again, or '?' for help with the codes displayed in the above output.
Select a Solution:
```

Specific sections of solution documents may be selected for viewing.

```
Select a Solution: 1

Type the number of the section to view or 'e' to return to the previous menu.
 1 Title
 2 Issue
 3 Environment
 4 Resolution
 5 Display all sections
End of options.
Section: 1

Title
===============================================================================
How to register and subscribe a system to Red Hat Network (RHN) using Red Hat
 Subscription Manager (RHSM)?
URL:        https://access.redhat.com/site/solutions/253273
(END) q
```

Directly access Knowledgebase articles by document ID

Locate online articles directly using the tool's **kb** command with the Knowledgebase document ID. Returned documents scroll on the screen without pagination, allowing a user to redirect the output using other local commands. This example views the document with the **less** command:

```
[student@demo ~]$ redhat-support-tool kb 253273 | less

Title:   How to register and subscribe a system to Red Hat Network (RHN) using Red Hat
 Subscription Manager (RHSM)?
ID:      253273
State:   Verified: This solution has been verified to work by Red Hat Customers and
 Support Engineers for the specified product version(s).
URL:     https://access.redhat.com/site/solutions/253273
: q
```

Documents retrieved in unpaginated format are easy to send to a printer, convert to PDF or other document format, or to redirect to a data entry program for an incident tracking or change management system, using other utilities installed and available in Red Hat Enterprise Linux.

Using redhat-support-tool to manage support cases

One benefit of a product subscription is access to technical support through Red Hat Customer Portal. Depending on the system's subscription support level, Red Hat may be contacted through online tools or by phone. See **https://access.redhat.com/site/support/policy/support_process** for links to detailed information about the support process.

Preparing a bug report

Before contacting Red Hat Support, gather relevant information for a bug report.

Define the problem. Be able to clearly state the problem and its symptoms. Be as specific as possible. Detail the steps that will reproduce the problem.

Gather background information. Which product and version is affected? Be ready to provide relevant diagnostic information. This can include output of **sosreport**, discussed earlier in this section. For kernel problems, this could include the system's **kdump** crash dump or a digital photo of the kernel backtrace displayed on the monitor of a crashed system.

Determine the severity level. Red Hat uses four severity levels to classify issues. *Urgent* and *High* severity problem reports should be followed by a phone call to the relevant local support center (see **`https://access.redhat.com/site/support/contact/technicalSupport`**).

Severity	Description
Urgent (Severity 1)	A problem that severely impacts use of the software in a production environment (such as loss of production data, or production systems are not functioning). The situation halts business operations and no procedural workaround exists.
High (Severity 2)	A problem where the software is functioning, but use in a production environment is severely reduced. The situation is causing a high impact to portions of the business operations and no procedural workaround exists.
Medium (Severity 3)	A problem that involves partial, noncritical loss of use of the software in a production environment or development environment. For production environments, there is a medium-to-low impact on the business, but the business continues to function, including by using a procedural workaround. For development environments, where the situation is causing the project to no longer continue or migrate into production.
Low (Severity 4)	A general usage question, reporting of a documentation error, or recommendation for a future product enhancement or modification. For production environments, there is low-to-no impact on the business or the performance or functionality of the system. For development environments, there is a medium-to-low impact on the business, but the business continues to function, including by using a procedural workaround.

Managing a bug report with `redhat-support-tool`

Subscribers may create, view, modify, and close Red Hat Support cases using **`redhat-support-tool`**. When support cases are opened or maintained, users may include files or documentation, such as diagnostic reports (**`sosreport`**). The tool uploads and attaches files to online cases. Case details including *product*, *version*, *summary*, *description*, *severity*, and *case group* may be assigned with command options or letting the tool prompt for required information. In the following example, the **`--product`** and **`--version`** options are specified, but **`redhat-support-tool`** would provide a list of choices for those options if the **`opencase`** command did not specify them.

```
[student@demo ~]$ redhat-support-tool
Welcome to the Red Hat Support Tool.
Command (? for help): opencase --product="Red Hat Enterprise Linux" --version="7.0"
Please enter a summary (or 'q' to exit): System fails to run without power
Please enter a description (Ctrl-D on an empty line when complete):
When the server is unplugged, the operating system fails to continue.
 1   Low
 2   Normal
 3   High
 4   Urgent
Please select a severity (or 'q' to exit): 4
Would you like to assign a case group to this case (y/N)? N
Would see if there is a solution to this problem before opening a support case? (y/N) N
--------------------------------------------------------------------------------
```

```
Support case 01034421 has successfully been opened.
```

Including diagnostic information by attaching a SoS report archive

Including diagnostic information when a support case is first created contributes to quicker problem resolution. The **sosreport** command generates a compressed tar archive of diagnostic information gathered from the running system. The **redhat-support-tool** prompts to include one if an archive has been created previously:

```
Please attach a SoS report to support case 01034421. Create a SoS report as
the root user and execute the following command to attach the SoS report
directly to the case:
 redhat-support-tool addattachment -c 01034421 path to sosreport

Would you like to attach a file to 01034421 at this time? (y/N) N
Command (? for help):
```

If a current SoS report is not already prepared, an administrator can generate and attach one later, using the tool's **addattachment** command as advised previously.

Support cases can also be viewed, modified, and closed by you as the subscriber:

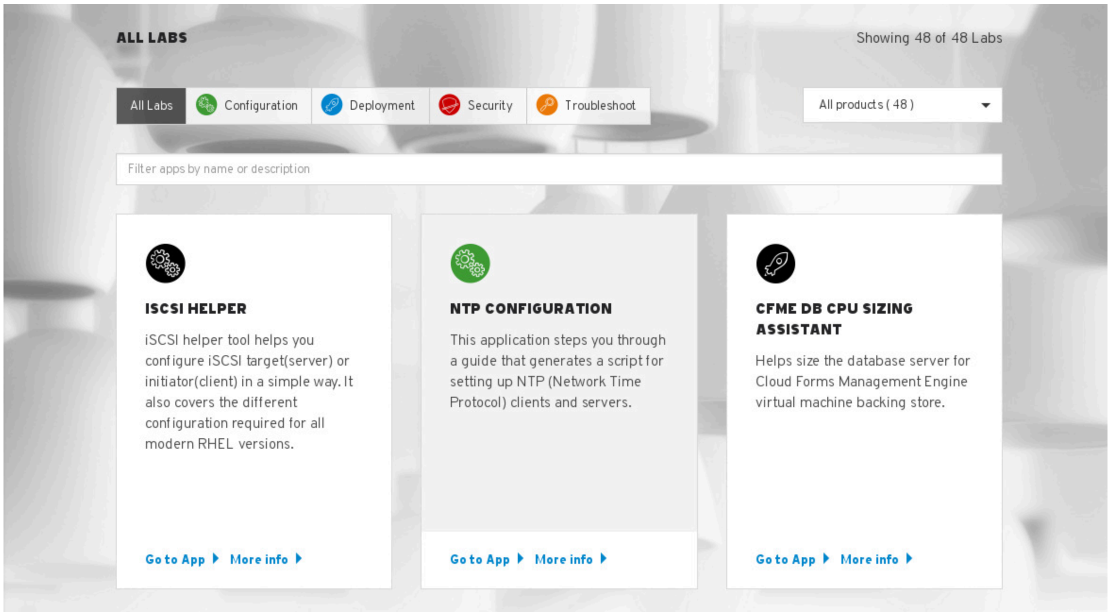
```
Command (? for help): listcases

Type the number of the case to view or 'e' to return to the previous menu.
 1 [Waiting on Red Hat]  System fails to run without power
No more cases to display
Select a Case: 1

Type the number of the section to view or 'e' to return to the previous menu.
 1 Case Details
 2 Modify Case
 3 Description
 4 Recommendations
 5 Get Attachment
 6 Add Attachment
 7 Add Comment
End of options.
Option: q

Select a Case: q

Command (? for help):q

[student@demo ~]$ redhat-support-tool modifycase --status=Closed 01034421
Successfully updated case 01034421
[student@demo ~]$
```

The Red Hat Support Tool has advanced application diagnostic and analytic capabilities. Using kernel crash dump core files, **redhat-support-tool** can create and extract a *backtrace*, a report of the active stack frames at the point of a crash dump, to provide onsite diagnostics and open a support case.

The tool also provides log file analysis. Using the tool's **analyze** command, log files of many types, including operating system, JBoss, Python, Tomcat, oVirt, and others, can be parsed to recognize problem symptoms, which can then be viewed and diagnosed individually. Providing preprocessed analysis, as opposed to raw data such as crash dump or log files, allows support cases to be opened and made available to engineers more quickly.

Labs

Red Hat Access Labs, which can be found at **`https://access.redhat.com/labs`**, provides various web-based tools to aid in configuration, deployments, security, and troubleshooting. This includes test scripts for security exploits such as Shellshock and Heartbleed, but also configuration scripts for NFS servers and clients, logfile analyzers, a tool to help in submitting an architecture review for high-availability clusters, and many more.



ISCSI HELPER

iSCSI helper tool helps you configure iSCSI target(server) or initiator(client) in a simple way. It also covers the different configuration required for all modern RHEL versions.

NTP CONFIGURATION

This application steps you through a guide that generates a script for setting up NTP (Network Time Protocol) clients and servers.

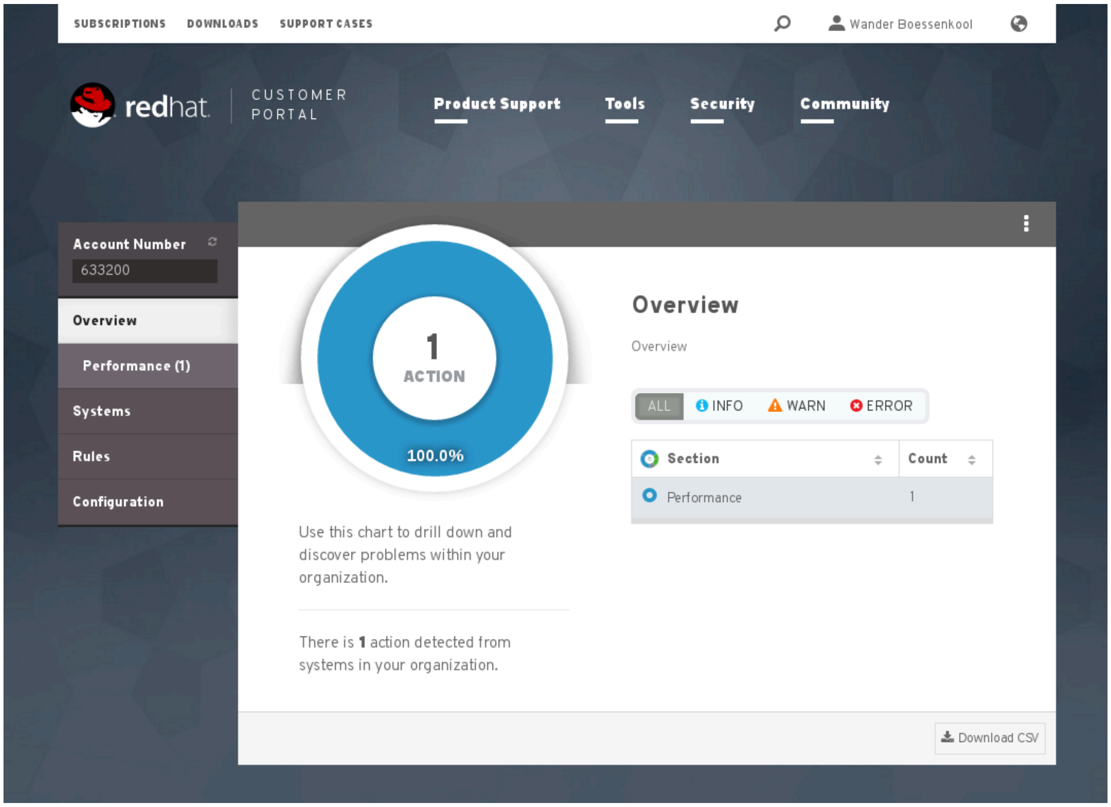
CFME DB CPU SIZING ASSISTANT

Helps size the database server for Cloud Forms Management Engine virtual machine backing store.

Figure 1.2: Labs at the Red Hat Customer Portal

Red Hat Insights

Red Hat Insights is a hosted service that gives system administrators and managers a tool to help them proactively manage their systems. Red Hat Insights (securely) uploads key information from a system to Red Hat, where it is analyzed, and a set of tailored recommendations will be made. These recommendations can help keep systems stable and performing, spotting any potential issues and giving remediation advice before they can become larger problems that cause issues.



Account Number
633200

Overview

Use this chart to drill down and discover problems within your organization.

There is **1** action detected from systems in your organization.

Section	Count
Performance	1

Figure 1.3: Red Hat Insights

The web interface at **`https://access.redhat.com/insights`** provides an overview of all possible issues affecting registered systems, ordered by severity and type. Administrators can drill down into issues to get tailored recommendations. Administrators can also choose to permanently ignore certain rules.

Registering systems with Red Hat Insights

Registering systems with Red Hat Insights requires only two steps:

1. Ensure that the *redhat-access-insights* package is installed:

```
[root@demo ~]# yum -y install redhat-access-insights
```

2. Register the system with Red Hat Insights.

```
[root@demo ~]# redhat-access-insights --register
Successfully registered demo.lab.example.com

Attempting to download collection rules
Attempting to download collection rules GPG signature from https://cert-
api.access.redhat.com/r/insights/v1/static/uploader.json.asc
Successfully downloaded GPG signature
Verifying GPG signature of Insights configuration
Starting to collect Insights data
Uploading Insights data, this may take a few minutes
Upload completed successfully
```

Immediately after registering a system, Insights data becomes available in the web interface.

References

sosreport(1) man page

> Red Hat Access: Red Hat Support Tool
> https://access.redhat.com/site/articles/445443

> Red Hat Support Tool First Use
> https://access.redhat.com/site/videos/534293

> Contacting Red Hat Technical Support
> https://access.redhat.com/site/support/policy/support_process/

> Help - Red Hat Customer Portal
> https://access.redhat.com/site/help/

> Red Hat Access Labs
> https://access.redhat.com/labs

> Red Hat Insights
> https://access.redhat.com/insights

Guided Exercise: Using Red Hat Resources

In this lab, you will generate and inspect a **sosreport** on your **servera** system.

Resources	
Machines	• **workstation** • **servera**

Outcome(s)
You should be able to generate a **sosreport** to aid Red Hat Support in resolving an issue.

While dealing with an issue on your **servera** system, you have contacted Red Hat Support for assistance. Support has requested that you generate a **sosreport** with the **xfs** module active, and the **xfs.logprint** option enabled.

For the purpose of this exercise, your name is "student", and your case number is 123456.

1. Verify that you have the *sos* package installed on your **servera** system. If not, install it.

    1.1. Verify that you have the *sos* package installed on your **servera** system.

    ```
    [root@servera ~]# rpm -q sos
    sos-3.2-35.el7.noarch
    ```

    1.2. If the previous command returned **package sos is not installed**, install the *sos* package.

    ```
    [root@servera ~]# yum -y install sos
    ```

2. View the available options, plug-ins, and plug-in options for **sosreport**.

    2.1. View the available options for **sosreport**.

    ```
    [root@servera ~]# sosreport --help | less
    ```

    2.2. View the available plug-ins and plug-in options for **sosreport**.

    ```
    [root@servera ~]# sosreport -l | less
    ```

3. Generate a **sosreport** on **servera**. Make sure to include the **xfs.logprint** option.

    3.1. Run the **sosreport** tool with the **-k xfs.logprint** option. Use the interactive prompts to enter all required information. This process might take some time to complete.

    ```
    [root@servera ~]# sosreport -k xfs.logprint
    sosreport (version 3.2)

    This command will collect diagnostic and configuration information from
    ```

```
this Red Hat Enterprise Linux system and installed applications.

An archive containing the collected information will be generated in
/var/tmp and may be provided to a Red Hat support representative.

Any information provided to Red Hat will be treated in accordance with
the published support policies at:

  https://access.redhat.com/support/

The generated archive may contain data considered sensitive and its
content should be reviewed by the originating organization before being
passed to any third party.

No changes will be made to system configuration.

Press ENTER to continue, or CTRL-C to quit. Enter
Please enter your first initial and last name [servera.lab.example.com]: student
Please enter the case id that you are generating this report for []: 123456

 Setting up archive ...
 Setting up plugins ...
 Running plugins. Please wait ...

  Running 1/82: abrt...
  Running 2/82: acpid...
...
  Running 81/82: xfs...
  Running 82/82: yum...

Creating compressed archive...

Your sosreport has been generated and saved in:
  /var/tmp/sosreport-student.123456-20151210132339.tar.xz

The checksum is: d0459a202c5c456be00a3dac7b92567a

Please send this file to your support representative.
```

4. Copy the generated file to the home directory for **student** on **workstation**, and extract it.

   4.1. Copy the generated file to the home directory for **student** on **workstation**.

```
[student@workstation ~]$ scp root@servera:/var/tmp/sosreport-
student.123456*.tar.xz .
```

   4.2. Extract the generated file. You can safely ignore any errors related to the creation of device nodes.

```
[student@workstation ~]$ tar xvf sosreport-student.123456*.tar.xz
```

5. Inspect the extracted **sosreport**. Some files of interest:

   - **etc/**
   - **sos_commands/**
   - **sos_reports/sos.html**

- **`sos_commands/xfs/xfs_logprint_-c.dev.vda1`**

  This file was generated by the **`-k xfs.logprint`** option.

Lab: What Is Troubleshooting?

In this lab, you will solve a problem with FTP transfers on **servera**.

Resources	
Machines	• **workstation** • **servera**

Outcome(s)

You should be able to solve a FTP connectivity and file transfer problem using the scientific method.

Before you begin

Prepare your systems for this exercise by running the command **lab troubleshootingintro setup** on your **workstation** machine.

```
[student@workstation ~]$ lab troubleshootingintro setup
```

One of your users has reported a problem with the FTP server running on **servera**. The user experiences the following symptoms:

- The user is unable to connect from **workstation** using **lftp**.
- The user can connect to **localhost** using **lftp** when logged into a shell on **servera**.
- When connected, the file **pub/noclip** does transfer, but the file **pub/getall** does not.

For your testing purposes, **lftp** has been installed on both **workstation** and **servera**. Remember that **lftp** will not attempt to connect until the first command has been issued.

The FTP daemon (**vsftpd.service**) on **servera** logs all file transfers to **/var/log/xferlog**. If the last character on a line in that file is **c**, the transfer completed successfully; if the last character is **i**, the transfer was incomplete.

1. Attempt to recreate the issue.
2. Collect information about the FTP service running on **servera**. Include network ports, firewall information, document root, file permissions, SELinux denials, etc.
3. Form a hypothesis as to what the issue(s) might be.
4. Implement a fix for all problems found.
5. Test that the reported issues have now been resolved.
6. Grade your work by running the command **lab troubleshootingintro grade** from **workstation**.

   ```
   [student@workstation ~]$ lab troubleshootingintro grade
   ```

7. Reset all of your machines to provide a clean environment for the next exercises.

Solution

In this lab, you will solve a problem with FTP transfers on **servera**.

Resources	
Machines	• **workstation** • **servera**

Outcome(s)

You should be able to solve a FTP connectivity and file transfer problem using the scientific method.

Before you begin

Prepare your systems for this exercise by running the command **lab troubleshootingintro setup** on your **workstation** machine.

```
[student@workstation ~]$ lab troubleshootingintro setup
```

One of your users has reported a problem with the FTP server running on **servera**. The user experiences the following symptoms:

- The user is unable to connect from **workstation** using **lftp**.
- The user can connect to **localhost** using **lftp** when logged into a shell on **servera**.
- When connected, the file **pub/noclip** does transfer, but the file **pub/getall** does not.

For your testing purposes, **lftp** has been installed on both **workstation** and **servera**. Remember that **lftp** will not attempt to connect until the first command has been issued.

The FTP daemon (**vsftpd.service**) on **servera** logs all file transfers to **/var/log/xferlog**. If the last character on a line in that file is **c**, the transfer completed successfully; if the last character is **i**, the transfer was incomplete.

1. Attempt to recreate the issue.

   1.1. Attempt to use **lftp** from **workstation** to connect to the FTP service running on **servera**.

   ```
   [student@workstation ~]$ lftp servera
   lftp servera:~> ls
   'ls' at 0 [Delaying before reconnect: 30]Ctrl+C
   Interrupt
   lftp servera:~> bye
   ```

   1.2. Attempt to use **lftp** from **servera** to connect to the FTP service running on **servera**/**localhost**.

   ```
   [student@servera ~]$ lftp servera
   lftp servera:~> ls
   drwxr-xr-x    2 0        0             32 Dec 11 09:42 pub
   lftp servera:~>
   ```

1.3. Attempt to view the contents of the **pub/noclip** file.

```
lftp servera:~> cat pub/noclip
idspispopd
11 bytes transferred
lftp servera:~>
```

1.4. Attempt to view the contents of the **pub/getall** file.

```
lftp servera:~> cat pub/getall
cat: Access failed: 550 Failed to open file. (pub/getall)
lftp servera:~> bye
```

2. Collect information about the FTP service running on **servera**. Include network ports, firewall information, document root, file permissions, SELinux denials, etc.

2.1. Collect information on where **vsftpd** is listening on the network.

```
[root@servera ~]# ss -tulpn | grep ftp
tcp   LISTEN  0  32 :::21   :::*  users:(("vsftpd",pid=30050,fd=3))
```

This shows that **vsftpd** is listening on the default FTP port (**tcp:21**), and accepting connections from all IP addresses.

2.2. View the firewall configuration on **servera**.

```
[root@servera ~]# firewall-cmd --list-all
public (default, active)
  interfaces: eth0 eth1
  sources:
  services: dhcpv6-client ssh
  ports:
  masquerade: no
  forward-ports:
  icmp-blocks:
  rich rules:
```

This shows that the **ftp** service is not opened up in the firewall. That explains why remote connections failed, but local connections worked. Make a note of this.

2.3. View the contents of the FTP document root. The default location is **/var/ftp**.

```
[root@servera ~]# ls -lR /var/ftp
/var/ftp:
total 0
drwxr-xr-x. 2 root root 32 Dec 11 10:42 pub

/var/ftp/pub:
total 8
-rw-r--r--. 1 root root  6 Dec 11 10:42 getall
-rw-r--r--. 1 root root 11 Dec 11 10:42 noclip
```

File permissions appear to be correct.

2.4. Check for any SELinux denials in the last day.

```
[root@servera ~]# ausearch -m avc -i -ts today
...
type=SYSCALL msg=audit(12/11/2015 11:01:01.997:2031) : arch=x86_64 syscall=open
 success=no exit=-13(Permission denied) a0=0x7f91ba3da4f0 a1=O_RDONLY|O_NONBLOCK
 a2=0x7f91ba3d9880 a3=0x566a9edd items=0 ppid=30406 pid=30408 auid=unset uid=ftp
 gid=ftp euid=ftp suid=ftp fsuid=ftp egid=ftp sgid=ftp fsgid=ftp tty=(none)
 ses=unset comm=vsftpd exe=/usr/sbin/vsftpd subj=system_u:system_r:ftpd_t:s0-
s0:c0.c1023 key=(null)
type=AVC msg=audit(12/11/2015 11:01:01.997:2031) : avc:  denied
 { open } for  pid=30408 comm=vsftpd path=/pub/getall dev="vda1"
 ino=16828424 scontext=system_u:system_r:ftpd_t:s0-s0:c0.c1023
 tcontext=unconfined_u:object_r:tmp_t:s0 tclass=file
...
```

This shows that the file **`/var/ftp/pub/getall`** has a SELinux context of **`tmp_t`**. This is most likely incorrect.

3. Form a hypothesis as to what the issue(s) might be.

3.1. The firewall is not opened for the FTP service, resulting in failed connections from remote machines.

3.2. The file **`/var/ftp/pub/getall`** has an incorrect SELinux context, disallowing the FTP daemon read access to the file.

4. Implement a fix for all problems found.

4.1. Open up the firewall for FTP services on **`servera`**.

```
[root@servera ~]# firewall-cmd --add-service=ftp
[root@servera ~]# firewall-cmd --permanent --add-service=ftp
```

4.2. Recursively reset the SELinux contexts on **`/var/ftp`**.

```
[root@servera ~]# restorecon -Rv /var/ftp
```

5. Test that the reported issues have now been resolved.

5.1. Attempt to use **`lftp`** from **`workstation`** to connect to the FTP service running on **`servera`**/**`localhost`**.

```
[student@workstation ~]$ lftp servera
lftp servera:~> ls
drwxr-xr-x    2 0        0             32 Dec 11 09:42 pub
lftp servera:~>
```

5.2. Attempt to view the contents of the **`pub/noclip`** file.

```
lftp servera:~> cat pub/noclip
idspispopd
11 bytes transferred
```

```
lftp servera:~>
```

5.3. Attempt to view the contents of the **pub/getall** file.

```
lftp servera:~> cat pub/getall
idkfa
6 bytes transferred
lftp servera:~> bye
```

6. Grade your work by running the command **lab troubleshootingintro grade** from **workstation**.

```
[student@workstation ~]$ lab troubleshootingintro grade
```

7. Reset all of your machines to provide a clean environment for the next exercises.

Summary

In this chapter, you learned:

- How to apply the scientific method to troubleshooting.
- How to use **journalctl** to read system logs.
- How to configure a persistent system journal.
- How to use **redhat-support-tool**.
- How to use **sosreport**.
- How to configure Red Hat Insights.

CHAPTER 2

BEING PROACTIVE

Overview	
Goal	Prevent small issues from becoming large problems by employing proactive system administration techniques.
Objectives	• Monitor systems for vital characteristics. • Configure systems to send logging messages to a centralized host. • Configure configuration management to aid in large-scale system administration. • Implement change tracking to keep systems homogenous.
Sections	• Monitoring Systems (and Guided Exercise) • Configuring Remote Logging (and Guided Exercise) • Using Configuration Management (and Guided Exercise) • Configuring Change Tracking (and Guided Exercise)
Lab	• Being Proactive

Monitoring Systems

Objectives

After completing this section, students should be able to monitor systems for vital characteristics.

System monitoring with cockpit

Cockpit is software developed by Red Hat that provides an interactive browser-based Linux administration interface. Its graphical interface allows beginner system administrators to perform common system administration tasks without the requisite skills on the command line. In addition to making systems easier to manage by novice administrators, Cockpit also makes system configuration and performance data accessible to them without knowledge of command-line tools.

Cockpit is made available via the *cockpit* package in the Red Hat Enterprise Linux 7 Extras repository. The installation of Cockpit is performed with the following command.

```
[root@demo ~]# yum install -y cockpit
```

Once installed on a system, the **cockpit** must be started before it can be accessed across the network.

```
[root@demo ~]# systemctl start cockpit
```

Cockpit can be accessed remotely via HTTPS using a web browser and connecting to port TCP port 9090. This port must be opened on the system's firewall in order for Cockpit to be accessed remotely. The port is predefined as the **cockpit** for **firewalld**, so access through the firewall can be granted as shown in the following example.

```
[root@demo ~]# firewall-cmd --add-service=cockpit --permanent
[root@demo ~]# firewall-cmd --reload
```

Once a connection is established with the Cockpit web interface, a user must authenticate in order to gain entry. Authentication is performed using the system's local OS account database. Access to privileged system management functions provided by the Cockpit interface, such as user creation, will require that the user log in as the **root** user.

The **Dashboard** screen in the Cockpit interface provides an overview of core system performance metrics. Metrics are reported on a per-second basis, and allows administrators to monitor the utilization of subsystems, such as CPU, memory, network, and disk.

Using Performance Co-Pilot

Red Hat Enterprise Linux 7 includes a program called Performance Co-Pilot, provided by the *pcp* RPM package. Performance Co-Pilot, or **pcp** for short, allows administrators to collect and query data from various subsystems. Performance Co-Pilot has also been backported into Red Hat Enterprise Linux 6, and is available in 6.6 and later.

Installing Performance Co-Pilot

Performance Co-Pilot is installed with the *pcp* package. After installing *pcp*, the machine will have the **pmcd** daemon necessary for collecting subsystem data. Additionally, the machine will also have various command-line tools for querying system performance data.

```
[root@demo ~]# yum -y install pcp
```

There are several services that are part of Performance Co-Pilot, but the one that collects system performance data locally is **pmcd**, the *Performance Metrics Collector Daemon*. This service must be running in order to query performance data with the Performance Co-Pilot command-line utilities.

```
[root@demo ~]# systemctl start pmcd

[root@demo ~]# systemctl status pmcd
● pmcd.service - Performance Metrics Collector Daemon
   Loaded: loaded (/usr/lib/systemd/system/pmcd.service; enabled; vendor preset:
 disabled)
   Active: active (exited) since Fri 2015-12-11 08:26:38 EST; 8s ago
... Output Truncated ...

[root@demo ~]# systemctl enable pmcd
Created symlink from /etc/systemd/system/multi-user.target.wants/pmcd.service to /usr/
lib/systemd/system/pmcd.service.
```

Using the *pcp* command-line utilities

The *pcp* package provides a variety of command-line utilities to gather and display data on a machine.

The **pmstat** command provides information similar to **vmstat**. Like **vmstat**, **pmstat** will support options to adjust the interval between collections (**-t**) or the number of samples (**-s**).

```
[root@demo ~]# pmstat -s 5
@ Fri Nov  7 15:00:08 2014
 loadavg                      memory      swap        io    system          cpu
   1 min   swpd   free   buff  cache   pi   po   bi   bo   in   cs  us  sy  id
    0.02      0 817044    688 480812    0    0    0    0   30   33   0   0  99
    0.02      0 817044    688 480812    0    0    0    0    7   14   0   0 100
    0.02      0 817044    688 480812    0    0    0    0    7   13   0   0 100
    0.02      0 817044    688 480812    0    0    0    0    9   15   0   0 100
    0.02      0 817044    688 480812    0    0    0    9   12   22   0   0 100
```

pmatop provides a **top**-like output of machine statistics and data. It includes disk I/O statistics and network I/O statistics, as well as the CPU, memory, and process information provided by other tools. By default, **pmatop** will update every five seconds.

```
[root@demo ~]# pmatop
ATOP - Fri Nov  7 17:43:19 2014         0:00:05 elapsed

PRC | sys    0.06s | user   0.23s | #proc    214 | #zombie    0
CPU | sys 0% | user      2% | irq       0% | idle     97% | wait      0% |
cpu | sys       0% | user      2% | irq       0% | idle     47% | cpu01     0% |
CPL | avg1    0.05 | avg5    0.08 | avg15   0.07 | csw       211 | intr      390
 |
MEM | tot    1885M | free    795M | cache   480M | buff      0M | slab    267M |
SWP | tot 0G | free      0G |              | vmcom     1G | vmlim     0G |
PAG | scan  0 | steal 0 | stall      0 | swin       0 | swout      0 |
DSK | vda          | busy      0% | read       0 | write      0 | avio    0 ms |
DSK | vdb          | busy      0% | read       0 | write      0 | avio    0 ms |
NET | transport    | tcpi      1M | tcpo      1M | udpi      0M | udpo      0M |
NET | network    | ipi       1M | ipo       1M | ipfrw     0M | deliv     1M |
NET | eth0         | pcki      2M | pcko      2M | si    0 Kbps | so    0 Kpbs |

  PID SYSCPU USRCPU VGROW    RGROW  RUID   THR ST EXC S CPU  CMD
27541  0.00s  0.21s      0K    264K  root     1  --  -  S 72% pmatop
 9002  0.02s  0.00s  0K  0K  root     1  --  -  R  6% pmdaproc
    1  0.00s  0.00s      0K      0K  root     1  --  -  S  0% systemd
    2  0.00s  0.00s      0K      0K  root     0  --  -  S  0% kthreadd
    3  0.00s  0.00s      0K      0K  root     0  --  -  S  0% ksoftirqd/0
    5  0.00s  0.00s  0K  0K  root     0  --  -  S  0% kworker/0:0H
    6  0.00s  0.00s  0K  0K  root     0  --  -  S  0% kworker/u4:0
```

Performance Co-Pilot also has a text-based query mechanism for interrogating individually tracked metrics. To obtain a list of the metrics stored in the Performance Co-Pilot database, use the **pminfo** command, then use the **pmval** command with the metric name to gather data about the desired item.

```
[root@demo ~]# pminfo
... Output Truncated ...
proc.nprocs
proc.psinfo.pid
... Output Truncated ...

[root@demo ~]# pminfo -dt proc.nprocs

proc.nprocs [instantaneous number of processes]
    Data Type: 32-bit unsigned int  InDom: PM_INDOM_NULL 0xffffffff
    Semantics: instant  Units: none

[root@demo ~]# pmval -s 5 proc.nprocs

metric:    proc.nprocs
host:      server1.example.com
semantics: instantaneous value
units:     none
samples:   5
interval:  1.00 sec
        133
        133
        133
        133
        130
```

Retrieving historical performance data

Performance Co-Pilot also has the ability to store data in a log. This facility is provided by the **pmlogger** service. This service must be enabled for system performance data to be archived by Performance Co-Pilot.

```
[root@demo ~]# systemctl start pmlogger

[root@demo ~]# systemctl status pmlogger
● pmlogger.service - Performance Metrics Archive Logger
   Loaded: loaded (/usr/lib/systemd/system/pmlogger.service; enabled; vendor preset:
 disabled)
   Active: active (exited) since Fri 2015-12-11 08:28:38 EST; 6s ago
```

```
... Output Truncated ...

[root@demo ~]# systemctl enable pmlogger
Created symlink from /etc/systemd/system/multi-user.target.wants/pmlogger.service to /
usr/lib/systemd/system/pmlogger.service.
```

By default, **pmlogger** collects data every second and stores logged data in the **/var/log/pcp/pmlogger/*HOSTNAME*** directory. After the data has been collected into a **pmlogger** archive, tools, like **pmval**, may be used to query data from it.

The log file name will begin with the ISO-formatted date. In addition to the log file, several other files are created to store metadata and index information.

Note

When exporting the log to another system, be sure to copy the log file and the associated **.meta** files. Lacking either one of these files will prevent the analysis tools from using the log data.

Once the log has been created, command-line **pcp** tools use the **-a** option to indicate that they should be run against a data archive instead of live data. The **pmval** command also has options to specify the start and end times that should be used if an administrator desired to narrow the data to a specific time window.

```
[root@demo ~]# pmval -a /var/log/pcp/pmlogger/serverX.example.com/20150224.00.10.0
 kernel.all.load
...
03:03:46.197       0.1100        0.1700        0.1200
03:03:47.197       0.1100        0.1700        0.1200
03:03:48.197       0.1100        0.1700        0.1200
03:03:49.197       0.1100        0.1700        0.1200
...
```

If an administrator is interested in the load averages from 03:03:00 until 03:04:00 on February 24, 2015:

```
[root@demo ~]# pmval -a /var/log/pcp/pmlogger/serverX.example.com/20150224.00.10.0
 kernel.all.load -S '@ Tue Feb 24 03:03:00 2015' -T '@ Tue Feb 24 03:04:00 2015'
metric:    kernel.all.load
archive:   /var/log/pcp/pmlogger/server0.example.com/20150224.00.10.0
host:      server0.example.com
start:     Tue Feb 24 03:03:00 2015
end:       Tue Feb 24 03:04:00 2015
semantics: instantaneous value
units:     none
samples:   61
interval:  1.00 sec

                 1 minute      5 minute     15 minute
03:03:00.000       0.3000        0.2100        0.1200
03:03:01.000       0.3000        0.2100        0.1200
03:03:02.000       0.3000        0.2100        0.1200
... Output Truncated ...
```

Alternate time specifications can be found in the **PCPIntro**(1) man page.

Monitoring of remote systems with Nagios

When dealing with large IT infrastructure, most medium and large-scale enterprises require a centralized monitoring solution capable of polling the performance of remote systems over the network. Some organizations rely on network monitoring/management systems and services from vendors like CA, HP, IBM, and others. Others prefer open source, GPL-friendly options in this area, such as Nagios.

Nagios allows administrators to monitor the system and network performance on networked hosts from a central server. It relies on both active and passive monitoring techniques, some of which may involve the installation of agents on the monitored systems.

While **Nagios** is a great free open source monitoring tool, it is currently provided via the EPEL (Extra Packages for Enterprise Linux) repository from the Fedora project. As such, it is not supported by Red Hat.

Nagios is a modular system, consisting of a core **nagios** package and additional functionality in the form of plug-ins. Plugins can be run on local machines to provide information that is not readily available via the network, such as disk space usage, etc. The configuration is very flexible, allowing definitions of time periods, admin groups, system groups, and even custom command sets.

There is a web-based interface on the main Nagios server that can be used to configure tests and settings for Nagios itself or for the hosts it is monitoring. Administrators can set thresholds for monitored system performance indicators so that alerts are generated by Nagios when these performance metrics fall outside of normal operating ranges.

References

For more information on Cockpit, see

> Cockpit Project
> http://www.cockpit-project.org

For more information on Performance Co-Pilot, see the **pmstat**(1), **pmatop**(1), **pminfo**(1), **pmval**(1), and **PCPIntro**(1) man pages.

For more information on Nagios, see

> Nagios Project
> http://www.nagios.org

Guided Exercise: Monitoring Systems

In this lab, you will monitor current and historical system performance using Performance Co-Pilot.

Resources	
Files	`/root/cpuidle`
Machines	`servera`

Outcome(s)

You should be able to display current and historical performance indicators on a system using Performance Co-Pilot.

Before you begin

Log into **`servera`** as the **`root`** user.

In this exercise, you will install and configure the Performance Co-Pilot software on **`servera`**. You will utilize the utilities provided by the software to gather current performance statistics of the system. You will also leverage the archival feature of the software and retrieve historical performance statistics.

1. Install the Performance Co-Pilot software on **`servera`** by installing the *pcp*.

```
[root@servera ~]# yum install -y pcp
```

2. Start and enable the **pmcd** service to enable collection of performance metrics.

```
[root@servera ~]# systemctl start pmcd
[root@servera ~]# systemctl enable pmcd
Created symlink from /etc/systemd/system/multi-user.target.wants/pmcd.service to /
usr/lib/systemd/system/pmcd.service.
[root@servera ~]# systemctl status pmcd
● pmcd.service - Performance Metrics Collector Daemon
   Loaded: loaded (/usr/lib/systemd/system/pmcd.service; enabled; vendor preset:
 disabled)
   Active: active (exited) since Fri 2015-12-11 08:26:38 EST; 8s ago
     Docs: man:pmcd(8)
 Main PID: 1896 (code=exited, status=0/SUCCESS)
   CGroup: /system.slice/pmcd.service
           ├─1942 /usr/libexec/pcp/bin/pmcd
           ├─1945 /var/lib/pcp/pmdas/root/pmdaroot
           ├─1946 /var/lib/pcp/pmdas/proc/pmdaproc -d 3
           ├─1947 /var/lib/pcp/pmdas/xfs/pmdaxfs -d 11
           └─1948 /var/lib/pcp/pmdas/linux/pmdalinux

Dec 14 08:26:38 servera.lab.example.com systemd[1]: Starting Performance Metrics
 Collector Daemon...
Dec 14 08:26:38 servera.lab.example.com pmcd[1896]: Starting pmcd ...
Dec 14 08:26:38 servera.lab.example.com systemd[1]: Started Performance Metrics
 Collector Daemon.
```

3. Start and enable the **`pmlogger`** service to configure persistent logging of metrics.

```
[root@servera ~]# systemctl start pmlogger
[root@servera ~]# systemctl enable pmlogger
Created symlink from /etc/systemd/system/multi-user.target.wants/pmlogger.service
 to /usr/lib/systemd/system/pmlogger.service.
[root@servera ~]# systemctl status pmlogger
● pmlogger.service - Performance Metrics Archive Logger
   Loaded: loaded (/usr/lib/systemd/system/pmlogger.service; enabled; vendor preset:
 disabled)
   Active: active (exited) since Fri 2015-12-11 08:28:38 EST; 6s ago
     Docs: man:pmlogger(1)
 Main PID: 2535 (code=exited, status=0/SUCCESS)
   CGroup: /system.slice/pmlogger.service
           └─6489 /usr/libexec/pcp/bin/pmlogger -P -r -T24h10m -c config.default -m
 pmlogger_check 20151211.08.28

Dec 14 08:28:38 servera.lab.example.com systemd[1]: Starting Performance Metrics
 Archive Logger...
Dec 14 08:28:38 servera.lab.example.com pmlogger[2535]: /usr/share/pcp/lib/pmlogger:
 Warning: Performance Co-Pilot archive ...bled.
Dec 14 08:28:38 servera.lab.example.com pmlogger[2535]: To enable pmlogger, run the
 following as root:
Dec 14 08:28:38 servera.lab.example.com pmlogger[2535]: # /usr/bin/systemctl enable
 pmlogger.service
Dec 14 08:28:38 servera.lab.example.com pmlogger[2535]: Starting pmlogger ...
Dec 14 08:28:38 servera.lab.example.com systemd[1]: Started Performance Metrics
 Archive Logger.
Hint: Some lines were ellipsized, use -l to show in full.
```

4. Query the **pmcd** daemon to display the per-CPU idle time for one minute.

```
[root@servera ~]# pmval -T 1minute kernel.percpu.cpu.idle
metric:    kernel.percpu.cpu.idle
host:      servera.lab.example.com
semantics: cumulative counter (converting to rate)
units:     millisec (converting to time utilization)
samples:   61
interval:  1.00 sec

              cpu0                 cpu1
               0.9997               0.9997
               0.9997               0.9997
... Output omitted ...
```

5. Retrieve the location of the archive log that the primary **pmlogger** process is writing to.

```
[root@servera ~]# pcp | grep 'primary logger'
 pmlogger: primary logger: /var/log/pcp/pmlogger/
servera.lab.example.com/20151211.08.28
```

6. Using the output from the previous command, determine the most recent iteration of the archive log by examining the date and time data contained in the file name.

```
[root@servera ~]# ls -1 /var/log/pcp/pmlogger/
servera.lab.example.com/20151211.08.28.[0-9]* | tail -1
/var/log/pcp/pmlogger/servera.lab.example.com/20151211.08.28.0
```

7. Use the **pmval** command to obtain historical statistics of per-CPU idle time at one minute intervals from the most recent archive log. Save the output into the **/root/cpuidle** file.

```
[root@servera ~]# pmval -t1minute kernel.percpu.cpu.idle -a /var/log/pcp/pmlogger/
servera.lab.example.com/20151211.08.28.0 > /root/cpuidle

pmval: pmFetch: End of PCP archive log
```

8. Grade your work, then clean up your systems for the next exercise.

   8.1. Grade your work.

   ```
   [student@workstation ~]$ lab pcp grade
   ```

   8.2. Clean up your systems for the next exercise.

   ```
   [student@workstation ~]$ lab pcp reset
   ```

Configuring Remote Logging

Objectives

After completing this section, students should be able to configure systems for remote logging to a central log host.

Remote logging

Standard system log management configuration rotates log files every week and retains them for four rotations. It is often desirable to maintain logs longer than the four-week default, especially when establishing system performance trends related to tasks, such as month-end financial closings, which are executed just once a month. By sending log messages to a remote log host with dedicated mass storage, administrators can maintain large archives of system logs for their systems without changing the default log rotation configuration, which is intended to keep logs from overconsuming disk storage.

Central collection of system log messages can also be very useful for monitoring the state of systems and for quickly identifying problems. It also provides a backup location for log messages in case a system suffers a catastrophic hard drive failure or other problems, which cause the local logs to no longer be available. In these situations, the copy of the log messages which reside on the central log host can be used to help diagnose the issue that caused the problem.

Standardized system logging is implemented in Red Hat Enterprise Linux 7 by the **rsyslog** service. System programs can send syslog messages to the local **rsyslogd** service, which will then redirect those messages to files in **/var/log**, remote log servers, or other databases based on the settings in its configuration file, **/etc/rsyslog.conf**.

Log messages have two characteristics that are used to categorize them. The *facility* of a log message indicates the type of message it is. The *priority*, on the other hand, indicates the importance of the event logged in the message.

Syslog Priority Levels

Priority	*Meaning*
emerg	System is unusable
alert	Immediate action required
crit	Critical condition
err	Error condition
warning	Warning condition
notice	Normal but significant condition
info	Informational messages
debug	Debugging messages

Configuring a central log host

The implementation of a central log host requires the configuration of the **rsyslog** service on two types of systems: the remote systems where the log messages originate from and the central log host receiving the messages. On the central log host, the **rsyslog** service needs to be configured so that log messages from remote hosts are accepted.

To configure the **rsyslog** service on the central log host to accept remote logs, uncomment either the TCP or UDP reception lines in the modules section in the **/etc/rsyslog.conf** file.

For UDP reception:

```
# Provides UDP syslog reception
$ModLoad imudp.so
$UDPServerRun 514
```

or for TCP reception:

```
# Provides TCP syslog reception
$ModLoad imtcp.so
$InputTCPServerRun 514
```

TCP provides more reliable delivery of remote log messages, but UDP is supported by a wider variety of operating systems and networking devices.

Important

Plain TCP transport of syslog messages is fairly widely implemented but not yet standardized. Most implementations currently use port 514/TCP, which is the legacy **rshd** port. If the system has the *rsh-server* package installed and is using the old insecure **rshd** service, it will conflict with using port 514/TCP for plain TCP syslog reception. Configure the log server to use a different port by changing the setting for **$InputTCPServerRun**.

The rules contained in **/etc/rsyslog.conf** are configured by default to accommodate the logging of messages on a single host. Therefore, it sorts and bundles messages by facility. For example, mail messages are funneled into **/var/log/maillog** while messages generated by the **crond** daemon are consolidated into **/var/log/cron** to facilitate locating each type of message.

While sorting of messages by facility is ideal on a single host, it produces an undesirable result on a central log host since it causes messages from different remote hosts to be mixed with each other. On a central log host, it is usually more optimal for log messages from remote systems to remain separate from each other. This separation can be achieved by defining dynamic log file names using the template function of **rsyslog**.

Templates are defined in **/etc/rsyslog.conf** and can be used to generate rules with dynamic log file names. A template definition consists of the **$template** directive, followed by a template name, and then a string representing the template text. The template text can be made dynamic by making use of values substituted from the properties of a log message. For example, to direct **cron** syslog messages from different systems to different files on a central log host, use the following template to generate dynamic log file names based on the **HOSTNAME** property of each message:

```
$template DynamicFile,"/var/log/loghost/%HOSTNAME%/cron.log"
```

The dynamic file name created using the template definition can then be referenced by the template name in a rule as follows:

```
cron.*   ?DynamicFile
```

On systems performing extremely verbose logging, it may be desirable to turn off syncing of the log file after each write operation in order to improve performance. The syncing of a log file after every logging can be omitted by prefixing the log file name with the minus (-) sign in a logging rule. However, the trade off of improved performance does create the possibility of log data loss if the system crashes immediately after a write attempt.

The following is another example of the use of templates to generate dynamic log file names. In this example, remote log messages will be sorted by their host name and facility values by referencing the **HOSTNAME** and **syslogfacility-test** properties. Log messages will be written to the dynamically generated log file names and no syncing will be performed after the write operation.

```
$template DynamicFile,"/var/log/loghost/%HOSTNAME%/%syslogfacility-text%.log"
*.*   -?DynamicFile
```

Note

A full list of the syslog messages properties made available by **rsyslog** can be found in the **Available Properties** section of the **rsyslog.conf**(5) man page.

Once syslog reception has been activated and the desired rules for log separation by host has been created, restart the **rsyslog** service for the configuration changes to take effect. In addition, add the necessary UDP and/or TCP firewall rules to allow incoming syslog traffic and then reload **firewalld**.

```
[root@loghost ~]# systemctl restart rsyslog
[root@loghost ~]# firewall-cmd --add-port=514/udp --permanent
[root@loghost ~]# firewall-cmd --add-port=514/tcp --permanent
[root@loghost ~]# firewall-cmd --reload
```

When new log files are created, they may not be included by the log host's existing log rotation schedule. This should be remedied to ensure that the new log files do not grow to unmanageable sizes. For instance, to include the new log files from the previous examples in log rotation, add the following entry to the list of log files in the **/etc/logrotate.d/syslog** configuration file.

```
/var/log/loghost/*/*.log
```

Redirecting logging to central log host

Once the central log host is configured to accept remote logging, the **rsyslog** service can to be configured on remote systems to send logs to the central log host. To configure a machine to send logs to a remote **rsyslog** server, add a line to the rules section in the **/etc/rsyslog.conf** file. In place of the file name, use the IP address of the remote **rsyslog** server. To use UDP, prefix the IP address with a single @ sign. To use TCP, prefix it with two @ signs (@@).

For instance, to have all messages with **info** or higher priority sent to **loghost.example.com** via UDP, use the following line:

```
*.info    @loghost.example.com
```

To have *all* messages sent to loghost.example.com via TCP, use the following line:

```
*.*    @@loghost.example.com
```

Optionally, the log host name can be appended with **`:`*`PORT`***, where *PORT* is the port that the remote **`rsyslog`** server is using. If no port is given, it assumes the default port 514.

After adding the rule(s), restart the **`rsyslog`** service and send a test message using the **`logger`** command:

```
[root@logclient ~]# logger "Test from logclient"
```

Check the logs on the remote server to ensure the message was received.

References

For more information, see the **`rsyslog.conf`**(5), **`rsyslogd`**(8), and **`logger`**(1) man pages.

Guided Exercise: Configuring Remote Logging

In this lab, you will configure system logging to a central log host.

Resources	
Files	• **/etc/rsyslog.conf** • **/etc/logrotate.d/syslog**
Machines	• **servera** • **serverb**

Outcome(s)

You should be able to centralize remote system logging to a central log host.

Before you begin

Log in as the **root** user on **servera** and **serverb**.

Configure the **rsyslog** service on **servera** so that it can serve as a central log host. For more reliable syslog messages delivery, configure the central log host to accept message delivery from remote hosts using TCP.

To better organize messages, create a new rule which writes the syslog messages generated by each host to separate subdirectories under the **/var/log/loghost** directory. Subdirectories will be named after the host names of each host. Within each host's subdirectory, a separate log file will be maintained for messages of each syslog facility. For better performance, configure logging so that a sync is not performed after each message logged. Configure log rotation for the new log files.

Configure **serverb** for remote logging to the central log host. Test the configuration by generating a test message and verifying its presence in the appropriate log on **servera**.

1. Verify that the **rsyslog** service is running and enabled to start upon boot.

```
[root@servera ~]# systemctl is-active rsyslog
active
[root@servera ~]# systemctl is-enabled rsyslog
enabled
```

2. Configure the **rsyslog** service on **servera** so that syslog messages from remote hosts are accepted using TCP and that the messages are written to separate files for each host.

   2.1. Enable TCP syslog reception in **/etc/rsyslog.conf** by uncommenting the following lines.

```
$ModLoad imtcp
$InputTCPServerRun 514
```

   2.2. Under the "#### **RULES** ####" section, add the following rule for dynamic log file names and then create a rule to use the dynamic file name to separate log messages by host name and facility.

```
$template DynamicFile,"/var/log/loghost/%HOSTNAME%/%syslogfacility-text%.log"
*.* -?DynamicFile
```

2.3. Restart the **rsyslog** service for the configuration changes to take effect.

```
[root@servera ~]# systemctl restart rsyslog
```

3. Add the following entry to the list of files in the **/etc/logrotate.d/syslog** configuration file so that the new log files are placed into the log rotation schedule.

```
/var/log/loghost/*/*.log
```

4. Modify the firewall on **servera** to allow incoming syslog messages from remote hosts to be delivered using TCP.

4.1. Allow incoming traffic on TCP port 514.

```
[root@servera ~]# firewall-cmd --add-port=514/tcp --permanent
success
```

4.2. Reload **firewalld** for the firewall change to take effect.

```
[root@servera ~]# firewall-cmd --reload
```

5. Configure **serverb** to log syslog messages remotely to **servera** using the TCP protocol.

5.1. On **serverb**, add the following rule to **/etc/rsyslog.conf** so that all syslog messages are delivered remotely to **servera** using TCP on port 514.

```
*.* @@servera.lab.example.com:514
```

5.2. Restart the **rsyslog** service on **serverb** for the configuration changes to take effect.

```
[root@serverb ~]# systemctl restart rsyslog
```

6. Verify remote logging from **serverb** to the central log host running on **servera** is working as expected.

6.1. On **serverb**, generate a couple of syslog messages with different facilities.

```
[root@serverb ~]# logger -p user.info "Test user.info message from serverb"
[root@serverb ~]# logger -p authpriv.crit "Test authpriv.crit message from
 serverb"
```

6.2. On **servera**, verify that the syslog message appears in the appropriate files under the **/var/log/loghost/serverb** directory.

```
[root@servera ~]# grep 'user\.info' /var/log/loghost/serverb/user.log
```

```
Dec 11 00:44:09 serverb root: Test user.info message from serverb
[root@servera ~]# grep 'authpriv\.crit' /var/log/loghost/serverb/authpriv.log
Dec 11 00:44:40 serverb root: Test authpriv.crit message from serverb
```

7. Grade your work, then clean up your systems for the next exercise.

   7.1. Grade your work.

   ```
   [student@workstation ~]$ lab loghost grade
   ```

   7.2. Clean up your systems for the next exercise.

   ```
   [student@workstation ~]$ lab loghost reset
   ```

Using Configuration Management

Objectives

After completing this section, students should be able to implement the Puppet client for configuration management.

Configuration management Solutions

As an organization's IT infrastructure grows, many administrators quickly find that the maintenance of system configurations becomes unmanageable. Often, this is a result of administrators managing the configuration of their systems manually. As an organization's server population increases, a manual approach to system configuration not only fails to scale, but also increasingly becomes the main point of entry for human errors, which negatively impact system stability and function.

A better alternative to the manual maintenance of system configuration is to use a configuration management tool. Even simple configuration management tools can greatly enhance operational stability and efficiency. These tools can greatly expedite the deployment of configuration changes to a large number of systems. Since the changes are made in a programmatic and automated fashion, the accuracy of the configuration changes are also guaranteed.

An example of a simple configuration management tool is the Configuration Channel feature offered in Red Hat Satellite 5. Red Hat Network Configuration Channels provide an easy way to deploy configuration files in an enterprise environment. Rather than making configuration file changes on individual systems, administrators can instead make changes to a configuration file maintained in a Configuration Channel hosted on the Red Hat Satellite 5 servers. The changes can then be deployed to client systems subscribed to the channel.

In recent years, simple tools for managing configuration files have been supplanted by configuration management systems. These systems have matured significantly over the years and now there are many open source solutions available to meet an organization's configuration management needs. The following are some configuration management tool options that are available to administrators.

- Organizations with simple configuration management needs may find Ansible suitable. Ansible is based on the Python programming language. It's agentless and relies on SSH to push configurations to remote systems.
- Chef and Puppet are heavyweight configuration management solutions that may better suit enterprises with rigorous configuration management requirements. Both are based on the Ruby programming language. In addition, both Chef and Puppet utilize a client-server architecture, so typical implementations require the installation of agents on managed systems.
- The latest version of Red Hat Satellite, Satellite 6, offers much more than configuration file management. It incorporates the use of Puppet and provides a full-fledged configuration management system to enterprise environments.

Using Puppet for configuration management

Puppet allows system administrators to write *infrastructure as code* using a descriptive language to configure machines, instead of using individualized and customized scripts to do so. The

Puppet domain-specific language (DSL) is used to describe the state of the machine, and Puppet can enforce this state. That means that if an administrator mistakenly changes something on the machine, Puppet can enforce the state and return the machine to the desired state. Thus, not only can Puppet code be used to configure a system initially, but it can also be used to keep the state of the system in line with the desired configuration.

A traditional system administrator would need to log into every machine to perform system administration tasks, but that does not scale well. Perhaps the system administrator would create an initial configuration script (e.g., a Kickstart file) for a machine, but once the initial script has run, the machine can (and will) begin to diverge from that initial script configuration. In the best-case scenario, the system administrator would need to periodically check the machine to verify the configuration. In the worst-case scenario, the system administrator would need to figure out why the machine is no longer functioning or rebuild the machine and redeploy the configuration.

Puppet can be used to set a desired state and have the machine converge to that state. The system administrator no longer needs a script for initial configuration and a separate script (or worse yet, human interaction) for verification of the state. Puppet manages system configuration and verifies that the system is in the desired state. This can scale much better than the traditional system administrator using individual scripts for each system.

Puppet architecture

Puppet uses a server/client model. The server is called a Puppet *master*. The Puppet master stores *recipes* or *manifests* (code containing resources and desired states) for the clients. The clients are called Puppet *nodes* and run the Puppet *agent* software. These nodes normally run a Puppet daemon (**agent**) that is used to connect to the Puppet master. The nodes will download the recipe assigned to the node from the Puppet master and apply the configuration if needed.

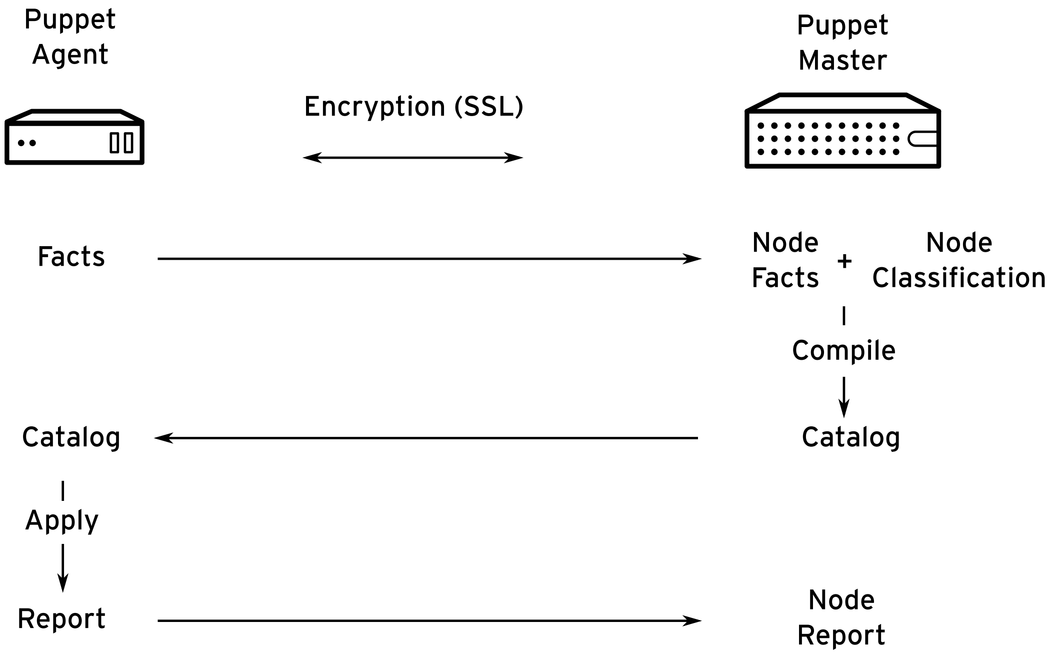


Figure 2.1: A Puppet run

A Puppet run starts with the Puppet node (*not* the Puppet master). By default, the Puppet agent starts a Puppet run every 30 minutes. This run uses secure transmission (SSL) services to pass data back and forth between the Puppet node and the master. The node starts by gathering facts about the system using the **facter** command. Facter includes information on block devices, file systems, network interface, MAC addresses, IP addresses, memory, operating system, CPUs, virtualization, etc. These facts are sent from the node to the master.

Once the Puppet master receives the facts from the Puppet node, the Puppet master compiles a *catalog*, which describes the desired state for each resource configured for the node. The Puppet master checks the host name of the node and matches it to the specific node configuration (called *node classification*) or uses the default configuration if the node does not match. This catalog may include dependency information for the resources (e.g., should Puppet install the package first, or start the service first?).

Once the catalog is compiled, the Puppet master sends the catalog to the node. Puppet will then apply the catalog on the Puppet node, configuring all resources defined in the catalog. Puppet is *idempotent*; Puppet can apply the catalog to a node multiple times without affecting the resultant state.

Once the catalog is applied to the node by the Puppet agent, the node will report back to the Puppet master with the details of the run. This report includes information on what changes were made to the node (if any), and whether the run completed successfully. Puppet's reporting infrastructure has an API, so other applications can download reports from the Puppet master for storage or further analysis.

Configuring a Puppet client

Although Puppet can run in *standalone mode*, where all Puppet clients have Puppet modules locally that are applied to the system, most system administrators find that Puppet works best using a centralized Puppet master. The first step in deploying a Puppet client is to install the *puppet* package:

```
[root@demo ~]# yum install -y puppet
```

Once the *puppet* package is installed, the Puppet client must be configured with the host name of the Puppet master. The host name of the Puppet master should be placed in the **/etc/puppet/puppet.conf** file, under the **[agent]** section. The following snippet shows an example where the Puppet master is named **puppet.lab.example.com**:

```
[agent]
    server = puppet.lab.example.com
```

Note

When an explicit Puppet master is not defined, Puppet uses a default host name of **puppet**. If the DNS search path includes a host named **puppet**, this host will be used automatically.

The final step to be taken on the Puppet client is to start the Puppet agent service and configure it to run at boot time.

```
[root@demo ~]# systemctl start puppet.service
```

```
[root@demo ~]# systemctl enable puppet.service
ln -s '/usr/lib/systemd/system/puppet.service'
 '/etc/systemd/system/multi-user.target.wants/puppet.service'
```

When the Puppet agent service starts for the first time, it will generate a host certificate and send a certificate-signing request to the Puppet master. Once the client certificate request has been sent, the request to sign the client certificate can be seen on the Puppet master. By default, signing of client certificates is performed manually on the Puppet master. However, the Puppet master can also be configured to autosign client certificates so that human intervention is not required for new client registrations. Once the certificate has been signed by the Puppet master, the Puppet agent will receive a catalog and apply any changes needed.

Note

The *puppet* package is offered by the Satellite Tools 6.1 repository.

References

For more information on Puppet, see

> Puppet Introduction
> https://docs.puppetlabs.com/puppet/3.6/reference/index.html

For more information on **facter**, see the **facter**(8) man page.

For more information on configuring a Puppet client, see

> Installing Puppet: Post-Install Tasks
> https://docs.puppetlabs.com/guides/install_puppet/post_install.html

Guided Exercise: Using Configuration Management

In this lab, you will configure a system for configuration management using Puppet.

Resources	
Files	**/etc/puppet/puppet.conf**
Machines	• **servera** • **serverb**

Outcome(s)

You should be able to install and configure Puppet on a system and then use the Puppet agent to configure the web service on the system.

Before you begin

Prepare your systems for this exercise by running the command **lab puppet setup** on your **workstation** machine. This will configure the necessary repositories on your systems, and configure **servera** to act as a puppet master.

```
[student@workstation ~]$ lab puppet setup
```

1. Install the Puppet agent software on **serverb** to implement it as the Puppet client.

```
[root@serverb ~]# yum install -y puppet
```

2. Configure the Puppet agent to point to the Puppet master, **servera.lab.example.com**, by adding the following entry to the **agent** section of the **/etc/puppet/puppet.conf** configuration file.

```
server = servera.lab.example.com
```

3. Determine whether the *httpd* package is installed on this system.

```
[root@serverb ~]# rpm -q httpd
package httpd is not installed
```

4. Enable the **puppet** service so that the Puppet agent starts upon boot.

```
[root@serverb ~]# systemctl enable puppet
Created symlink from /etc/systemd/system/multi-user.target.wants/puppet.service to /
usr/lib/systemd/system/puppet.service.
```

5. In a separate terminal window, use the **journalctl** command to monitor new log entries generated by the Puppet agent.

```
[root@serverb ~]# journalctl -ef
```

6. In the original terminal window, start the **puppet** service. The system will register with the Puppet master and retrieve and apply its configuration.

```
[root@serverb ~]# systemctl start puppet
```

Observe the activities of the Puppet agent by monitoring the new log entries appearing in the the system journal.

```
Jan 12 23:13:42 serverb.lab.example.com systemd[1]: Started Puppet agent.
Jan 12 23:13:42 serverb.lab.example.com systemd[1]: Starting Puppet agent...
Jan 12 23:13:43 serverb.lab.example.com puppet-agent[1579]: Starting Puppet client
 version 3.6.2
Jan 12 23:13:46 serverb.lab.example.com yum[1741]: Installed: apr-1.4.8-3.el7.x86_64
Jan 12 23:13:46 serverb.lab.example.com yum[1741]: Installed: apr-
util-1.5.2-6.el7.x86_64
Jan 12 23:13:47 serverb.lab.example.com yum[1741]: Installed: httpd-
tools-2.4.6-40.el7.x86_64
Jan 12 23:13:47 serverb.lab.example.com yum[1741]: Installed:
 mailcap-2.1.41-2.el7.noarch
Jan 12 23:13:47 serverb.lab.example.com useradd[1751]: new group: name=apache,
 GID=48
Jan 12 23:13:47 serverb.lab.example.com useradd[1751]: new user: name=apache,
 UID=48, GID=48, home=/usr/share/httpd, shell=/sbin/nologin
Jan 12 23:13:49 serverb.lab.example.com systemd[1]: Reloading.
Jan 12 23:13:49 serverb.lab.example.com yum[1741]: Installed:
 httpd-2.4.6-40.el7.x86_64
Jan 12 23:13:50 serverb.lab.example.com puppet-agent[1599]: (/Stage[main]/Main/
Node[serverb.lab.example.com]/Package[httpd]/ensure) created
Jan 12 23:13:50 serverb.lab.example.com systemd[1]: Starting The Apache HTTP
 Server...
Jan 12 23:13:50 serverb.lab.example.com systemd[1]: Started The Apache HTTP Server.
Jan 12 23:13:50 serverb.lab.example.com systemd[1]: Reloading.
Jan 12 23:13:50 serverb.lab.example.com puppet-agent[1599]: (/Stage[main]/Main/
Node[serverb.lab.example.com]/Service[httpd]/ensure) ensure changed 'stopped' to
 'running'
Jan 12 23:13:50 serverb.lab.example.com puppet-agent[1599]: Finished catalog run in
 5.66 seconds
```

7. Verify that the applied configuration resulted in the installation of the *httpd* and the enabling and starting of the **httpd** service.

```
[root@serverb ~]# rpm -q httpd
httpd-2.4.6-40.el7.x86_64
[root@serverb ~]# systemctl is-active httpd
active
[root@serverb ~]# systemctl is-enabled httpd
enabled
```

8. Grade your work, then clean up your systems for the next exercise.

   8.1. Grade your work.

```
[student@workstation ~]$ lab puppet grade
```

8.2. Clean up your systems for the next exercise.

```
[student@workstation ~]$ lab puppet reset
```

Configuring Change Tracking

Objectives

After completing this section, students should be able to implement change tracking to keep systems homogeneous.

Using intrusion detection software to monitor changes

System stability is put at risk when configuration files are deleted or modified without authorization or careful supervision. How can a change to an important file or directory be detected? This problem can be solved by using intrusion detection software to monitor files for changes. *Advanced Intrusion Detection Environment* (AIDE) can be configured to monitor files for a variety of changes including permission or ownership changes, timestamp changes (modification or access timestamps), or content changes.

Installing AIDE

To get started with AIDE, install the RPM package that provides the AIDE software. This package provides useful documentation on tuning the software to monitor the specific changes of interest.

```
[root@demo ~]# yum install -y aide
```

Configuring AIDE

Once the software is installed, it needs to be configured. The **/etc/aide.conf** file is the primary configuration file for AIDE. It has three types of configuration directives: configuration lines, selection lines, and macro lines.

Configuration lines take the form **param = value**. When *param* is not a built-in AIDE setting, it is a group definition that lists which changes to look for. For example, the following group definition can be found in **/etc/aide.conf** installed by default:

```
PERMS = p+i+u+g+acl+selinux
```

The previous line defines a group called **PERMS** that looks for changes in file permissions (**p**), inode (**i**), user ownership (**u**), group ownership (**g**), ACLs (**acl**), or SELinux context (**selinux**).

Selection lines define which checks are performed on matched directories. The following lines are examples of selection lines:

```
/dir1   group
=/dir2  group
!/dir3
```

The first line performs the group of checks on **/dir1** and all of the files and directories below it. The second line performs the group of checks specified on **/dir2**. The equal sign specifies the check is to be done on the directory only and does not recurse below it. The third line in the previous example excludes **/dir3** and all of the files below it from any checks.

The third type of directive are macro lines. They define variables and their definition have the following syntax:

```
@@define VAR value
```

@@{VAR} is a reference to the macro defined previously.

The **aide.conf** configuration file can also include comments. Comments are lines that start with the # character.

Initialize the AIDE database

Execute the **aide --init** command to initialize the AIDE database. AIDE will scan the file system and record all of the current information about the files and directories specified in the **/etc/aide.conf** configuration file.

```
[root@demo ~]# aide --init
```

The previous command creates the database in a file called **/var/lib/aide/aide.db.new.gz**. This file has to be renamed to **/var/lib/aide/aide.db.gz** because this is where AIDE expects the database to be when performing file system checks.

Since the database can be updated from the machine, it may be recommended to copy the **aide.db.gz** to another system.

> Note
>
> When files that contain binary executables are captured in an AIDE database, the **prelink cron** job should be disabled. Otherwise, the first time **prelink** is executed, the binaries will show up as changed files.

Check for file system changes

After the database is created, a file system check can be performed by executing the **aide --check** command. This command will scan the file system and compare the current state of files with the information in the AIDE database. Any differences that are found will be displayed, showing both the original file state and the new condition.

```
[root@demo ~]# aide --check
```

System auditing with auditd

auditd is the user-space component of the Linux auditing subsystem. When **auditd** is running, audit messages sent by the kernel will be collected in the log file configured for **auditd** (normally **/var/log/audit/audit.log**). If **auditd** is not running for any reason, kernel audit messages will be sent to **rsyslog**.

Without any extra configuration, only a limited number of messages is passed through the audit system, mainly authentication/authorization messages (users logging in, **sudo** being used, etc.) and SELinux messages. With a tool called **auditctl**, (security) administrators can add auditing rules on any system call they want, including any and all file operations.

Writing custom audit rules

Auditing rules can be added from the command line using the **auditctl** tool. By default, auditing rules will be added to the bottom of the current list, but rules can be inserted at the top as well.

Important

Since audit rules work on a first-match-wins basis, ordering is important for rules to function as intended. For example, if a rule is configured to log all access to **/etc** and another rule is configured to log all access to **/etc/sysconfig** with a different key, the second rule would never be triggered since the first rule already covers that access. Switching the rules around would make the rules work as intended.

One of the easiest rules to add is a *watch* rule. Watches can be set on files and directories, and can optionally just trigger on certain types of access (read, write, attribute change, and execute). Watches are triggered on the **open** system call, and not on individual **read** or **write** calls.

Some examples of watches:

-
  ```
  auditctl -w /etc/passwd -p wa -k user-edit
  ```

  Adds a watch to **/etc/passwd**, for write and attribute change access, and adds a custom key of **user-edit** to all log messages.

-
  ```
  auditctl -w /etc/sysconfig/ -p rwa -k sysconfig-access
  ```

  Adds a recursive watch to the **/etc/sysconfig** directory and all files and directories beneath it. Watches for read, write, and attribute change access. Labels log messages with a custom key of **sysconfig-access**.

-
  ```
  auditctl -w /bin -p x
  ```

  Audits all executions of binaries under **/bin**.

Removing rules

File system watch rules can be removed by using the **-W** option instead of **-w** with the original rule. Rules added with **-a** or **-A** can be removed by replacing **-[aA]** with **-d**.

To remove all rules, use the **auditctl -D** command.

Persistent rules

To make auditing rules persistent, add them to the file **/etc/audit/rules.d/audit.rules**. This file contains **auditctl** commands as they would be entered on the command line, but without the **auditctl** command itself in front. Empty lines are permitted, and comments are indicated by starting a line with the hash character (**#**).

Whenever **auditd** is started, it will activate all rules from **/etc/audit/rules.d/audit.rules**. The **auditctl** command can be directed to parse a file with the **-R** option.

Important

Due to a bug, **auditd** cannot currently be restarted with **systemctl restart auditd**. Therefore, it is necessary to use **service auditd restart** in order to reload changes to persistent rules.

Reading audit messages

Audit messages logged to **/var/log/audit/audit.log** carry a lot of information with them. Different types of messages consist of different fields, but each field is labeled. The following is a dissection of a typical audit record straight from **/var/log/audit/audit.log**.

```
type=SYSCALL❶ msg=audit(1371716130.596:28708)❷: arch=c000003e syscall=2 success=yes
 exit=4 a0=261b130 a1=90800 a2=e a3=19 items=1 ppid=2548 pid=26131 auid=500❸ uid=0
 gid=0 euid=0❹ suid=0 fsuid=0 egid=0 sgid=0 fsgid=0 tty=pts0 ses=1 comm="aureport"❺
 exe="/sbin/aureport"❻ subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
 key="audit-access"❼
type=CWD msg=audit(1371716130.596:28708):  cwd="/root"
type=PATH msg=audit(1371716130.596:28708): item=0 name="/var/log/audit" inode=17998
 dev=fd:01 mode=040750 ouid=0 ogid=0 rdev=00:00 obj=system_u:object_r:auditd_log_t:s0
```

❶ The *type* of audit message. In this case, a system call is audited, but many different types can be found.

❷ The timestamp and ID of the audit message. The part before the colon (in this case, **1371716130.596** is the timestamp in number of seconds since the epoch, while the part after the colon (**28708**) is the audit event number.

To convert a timestamp from epoch time to local time zone, use use the command **date --date=@<*epoch-time*>**.

As shown in this example, a single *event* might trigger multiple lines of logging, with different *types*.

❸ The original, or *Audit*, UID of the user that triggered this audit message. The audit system will remember the original UID even when a user elevates their privileges through tools like **su** or **sudo**.

❹ The *Effective* UID of the user that triggered this audit message. This might be different from the *Audit UID*, and even the regular UID; for example, when running a binary with the **SUID** or **SGID** bit set.

❺ The name of the executable that triggered this audit message, as it would be reported by tools like **ps** and **top**.

❻ The name of the binary on disk for the process that triggered this audit message.

❼ An identifier that can be used when searching for events. Keys are typically set by custom auditing rules to make it easier to filter for certain types of events.

Searching for events

The auditing system ships with a powerful tool for searching audit logs: **ausearch**. Not only does **ausearch** let administrators easily search for and filter on various types of events, it can also interpret events by translating numeric values into (more) readable values like usernames or system call names. The following table lists some of the more common options to **ausearch**, but many more exist, including options to search for events based on user, terminal, or even virtual machine.

Option	Description
-i	Interpret log line, translate numeric values into names.
--raw	Print raw log entries, do not put record separators between entries.

Option	Description
-a *<EVENT-ID>*	Show all lines for the event with ***<EVENT-ID>*** as the event ID.
--file *<FILENAME>*	Search for all events touching a specific file name.
-k *<KEY>*	Search for all events labeled with ***<KEY>***.
--start *[start-date] [start-time]*	Only search for events after ***start-date*** and ***start-time***. If a starting time is left off, **midnight** is assumed. Omitting a starting date will assume **today**. The time format that should be used depends on a system's current locale. Other values that may be used include **recent** (past 10 minutes), **this-week**, **this-month**, and **this-year**. **--end** can also be used with the same syntax.

As an example, **ausearch** can be used to show an interpreted version of the previous example searched by the key value of **audit-access**.

```
[root@demo ]# ausearch -i -k audit-access
----
type=PATH msg=audit(06/20/2013 10:15:30.596:28708) : item=0 name=/var/log/
audit inode=17998 dev=fd:01 mode=dir,750 ouid=root ogid=root rdev=00:00
 obj=system_u:object_r:auditd_log_t:s0
type=CWD msg=audit(06/20/2013 10:15:30.596:28708) :  cwd=/root
type=SYSCALL msg=audit(06/20/2013 10:15:30.596:28708) : arch=x86_64 syscall=open
 success=yes exit=4 a0=261b130 a1=90800 a2=e a3=19 items=1 ppid=2548 pid=26131
 auid=student uid=root gid=root euid=root suid=root fsuid=root egid=root
 sgid=root fsgid=root tty=pts0 ses=1 comm=aureport exe=/sbin/aureport
 subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 key=audit-access
```

References

For more information on **aide**, see the **aide**(1) and **aide.conf**(5) man pages.

For more information on **auditd**, see the **auditd**(8), **auditd.conf**(5), **auditctl**(8), and **audit.rules**(7), and **ausearch**(8) man pages.

Guided Exercise: Configuring Change Tracking

In this lab, you will configure and perform system auditing using the Linus audit system.

Resources	
Files	**`/etc/audit/rules.d/audit.rules`**
Machines:	• **`servera`** • **`serverb`**

Outcome(s)

You should be able to detect changes to the content and attributes of files and directories located on a file system.

Before you begin

Log into **`servera`** as the root user.

1. Verify that the audit daemon is enabled and running on **`servera`**.

```
[root@servera ~]# systemctl status auditd
● auditd.service - Security Auditing Service
   Loaded: loaded (/usr/lib/systemd/system/auditd.service; enabled; vendor preset:
 enabled)
   Active: active (running) since Mon 2015-12-14 16:45:54 EST; 6h ago
  Process: 469 ExecStartPost=/sbin/augenrules --load (code=exited, status=0/SUCCESS)
 Main PID: 468 (auditd)
   CGroup: /system.slice/auditd.service
           └─468 /sbin/auditd -n

Dec 14 16:45:54 servera.ilt.example.com augenrules[469]: No rules
Dec 14 16:45:54 servera.ilt.example.com augenrules[469]: enabled 0
Dec 14 16:45:54 servera.ilt.example.com augenrules[469]: flag 1
Dec 14 16:45:54 servera.ilt.example.com augenrules[469]: pid 0
Dec 14 16:45:54 servera.ilt.example.com augenrules[469]: rate_limit 0
Dec 14 16:45:54 servera.ilt.example.com augenrules[469]: backlog_limit 320
Dec 14 16:45:54 servera.ilt.example.com augenrules[469]: lost 0
Dec 14 16:45:54 servera.ilt.example.com augenrules[469]: backlog 0
Dec 14 16:45:54 servera.ilt.example.com auditd[468]: Init complete, auditd 2.4.1
 listening for events (startup state enable)
Dec 14 16:45:54 servera.ilt.example.com systemd[1]: Started Security Auditing
 Service.
```

2. Create a rule to audit the changes to the file and directory contents located under the **`/etc`** directory. To facilitate the identification of records created by the rule, associate the records with the keyword **`etc_content`**.

```
[root@servera ~]# auditctl -w /etc -p w -k etc_content
```

3. Create another rule to audit the changes to the attributes of files and directories located under the **`/etc`** directory. To facilitate the identification of records created by the rule, associate the records with the keyword **`etc_attribute`**.

```
[root@servera ~]# auditctl -w /etc -p a -k etc_attribute
```

4. Make the audit rules persistent by adding the following entries to the end of **/etc/audit/rules.d/audit.rules**.

```
-w /etc -p w -k etc_content
-w /etc -p a -k etc_attribute
```

5. Copy the contents of the **/etc/shadow** file into a new file named **/etc/shadow.copy**.

```
[root@servera ~]# cp /etc/shadow /etc/shadow.copy
```

6. Change the permissions on the **/etc/shadow.copy** file so that it is readable by anyone.

```
[root@servera ~]# chmod +r /etc/shadow.copy
```

7. Use the **ausearch** command to extract the audit records for changes to the contents of files and directories under the **/etc** directory.

```
[root@servera ~]# ausearch -k etc_content
```

8. Use the **ausearch** command to extract the audit records for changes to the attributes of files and directories under the **/etc** directory.

```
[root@servera ~]# ausearch -k etc_attribute
```

9. Grade your work, then clean up your systems for the next exercise.

   9.1. Grade your work.

   ```
   [student@workstation ~]$ lab audit grade
   ```

   9.2. Clean up your systems for the next exercise.

   ```
   [student@workstation ~]$ lab audit reset
   ```

Lab: Being Proactive

In this lab, you will configure systems for monitoring, remote logging, configuration management, and auditing.

Resources	
Files	• **/etc/puppet/puppet.conf** • **/etc/rsyslog.conf** • **/etc/logrotate.d/syslog** • **/etc/audit/rules.d/audit.rules**
Machines	• **servera** • **serverb**

Outcome(s)

You should be able to configure a server as a central log host and a Puppet master. You should also be able to configure another host for monitoring, logging to the log host, communicating with the Puppet master, and auditing for file system access.

Before you begin

Prepare your systems for this exercise by running the command **lab proactive setup** on your **workstation** machine.

```
[student@workstation ~#$ lab proactive setup
```

On **servera**, configure the **rsyslog** service so that it serves as a central log host. For more reliable syslog messages delivery, configure the central log host to accept message delivery from remote hosts using TCP. To better organize messages, create a new rule which writes the syslog messages generated by each host to separate files under the **/var/log/loghost** directory. Each log file should be named after the host name of each system generating the messages. For better performance, configure logging so that a sync is not performed after each message logged. Add the new log files to the log rotation schedule so they do not grow to unmanageable size.

On **serverb**, create an audit rule to monitor the execution of the **/usr/sbin/useradd** binary. Each execution should be logged with the label **add_user**. Make the audit rule persistent so it is applied at each system boot. Also, install and configure the Performance Co-pilot software to run on **serverb** so that it can be utilized for querying both current and historical system performance statistics. To centralize logging, configure **serverb** for remote logging to the central log host. Lastly, configure **serverb** for configuration management by the Puppet master running on **servera**.

1. Configure the **rsyslog** service on **servera** so that syslog messages from remote hosts are accepted using the TCP protocol and that the messages are written to separate files for each host.

2. Modify the firewall on **servera** to allow incoming syslog messages from remote hosts to be delivered using TCP.

3. Add the following entry to the list of files in the **/etc/logrotate.d/syslog** configuration file so that the new log files are placed into the log rotation schedule.

   ```
   /var/log/loghost/*.log
   ```

4. Configure **serverb** to log syslog messages remotely to **servera** using the TCP protocol.

5. On **serverb**, create a rule to audit execution access to the **/usr/sbin/useradd** file.

6. On **serverb**, make the audit rule persistent so it is applied at each system boot.

7. Install the Performance Co-pilot software on **serverb** by installing the *pcp* package.

8. Start and enable the **pmcd** service to collect system performance metrics.

9. Start and enable the **pmlogger** service to archive system performance metrics.

10. Install the Puppet agent software on **serverb** to implement it as the Puppet client.

11. Configure the Puppet agent to point to the Puppet master, **servera.lab.example.com**, by adding the following entry to the **agent** section of the **/etc/puppet/puppet.conf** configuration file.

12. Enable the **puppet** service so that the Puppet agent starts upon boot.

13. Start the **puppet** service. The system will register with the Puppet master and retrieve and apply its configuration.

14. Grade your work, then clean up your systems for the next exercise.

    14.1. Grade your work.

    ```
    [student@workstation ~]$ lab proactive grade
    ```

    14.2.Reset all of your virtual machines to provide a clean environment for the next exercises.

Solution

In this lab, you will configure systems for monitoring, remote logging, configuration management, and auditing.

Resources	
Files	• **/etc/puppet/puppet.conf** • **/etc/rsyslog.conf** • **/etc/logrotate.d/syslog** • **/etc/audit/rules.d/audit.rules**
Machines	• **servera** • **serverb**

Outcome(s)

You should be able to configure a server as a central log host and a Puppet master. You should also be able to configure another host for monitoring, logging to the log host, communicating with the Puppet master, and auditing for file system access.

Before you begin

Prepare your systems for this exercise by running the command **lab proactive setup** on your **workstation** machine.

```
[student@workstation ~#$ lab proactive setup
```

On **servera**, configure the **rsyslog** service so that it serves as a central log host. For more reliable syslog messages delivery, configure the central log host to accept message delivery from remote hosts using TCP. To better organize messages, create a new rule which writes the syslog messages generated by each host to separate files under the **/var/log/loghost** directory. Each log file should be named after the host name of each system generating the messages. For better performance, configure logging so that a sync is not performed after each message logged. Add the new log files to the log rotation schedule so they do not grow to unmanageable size.

On **serverb**, create an audit rule to monitor the execution of the **/usr/sbin/useradd** binary. Each execution should be logged with the label **add_user**. Make the audit rule persistent so it is applied at each system boot. Also, install and configure the Performance Co-pilot software to run on **serverb** so that it can be utilized for querying both current and historical system performance statistics. To centralize logging, configure **serverb** for remote logging to the central log host. Lastly, configure **serverb** for configuration management by the Puppet master running on **servera**.

1. Configure the **rsyslog** service on **servera** so that syslog messages from remote hosts are accepted using the TCP protocol and that the messages are written to separate files for each host.

   1.1. Enable TCP syslog reception in **/etc/rsyslog.conf** by uncommenting the following lines.

   ```
   $ModLoad imtcp
   ```

```
$InputTCPServerRun 514
```

1.2. Under the "**#### RULES ####**" section, add the following rule for dynamic log file names.

```
$template DynamicFile,"/var/log/loghost/%HOSTNAME%.log"
*.* -?DynamicFile
```

1.3. Restart the **rsyslog** service for the configuration changes to take effect.

```
[root@servera ~]# systemctl restart rsyslog
```

2. Modify the firewall on **servera** to allow incoming syslog messages from remote hosts to be delivered using TCP.

2.1. Allow incoming traffic on TCP port 514.

```
[root@servera ~]# firewall-cmd --add-port=514/tcp --permanent
success
```

2.2. Reload **firewalld** for the firewall change to take effect.

```
[root@servera ~]# firewall-cmd --reload
```

3. Add the following entry to the list of files in the **/etc/logrotate.d/syslog** configuration file so that the new log files are placed into the log rotation schedule.

```
/var/log/loghost/*.log
```

4. Configure **serverb** to log syslog messages remotely to **servera** using the TCP protocol.

4.1. On **serverb**, add the following rule to **/etc/rsyslog.conf** so that all syslog messages are delivered remotely to **servera** using TCP on port 514.

```
*.* @@servera.lab.example.com:514
```

4.2. Restart the **rsyslog** service on **serverb** for the configuration changes to take effect.

```
[root@serverb ~]# systemctl restart rsyslog
```

5. On **serverb**, create a rule to audit execution access to the **/usr/sbin/useradd** file.

```
[root@serverb ~]# auditctl -w /usr/sbin/useradd -p x -k add_user
```

6. On **serverb**, make the audit rule persistent so it is applied at each system boot.

```
[root@serverb ~]# echo '-w /usr/sbin/useradd -p x -k add_user' >> /etc/audit/
rules.d/audit.rules
```

7. Install the Performance Co-pilot software on **serverb** by installing the *pcp* package.

```
[root@serverb ~]# yum install -y pcp
```

8. Start and enable the **pmcd** service to collect system performance metrics.

```
[root@serverb ~]# systemctl start pmcd
[root@serverb ~]# systemctl enable pmcd
Created symlink from /etc/systemd/system/multi-user.target.wants/pmcd.service to /
usr/lib/systemd/system/pmcd.service.
[root@serverb ~]# systemctl status pmcd
● pmcd.service - Performance Metrics Collector Daemon
   Loaded: loaded (/usr/lib/systemd/system/pmcd.service; enabled; vendor preset:
 disabled)
   Active: active (exited) since Fri 2015-12-11 08:26:38 EST; 8s ago
     Docs: man:pmcd(8)
 Main PID: 1896 (code=exited, status=0/SUCCESS)
   CGroup: /system.slice/pmcd.service
           ├─1942 /usr/libexec/pcp/bin/pmcd
           ├─1945 /var/lib/pcp/pmdas/root/pmdaroot
           ├─1946 /var/lib/pcp/pmdas/proc/pmdaproc -d 3
           ├─1947 /var/lib/pcp/pmdas/xfs/pmdaxfs -d 11
           └─1948 /var/lib/pcp/pmdas/linux/pmdalinux

Dec 14 08:26:38 serverb.lab.example.com systemd[1]: Starting Performance Metrics
 Collector Daemon...
Dec 14 08:26:38 serverb.lab.example.com pmcd[1896]: Starting pmcd ...
Dec 14 08:26:38 serverb.lab.example.com systemd[1]: Started Performance Metrics
 Collector Daemon.
```

9. Start and enable the **pmlogger** service to archive system performance metrics.

```
[root@serverb ~]# systemctl start pmlogger
[root@serverb ~]# systemctl enable pmlogger
Created symlink from /etc/systemd/system/multi-user.target.wants/pmlogger.service
 to /usr/lib/systemd/system/pmlogger.service.
[root@serverb ~]# systemctl status pmlogger
● pmlogger.service - Performance Metrics Archive Logger
   Loaded: loaded (/usr/lib/systemd/system/pmlogger.service; enabled; vendor preset:
 disabled)
   Active: active (exited) since Fri 2015-12-11 08:28:38 EST; 6s ago
     Docs: man:pmlogger(1)
 Main PID: 2535 (code=exited, status=0/SUCCESS)
   CGroup: /system.slice/pmlogger.service
           └─6489 /usr/libexec/pcp/bin/pmlogger -P -r -T24h10m -c config.default -m
 pmlogger_check 20151211.08.28

Dec 14 08:28:38 serverb.lab.example.com systemd[1]: Starting Performance Metrics
 Archive Logger...
Dec 14 08:28:38 serverb.lab.example.com pmlogger[2535]: /usr/share/pcp/lib/pmlogger:
 Warning: Performance Co-Pilot archive ...bled.
Dec 14 08:28:38 serverb.lab.example.com pmlogger[2535]: To enable pmlogger, run the
 following as root:
```

```
Dec 14 08:28:38 serverb.lab.example.com pmlogger[2535]: # /usr/bin/systemctl enable
 pmlogger.service
Dec 14 08:28:38 serverb.lab.example.com pmlogger[2535]: Starting pmlogger ...
Dec 14 08:28:38 serverb.lab.example.com systemd[1]: Started Performance Metrics
 Archive Logger.
Hint: Some lines were ellipsized, use -l to show in full.
```

10. Install the Puppet agent software on **serverb** to implement it as the Puppet client.

```
[root@serverb ~]# yum install -y puppet
```

11. Configure the Puppet agent to point to the Puppet master, **servera.lab.example.com**, by adding the following entry to the **agent** section of the **/etc/puppet/puppet.conf** configuration file.

```
server = servera.lab.example.com
```

12. Enable the **puppet** service so that the Puppet agent starts upon boot.

```
[root@serverb ~]# systemctl enable puppet
Created symlink from /etc/systemd/system/multi-user.target.wants/puppet.service to /
usr/lib/systemd/system/puppet.service.
```

13. Start the **puppet** service. The system will register with the Puppet master and retrieve and apply its configuration.

```
[root@serverb ~]# systemctl start puppet
```

14. Grade your work, then clean up your systems for the next exercise.

14.1. Grade your work.

```
[student@workstation ~]$ lab proactive grade
```

14.2. Reset all of your virtual machines to provide a clean environment for the next exercises.

Summary

In this chapter, you learned:

- A list of the system performance data collected by the Performance Metrics Collector Daemon, **pmcd**, can be displayed with **pminfo**.
- The **pmval** command can be used to retrieve current and historical system performance data collected by **pmcd**.
- Implementation of a central log host requires the configuration of rsyslog on the central log host as well as the remote clients.
- Log data from remote hosts can be logged separately on the central log host with the use of templates to generate dynamic log file names.
- Puppet agent, by default, attempts to find the Puppet master using the name **puppet.lab.example.com**.
- To point the Puppet agent to a Puppet master of a different name, configure the setting using the **server** parameter in the **agent** section of **/etc/puppet/puppet.conf**.
- The AIDE database is initialized with **aide --init**.
- Changes to the file system can be detected by comparison against the AIDE database using **aide --check**.
- Audit rules are created using the **auditctl** command.
- Events in audit logs can be searched using the **ausearch** command.

CHAPTER 3

TROUBLESHOOTING BOOT ISSUES

Overview	
Goal	Identify and resolve issues that can affect a system's ability to boot.
Objectives	• Fix boot issues on traditional BIOS systems. • Fix boot problems on UEFI systems. • Identify and resolve service failures affecting boot. • Regain root control of a system.
Sections	• Resolving Boot Loader Issues on BIOS Systems (and Guided Exercise) • Resolving Boot Loader Issues on UEFI Systems (and Quiz) • Dealing with Failing Services (and Guided Exercise) • Recovering a Root Password (and Guided Exercise)
Lab	• Troubleshooting Boot Issues

Resolving Boot Loader Issues on BIOS Systems

Objectives

After completing this section, students should be able to repair boot issues on traditional BIOS systems.

A traditional boot sequence

A *Basic Input Output System* (BIOS)-powered system goes through a number of steps from being powered down to having a running Red Hat Enterprise Linux system. The following list gives a high-level overview of the steps being taken. This list also applies to virtual machines that emulate a traditional BIOS system. This list assumes that the **grub2** boot loader is being used. For different boot loaders, the list will differ.

1. The BIOS firmware is started and performs a *Power On Self Test* (POST).

2. The BIOS scans for (possible) boot devices, and orders them according to a user-set preference.

3. The boot devices are scanned in order for a boot firmware (such as a PXE ROM on network cards), an *Master Boot Record* (MBR), or a partition marked as bootable. If found, the BIOS executes it.

4. The first-stage boot loader stored in the MBR loads the stage 1.5 (drivers) and stage 2 boot loader from disk and executes it.

5. The boot loader loads a configuration file from disk. In the case of **grub2**, this will be **/boot/grub2/grub.cfg**.

6. The configuration file is parsed, and based on its contents, a boot entry is selected automatically or by the user.

7. The kernel and initial ramdisk referenced in the boot entry are loaded from disk, and control is handed over to the kernel.

8. The kernel starts, and initializes hardware using the drivers found in the initial ramdisk. A simple init system is also started from the ramdisk.

9. The scripts in the initial ramdisk mount the root file system of the target system, then switch root to the newly mounted file system, and hand over control to **/sbin/init** on the target root file system.

10. The init system mounts file systems and starts services according to its configuration.

GRUB2

The boot loader used on Red Hat Enterprise Linux is **grub2**, the second major version of the *GRand Unified Bootloader*. **grub2** stores file on a BIOS system in a number of different locations:

/boot/

Kernels and initial ramdisks. Subdirectories contain other files.

/boot/grub2/
Configuration files, extension modules, and themes.

/boot/grub2/grub.cfg
The main **grub2** configuration file. This file is autogenerated, and should normally not be edited manually.

As a convenience, **/etc/grub2.cfg** is a symbolic link to this file.

/etc/grub.d/
This directory contains a helper script to generate a main **grub2** configuration file.

/etc/default/grub
This file contains variables used in the generation of the main **grub2** configuration file.

/boot/grub2/grubenv
A file of exactly 1 KiB, used as storage for variables, such as the default or "saved" boot entry.

The **grub2-editenv** tool can be used to read and modify these settings.

The Master Boot Record (MBR)

In order to boot **grub2** from disk on a BIOS system, there are two options: Store the first part of the **grub2** boot loader in the *Master Boot Record* (MBR) of a disk, or store it in the first sector of a partition that is marked as "bootable".

The problem with the MBR

Using the MBR has a big restriction: A typical MBR is only 512 bytes in size, and part of that space is used for the partition table for that disk, leaving only 446 bytes for the boot loader.

To work around this issue, **grub2** can use the "boot track", "MBR gap", or "embedding area" on the disk to store the extra modules and files it needs. This is the space between the MBR and the first partition on the disk. In order to work reliably, there needs to be at least 31 KiB of space available this way (63 sectors on a 512-byte sector disk). If a disk has been partitioned by **anaconda**, the first partition will usually start at sector 2048, leaving roughly 1 MiB of space for the embedding area, ample room for **grub2**.

Configuring grub2

Under normal circumstances, an administrator should not have to manually configure the **grub2** boot loader. When a new kernel is installed, it will be added to the configuration automatically, and when a kernel is removed, the corresponding entry in the boot loader menu is removed automatically as well.

An administrator might want to tweak some parameters that are passed into a kernel at startup by **grub2**. The best way to do this is by editing **/etc/default/grub**, and then forcing a recreation of the main **grub2** configuration file.

The settings in **/etc/default/grub** that are of interest are listed here:

GRUB_TIMEOUT
The number of seconds the **grub2** menu is displayed before the default entry is booted automatically.

GRUB_DEFAULT

The number of the default entry that should be started whenever a user does not select another entry. **grub2** starts counting at **0**.

If this variable is set to the string **saved**, the entry will be taken from **/boot/grub2/grubenv**.

GRUB_CMDLINE_LINUX

This variable contains a list of extra kernel command-line options that should be added to every single Linux kernel. Typical uses include "**rhgb quiet**" for a graphical boot, "**console=xxxxxx**" for selecting a kernel console device, and "**crashkernel=xxxx**" for configuring automatic kernel crash dumping.

After updating the file **/etc/default/grub**, changes can be applied by running the command **grub2-mkconfig -o /boot/grub2/grub.cfg**. This will generate a fresh configuration file using the scripts in **/etc/grub.d/** and the settings in **/etc/default/grub**. If no output file is specified with the **-o** option, a configuration file will be written to standard output.

Reinstalling grub2 into the MBR

If for some reason the MBR, or the embedding area, has become damaged, an administrator will have to reinstall **grub2** into the MBR. Since this implies that the system is currently unbootable, this is typically done from within a Live CD, or from within the *rescue* environment provided by the **anaconda** installer.

The following procedure explains how to boot into a rescue environment, and reinstall **grub2** into the MBR from there. If an administrator is still logged into a running system, the procedure can be shortened to just the **grub2-install** command.

1. Boot from an installation source; this can be a DVD image, a netboot CD, or from PXE providing a RHEL installation tree.

2. In the boot menu for the installation media, select the **Rescue an installed system** option, or edit the kernel command line to include the word **rescue**.

3. When prompted about mounting the disks for the target system to be rescued, select option 1 (**Continue**). This will mount the system under **/mnt/sysimage**.

4. Press **Enter** to obtain a shell when prompted. This shell will live inside the installation/rescue environment, with the target system mounted under **/mnt/sysimage**.

   This shell has a number of tools available for rescuing a system, such as all common file system, disk, LVM, and networking tools. The various **bin** directories of the target system are added to the default executable search path (**${PATH}**) as well.

5. **chroot** into the **/mnt/sysimage** directory.

   ```
   sh-4.2# chroot /mnt/sysimage
   ```

6. Verify that **/boot** is mounted. Depending on the type of installation, **/boot** can be on a separate partition, or it can be part of the root file system.

   ```
   sh-4.2# ls -l /boot
   ```

7. Use the command **grub2-install** to rewrite the boot loader sections of the MBR and the embedding area. This command will need the block device that represents the main disk as an argument. If unsure about the name of the main disk, use **lsblk** and **blkid** to identify it.

   ```
   sh-4.2# grub2-install /dev/vda
   ```

8. Reboot the system. This can be done by exiting both the **chroot** shell and the rescue shell.

9. Wait for the system to reboot twice. After the first reboot, the system will perform an extensive SELinux file system relabel, then reboot again automatically to ensure that proper contexts are being used. This relabel is triggered automatically by the **anaconda** rescue system creating the file **/.autorelabel**.

References

The chapter on the **grub2** boot loader in the Red Hat Enterprise Linux System Administration Guide found on **https://access.redhat.com/documentation**

grub2-install(1) and **grub2-mkconfig**(1) man and info pages

Guided Exercise: Resolving Boot Loader Issues on BIOS Systems

In this lab, you will restore the boot loader on a BIOS-based machine that is refusing to boot.

Resources	
Machines	• **workstation** • **servera**

Outcome(s)

You should be able to fix or reinstall the boot loader on a BIOS-based system using **rescue** mode from **anaconda**.

Before you begin

From your **workstation** system, run the command **lab biosbootbreak setup**. This command will modify your **servera** system so that it is unable to boot, then reboot it.

```
[student@workstation ~]$ lab biosbootbreak setup
```

Your **servera** is unable to boot. Investigate and fix the issue.

1. View the console for your **servera** system. What information is displayed, and what can be deducted from this information?

   1.1.
   ```
   Booting from Hard Disk...
   ```

   Followed by a blinking cursor.

   1.2. This would indicate an issue in the contents of the MBR. If the problem had been in **grub2.cfg**, there would have been a more informative error message.

2. Reboot your **servera** into the **anaconda** rescue system.

   2.1. Reboot, or power cycle, your **servera** machine.

   2.2. When the boot menu is displayed select the entry to boot from the network.

   2.3. Highlight the **Boot into rescue environment** line and press **Enter**.

   2.4. Wait for **anaconda** to finish starting, then when prompted about mounting your system under **/mnt/sysimage**, press **1** and hit **Enter** to continue.

   2.5. When prompted, press **Enter** to open a shell on your system.

3. Reinstall **grub2** into the MBR of **/dev/vda**. You will have to **chroot** to the mounted system on **/mnt/sysimage** in order to proceed.

   3.1. **chroot** into the system at **/mnt/sysimage**.

```
sh-4.2# chroot /mnt/sysimage
```

3.2. Reinstall the **grub2** boot loader on **`/dev/vda`**.

```
sh-4.2# grub2-install /dev/vda
Installing for i386-pc platform.
Installation finished. No error reported.
```

4. Exit from **anaconda**, and allow the system to reboot normally.

4.1. Press **`Ctrl`**+**`D`** to exit from the **`chroot`** shell.

4.2. Press **`Ctrl`**+**`D`** to exit from **anaconda**. The system will now reboot *twice*. The second reboot is forced by the system after performing a complete SELinux relabel.

Important

During the first reboot, your system may appear to become unresponsive; this is the SELinux relabel being performed in the background. Do not interrupt the system, or it will restart the relabel at the next reboot.

Resolving Boot Loader Issues on UEFI Systems

Objectives

After completing this section, students should be able to fix boot problems on UEFI systems.

What is UEFI?

Introduced in 2005, the *Unified Extensible Firmware Interface* (UEFI) replaces the older *Extensible Firmware Interface* (EFI) and the system BIOS. A replacement for the older BIOS was sought to work around the limits the BIOS imposed, such as 16-bit processor mode and only 1 MiB of addressable space.

UEFI also allows booting from disks larger than 2 TiB, as long as they are partitioned using a *GUID Partition Table* (GPT).

One of the big differences between UEFI systems and BIOS systems is the way they boot. While on a BIOS system the BIOS has to search for bootable devices, operating systems can "register" themselves with the UEFI firmware. This makes it easier for end users to select which operating system they want to boot in a multiboot environment.

grub2 and UEFI

In order to boot a UEFI system using **grub2**, an *EFI System Partition* (ESP) needs to be present on disk. This partition should have the GPT UUID **C12A7328-F81F-11D2-BA4B-00A0C93EC93B**, or the MBR type **0xEF**. This partition needs to be formatted with a FAT file system, and should be large enough to hold all bootable kernels and the boot loader itself. A size of 512 MiB is recommended, but smaller sizes will work. This partition should be mounted at **/boot/efi**, and will normally be created by the **anaconda** installer.

Main grub2 difference between BIOS and UEFI

While most of the configuration syntax and tools remain the same between BIOS and UEFI boots, some small changes do apply:

linux16/initrd16 vs. **linuxefi/initrdefi**

The configuration commands to load a kernel and initial ramdisk switch from **linux16** and **initrd16** to **linuxefi** and **initrdefi** respectively.

This change is necessary since kernels need to be loaded differently on an UEFI system than on a BIOS system. The **grub2-mkconfig** command automatically recognizes an UEFI system and makes the correct changes.

/boot/grub2 vs. **/boot/efi/EFI/redhat**

On UEFI systems, the directory holding all **grub2** configuration files and extras is moved to **/boot/efi/EFI/redhat**. This is a location on the ESP, making it accessible to the UEFI firmware.

grub2-install

While the **grub2-install** command is more than capable of installing a boot loader onto an UEFI system, it should *never* be used directly.

Calling **`grub2-install`** on an UEFI system will generate a new **`/boot/efi/EFI/redhat/grubx64.efi`** file. On Red Hat Enterprise Linux 7 systems, this file should be the prebaked version from the *grub2-efi* package.

The **`grub2-install`** command will also register a bootable target in the UEFI firmware using this updated **`grubx64.efi`** application, instead of the **`shim.efi`** application.

> Warning
>
> Using **`grub2-install`** will install a custom **`grubx64.efi`**. If a system was set up for secure boot, this will cause the boot to fail.

`/boot/grub2/grub.cfg` vs. **`/boot/efi/EFI/redhat/grub.cfg`**

Included in the move from **`/boot/grub2`** to **`/boot/efi/EFI/redhat/`** is the move of the main **grub2** configuration file.

The symbolic link **`/etc/grub.cfg`** is also moved to **`/etc/grub2-efi.cfg`**.

Reinstalling grub2 on UEFI-based machines

If, for some reason, any of the files needed to boot in **`/boot/efi`** have been removed, or if the partition has been recreated, these files can be recovered by reinstalling both the *grub2-efi* and *shim* packages.

```
[root@demo ~]# yum reinstall grub2-efi shim
```

If **`/boot/efi/EFI/redhat/grub.cfg`** has been removed, it can be restored with the following command:

```
[root@demo ~]# grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg
```

If the **`Red Hat`** entry has been removed from the UEFI boot menu, booting from the disk that has the installation of Red Hat Enterprise Linux 7 will automatically add it back.

The UEFI boot chain

In order to work with Secure Boot systems, where signatures on boot loaders, kernels, etc., can be checked by the firmware to authenticate that only unmodified software is run, Red Hat Enterprise Linux 7 utilizes the **shim** UEFI boot loader.

The **`shim.efi`** application is signed with a key trusted by most UEFI firmware. When started, it will attempt to load **`grubx64.efi`** using the normal UEFI firmware calls. If the UEFI firmware refuses to load the application due to a bad signature, **shim** will attempt to verify the application using other keys compiled into **shim**, or a user-generated key stored in the UEFI NVRAM (*Machine Owner Key* (MOK)). When attempting to start a signed application for which no keys have been registered, the **MokManager.efi** application is started automatically so that an end user can register a personal key.

The **shim.efi** application also registers a new UEFI system call that can be used by **grub2** to verify kernel signatures.

If **`shim.efi`** is not registered with the UEFI firmware, booting from the disk that contains the ESP will launch the **`/boot/efi/EFI/BOOT/BOOTX64.efi`** application. This in turn will launch the **fallback.efi** application, which automatically registers the **shim.efi** application and boots the system, based on the settings in **/boot/efi/EFI/redhat/BOOT.CSV**.

Managing UEFI boot targets

In order to manage the logical boot order and available boot targets, Red Hat Enterprise Linux 7 ships with a tool called **efibootmgr** in the package *efibootmgr*. On UEFI-based systems, this package will be installed by default.

The **efibootmgr** tools allow administrators to manage the list of available UEFI boot targets, edit individual entries, and more.

Viewing current targets

To view the current targets, the tool can be invoked without any options:

```
[root@demo ~]# efibootmgr
BootCurrent: 0016
Timeout: 0 seconds
BootOrder: 0016,000A,000D,0009,000B,000C,000E,0008,0007,0014,0015,0013❶
Boot0000  Setup
Boot0001  Boot Menu
Boot0002  Diagnostic Splash Screen
Boot0003  Lenovo Diagnostics
...
Boot0013* PCI LAN
Boot0014  Other CD
Boot0015  Other HDD
Boot0016* Red Hat❷
```

❶ This list defines the ordering in which boot entries will be tried, with the leftmost entry being tried first, and the rightmost entry being tried last. The number refers to the various **Boot*XXXX*** entries that follow it.

❷ This is an individual boot entry; in this case, number **0x0016**. Boot entry numbers are always presented (and used) as hexadecimal numbers. The asterisks (*****) show that this entry is **active**, i.e., it can be selected and booted.

To view more information about boot entries, the **-v** option can be used.

```
[root@demo ~]# efibootmgr
BootCurrent: 0016
Timeout: 0 seconds
BootOrder: 0016,000A,000D,0009,000B,000C,000E,0008,0007,0014,0015,0013
Boot0000  Setup FvFile(721c8b66-426c-4e86-8e99-3457c46ab0b9)
...
Boot0016* Red Hat HD(1,GPT,c13391b4-a66f-4b9b-9deb-39da588d33a3,0x800,0x64000)/File(\EFI
\redhat\shim.efi)
```

The entry for **Boot0016** now shows which hard disk, partition, and file on that partition to use to boot as well.

Removing an entry

To remove an entry from the list completely, the **-B** option can be used. For example, to delete entry **1E**, the following command can be used:

```
[root@demo ~]# efibootmgr -b 1E -B
```

Selecting a temporary boot target

To override the normal boot ordering for a single boot, the **`-n`** option can be used. For example, to force the system to boot into entry **`2C`** the next time it is started, the following command can be used.

```
[root@demo ~]# efibootmgr -n 2c
```

Adding an entry

Entries can be added using the **`-c`** option. Unless otherwise specified, this will assume the ESP is **`/dev/sda1`**, the intended label is **`Linux`**, and the boot loader application is called **`elilo.efi`**.

The following example adds a new entry named **`Yippie`**, with **`/dev/sda2`** as the ESP, and with the application **`/EFI/yippie.efi`** on the ESP. Notice how normal forward slashes (**`/`**) have to be replaced by backslashes (**`\`**).

```
[root@demo ~]# efibootmgr -c -d /dev/sda -p 2 -L "Yippie" -l "\EFI\yippie.efi"
```

Consult the man page for **`efibootmgr`** for more information on selecting temporary boot targets, adding and removing entries, and updating entries.

References

The sections on “UEFI Secure Boot” in the Red Hat Enterprise Linux System Administration Guide at **`https://access.redhat.com/documentation`**

`efibootmgr`(1) man page

Quiz: Resolving Boot Loader Issues on UEFI Systems

Choose the correct answer to the following questions:

1. What is the main use of the **shim.efi** application?

   a. To verify that the UEFI firmware is unmodified.
   b. To verify the signature on **grubx64.efi** and load it.
   c. To load **fallback.efi** and register it with the UEFI firmware.
   d. To multiboot into another installed operating system.

2. The tool to manage UEFI boot entries is called:

   a. **`grub2-install`**
   b. **`fallback.efi`**
   c. **`efibootmgr`**
   d. **`grub2`**

3. On an UEFI-enabled system, the location for the main **grub2** configuration file is:

   a. **`/boot/efi/EFI/redhat/grub.cfg`**
   b. **`/boot/grub2/grub.cfg`**
   c. **`/boot/efi/grub.cfg`**
   d. **`/boot/efi/redhat/grub.cfg`**

Solution

Choose the correct answer to the following questions:

1. What is the main use of the **shim.efi** application?

   a. To verify that the UEFI firmware is unmodified.
   b. **To verify the signature on grubx64.efi and load it.**
   c. To load **fallback.efi** and register it with the UEFI firmware.
   d. To multiboot into another installed operating system.

2. The tool to manage UEFI boot entries is called:

   a. `grub2-install`
   b. `fallback.efi`
   c. **`efibootmgr`**
   d. `grub2`

3. On an UEFI-enabled system, the location for the main **grub2** configuration file is:

   a. **`/boot/efi/EFI/redhat/grub.cfg`**
   b. `/boot/grub2/grub.cfg`
   c. `/boot/efi/grub.cfg`
   d. `/boot/efi/redhat/grub.cfg`

Dealing with Failing Services

Objectives

After completing this section, students should be able to identify and resolve service failures affecting the boot process.

Service dependencies

When a Red Hat Enterprise Linux 7 system starts up, a number of services are also automatically started based on the selected boot target. Each of these services can have dependencies on other services.

These dependencies are listed in the units configuration file on disk, a drop-in configuration file (typically a file in **/etc/systemd/system/<*UNITNAME*>.d/**), or using a **requires** or **wants** subdirectory such as **/etc/systemd/system/<*UNITNAME*>.requires** or **/etc/systemd/system/<*UNITNAME*>.wants**.

Dependency types

The following list details the different types of dependencies a unit can have. For more detailed information, consult the **systemd.unit**(5) man page.

Requires=

Starting this unit will automatically add the required units into the transaction. Stopping the required unit will stop this unit as well. Unless **After=** or **Before=** are also used, the units will be started at the same time.

This type of dependency can also be configured by creating a **/etc/systemd/system/<*UNITNAME*>.requires** directory, and adding symbolic links to the required units in that directory.

RequiresOverridable=

Like **Requires**, but failed requirements will not cause the unit to fail when explicitly started by an administrator. When started as a dependency, for example during boot, failed requirements will still cause this unit to fail.

Requisite= , **RequisiteOverridable=**

Like **Requires** and **RequiresOverridable**, but stronger. Where a **Requires** will start a required unit if it is not already running, a **Requisite** will fail.

Wants=

A weaker version of **Requires**. When a wanted unit fails to start, this unit itself will still start.

Wants can also be configured by creating symbolic links to the wanted units in **/etc/systemd/system/<*UNITNAME*>.wants/**.

Conflicts=

A negative dependency. Starting this unit will stop the conflicting units (if possible), and vice versa.

Before= , **After=**

These two entries configure ordering. When they are not used, required units are started at the same time. If unit **a.service** has a line **Before=b.service**, the execution of **b.service** will be delayed until **a.service** has finished starting.

After= indicates that the listed units have to have finished starting before this unit can be started.

Inspecting dependencies

There are three main ways of viewing a unit's dependencies: manually, automated, and graphically. To manually view the dependencies for a unit, the command **systemctl show** ***UNIT*** can be used. An administrator would then look for all the relevant configuration lines, and repeat this process for all listed dependencies.

A more automated way is using the **systemctl list-dependencies** ***UNIT*** command. This will show a tree of all listed dependencies, and their dependencies, etc. On a terminal capable of displaying colored output, each line will also show either a green or a red dot, indicating if that unit is currently active or not.

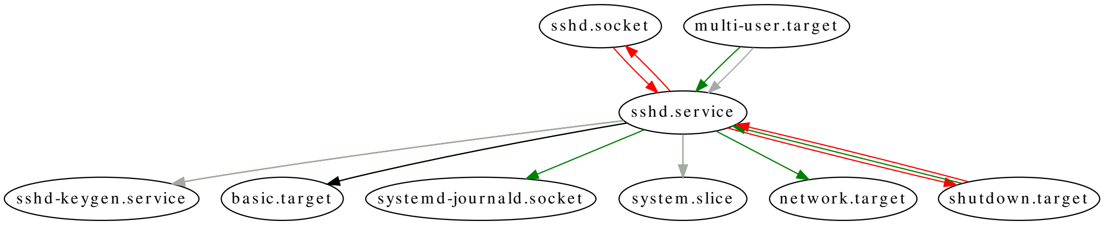
```
[root@demo ~]# systemctl list-dependencies nfs-server.service
nfs-server.service
● ├─auth-rpcgss-module.service
● ├─nfs-config.service
● ├─nfs-idmapd.service
● ├─nfs-mountd.service
● ├─proc-fs-nfsd.mount
● ├─rpc-statd-notify.service
● ├─rpc-statd.service
● ├─system.slice
● ├─network.target
● └─rpcbind.target
```

Note

A fair amount of units have a dependency on a larger target like **basic.target**. Running **systemctl list-dependencies** on those units will also display all dependencies for that target in the output.

In some cases, especially when dealing with larger dependency chains, a graphical representation of those dependencies might be required. For those cases, the **systemd-analyze dot** command should be used, in combination with the **dot** command from the *graphviz* package. For example, to create a visual representation of the dependencies on the **sshd.service** unit:

```
[root@demo ~]# yum -y install graphviz
[root@demo ~]# systemd-analyze dot sshd.service | dot -Tsvg > sshd-dependencies.svg
   Color legend: black     = Requires
                 dark blue = Requisites
                 dark grey = Wants
                 red       = Conflicts
                 green     = After
```



Dependency issues

It is possible for an administrator to configure dependencies in such a way that the system can never start while adhering to those dependencies; for example, two units that have a **Requires** and a **Before** both pointing to each other. In order to keep systems bootable, **systemd** will detect these kind of issues and try to resolve them. The method that is used to resolve these issues is fairly straightforward: One of the offending units is removed from the transaction at a time until a startable configuration is reached. This can mean that a number of services that an administrator thought were being started are not being started. Only a short message is output to the console at boot time for these issues, so a proper reading of the system logs can be required.

After spotting, and verifying, these types of issues, an administrator will need to modify the dependencies listed in the affected units to remedy the situation. In many cases, changing a **Requires** to a **Wants**, or fixing an incorrect use of **Before** or **After**, can remedy the situation.

Important

When changing configuration files on disk, do not forget to execute **systemctl daemon-reload** to inform **systemd** of the changes.

Early root shell

In some cases, startup tasks can take far longer than anticipated, or even fail to complete after minutes or hours. To help debug these types of problems, the **systemctl list-jobs** command can show all current jobs that **systemd** is executing, allowing an administrator to stop or kill them, or fix the reason these jobs are taking so long.

To obtain a root shell *during* startup, an administrator can enable the **debug-shell.service** service. This service will start a virtual console with a logged-in root shell attached to it on **/dev/tty9**, very early during boot, during the startup of the **sysinit.target** target.

Using this early root shell, an administrator can analyze, debug, and sometimes remedy a failing service or unit.

Warning

Do *NOT* leave the **debug-shell.service** service enabled and running after debugging is done. Anyone with console access, either physical or via a management card or KVM switch, will have full, unrestricted root access to the machine.

References

> Overview of systemd for RHEL7
> https://access.redhat.com/articles/754933

systemd-analyze(1) and **systemd.unit**(5) man pages

Guided Exercise: Dealing with Failing Services

In this lab, you will resolve an issue where two services fail to start at boot.

Resources	
Machines	• **workstation** • **servera**

Outcome(s)

You should be able to resolve an issue where two services fail to start at boot.

Before you begin

Set up your systems for this exercise by running the command **lab bootservicefail setup** on your **workstation** system.

```
[student@workstation ~#$ lab bootservicefail setup
```

As part of a larger project, your **servera** machine is set up to serve both web and FTP content, using the **httpd** and **vsftpd** daemons respectively. The specifications for this project demanded that both daemons would always be running at the same time, on the same machine.

One of your former colleagues has attempted to satisfy this requirement, but now both services refuse to start at boot time. You have been called in to remedy this issue. For now, you do not need to ensure that the two services will always be running simultaneously.

1. Attempt to recreate the issue by checking the status of both **httpd** and **vsftpd** on your **servera** machine.

   1.1. Check the status of the **httpd** service. Since some status lines may contain useful information past the 80-character limit, use the **-l** option to show full status lines.

```
[root@servera ~]# systemctl status -l httpd
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; vendor
 preset: disabled)
  Drop-In: /etc/systemd/system/httpd.service.d
           └─10-dependencies.conf
   Active: inactive (dead)
     Docs: man:httpd(8)
           man:apachectl(8)

Dec 14 10:40:00 servera.lab.example.com systemd[1]: Found ordering cycle on
 httpd.service/start
Dec 14 10:40:00 servera.lab.example.com systemd[1]: Found dependency on
 vsftpd.service/start
Dec 14 10:40:00 servera.lab.example.com systemd[1]: Found dependency on
 httpd.service/start
Dec 14 10:40:00 servera.lab.example.com systemd[1]: Breaking ordering cycle by
 deleting job vsftpd.service/start
```

   1.2. Check the status of the **vsftpd** service. Since some status lines may contain useful information past the 80-character limit, use the **-l** option to show full status lines.

```
[root@servera ~]# systemctl status -l vsftpd
● vsftpd.service - Vsftpd ftp daemon
   Loaded: loaded (/usr/lib/systemd/system/vsftpd.service; enabled; vendor
 preset: disabled)
  Drop-In: /etc/systemd/system/vsftpd.service.d
           └─10-dependencies.conf
   Active: inactive (dead)
....
Dec 14 10:40:00 servera.lab.example.com systemd[1]: Job vsftpd.service/start
 deleted to break ordering cycle starting with httpd.service/start
```

2. It looks like both services have a hard start dependency on each other. Looking at the output of the previous step, which files might be responsible for this dependency? Investigate the contents of these files.

   2.1. From the output of the **systemctl status** commands, you can see two nonstandard drop-in files have been added to the **systemd** service specifications:

   - **/etc/systemd/system/httpd.service.d/10-dependencies.conf**
   - **/etc/systemd/system/vsftpd.service.d/10-dependencies.conf**

   2.2. Inspect the contents of these two files.

```
[root@servera ~]# cat /etc/systemd/system/httpd.service.d/10-dependencies.conf
[Unit]
After=vsftpd.service
Requires=vsftpd.service
[root@servera ~]# cat /etc/systemd/system/vsftpd.service.d/10-dependencies.conf
[Unit]
After=httpd.service
Requires=httpd.service
```

3. Form a hypothesis as to why these two services fail to start.

   3.1. Both services have a **Requires** for the other service, but this only ensures that if one of them is started, the other is started as well.

   3.2. Both services have an **After** requirement on the other service. This will cause a cyclic dependency, which **systemd** will break by simply not starting these services.

4. Describe two possible solutions.

   4.1. Removing both drop-in directories/files entirely, then reloading **systemd**, and finally starting the services. This approach will remove all dependencies.

   4.2. Changing one of the two **After** lines to **Before** will ensure the services are always started together, without causing a cyclic dependency.

5. Attempt to fix this issue by changing the **After** line for the **httpd.service** service to **Before**. Reload **systemd** and start both services to verify.

   5.1. Change the **After** line for the **httpd.service** service to **Before**. Open **/etc/systemd/system/httpd.service.d/10-dependencies.conf** in a text editor, and edit it so that it reads like this:

```
[Unit]
Before=vsftpd.service
Requires=vsftpd.service
```

5.2. Instruct **systemd** to reload all configuration files from disk.

```
[root@servera ~]# systemctl daemon-reload
```

5.3. Start both the **`httpd.service`** and **`vsftpd.service`** services.

```
[root@servera ~]# systemctl start httpd.service vsftpd.service
```

5.4. Reboot your **`servera`** system to verify that both services will start during boot as well.

6. Once you are done, verify your work by running the command **`lab bootservicefail grade`** from your **`workstation`** machine.

```
[student@workstation ~]$ lab bootservicefail grade
```

7. After grading your systems successfully, reset your machines to the state they had before starting the lab, either by resetting your virtual machines or by running the following command on your **`workstation`** system.

```
[student@workstation ~]$ lab bootservicefail reset
```

Recovering a root Password

Objectives

After completing this section, students should be able to regain root control of a system.

Recovering a lost root password

When the **root** password for a system has been lost or forgotten, there are various ways in which an administrator can reset it. Some of these methods can be used remotely--for example, over an SSH connection--while others require access to a physical console.

If the **root** account is still logged into an unlocked terminal, that session can be used to reset the password; similarly, if an account for which the password is known has **sudo** access to a shell, or to the **passwd** command, that access can be used to reset the root password.

Some slightly more complex methods include manually editing the **/etc/shadow** file with a known password hash from an account that has **sudo** access to an editor, or editing virtual machine disk images with the **guestfish** command.

If none of the previous methods are available, other methods still are. The **anaconda** rescue mode will allow somebody with physical access to a machine to reset the **root** password (as long as the disk is not encrypted or the encryption password is known, and the BIOS/UEFI firmware allows booting from another device, or the BIOS password is known).

Resetting a root password without external media

A common method to reset a lost **root** password without using external media involves “breaking into” the initial ramdisk (initramfs) startup sequence. This method only requires access to the physical console (or access to a remote management card or KVM switch), and the knowledge of any disk encryption passwords and boot loader passwords being used.

The procedure for resetting a lost **root** password using this method entails the following steps:

1. Reboot the system and interrupt the boot loader countdown timer by pressing any key except **Enter**.

2. Find the entry that is normally booted, and change it so that it will halt execution during the initial ramdisk, and present the user with a shell.

    2.1. Use the cursor keys to highlight the entry that would normally be booted, and press **e**.

    2.2. Use the cursor keys to move to the line that has the kernel and kernel arguments. This line normally starts with **linux16** or **linuxefi**.

    2.3. Move the cursor to the end of the line by pressing **End**, and add **rd.break**.

> ### Note
>
> The virtual machine images used in the classroom have a **`console=`** setting at the end of the line that configures a serial console. If you are not using a serial console, remove this setting to have the initial ramdisk use the regular virtual console.

2.4. Press **`Ctrl`**+**`X`** to boot using the modified stanza.

3. The system will now boot, but halt the process early on during the execution of the initial ramdisk. If the system seems to be doing nothing, press **`Enter`**; a prompt might be obscured by kernel output.

4. Mount the root file system read-write. This file system is mounted under **`/sysroot`**.

```
switch_root:/# mount -oremount,rw /sysroot
```

5. Change the working root to the **`/sysroot`** directory.

```
switch_root:/# chroot /sysroot
```

6. Reset the **`root`** password to something known.

```
sh-4.2# echo redhat | passwd --stdin root
```

7. Fix any SELinux issues. An easy, but long, solution is to **`touch /.autorelabel`**. This will force the system to relabel all mounted file systems during boot, then reboot. A quicker, but more involved method, is outlined in the following substeps.

   7.1. Force the system to load the currently configured SELinux policy.

   ```
   sh-4.2# load_policy -i
   ```

   7.2. Restore all SELinux contexts under **`/etc/`**.

   ```
   sh-4.2# restorecon -Rv /etc
   ```

8. Reboot the system by exiting from both the **`chroot`** shell, and from the **`rd.break`** shell. This can be done either by typing **`exit`**, or by pressing **`Ctrl`**+**`D`**.

Note

The ordering of the previous steps might seem slightly “off” to some administrators. Why isn't the SELinux policy loaded before the **`passwd`** command is executed? Wouldn't this save the hassle of running the **`restorecon`** command?

The answer lies in the fact that the SELinux policy, once loaded, disallows the changing of passwords from the **`rd.break`** shell, even after running a **`setenforce 0`** command. Loading the policy afterward allows an administrator to change the password, and fix the SELinux contexts without having to go through a complete relabel.

References

> Resetting the Root Password of RHEL-7 / systemd
> https://access.redhat.com/solutions/918283

Guided Exercise: Recovering a root Password

In this lab, you will reset a lost root password.

Resources	
Machines	• **`workstation`** • **`servera`**

Outcome(s)

You should be able to reset a lost root password without having another form of root access to a machine.

Before you begin

Prepare your systems for this exercise by running the command **`lab rootpw setup`** from your **`workstation`** machine.

```
[student@workstation ~#$ lab rootpw setup
```

One of your esteemed colleagues has recently left your company. One of their final projects included setting up your **`servera`** machine. While your former colleague did an admirable job in configuring the server, the documentation for **`servera`** is lacking.

One of the pieces of information missing from the documentation is the password for the **`root`** account on **`servera`**. You have been asked to reset this password to **`redhat`**.

Important

For this exercise, assume that you do not have passwordless **ssh** access to the **`root`** account on **`servera`**, and that the **`student`** account on **`servera`** does not have full **sudo** access.

1. Open a console to **`servera`**, and reboot it. Break into the **grub** menu, and force the system to pause bootup during the execution of the initial ramdisk.

   1.1. Reboot your **`servera`** machine from the console.

   ```
   [root@servera ~]# reboot
   ```

   1.2. Press any key to pause the **grub** menu countdown timer when it appears.

   1.3. Highlight the default entry, and press **`e`** to edit it.

   1.4. Scroll down to the line that starts with **`linux16`**, press **End** to jump to the end of the line, remove the last **`console=`** setting, and append **`rd.break`**.

   1.5. Press **`Ctrl`**+**`X`** to boot with these modified settings.

2. Change the root password on your system to **`redhat`**, taking care not to disturb any SELinux contexts.

2.1. Remount the file system on **/sysroot** read-write.

```
switch_root:/# mount -o remount,rw /sysroot
```

2.2. Switch root to the **/sysroot** directory.

```
switch_root:/# chroot /sysroot
```

2.3. Set the **root** password to **redhat**.

```
sh-4.2# echo redhat | passwd --stdin root
```

2.4. Load the default SELinux policy. If this fails, ensure that the system will perform a full relabel on the next reboot.

```
sh-4.2# load_policy -i
```

If the previous command failed:

```
sh-4.2# touch /.autorelabel
```

2.5. Recursively restore the SELinux contexts on **/etc**.

```
sh-4.2# restorecon -Rv /etc
```

Note

Loading the SELinux policy before changing the root password will cause a denial when trying to update the root password. Setting the password, then loading the policy and fixing the contexts, does work. If you forget to update the contexts, no local users can log in after a reboot, since there will be an invalid security context on **/etc/shadow**.

3. Reboot your **servera** system, and verify that you can log in as **root** using the password **redhat**.

3.1. Reboot your **servera** system by pressing **Ctrl**+**D** twice.

```
sh-4.2# Ctrl+D
```

```
switch_root:/# Ctrl+D
```

If you had to touch **/.autorelabel** in the previous step, your system will now reboot twice; otherwise, a single reboot will occur.

3.2. Attempt to log into the console as **root** using the password **redhat**.

4. From **workstation**, grade your work by running the command **lab rootpw grade**.

```
[student@workstation ~]$ lab rootpw grade
```

Lab: Troubleshooting Boot Issues

In this lab, you will fix an issue where a system fails to boot past the boot loader screen.

Resources	
Machines	• **workstation** • **servera**

Outcome(s)

You should be able to repair issues where a system fails to boot past the boot loader.

Before you begin

Prepare your systems for this exercise by running the command **lab bootissues setup** on your **workstation** system. This will modify the ability of your **servera** system to boot, then reboot it.

```
[student@workstation ~]$ lab bootissues setup
```

Someone decided to “streamline” the boot process of your **servera** machine. While doing this, they accidentally broke the ability of said system to actually boot its default target.

You have been assigned the task of making **servera** bootable again.

1. Open a console to your **servera** machine and assess the problem.

2. Temporarily boot your **servera** system into a working configuration.

3. Permanently fix the issue.

4. Grade your work by running the command **lab bootissues grade** from your **workstation** machine.

   ```
   [student@workstation ~]$ lab bootissues grade
   ```

5. Reset all of your machines to provide a clean environment for the next exercises.

Solution

In this lab, you will fix an issue where a system fails to boot past the boot loader screen.

Resources	
Machines	• **workstation** • **servera**

Outcome(s)

You should be able to repair issues where a system fails to boot past the boot loader.

Before you begin

Prepare your systems for this exercise by running the command **lab bootissues setup** on your **workstation** system. This will modify the ability of your **servera** system to boot, then reboot it.

```
[student@workstation ~]$ lab bootissues setup
```

Someone decided to "streamline" the boot process of your **servera** machine. While doing this, they accidentally broke the ability of said system to actually boot its default target.

You have been assigned the task of making **servera** bootable again.

1. Open a console to your **servera** machine and assess the problem.

    1.1. What is displayed on the console of **servera**? If a **grub** menu is displayed, attempt to boot the default (top) entry.

    ```
    error: you need to load the kernel first
    ```

    1.2. What can this indicate?

    This typically indicates that there's no kernel defined in the menu entry, or the ramdisk is being loaded before the kernel is.

2. Temporarily boot your **servera** system into a working configuration.

    2.1. There are two options here. Either see if a second entry will boot, in this case the **rescue** entry, or manually edit the configuration in memory to temporarily solve the issue.

    2.2. Use the **grub** menu to boot the second (rescue) entry.

3. Permanently fix the issue.

    3.1. Recreate a working **/boot/grub2/grub.cfg** using the list of installed kernels.

    ```
    [root@servera ~]# grub2-mkconfig -o /boot/grub2/grub.cfg
    ```

    3.2. Verify your work by rebooting your **servera** machine.

```
[root@servera ~]# reboot
```

4. Grade your work by running the command **lab bootissues grade** from your **workstation** machine.

   ```
   [student@workstation ~]$ lab bootissues grade
   ```

5. Reset all of your machines to provide a clean environment for the next exercises.

Summary

In this chapter, you learned:

- How to recreate a **grub2** configuration file.
- How to reinstall **grub2** into an MBR using the command **`grub2-install`**.
- How to reconfigure an UEFI system to use the **`shim`** EFI boot loader by removing any existing EFI boot entries using **`efibootmgr`**, and rebooting from the system disk.
- How to inspect and configure **`systemd`** unit dependencies using **`systemctl list-dependencies`**.
- How to obtain an early root shell by enabling the **`debug-shell.service`** service, allowing for early diagnostics using **`systemctl list-jobs`**.
- How to recover a lost **`root`** password using various methods, including using **`rd.break`** on the kernel command line.

CHAPTER 4

IDENTIFYING HARDWARE ISSUES

Overview	
Goal	Identify hardware problems that can affect a system's ability to operate normally.
Objectives	• Identify hardware and hardware problems. • Manage kernel modules and their parameters. • Identify and resolve virtualization issues.
Sections	• Identifying Hardware Issues (and Guided Exercise) • Managing Kernel Modules (and Guided Exercise) • Handling Virtualization Issues (and Guided Exercise)
Lab	• Identifying Hardware Issues

Identifying Hardware Issues

Objectives

After completing this section, students should be able to identify hardware and hardware problems.

Identifying hardware

An important step in troubleshooting potential hardware issues is knowing exactly which hardware is present in a system. For virtual systems, this might seem less useful than for a physical system, but it can tell an administrator if the correct virtual devices have been added.

Identifying CPUs

The CPU(s) in a running system can be identified with the **lscpu** command from the *util-linux* package.

```
[root@demo ~]# lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                8
On-line CPU(s) list:   0-7
Thread(s) per core:    2
Core(s) per socket:    4
Socket(s):             1
NUMA node(s):          1
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 60
Model name:            Intel(R) Core(TM) i7-4810MQ CPU @ 2.80GHz
Stepping:              3
CPU MHz:               3004.750
CPU max MHz:           3800.0000
CPU min MHz:           800.0000
BogoMIPS:              5587.29
Virtualization:        VT-x
L1d cache:             32K
L1i cache:             32K
L2 cache:              256K
L3 cache:              6144K
NUMA node0 CPU(s):     0-7
```

Another useful piece of information is what *flags* a CPU supports. These flags indicate whether a CPU supports certain extended technologies, such as AES acceleration, hardware-assisted virtualization, and many more. These flags can be inspected in **/proc/cpuinfo**.

Note

The fact that a CPU supports a certain flag does not always mean that the feature is available. For example, the **vmx** flag on a Intel CPU indicates that the processor is capable of supporting hardware virtualization, but the feature itself might be disabled in the system firmware.

Identifying memory

The **dmidecode** tool can be used to retrieve information about physical memory banks, including the type, speed, and location of the bank. To retrieve this information, use the command **dmidecode -t memory**.

```
[root@demo ~]# dmidecode -t memory
# dmidecode 2.12

...

Handle 0x0007, DMI type 17, 34 bytes
Memory Device
 Array Handle: 0x0005
 Error Information Handle: Not Provided
 Total Width: 64 bits
 Data Width: 64 bits
 Size: 8192 MB
 Form Factor: SODIMM
 Set: None
 Locator: ChannelA-DIMM1
 Bank Locator: BANK 1
 Type: DDR3
 Type Detail: Synchronous
 Speed: 1600 MHz
 Manufacturer: Hynix/Hyundai
 Serial Number: 0E80EABA
 Asset Tag: None
 Part Number: HMT41GS6BFR8A-PB
 Rank: Unknown
 Configured Clock Speed: 1600 MHz
...
```

Identifying disks

To identify physical disks, an administrator can use the command **lsscsi** from the *lsscsi* package. This tool can list all physical SCSI (and USB, SATA, and SAS) drives attached to a system.

```
[root@demo ~]# lsscsi -v
[0:0:0:0]    disk    ATA      SAMSUNG MZ7LN512 4L0Q  /dev/sda
  dir: /sys/bus/scsi/devices/0:0:0:0  [/sys/devices/pci0000:00/0000:00:1f.2/ata1/host0/
target0:0:0/0:0:0:0]
[5:0:0:0]    cd/dvd  HL-DT-ST DVDRAM GUB0N     LV20  /dev/sr0
  dir: /sys/bus/scsi/devices/5:0:0:0  [/sys/devices/pci0000:00/0000:00:1f.2/ata6/host5/
target5:0:0/5:0:0:0]
```

For more information, the **hdparm** command from the *hdparm* package can be used on individual disks.

```
[root@demo ~]# hdparm -I /dev/sda
/dev/sda:

ATA device, with non-removable media
 Model Number:       SAMSUNG MZ7LN512HCHP-000L1
 Serial Number:      S1ZKNXAG806853
 Firmware Revision:  EMT04L0Q
 Transport:          Serial, ATA8-AST, SATA 1.0a, SATA II Extensions, SATA Rev 2.5, SATA
 Rev 2.6, SATA Rev 3.0
...
```

Identifying PCI hardware

Attached PCI hardware can be identified with the **lspci** command. Adding one or more **-v** options will increase the verbosity.

```
[root@demo ~]# lspci
00:00.0 Host bridge: Intel Corporation Xeon E3-1200 v3/4th Gen Core Processor DRAM
 Controller (rev 06)
00:01.0 PCI bridge: Intel Corporation Xeon E3-1200 v3/4th Gen Core Processor PCI Express
 x16 Controller (rev 06)
00:02.0 VGA compatible controller: Intel Corporation 4th Gen Core Processor Integrated
 Graphics Controller (rev 06)
....
```

Identifying USB hardware

USB hardware can be identified using the **lsusb** command. Just like with **lspci**, verbosity can be increased by adding **-v** options.

```
[root@demo ~]# lsusb
Bus 003 Device 002: ID 8087:8000 Intel Corp.
Bus 003 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 003: ID 1058:083a Western Digital Technologies, Inc.
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 001 Device 009: ID 05e3:0608 Genesys Logic, Inc. Hub
Bus 001 Device 008: ID 17a0:0001 Samson Technologies Corp. C01U condenser microphone
Bus 001 Device 006: ID 04f2:b39a Chicony Electronics Co., Ltd
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
....
```

Hardware error reporting

Modern systems can typically keep a watch on various hardware failures, alerting an administrator when a hardware fault occurs. While some of these solutions are vendor-specific, and require a remote management card, others can be read from the OS in a standardized fashion.

Red Hat Enterprise Linux 7 provides two mechanisms for logging hardware faults, **mcelog** and **rasdaemon**.

mcelog

mcelog provides a framework for catching, and logging *machine check exceptions* on x86 systems. On supported systems, it can also automatically mark bad areas of RAM so that they will not be used.

To install and enable **mcelog**, follow the following procedure:

1. Install the *mcelog* package.

   ```
   [root@demo ~]# yum install mcelog
   ```

2. Start and enable the **mcelog.service** service.

   ```
   [root@demo ~]# systemctl enable mcelog
   [root@demo ~]# systemctl start mcelog
   ```

From now on, hardware errors caught by the **mcelog** daemon will show up in the system journal. Messages can be queried using the command **journalctl -u mcelog.service**. If the **abrt** daemon is installed and active, it will also trigger on various **mcelog** messages.

Alternatively, for administrators who do not wish to run a separate service, a **cron** is set up, but commented out, in **/etc/cron.hourly/mcelog.cron** that will dump events into **/var/log/mcelog**.

rasdaemon

rasdaemon is a more modern replacement for **mcelog** that hooks into the kernel trace subsystem. RAS stands for Reliability, Availability, and Serviceability. **rasdaemon** hooks into the *Error Detection And Correction* (EDAC) mechanism for DIMM modules and reports them to user space, and also the RAS messages that come from the kernel tracing subsystem.

To enable **rasdaemon**, use the following steps:

1. Install the *rasdaemon* package.

```
[root@demo ~]# yum install rasdaemon
```

2. Start and enable the **rasdaemon.service** service.

```
[root@demo ~]# systemctl enable rasdaemon
[root@demo ~]# systemctl start rasdaemon
```

Information about the various memory banks can be queried using the **ras-mc-ctl** tool. Of special interest are **ras-mc-ctl --status** to show the current status, and **ras-mc-ctl --errors** to view any logged errors.

> **Note**
>
> If no errors or events have been logged, an error message about a missing database might be presented when running the previous commands.

Memory testing

When a physical memory error is suspected, an administrator might want to run an exhaustive memory test. In these cases, the *memtest86+* package can be installed.

Since memory testing on a live system is less than ideal, the *memtest86+* package will install a separate boot entry that runs **memtest86+** instead of a regular Linux kernel. The following steps outline how to enable this boot entry.

1. Install the *memtest86+* package; this will install the **memtest86+** application into **/boot**.

```
[root@demo ~]# yum install memtest86+
```

2. Run the command **memtest-setup**. This will add a new template into **/etc/grub.d/** to enable **memtest86+**.

```
[root@demo ~]# memtest-setup
```

3. Update the **grub2** boot loader configuration.

```
[root@demo ~]# grub2-mkconfig -o /boot/grub2/grub.cfg
```

References

> RHEL7: Which of mcelog and rasdaemon should I use for monitoring of hardware, for which usecases?
> https://access.redhat.com/solutions/1412953

The section on checking for hardware errors in the RHEL7 System Administrator's Guide on **`https://access.redhat.com/docs`**.

Guided Exercise: Identifying Hardware Issues

In this lab, you will identify a possible issue with the memory on a machine.

Resources	
Machines	• **servera**

Outcome(s)

You should be able to identify hardware issues using journal entries.

Before you begin

On your **workstation** system, run the command **lab hardwareissues setup** to prepare your **servera** system for this exercise.

```
[student@workstation ~]$ lab hardwareissues setup
```

One of your junior admins has provided you with a text dump of a **systemd-journald** journal file from a machine that has been experiencing strange, random errors. He has asked you to inspect this journal for any possible signs of hardware failure, since the machine kept experiencing these issues even after a complete wipe and reinstall.

This file is available on your **servera** machine as **/root/logdump**.

1. Inspect the log file provided to you on **servera**, and try to find any hardware errors.

   1.1. Inspect the file, verifying that it is readable and a log file.

   ```
   [root@servera ~]# less /root/logdump
   ```

   1.2. Narrow down your view to any events that are generated from the **mcelog.service** service. Does this provide relevant information?

   ```
   [root@servera ~]# grep mcelog /root/logdump | less
   ```

   The following lines, and all their repeats, seem to indicate a memory issue:

   ```
   Dec 03 12:22:41 ouch.example.com mcelog[1037]: MCA: MEMORY CONTROLLER
    RD_CHANNEL1_ERR
   Dec 03 12:22:41 ouch.example.com mcelog[1037]: Transaction: Memory read error
   Dec 03 12:22:41 ouch.example.com mcelog[1037]: STATUS 8c00004000010091 MCGSTATUS
    0
   Dec 03 12:22:41 ouch.example.com mcelog[1037]: MCGCAP 1000c1d APICID 20 SOCKETID
    1
   Dec 03 12:22:41 ouch.example.com mcelog[1037]: CPUID Vendor Intel Family 6 Model
    62
   Dec 03 12:22:41 ouch.example.com mcelog[1037]: Hardware event. This is not a
    software error.
   ```

2. Since the machine seems to have a memory issue, you have decided to run an exhaustive memory check on it. Pretend that **servera** is the machine in question, and prepare it to run **memtest86+**.

Important

Running **memtest86+** on a virtual machine is not recommended outside of the classroom. Under normal circumstances, **memtest86+** should be run on the hypervisor machine.

2.1. Install the *memtest86+* package on your **servera** machine.

```
[root@servera ~]# yum install memtest86+
```

2.2. Run the **memtest-setup** command to add a new template to **/etc/grub.d/**

```
[root@servera ~]# memtest-setup
```

2.3. Generate a fresh **grub2** configuration that includes **memtest86+**.

```
[root@servera ~]# grub2-mkconfig -o /boot/grub2/grub.cfg
```

3. Reboot **servera** and select **memtest86+** from the **grub2** menu. Let it run for a few seconds, explore some of the options, and then reboot back into Red Hat Enterprise Linux 7.

4. Verify your work by running the command **lab hardwareissues grade** on your **workstation** system.

```
[student@workstation ~]$ lab hardwareissues grade
```

Managing Kernel Modules

Objectives

After completing this section, students should be able to manage kernel modules and their parameters.

Managing kernel modules

The Linux kernel has large parts of its functionality split out into modules, small pieces of code that can be loaded and unloaded at will. This helps keeps the base kernel image smaller, so that only code that is actually needed is loaded in memory.

Most device drivers are split out into kernel modules, but also file system drivers, encryption algorithms, and other parts of the kernel. On disk, these modules can be found in the directory **/lib/modules/*<KERNEL_VERSION>*/**. All loadable modules have a file name that ends in **.ko**.

Apart from their regular names, device driver modules also have aliases built in, such as **usb:v0421p0434d0100dc*dsc*dp*ic*isc*ip*in*** or **pci:v00008086d0000162Bsv*sd*bc03sc*i***. These aliases allow the kernel to automatically load the needed hardware drivers based on their USB or PCI vendor and device IDs.

Viewing loaded modules

The **lsmod** command can be used to view a list of currently loaded modules. The output contains four columns: the name of the loaded module, its size in bytes, the number of times this module is used by another module, and (optionally) the list of other modules using this module.

Loading and unloading modules

While at a low level, the commands **insmod** and **rmmod** are used to load and unload modules respectively, they should *never* be used by an administrator directly. Instead, administrators should use the **modprobe** command and the **modprobe -r** command. This wrapper tool handles any module dependencies that exist, and will use the module configuration files in **/etc/modprobe.d/** for any options or aliases.

By adding the **-v** option to a **modprobe** command, the actual **insmod** and **rmmod** commands being used will be shown.

Note

The exit codes for the **modprobe** command do not reflect whether an action succeeded, but whether the desired end result is achieved. For example, when loading a module that is already loaded, it will return **0**, even though no action has been performed.

Module options

A lot of kernel modules can take options, altering how they behave. These parameters are exposed in two ways, using the **modinfo -p** command, and in the **/sys/module** virtual file system.

Using **modinfo -p *<MODULE>***, an administrator can view a list of supported options for a module. For example:

```
[root@demo ~]# modinfo -p st
buffer_kbs:Default driver buffer size for fixed block mode (KB; 32) (int)
max_sg_segs:Maximum number of scatter/gather segments to use (256) (int)
try_direct_io:Try direct I/O between user buffer and tape drive (1) (int)
debug_flag:Enable DEBUG, same as setting debugging=1 (int)
try_rdio:Try direct read i/o when possible (int)
try_wdio:Try direct write i/o when possible (int)
```

This lists the option name, the option description, and the value type (integer, Boolean, string, etc.). A more detailed description of these options and their possible values is often available in dedicated documentation files in the *kernel-doc* package.

Another method of working with module options is through the **/sys/module/** directory. Most modules with options will expose these in a directory called **/sys/module/*<MODULENAME>*/parameters**, where individual options become separate files. If an option is modifiable at runtime, the corresponding file will be writable for **root**. If an option is only configurable at module load time, the corresponding file will be read-only.

Managing module options

To set a module option when the module is loaded with **modprobe**, it is enough to simply add **name=value** to the **modprobe** command. For example, to set the option **buffer_kbs** for the **st** module to 64, the following command can be used:

```
[root@demo ~]# modprobe -v st buffer_kbs=64
insmod /lib/modules/3.10.0-327.el7.x86_64/kernel/drivers/scsi/st.ko buffer_kbs=64
```

To automatically set this option every single time that this module is loaded, even when it is loaded automatically, a configuration file in **/etc/modprobe.d/** can be used. This file will have to have a file name ending in **.conf**, and files are parsed in alphanumeric order. Options are added with an **options** line like this:

```
options st buffer_kbs=64 max_sg_segs=512
```

It is recommended to group options into files corresponding to their module, e.g., **/etc/modprobe.d/st.conf**, or in files related to a function, e.g., **/etc/modprobe.d/throughput.conf**.

References

The chapter on working with kernel modules in the Red Hat Enterprise Linux 7 System Administrator's Guide on **https://access.redhat.com/docs**.

lsmod(8) and **modprobe**(8) man pages.

Guided Exercise: Managing Kernel Modules

In this lab, you will configure parameters for a kernel module.

Resources	
Machines	• **servera** • **workstation**

Outcome(s)
You should be able to configure module options for loadable kernel modules.

Your **servera** system has a SAS RAID array attached that uses the **megaraid_sas.ko** kernel module. Unfortunately, performance has been unpredictable, and sometimes error messages regarding MSI-X interrupts have shown up in your logs.

After opening up a support case, you have been asked to disable MSI-X interrupt handling in the driver to see if this fixes the issue.

1. View the list of possible kernel module options for the **megaraid_sas.ko** kernel module, find the option corresponding to MSI-X interrupt handling, load the module, and verify that the stated default is indeed applied. Unload the module afterwards.

   1.1. View the options for the **megaraid_sas.ko** kernel module:

   ```
   [root@servera ~]# modinfo -p megaraid_sas
   ```

   1.2. Find the option corresponding to MSI-X interrupt handling.

   ```
   [root@servera ~]# modinfo -p megaraid_sas | grep -i msi
   msix_disable:Disable MSI-X interrupt handling. Default: 0 (int)
   msix_vectors:MSI-X max vector count. Default: Set by FW (int)
   ```

   Of these two, **msix_disable** seems to be the option needed.

   1.3. Load the **megaraid_sas.ko** kernel module manually.

   ```
   [root@servera ~]# modprobe -v megaraid_sas
   ```

   1.4. Verify that the listed default (**0**) for the **msix_disable** option is indeed the active value.

   ```
   [root@servera ~]# cat /sys/module/megaraid_sas/parameters/msix_disable
   0
   ```

   1.5. Unload the **megaraid_sas.ko** module.

   ```
   [root@servera ~]# modprobe -rv megaraid_sas
   ```

2. Configure the **megaraid_sas.ko** module to always use the setting **msix_disable=1**. This option should be applied when the module is loaded manually, as well as when the module is loaded automatically.

   This configuration should be easy to copy to other systems, so a separate configuration file is needed.

   Verify that your new configuration works as intended.

   2.1. Create a new file called **/etc/modprobe.d/megaraid.conf** with the following content:

   ```
   options megaraid_sas msix_disable=1
   ```

   2.2. Load the **megaraid_sas.ko** kernel module.

   ```
   [root@servera ~]# modprobe -v megaraid_sas
   ```

   2.3. Verify that your new setting has been applied correctly.

   ```
   [root@servera ~]# cat /sys/module/megaraid_sas/parameters/msix_disable
   1
   ```

3. Grade your work by running the command **lab kernelmodules grade** on your **workstation** system.

   ```
   [student@workstation ~]$ lab kernelmodules grade
   ```

4. Clean up your work by running the command **lab kernelmodules reset** on your **workstation** system.

   ```
   [student@workstation ~]$ lab kernelmodules reset
   ```

Handling Virtualization Issues

Objectives

After completing this section, students should be able to identify and resolve virtualization issues.

Virtualization issues

When running (multiple) virtual machines using KVM and **libvirt**, a number of issues can arise. Some of these are hardware/firmware-related, others can be purely software/configuration-related. This section identifies a number of possible issues, their symptoms, and their fixes.

Hardware virtualization support

KVM requires hardware virtualization support both on the CPU, and in the system firmware (BIOS/UEFI). CPU support for KVM can be queried by looking at the **flags** section of **/proc/cpuinfo**. For Intel machines, the **vmx** flag should be available, and for AMD-based machines, the **svm** flag should be available.

If the required virtualization flag is available, support for KVM can be tested by loading the **kvm-intel** or **kvm-amd** kernel module manually. If the module loads without errors, or is already loaded, hardware virtualization support should be available. If loading the module fails with the following message, either the CPU does not support hardware virtualization or the feature is disabled in the system firmware:

```
[root@demo ~]# modprobe -v kvm-intel
modprobe: ERROR: could not insert 'kvm_intel': Operation not supported
```

Support for hardware virtualization can also be checked with the command **virsh capabilities**; this will show all the possible types of virtual machine that can be run.

```
[root@demo ~]# virsh capabilities
  <host>
    ....
  </host>
  <guest>
    <os_type>hvm</os_type>
    <arch name='x86_64'>❶
      <wordsize>64</wordsize>
      <emulator>/usr/bin/qemu-system-x86_64</emulator>
      <machine canonical='pc-i440fx-2.3' maxCpus='255'>pc</machine>
        ....
      <machine maxCpus='255'>pc-0.13</machine>
      <domain type='qemu'/>❷
      <domain type='kvm'>❸
        <emulator>/usr/bin/qemu-kvm</emulator>
        <machine canonical='pc-i440fx-2.3' maxCpus='255'>pc</machine>
        ....
      </domain>
    </arch>
    <features>
      <cpuselection/>
      <deviceboot/>
      <disksnapshot default='on' toggle='no'/>
      <acpi default='on' toggle='yes'/>
      <apic default='on' toggle='no'/>
    </features>
  </guest>
</capabilities>
```

1. This indicates what type of hardware will be virtualized; in this case, a 64-bit x86 machine. The output will contain multiple of these blocks in some cases; for example, both 32-bit and 64-bit VMs can be emulated.
2. This block indicates that this type of machine can be *emulated* using a **qemu** software CPU. Emulation is a lot slower than full hardware virtualization, in orders of magnitude.
3. This block indicates support for hardware virtualization using KVM. If it is missing, this type of machine cannot be virtualized in hardware, but requires emulation.

Important

If hardware virtualization is not available, and cannot be turned on in the system firmware, virtualized machines will run on an emulated processor, which is orders of magnitude slower than a hardware-virtualized CPU.

Overcommitted resources

libvirt allows administrators to assign more virtual resources to VMs than are actually available on the host system. In most cases, this can be done without repercussion (to a certain degree). Typical VM CPU usage is not 100%, so assigning more virtual CPUs than there are physical cores is not an issue, as long as VM CPU usage stays within limits. Memory can be overcommitted, and virtual machine memory can be swapped out to disk, or usage can be compressed by *Kernel Samepage Merging* (KSM), where duplicate memory pages are reduced to a single page and split out again when one VM writes to it. Sparse disk images can have a total size larger than the actual available space on the underlying storage, as long as the total used size does not exceed the actual available space on the host storage.

All of these types of overcommitting should not impact the overall performance of the running VMs, until a resource becomes stressed; for example, when a number of virtual machines start using all of their virtual CPUs at 100%, all VMs will feel the increased utilization, since VMs now have to compete for CPU.

When resources become scarce, overall performance and latency can suffer drastically. Not only does the host need to balance the available resources between the different VMs, it actually needs to use some of those resources itself to do the balancing.

Viewing resource usage

Since KVM virtual machines show up as regular processes on the host, normal performance metrics tools like **`top`** can be used to view resource usage. More specialized tools are also available in the form of **virt-manager**, and various **`virsh`** subcommands, such as **`virsh nodecpustats`**, **`virsh nodememstats`**, and **`virsh dommemstats`** ***`<DOMAIN>`***.

Various monitoring tools, such as **collectd** and **cockpit**, also have plug-ins to monitor virtual machine resource usage.

Dealing with scarce resources

There are a number of solutions that can be used to fight scarce resources. The first is to simply add more resources, either in the form of more memory, CPU, disk, etc., or by adding more hypervisor hosts and spreading out the virtual machines.

Another option is to limit the resource usage of various virtual machines, either by reducing the amount of resources allocated to them or by imposing limits using cgroups.

A third option is stopping unnecessary or noncritical virtual machines until resource usage has subsided again.

Libvirt XML configuration

libvirt stores virtual machine definitions (and other configurations) as XML. These XML files can be validated using the Relax NG schemas stored in **`/usr/share/libvirt/schemas`**.

If only **libvirt** tools like **`virsh`** and **virt-manager** have been used to make updates, all files should be valid XML, and validate against the provided schema files, but if an administrator has made manual changes to any of the files under **`/etc/libvirt/`**, issues might have been introduced.

An easy way to validate if a file is valid XML, and validates against the **libvirt** schema, requires two steps:

1. First, ensure that the file is valid XML. To do this, the **`xmllint`** tool can be used.

   ```
   [root@demo ~]# xmllint --noout <FILENAME>
   ```

   Typically, errors reported by **`xmllint`** should be fixed top-down, as any earlier errors can cause a fair number of false negatives further down the file.

2. After the file validates as well-structured XML, it can be validated to conform with the schema using the tool **`virt-xml-validate`**. This tool will (attempt) to parse the XML to find out the type of configuration it is, then validate it against the correct schema.

   ```
   [root@demo ~]# virt-xml-validate <FILENAME>
   ```

Libvirt networking

To provide virtual networks to virtual machines, **libvirt** uses software bridges. These bridges act as virtual switches, connecting all (virtual) network interfaces assigned to them.

These networks are defined in files in **`/etc/libvirt/qemu/networks/`**, and can be made to autostart with a symbolic link in **`/etc/libvirt/qemu/networks/autostart/`**. Normally, these files should not be edited manually, but rather with tools like **`virsh`** or **virt-manager**.

Apart from virtual networks created by **libvirt**, administrators can also configure regular bridges with one or more physical network interfaces assigned to them using **NetworkManager**.

Issues with virtual networking

If networking on virtual machines is not working, or not working as expected, a number of issues can be at play:

Virtual machine unreachable from the outside

> If a virtual machine cannot be connected to from outside the hypervisor machine, a number of issues can be at play: the virtual network can be of the type **NAT**; a firewall on the

hypervisor, or on the VM, can be blocking connections; or the client machine can lack a defined route to reach the VM.

Outside world unreachable from the virtual machine

If the virtual machine itself cannot reach the outside world, a number of issues can be present as well. The virtual network might be of the isolated type, or a firewall rule on the hypervisor might be blocking outgoing connections.

Connection issues both ways

In order to allow network traffic to virtual networks of the type **NAT** and **routed**, **libvirt** creates firewall rules using **iptables**. If an administrator clears all firewall rules, or adds blocking rules at the top of chains, this might interrupt regular network traffic to and from virtual machines. These issues can typically be solved by restarting the **libvirtd.service** **systemd** unit, or by restarting the individual **libvirt** networks that encounter the error.

References

The chapter on network configuration in the Red Hat Enterprise Linux Virtualization Deployment and Administration Guide available on **https://access.redhat.com/docs**.

The chapter on network bridging in the Red Hat Enterprise Linux Networking Guide available on **https://access.redhat.com/docs**.

Guided Exercise: Handling Virtualization Issues

In this lab, you will fix an invalid domain XML specification for a virtual machine.

Resources	
Files	**/root/somemachine.xml**
Machines	• **workstation**

Outcome(s)

You should be able to fix a malformed **libvirt** domain XML file.

Before you begin

On your **workstation** system, run the command **lab vmxml setup**.

```
[student@workstation ~]$ lab vmxml setup
```

One of your colleagues has been struggling to import a virtual machine on your **workstation** system. This virtual machine is defined in the file **/root/somemachine.xml**, but trying to import it with **virsh define /root/somemachine.xml** keeps returning errors about a malformed XML file.

You have been asked to investigate, and fix, this issue.

Important: For the purpose of this exercise, the virtual machine only needs to be defined, not started. The disk images referenced in the virtual machine definition do not exist, and do *not* need to be created.

1. Begin by recreating the issue. Take note of any errors, and try to interpret them.

   1.1. Attempt to define the virtual machine using the file provided.

   ```
   [root@workstation ~]# virsh define somemachine.xml
   error: Failed to define machine from somemachine.xml
   error: (domain_definition):20: Opening and ending tag mismatch: acpi line 17 and
    features
      </features>
   -------------^
   ```

   1.2. The first error, **Failed to define domain**, indicates that the operation did not successfully complete.

   The second error, **Opening and ending tag mismatch**, indicates that the XML in **somemachine** is malformed.

2. Use **xmllint** to verify the XML in **somemachine.xml**, then correct any errors.

   2.1. Use **xmllint** to verify **somemachine.xml**. Remember that the first error being reported can have a cascading effect, so fix errors one at a time, starting with the first error reported by **xmllint**.

```
[root@workstation ~]# xmllint somemachine.xml
```

It appears that there is an unclosed **<acpi>** tag on or around line 20.

2.2. Using your editor of choice, change line 17 in **somemachine.xml** to:

```
<acpi/>
```

2.3. Repeat the previous two steps, this time changing line 87 to include a closing quote:

```
<memballoon model='virtio'>
```

2.4. Run the **xmllint** command again to verify that there are no more lingering XML issues.

```
[root@workstation ~]# xmllint somemachine.xml
```

3. Verify that the **somemachine.xml** file now adheres to the **libvirt** XML schema.

```
[root@workstation ~]# virt-xml-validate somemachine.xml
somemachine.xml validates
```

4. Define a virtual machine using the **somemachine.xml** file.

```
[root@workstation ~]# virsh define somemachine.xml
Domain somemachine defined from somemachine.xml
```

5. Verify your work by running the **lab vmxml grade** command on **workstation**.

```
[student@workstation ~]$ lab vmxml grade
```

6. Clean up your **workstation** machine by running the command **lab vmxml reset**.

```
[student@workstation ~]$ lab vmxml reset
```

Lab: Identifying Hardware Issues

In this lab, you will configure options for the **usb-storage** kernel module.

Resources	
Machines	servera

Outcome(s)

You should be able to configure a module option for the **usb-storage** driver.

Before you begin

Ensure that your **servera** system is still up and running. If necessary, reset your **servera** system.

Your **servera** machine has been having issues with a removable storage device that uses the **usb-storage** driver. After some testing (and some Internet searching), you and your colleagues have determined that your device incorrectly reports itself as being write-protected.

The USB vendor ID for this device is **0513** and the device ID is **0132**.

Documentation for the **usb-storage** module option can be found in the file **kernel-parameters.txt** from the *kernel-doc* package.

You have been asked to configure **servera** in such a way that the **usb-storage** module will ignore the write-protect mode from that device.

1. Install the *kernel-doc* package, and search for options related to **usb-storage** in the **kernel-parameters.txt** file.
2. Create a permanent configuration to always enable the flag found in the previous step for the USB device with vendor ID **0513** and product ID **0132**.
3. Grade your work by running the command **lab hardwarelab grade** on your **workstation** system.

   ```
   [student@workstation ~]$ lab hardwarelab grade
   ```

4. Reset all of your machines to provide a clean environment for the next exercises.

Solution

In this lab, you will configure options for the **usb-storage** kernel module.

Resources	
Machines	servera

Outcome(s)

You should be able to configure a module option for the **usb-storage** driver.

Before you begin

Ensure that your **servera** system is still up and running. If necessary, reset your **servera** system.

Your **servera** machine has been having issues with a removable storage device that uses the **usb-storage** driver. After some testing (and some Internet searching), you and your colleagues have determined that your device incorrectly reports itself as being write-protected.

The USB vendor ID for this device is **0513** and the device ID is **0132**.

Documentation for the **usb-storage** module option can be found in the file **kernel-parameters.txt** from the *kernel-doc* package.

You have been asked to configure **servera** in such a way that the **usb-storage** module will ignore the write-protect mode from that device.

1. Install the *kernel-doc* package, and search for options related to **usb-storage** in the **kernel-parameters.txt** file.

    1.1. Install the *kernel-doc* package.

    ```
    [root@servera ~]# yum -y install kernel-doc
    ```

    1.2. Open the file **/usr/share/doc/kernel-doc*/Documentation/kernel-parameters.txt** and search for **usb-storage**. Which parameter could be useful, and which flag will need to be used?

    The **quirks** parameter looks interesting, with the **w** (**NO_WP_DETECT**) flag added.

2. Create a permanent configuration to always enable the flag found in the previous step for the USB device with vendor ID **0513** and product ID **0132**.

    2.1. Create a new file called **/etc/modprobe.d/usb-storage.conf** with the following content:

    ```
    options usb-storage quirks=0513:0132:w
    ```

3. Grade your work by running the command **lab hardwarelab grade** on your **workstation** system.

    ```
    [student@workstation ~]$ lab hardwarelab grade
    ```

4. Reset all of your machines to provide a clean environment for the next exercises.

Summary

In this chapter, you learned:

- How to identify CPUs using the **lscpu** command.
- How to identify hardware using **lsusb** and **lspci**.
- How to identify memory banks with **dmidecode -t memory**.
- How to identify hard drives with **hdparm -I**.
- How to start and enable the **mcelog.service** service to collect logging on hardware failures.
- How to start and enable the **rasdaemon.service** to collect logging on hardware failures.
- The three steps to enable **memtest86+**: Install the *memtest86+* package, run **memtest-setup**, and run **grub2-mkconfig -o /boot/grub2/grub.cfg**.
- How to use **lsmod** to view loaded kernel modules.
- How to use **modprobe** and **modprobe -r** to load and unload kernel modules.
- How to create files in **/etc/modprobe.d/** to persistently enable kernel module options.
- How to identify if a machine supports hardware virtualization using **virsh capabilities**.
- How to validate a **libvirt** configuration file using **xmllint** and **virt-xml-validate**.

CHAPTER 5

TROUBLESHOOTING STORAGE ISSUES

Overview	
Goal	Identify and fix issues related to storage.
Objectives	• Describe the functions of the layers of the Linux storage stack • Recover corrupted file systems. • Recover from broken LVM configurations. • Recover data from encrypted file systems. • Identify and fix iSCSI issues.
Sections	• Orientation to the Linux Storage Stack • Recovering from File System Corruption (and Guided Exercise) • Recovering from LVM Accidents (and Guided Exercise) • Dealing with LUKS Issues (and Guided Exercise) • Resolving iSCSI Issues (and Guided Exercise)
Lab	• Troubleshooting Storage Issues

Orientation to the Linux Storage Stack

Objectives

After completing this section, students should be able to describe the layers of the Linux storage subsystem, explain each layer's function, and identify some tools that can be used to examine activity at different layers.

The Linux storage stack

Troubleshooting storage issues in Red Hat Enterprise Linux should start with an understanding of how I/O is passed from applications through the Linux storage stack to storage devices.

Applications read and write from storage by sending standardized system calls to the kernel. The kernel processes these system calls through the *storage stack*, various layers of software and hardware, to ensure that application data may be successfully stored and accessed on the hardware storage device. The storage stack allows application programmers to access data in a standardized way while offloading the work of writing code to interact with specific filesystem implementations and storage devices to kernel programmers. The storage stack also provides a number of features to improve the performance and reliability of I/O operations.

This section will look at the various layers of the Linux storage stack in turn. The entire Linux storage stack can be represented by the following diagram:

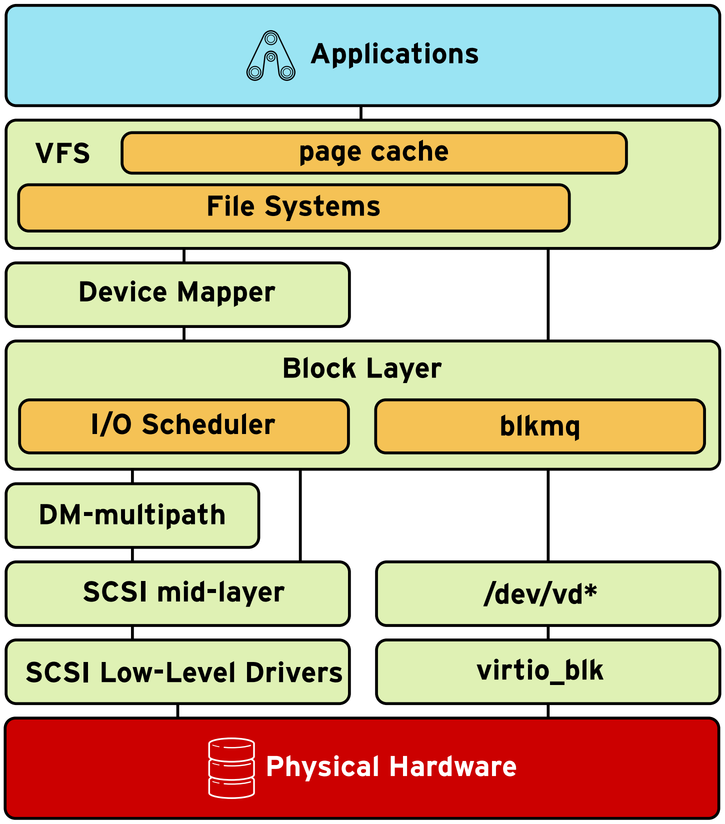


Figure 5.1: The Linux storage stack

The Virtual Filesystem (VFS)

The *Virtual Filesystem* (VFS) provides support for the standard POSIX system calls that applications use to read and write files. It implements a *common file model* so that applications can use the same system calls such as **creat()**, **open()**, **read()**, **write()**, and **mmap()** to access files without needing to deal with details of the underlying filesystem implementation.

This means that a user can work with files stored on different filesystems without needing to rewrite their applications or tools like **cp** or **mv** to support the new filesystem, as long as those programs use the common file model when working with files.

In turn, specific filesystem implementations such as XFS, ext4, FAT32, and so on can be plugged into VFS as code modules so that the common file model can be used to access data located on storage using those filesystems.

For example, Linux applications assume that directories can be handled like files. Different filesystems actually store directory data in different ways. VFS makes sure that applications can treat directories like files while working with the filesystem code to correctly handle the true implementation of directories in that filesystem.

VFS also maintains several caches to improve storage I/O performance, the inode cache, dentry cache, buffer cache, and page cache. Of these, the most important is the *page cache*. The page cache is dynamically allocated from free memory on the system, and is used to cache pages of data from files being read or written. Since the Linux 2.4 kernel, the *buffer cache* which caches disk blocks is mostly unified with the page cache, with the exception of a small and mainly unimportant amount of block data that's not backed by files (for metadata or raw block I/O). The inode cache and dentry cache are used to speed up access to inodes (file metadata) and directory entries respectively, and the memory usage of those caches is tracked in **/proc/slabinfo**.

Direct I/O (**O_DIRECT**) can be used by applications that implement their own caching, such as some databases, to bypass the page cache. Normally this is only done if the application is the only process working with a file and it can do a better job actively managing its cache than the system would do automatically. If this is done incorrectly or if other processes are accessing the file through the page cache, corruption and other problems are possible. There are a number of important restrictions and conditions on Direct I/O which will not be discussed here.

VFS tools

Issues at this layer typically have to do with performance problems due to memory shortages constraining the caches or **sysctl** tuning mistakes. To get an overview of cache usage, the command **free** can be run for a basic summary or **cat /proc/meminfo** for more detailed information.

```
[root@demo ~]# cat /proc/meminfo
MemTotal:        1884120 kB
MemFree:         1337828 kB
MemAvailable:    1604580 kB
Buffers:             872 kB
Cached:           373040 kB
```

Cache will be dynamically allocated from free memory as needed to support I/O, under the theory that unused memory is a wasted resource, and will be freed for use by applications if the system is under memory pressure. The page, inode, and dentry caches can be cleared by running:

```
[root@demo ~]# echo 3 > /proc/sys/vm/drop_caches
```

This will hurt system performance until the system can repopulate the caches with data being frequently accessed.

There are a large number of kernel **sysctl** tunables exposed in **/proc/sys/vm** that affect the performance of the disk caches and memory management in general. An in-depth discussion is beyond the scope of this course.

Filesystems

Filesystems provide the logical structures that determine how files and their data and metadata are organized and named on storage, and how they are protected against compromise and corruption. Different filesystems may provide different features, and may implement similar features in different ways. The filesystems usually used for Linux system disks such as XFS and ext4 will at least share a common subset of features required by the POSIX standard which allow them to integrate well with VFS and the common file model.

Filesystems may be backed by block storage (XFS, ext4, GFS2, FAT32), network storage (NFS, SMB), or they may be synthetic "pseudo-filesystems" (procfs, sysfs) or special purpose filesystems (tmpfs) backed by memory and presented as if they were storage by the kernel.

An example of a common troubleshooting scenario at the filesystem layer would be recovery of a corrupt filesystem after a system crash. Also, mistakes made with data alignment of filesystem structures to lower level block size when storage is formatted can cause performance issues.

Device mapper

Device mapper is a powerful mechanism in the kernel to create 1:1 mappings of blocks in one block device to blocks in another, logical block device. This layer is used in order to implement LVM, LUKS disk encryption, and device mapper RAID, among other things. Its use is optional, it is possible for a physical block device to be directly formatted with a filesystem without using device mapper at all.

For example, take an LVM logical volume named **/dev/mapper/myvg1-mylv1**, which has two physical volumes, **/dev/vdb1** and **/dev/vdb2**. When this logical volume was created using normal LVM utilities, at a lower level device-mapper mapped the physical block device partitions **/dev/vdb1** and **/dev/vdb2** to a logical device mapper device, **/dev/dm-0** in this case. The **dmsetup** command can be used to see the device mappings:

```
[root@servera ~]# dmsetup ls
myvg1-mylv1    (252:0)

[root@servera ~]# ls -l /dev/mapper/myvg1-mylv1
lrwxrwxrwx. 1 root root 7 Feb 26 09:40 /dev/mapper/myvg1-mylv1 -> ../dm-0

[root@servera ~]# dmsetup table /dev/mapper/myvg1-mylv1
0 1015808 linear 253:17 2048
1015808 1015808 linear 253:18 2048
```

This table says that **/dev/mapper/myvg1-mylv1** is made up of two mappings. The first is a 1:1 linear mapping of the block device with major/minor number 253:17 to the first 1015808 blocks of **/dev/mapper/myvg1-mylv1**. The second is a 1:1 linear mapping of the block device with major/

minor number 253:18 to the next 1015808 blocks of **/dev/mapper/myvg1-mylv1** (starting at block 1015808). The major:minor numbers 253:17 and 253:18 correspond to **/dev/vdb1** and **/dev/vdb2**:

```
[root@servera ~]# ls -l /dev/vdb*
brw-rw----. 1 root disk 253, 16 Feb 26 09:31 /dev/vdb
brw-rw----. 1 root disk 253, 17 Feb 26 09:40 /dev/vdb1
brw-rw----. 1 root disk 253, 18 Feb 26 09:40 /dev/vdb2
```

The resulting **/dev/dm-0** logical volume which is created by device mapper looks like this:

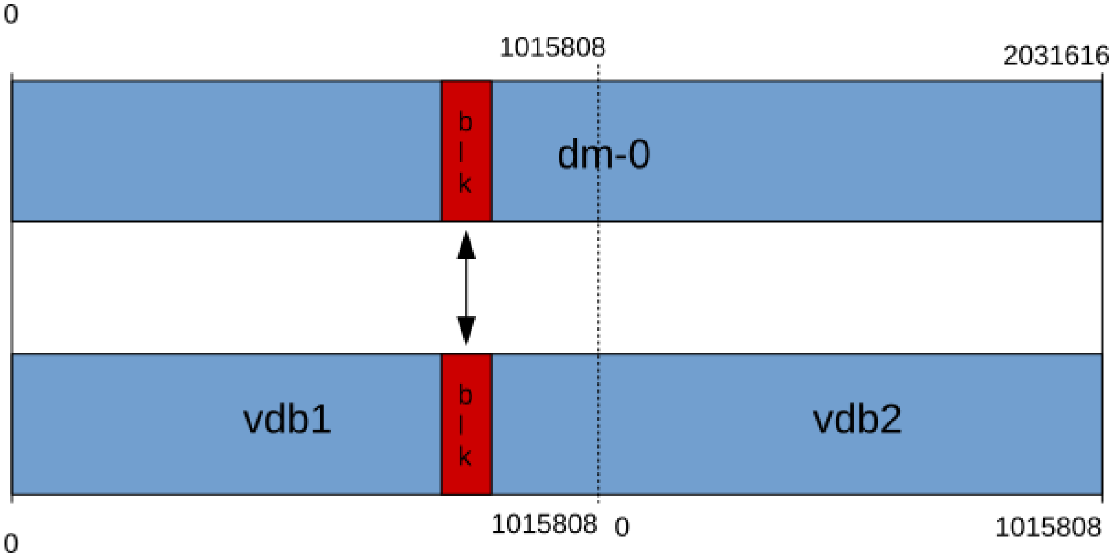


Figure 5.2: Device-Mapper mapping blocks between two storage devices and a DM device

The logical device created by device mapper in this example (**/dev/dm-0**) is therefore 2031616 sectors in size (1015808 sectors from **/dev/vdb1** and 1015808 sectors from **/dev/vdb2**).

The generic block layer and I/O schedulers

Applications perform read/write operations on files through the VFS and the filesystem. The VFS and the filesystem convert these requests to read and write files on the filesystem into requests to read and write blocks on the storage device. These requests are then passed to the *generic block layer* The job of the block layer is to organize these requests into a queue and submit them to the device driver.

Historically, storage hardware has been much slower than the system. The block layer's goal is to maximize performance by doing three things:

- *Merge* adjacent requests together to improve the amount of data being read or written per request
- *Sort* requests to minimize disk overhead
- *Prioritize* requests to manage latency and ensure that no request waits in a queue too long before being dispatched to the driver and out to storage

In Red Hat Enterprise Linux 7, most devices pass block I/O requests through one of three request-based I/O scheduler algorithms:

- **deadline** is the default for most block devices in RHEL 7 except for SATA devices and paravirtualized **virtio_blk** storage. It focuses on making sure that requests are sorted to be efficiently sent to the device unless requests have been waiting too long, in which case they are sent immediately, biasing read requests ahead of writes

- **cfq** ("completely fair queuing") is the default for SATA disks, and for most disks in RHEL 6 systems. Each process has a request queue for synchronous writes, there are a smaller number of queues for asynchronous I/O, and the scheduler processes each of them round-robin for a given time-slice.

- **noop** is a simple first-in, first-out scheduler that does basic merging of requests but no sorting before submitting them to the device. It is particularly good for truly random-access capable devices or ones which do their own scheduling (like enterprise storage controllers).

The standard I/O scheduler was designed for spinning hard drives, operating with latencies of milliseconds and hundreds of I/O operations per second. A new block I/O multi-queuing mechanism called **blk-mq** bypasses the traditional schedulers and allows certain device drivers to map I/O requests to multiple queues. The **virtio** paravirtualized **/dev/vd*** storage devices started using this subsystem in Red Hat Enterprise Linux 7.1 / *kernel-3.10.0-183.el7* instead of traditional I/O schedulers. In the long term, this new system is designed to handle storage with latencies of microseconds, millions of IOPS, and large internal parallelism.

Generic block layer and I/O scheduler tools

To see what scheduler is used with a particular device, inspect within sysfs for that block device:

```
[root@host1 ~]# cat /sys/block/sda/queue/scheduler
noop deadline [cfq]
```

It is possible to trace I/O through the block layer using the *blktrace* package. The command **blktrace /dev/sda** will trace I/O on **/dev/sda** and save the output to a file named something like **sda.blktrace.0** in the current directory. Then the **blkparse** and **btt** commands can be used to analyze the results of the trace.

Device mapper multipath

Device mapper multipath (DM Multipath) is used to configure multiple I/O paths between server nodes and storage arrays into a single device. These I/O paths can be physical Storage Area Network (SAN) connections which include separate cables, switches, and controllers. Multipathing aggregates the multiple I/O paths, creating a new device that consists of the aggregated paths.

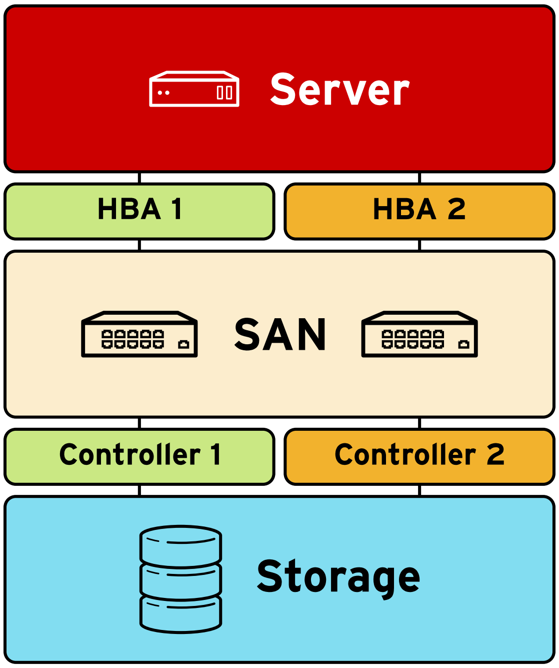


Figure 5.3: Device mapper multipath

In this diagram, a Red Hat Enterprise Linux server has two host bus adapters (HBAs) attached to a SAN. This is usually done using Fiber Channel physical connections. The SAN also has connections to two different storage controllers. As a result, this configuration allows for four possible "paths" to the back-end storage. This is incredibly useful for fault tolerance, but can also be used to improve performance.

In Red Hat Enterprise Linux, the kernel's device mapper mechanism is used to generate separate block devices to represent different paths to the same storage device. This is managed by the **multipathd** daemon and the **`multipath`** command-line tool, from the *device-mapper-multipath* package. These paths are then accessed through another multipath block device. That administrative block device will send the requests to one of the underlying block devices associated with it that represents a particular path.

For administrative purposes, multipath devices are created under **`/dev/mapper`**. They can be named **`/dev/mapper/mpath`*`N`*`[p`*`M`*`]`** if *user-friendly* names are chosen, or they can be named after the *World Wide ID* (WWID) of the storage device. An administrator can also set custom names for multipathed devices.

This layer only comes into play if the block device has multiple paths available and device mapper multipath is being used.

The SCSI mid-layer

The *SCSI mid-layer* is the common bridge layer between the *high-level SCSI drivers* that present the core device files used by storage and the low-level drivers that operate the individual host bus adapters and other hardware interface cards that communicate with the storage device.

There are four main high-level SCSI drivers. For block devices, the two most important are the SCSI disk (**`sd`**) driver and the SCSI CDROM (**`sr`**) driver. In addition, there is a high-level SCSI driver for character-based SCSI tape devices (**`st`**) and one for generic SCSI devices such as scanners (**`sg`**).

Anything that acts like a SCSI device and uses the SCSI protocol to communicate between the low-level driver and the high-level driver uses the SCSI mid-layer. This may include things that at first glance do not seem to be related to SCSI at all, such as SATA and USB storage and certain low-level paravirtualized storage devices. These will be presented to the system as SCSI devices and use appropriate device names (**/dev/sda**, for example).

Some storage devices bypass this layer. For example, the **/dev/vd*** devices are paravirtualized devices that talk directly to the **virtio_blk** low-level driver, which is not part of the SCSI stack and does not emulate the SCSI protocol.

Low-level drivers

Low-level drivers directly communicate with the specific physical hardware used by the system. In many cases, these are SCSI low-level drivers that can take advantage of the SCSI mid-layer. For example, these can include drivers for Qlogic (**qla2xxx**) or Adaptec (**aacraid**) or Emulex (**lpfc**) devices, or local SATA or USB devices through **libata** and **ahci** and so on. An iSCSI device that uses TCP/IP as a transport would have a driver at this level as well (such as **iscsi_tcp**) before passing its traffic to the network stack. Some paravirtualized devices may be presented as SCSI disks, such as those using the **virtio_scsi** and **vmw_pvscsi** drivers. However, there are low-level paravirtualized drivers such as **virtio_blk** that do not use the SCSI mid-layer at all but have an upper-layer driver that interacts directly with the scheduler at the block layer.

Whether using the SCSI mid-layer or not, the low-level driver receives I/O from the scheduler at the block layer and dispatches it to the underlying storage hardware. The driver takes the incoming I/O requests and transforms them into a form that is acceptable to the underlying hardware controller. The low-level driver does not queue I/O itself, but it does track the I/O requests "in flight". This is the lowest level of the storage stack that is still part of the Linux kernel.

Physical hardware

Host bus adapters (HBAs) and firmware

The *host bus adapter* (HBA) is the physical hardware on the server that is used to communicate with the storage device. This may be a SCSI or SATA host adapter or a Fiber Channel interface card. That device may have its own firmware that controls its operations. In a guest virtual machine, a paravirtualized **virtio_pci** device may be presented to the machine as its hardware.

The transport layer

The *transport layer* defines the wire protocol used by the HBA to communicate with attached storage. This layer could involve communication over a point-to-point SATA or SAS connection, USB, Fiber Channel, or even iSCSI over TCP/IP.

Physical storage devices

Finally, the actual *physical storage* is the device that actually contains the data. This might be a single hard drive with spinning magnetic platters, a solid state drive (SSD), optical storage such as a CD-ROM or DVD-ROM, or an enterprise storage array potentially made up of some combination of the above. These devices may have their own firmware or sophisticated storage operating systems.

References

Werner Fischer has published a more detailed diagram of the storage stack for recent Linux kernels, under the CC-BY-SA license, at

> Linux Storage Stack Diagram
> https://www.thomas-krenn.com/en/wiki/Linux_Storage_Stack_Diagram

> Overview of the Linux Virtual File System
> https://access.redhat.com/labs/psb/versions/kernel-3.10.0-327.el7/Documentation/filesystems/vfs.txt

- Also available on the local system as **`/usr/share/doc/kernel-doc-*/Documentation/filesystems/vfs.txt`**, provided by the *kernel-doc* package

> Using the Deadline IO scheduler
> https://access.redhat.com/solutions/32376

> Understanding the Deadline IO scheduler
> https://access.redhat.com/articles/425823

> Understanding the noop scheduler
> https://access.redhat.com/articles/46958

> Unable to change IO scheduler for virtio disk /dev/vda in RHEL 7.1
> https://access.redhat.com/solutions/1305843

> *Red Hat Enterprise Linux 7.2 Release Notes*: New Features. Storage: "Multiqueue I/O scheduling with blk-mq"
> https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/7.2_Release_Notes/storage.html#idp1704576

> SCSI subsystem documentation
> https://access.redhat.com/labs/psb/versions/kernel-3.10.0-327.el7/Documentation/scsi/scsi.txt

- Also available on the local system as **`/usr/share/doc/kernel-doc-*/Documentation/scsi/scsi.txt`**, provided by the *kernel-doc* package. Other kernel documentation which may be of interest is available in this directory as well.

`free`(1), **`dmsetup`**(8), **`multipath`**(8), **`blktrace`**(8), **`blkparse`**(1), and **`btt`**(1) man pages

Quiz: Orientation to the Linux Storage Stack

Choose the correct answer to the following questions:

1. Select all disk elevators.

   a. **`noop`**
   b. **`cfq`**
   c. **`virtio_blk`**
   d. **`deadline`**

2. Which of these devices communicate through the SCSI mid-layer?

   a. USB Mass Storage devices
   b. IDE hard drives
   c. SATA hard drives
   d. Paravirtualized **`virtio_blk`** devices.

3. Which kernel subsystem is used to map logical block devices to physical storage?

   a. **`DM-Multipath`**
   b. **`LVM`**
   c. **`device-mapper`**
   d. **`VFS`**

4. Which of the following statements about the kernel VFS layer are true?

   a. The VFS layer maintains various caches, such as the inode cache, dentry cache, and page cache.
   b. Applications can bypass the page cache by opening files with the **`O_DIRECT`** flag set.
   c. The VFS layer is only used for some common file systems like ext4, but not for others, such as XFS.
   d. The VFS layer provides a generic interface to user space applications across file system implementations.

Solution

Choose the correct answer to the following questions:

1. Select all disk elevators.

   a. **`noop`**
   b. **`cfq`**
   c. `virtio_blk`
   d. **`deadline`**

2. Which of these devices communicate through the SCSI mid-layer?

   a. **USB Mass Storage devices**
   b. **IDE hard drives**
   c. **SATA hard drives**
   d. Paravirtualized `virtio_blk` devices.

3. Which kernel subsystem is used to map logical block devices to physical storage?

   a. `DM-Multipath`
   b. `LVM`
   c. **`device-mapper`**
   d. `VFS`

4. Which of the following statements about the kernel VFS layer are true?

   a. **The VFS layer maintains various caches, such as the inode cache, dentry cache, and page cache.**
   b. **Applications can bypass the page cache by opening files with the `O_DIRECT` flag set.**
   c. The VFS layer is only used for some common file systems like ext4, but not for others, such as XFS.
   d. **The VFS layer provides a generic interface to user space applications across file system implementations.**

Recovering from File System Corruption

Objectives

After completing this section, students should be able to detect and recover from file system corruption.

File system choices

Red Hat Enterprise Linux offers system administrators a choice of several file systems to suit the needs of their system's workload requirements. In RHEL 6, ext4 was the default file system selection and XFS was offered as another option. However, with RHEL 7, XFS now supplants ext4 as the default file system selection.

Red Hat Enterprise Linux 7 provides a unified driver that supports the ext2, ext3, and ext4 file systems. While the ext2 file system is supported by this unified extended file system driver, ext2 is now considered deprecated. Therefore, administrators should avoid the use of the ext2 file system if at all possible.

File system journaling was introduced to Red Hat Enterprise Linux with the introduction of ext3. Journaling of file systems improves data integrity in the event of a power loss or system crash. After such an event, a journaled file system can use its journal to quickly recover unsaved information in an attempt to avoid file system metadata corruption.

XFS and ext4 file systems are also journaling file systems. While journaled file systems provide greater fault resiliency, they do not protect against file system corruption. File system corruption can still occur, even on journaled file systems, due to hardware failure, storage administration mistakes, and software bugs.

Identifying file system corruption

When file system corruption is encountered or suspected, administrators can use file system checking tools to perform a comprehensive consistency check on the entire file system. Once the nature and extent of the data corruption is identified, administrators can use file system repair tools to try to restore the consistency of the file system. The following are some good practices to follow when recovering from file system corruption.

- If a hardware failure is the cause of the data corruption, the hardware issue should be resolved prior to running a check or repair on the file system. For example, if a storage disk is failing, move the file system to a functional disk first before performing file system maintenance. The data migration can be performed with various common utilities such as **dd**. This will provide the functional hardware foundation that is required for file system maintenance.

- A file system check should be performed prior to a file system repair. This serves as a dry run for the file system repair. During this dry run, the file system check will report any file system issues discovered and corrective actions that can be taken.

- A file system should be unmounted prior to file system administration. In order to successfully restore file system consistency, file system repair tools need to have sole access to the file system. Unmounting a file system ensures that no storage operations, other than the file system repair, can be performed. Running a file system repair on a mounted file system can lead to further data corruption.

Checking ext3/ext4 file systems

After a power loss or system crash, the **e2fsck** utility will automatically be run on a system to perform journal recovery for ext3 and ext4 file systems. Replaying of the journal ensures a file system's consistency after a system crash or unclean shutdown. If file system metadata inconsistencies are recorded, then **e2fsck** will perform a full check of the file system after the journal is replayed. During the file system check, user intervention will be requested if any issues encountered are not safely resolvable by **e2fsck**.

Aside from the automatic execution of **e2fsck** at boot time, administrators can also manually execute **e2fsck** to initiate a file system check. To perform a file system check of an ext3 or ext4 file system, unmount the file system and run **e2fsck** with the **-n** option and then specifying the device name on which the file system resides.

```
[root@demo ~]# e2fsck -n /dev/vdb1
```

The **-n** option will instruct **e2fsck** to place the file system in read-only mode, as well as answer '*no*' to all questions raised during the file system check.

When executing a consistency check on an ext3 or ext4 file system, an administrator may be confronted with a corrupt superblock. If the primary superblock for the file system is corrupt, it may be necessary to use an alternate superblock to perform the file system check.

```
[root@demo ~]# e2fsck -n /dev/vdb1
e2fsck 1.42.9 (28-Dec-2013)
ext2fs_open2: Bad magic number in super-block
e2fsck: Superblock invalid, trying backup blocks...
e2fsck: Bad magic number in super-block while trying to open /dev/vdb1

The superblock could not be read or does not describe a correct ext2
file system.  If the device is valid and it really contains an ext2
file system (and not swap or ufs or something else), then the superblock
is corrupt, and you might try running e2fsck with an alternate superblock:
    e2fsck -b 8193 <device>
```

The location of backup superblocks varies depending on the file system's block size. To determine the location of backup superblocks, use the **dumpe2fs** utility.

```
[root@demo ~]# dumpe2fs /dev/vdb1 | grep 'Backup superblock'
dumpe2fs 1.42.9 (28-Dec-2013)
  Backup superblock at 32768, Group descriptors at 32769-32769
  Backup superblock at 98304, Group descriptors at 98305-98305
  Backup superblock at 163840, Group descriptors at 163841-163841
  Backup superblock at 229376, Group descriptors at 229377-229377
```

Once the location of alternative superblocks are identified, use the **-b** option to specify the alternate superblock to use during the file system check.

```
[root@demo ~]# e2fsck -n /dev/vdb1 -b 32768
e2fsck 1.42.9 (28-Dec-2013)
/dev/vdb1 was not cleanly unmounted, check forced.
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information
```

```
/dev/vdb1: 11/65536 files (0.0% non-contiguous), 8859/261888 blocks

[root@demo ~]# echo $?
0
```

The result of a file system check can be determined by the exit status of the **e2fsck** command. The exit status reported is the sum of the exit codes triggered in the following table.

Exit code	Description
0	No errors.
4	File system errors uncorrected.
8	Operational error.
16	Usage error.
32	Cancelled by user.
128	Shared library error.

Note

Both the **e2fsck** and the **dumpe2fs** utilities are provided by the *e2fsprogs* package.

Checking XFS file systems

Unlike ext3 and ext4 file systems, file system check and repair is not automatically initiated by the system at boot time on XFS file systems. To execute an XFS file system check, the administrator needs to manually execute the XFS file system check utility, which is provided by the *xfsprogs* package.

With earlier versions of the *xfsprogs* package, the **xfs_check** utility was provided for performing file system checks on XFS file systems. This utility was extremely slow and did not scale well for large file systems. Therefore, the **xfs_check** utility has since been deprecated and XFS file system checks are now performed using the **xfs_repair** utility.

The **xfs_repair** command is invoked with the **-n** option in order to perform file system checks. The **-n** option will instruct **xfs_repair** to scan the file system and only report on repairs to be made rather than performing any corrective action. The following command performs a file system check on the XFS file system residing on **/dev/vdb1**.

```
[root@demo ~]# xfs_repair -n /dev/vdb1
```

The **xfs_repair** command can only be executed on an XFS file system with a clean journal log. XFS file systems that do not have a clean journal log due to unclean system shutdown or system crash need to be mounted and unmounted first in order to be checked by **xfs_repair**.

Like ext3 and ext4 file systems, it's possible for file system checks to fail to execute due to a corrupt primary superblock. However, unlike **e2fsck**, **xfs_repair** does not require that the administrator specify the locations of the alternate superblocks. It will automatically scan the XFS file system until it locates a secondary superblock. Once located, the secondary superblock will be used to recover the primary superblock.

```
[root@demo ~]# xfs_repair -n /dev/vdb1
```

```
Phase 1 - find and verify superblock...
Phase 2 - using internal log
        - scan file system freespace and inode maps...
        - found root inode chunk
Phase 3 - for each AG...
        - scan (but don't clear) agi unlinked lists...
        - process known inodes and perform inode discovery...
        - agno = 0
        - agno = 1
        - agno = 2
        - agno = 3
        - process newly discovered inodes...
Phase 4 - check for duplicate blocks...
        - setting up duplicate extent list...
        - check for inodes claiming duplicate blocks...
        - agno = 0
        - agno = 1
        - agno = 2
        - agno = 3
No modify flag set, skipping phase 5
Phase 6 - check inode connectivity...
        - traversing file system ...
        - traversal finished ...
        - moving disconnected inodes to lost+found ...
Phase 7 - verify link counts...
No modify flag set, skipping file system flush and exiting.

[root@demo ~]# echo $?
0
```

An XFS file system check will return an exit code of **1** if file system corruption was detected. If no file system corruption was detected, an exit code of **0** will be returned.

Repairing file systems

Once a file system check has been performed, an administrator will have identified any file system errors which exist and will also have been informed of the corrective actions that can be taken to restore the integrity of the file system. Administrators can initiate file system repair by executing the repair utilities associated with a file system type.

Repairing ext3/ext4 file systems

ext3 and ext4 file systems are repaired using the same **e2fsck** command used to perform checks of the file systems. When the **-n** option is not specified, **e2fsck** will perform all corrective actions that can be performed safely to repair the file system. For operations that cannot be performed safely, the administrator will be prompted for input regarding whether the action should be executed.

As mentioned previously, the file system should be unmounted during file system repair to ensure file system consistency and to prevent further corruption. Depending on the nature of the file system corruption, administrators may choose to execute **e2fsck** with some of the following additional options.

Option	Description
-b *location*	Use alternate superblock at the specified location.
-p	Automatically repair the file system. Only prompt user for problems which cannot be safely fixed.
-v	Verbose mode.

Option	Description
-y	Run in noninteractive mode and answer **yes** to all questions. This option cannot be used in conjunction with the **-p** or **-n** options.

The following is an example repair on an ext4 file system using the **-y** and **-b** options to execute a noninteractive file system check using an alternate superblock.

```
[root@demo ~]# e2fsck /dev/vdb1 -b 98304
e2fsck 1.42.9 (28-Dec-2013)
/dev/vdb1 was not cleanly unmounted, check forced.
Resize inode not valid.  Recreate? yes

Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information
Free blocks count wrong for group #0 (28520, counted=28521).
Fix? yes

Free blocks count wrong (253028, counted=253029).
Fix? yes


/dev/vdb1: ***** FILE SYSTEM WAS MODIFIED *****
/dev/vdb1: 11/65536 files (0.0% non-contiguous), 8859/261888 blocks

[root@demo ~]# echo $?
1
```

The result of the file system repair can be determined by the exit code of the **e2fsck** command. The exit status reported is the sum of the exit codes triggered in the following table.

Exit code	Description
0	No errors.
1	File system errors corrected.
2	File system errors corrected. Reboot required.
4	File system errors uncorrected.
8	Operational error.
16	Usage error.
32	Cancelled by user.
128	Shared library error.

Repairing XFS file systems

XFS file systems are repaired using the same **xfs_repair** command used to perform checks of the file systems. When the **-n** option is not specified, **xfs_repair** will perform all corrective actions that can be performed safely to repair the file system. As previously mentioned, the file system needs to be unmounted during file system repair to ensure file system consistency as well as to prevent further corruption.

As previously stated, **xfs_repair** can only be executed on an XFS file system with a clean journal log. If mounting and unmounting an XFS file system still does not result in a clean log, it is possible that the log has been corrupted. Since a clean log is necessary, it may be necessary in these situations to utilize the **-L** option to **xfs_repair**. This option zeros out the journal log. While this step is necessary to proceed, administrators should take note that it can lead to further file system inconsistencies since it discards all file system metadata in the journal log.

Unlike **e2fsck**, **xfs_repair** is not an interactive utility. Once initiated, the XFS file system repair will perform all operations automatically without any input from the user. Also, in contrast with **e2fsck**, **xfs_repair** will always return an exit code of **0** upon completion of the file system repair, as demonstrated in the following example.

```
[root@demo ~]# xfs_repair /dev/vdb1
Phase 1 - find and verify superblock...
Phase 2 - using internal log
        - zero log...
        - scan file system freespace and inode maps...
        - found root inode chunk
Phase 3 - for each AG...
        - scan and clear agi unlinked lists...
        - process known inodes and perform inode discovery...
        - agno = 0
        - agno = 1
Invalid inode number 0x499602d2
xfs_dir_ino_validate: XFS_ERROR_REPORT
Metadata corruption detected at block 0xa7e0/0x1000
entry "subscription-manager" at block 0 offset 456 in directory inode 144973 references
 invalid inode 1234567890
        clearing inode number in entry at offset 456...
entry at block 0 offset 456 in directory inode 144973 has illegal name "/ubscription-
manager":         - process newly discovered i
nodes...
Phase 4 - check for duplicate blocks...
        - setting up duplicate extent list...
        - check for inodes claiming duplicate blocks...
        - agno = 0
        - agno = 1
Phase 5 - rebuild AG headers and trees...
        - reset superblock...
Phase 6 - check inode connectivity...
        - resetting contents of realtime bitmap and summary inodes
        - traversing file system ...
bad hash table for directory inode 144973 (no data entry): rebuilding
rebuilding directory inode 144973
        - traversal finished ...
        - moving disconnected inodes to lost+found ...
disconnected inode 145282, moving to lost+found
Phase 7 - verify and correct link counts...
done

[root@demo ~]# echo $?
0
```

During an XFS file system repair, it is possible to discover files and directories that have been allocated inodes and are in use, but are unreferenced by their parent directories. When these orphaned files and directories are discovered during file system checks, they are deposited in the **lost+found** directory located at the root of that file system. If files are missing after a file system repair, check in the **lost+found** directory to see if they have been deposited there.

File system backup and recovery

While the previously mentioned utilities can be used to detect and repair file system inconsistencies, administrators should not treat them as a replacement for a sound backup and recovery strategy. The **e2fsck** and **xfs_repair** commands are used to perform file system repair, but their outcomes are not guaranteed.

If severely damaged inodes or directories are encountered during file system repair, they may be permanently discarded if they cannot be fixed. Due to the possibility of such data loss, it is important that administrators still have recent backups of the file system's data to fall back on, so that they can properly recover lost data.

References

For more information, see the **e2fsck**(8), **dumpe2fs**(8), and **xfs_repair**(8) man pages.

Guided Exercise: Recovering from File System Corruption

In this lab, you will verify a file system and repair file system corruption.

Resources	
Files	**/root/etc.tgz**
Machines	**servera**

Outcome(s)

You should be able to check and repair an XFS file system.

Before you begin

Prepare your systems for this exercise by running the command **lab xfs setup** on your **workstation** machine.

```
[student@workstation ~#$ lab xfs setup
```

The **/dev/vdb1** partition on **servera** contains an XFS file system which holds the contents of the **/etc** directory from another system. Check the integrity of the XFS file system on **/dev/vdb1**. Repair any file system inconsistencies found and then mount the file system at **/mnt/etc_restore**. If the file system repair deposits any files in the file system's **lost+found** directory, use the backup file, **/root/etc.tgz**, to determine the proper location and name to restore the orphaned file to. Once validated, restore the orphaned file back to its proper location.

1. Perform a check of the XFS file system on the **/dev/vdb1** partition.

   1.1. Umount the **/dev/vdb1** file system since a file system check cannot be performed on a mounted file system.

   ```
   [root@servera ~]# umount /mnt/etc_restore
   ```

   1.2. Use the **xfs_repair** command with the **-n** option to perform the file system check with no corrective actions taken.

   ```
   [root@servera ~]# xfs_repair -n /dev/vdb1
   Phase 1 - find and verify superblock...
   Phase 2 - using internal log
           - scan filesystem freespace and inode maps...
           - found root inode chunk
   Phase 3 - for each AG...
           - scan (but don't clear) agi unlinked lists...
           - process known inodes and perform inode discovery...
           - agno = 0
           - agno = 1
   Invalid inode number 0x499602d2
   xfs_dir_ino_validate: XFS_ERROR_REPORT
   Metadata corruption detected at block 0xa7e0/0x1000
   entry "subscription-manager" at block 0 offset 456 in directory inode 144973
    references invalid inode 1234567890
           would clear inode number in entry at offset 456...
   ```

```
        - process newly discovered inodes...
Phase 4 - check for duplicate blocks...
        - setting up duplicate extent list...
        - check for inodes claiming duplicate blocks...
        - agno = 0
        - agno = 1
entry "subscription-manager" at block 0 offset 456 in directory inode 144973
 references invalid inode 1234567890
        would clear inode number in entry at offset 456...
No modify flag set, skipping phase 5
Phase 6 - check inode connectivity...
        - traversing filesystem ...
entry "subscription-manager" in directory inode 144973 points to non-existent
 inode 1234567890, would junk entry
bad hash table for directory inode 144973 (no data entry): would rebuild
        - traversal finished ...
        - moving disconnected inodes to lost+found ...
disconnected inode 145282, would move to lost+found
Phase 7 - verify link counts...
No modify flag set, skipping filesystem flush and exiting.
```

Note the errors reported as they present information that may be helpful for file system recovery.

2. Use the **xfs_repair** utility to repair the file system inconsistencies.

```
[root@servera ~]# xfs_repair /dev/vdb1
Phase 1 - find and verify superblock...
Phase 2 - using internal log
        - zero log...
        - scan filesystem freespace and inode maps...
        - found root inode chunk
Phase 3 - for each AG...
        - scan and clear agi unlinked lists...
        - process known inodes and perform inode discovery...
        - agno = 0
        - agno = 1
Invalid inode number 0x499602d2
xfs_dir_ino_validate: XFS_ERROR_REPORT
Metadata corruption detected at block 0xa7e0/0x1000
entry "subscription-manager" at block 0 offset 456 in directory inode 144973
 references invalid inode 1234567890
        clearing inode number in entry at offset 456...
entry at block 0 offset 456 in directory inode 144973 has illegal name "/
ubscription-manager":         - process newly discovered i
nodes...
Phase 4 - check for duplicate blocks...
        - setting up duplicate extent list...
        - check for inodes claiming duplicate blocks...
        - agno = 0
        - agno = 1
Phase 5 - rebuild AG headers and trees...
        - reset superblock...
Phase 6 - check inode connectivity...
        - resetting contents of realtime bitmap and summary inodes
        - traversing filesystem ...
bad hash table for directory inode 144973 (no data entry): rebuilding
rebuilding directory inode 144973
        - traversal finished ...
        - moving disconnected inodes to lost+found ...
disconnected inode 145282, moving to lost+found
Phase 7 - verify and correct link counts...
```

```
done
```

3. Mount the repaired file system at **/mnt/etc_restore** so the results of the file system repair can be analyzed.

```
[root@servera ~]# mount /dev/vdb1 /mnt/etc_restore
```

4. Determine if the file system repair deposited any files in the file system's **lost+found** directory.

```
[root@servera ~]# ls -la /mnt/etc_restore/lost+found
total 4
drwxr-xr-x. 2 root root 19 Jan 19 23:07 .
drwxr-xr-x. 4 root root 33 Jan 19 22:59 ..
-rw-r--r--. 1 root root 62 Oct 13 11:00 145282
```

5. Determine the correct name and location of the orphaned file in **lost+found** using the output previously generated during the file system repair.

   5.1. Based on output generated during the file system repair, **subscription-manager** should be the name of the orphaned file.

   ```
   ... Output omitted ...
   Invalid inode number 0x499602d2
   xfs_dir_ino_validate: XFS_ERROR_REPORT
   Metadata corruption detected at block 0x258e0/0x1000
   entry "subscription-manager" at block 0 offset 456 in directory inode 144973
    references invalid inode 1234567890
           clearing inode number in entry at offset 456...
   entry at block 0 offset 456 in directory inode 144973 has illegal name "/
   ubscription-manager":         - process newly discovered i
   nodes...
   ... Output omitted ...
   ```

   5.2. Based on output generated during the file system repair, determine the directory name for the **subscription-manager** file. Use the **find** command to locate the directory with inode number **144973**.

   ```
   ... Output omitted ...
   Invalid inode number 0x499602d2
   xfs_dir_ino_validate: XFS_ERROR_REPORT
   Metadata corruption detected at block 0x258e0/0x1000
   entry "subscription-manager" at block 0 offset 456 in directory inode 144973
    references invalid inode 1234567890
           clearing inode number in entry at offset 456...
   entry at block 0 offset 456 in directory inode 144973 has illegal name "/
   ubscription-manager":         - process newly discovered i
   nodes...
   ... Output omitted ...
   ```

   ```
   [root@servera ~]# find /mnt/etc_restore -inum 144973
   /mnt/etc_restore/etc/security/console.apps
   ```

6. Utilize the backup of the file system's content located at **/root/etc.tgz** to validate the proper name and the location of the orphaned file.

   6.1. Extract the backup file so that its contents can be used for comparison.

   ```
   [root@servera ~]# tar -C /tmp -xzf /root/etc.tgz
   ```

   6.2. Verify the contents of the orphaned file against the backup copy of the identified file.

   ```
   [root@servera ~]# diff -s /tmp/etc/security/console.apps/subscription-manager /
   mnt/etc_restore/lost+found/145282
   Files /tmp/etc/security/console.apps/subscription-manager and /mnt/etc_restore/
   lost+found/145282 are identical
   ```

7. Once validated, restore the orphaned file back to its proper location.

   ```
   [root@servera ~]# mv /mnt/etc_restore/lost+found/145282 /mnt/etc_restore/etc/
   security/console.apps/subscription-manager
   ```

8. Grade your work, then clean up your systems for the next exercise.

   8.1. Grade your work.

   ```
   [student@workstation ~]$ lab xfs grade
   ```

   8.2. Clean up your systems for the next exercise.

   ```
   [student@workstation ~]$ lab xfs reset
   ```

Recovering from LVM Accidents

Objectives

After completing this section, students should be able to revert storage administration mistakes make on LVM volumes.

Logical volume management

Logical volume management (LVM) provides administrators with a powerful storage virtualization framework. One or more data bearers (physical volumes) get aggregated into a storage pool (volume group), from which volumes can get carved to act as block devices (logical volumes). Logical volume management also allows data to be striped and/or mirrored between physical volumes.

Just like **dm-multipath**, logical volume management uses the kernel *Device Mapper* subsystem to create the LVM devices. Logical volumes are created in **/dev/** as **dm-*** device nodes, but with symlinks in both **/dev/mapper** and **/dev/*<vgname>*/** using more administrator-friendly (and persistent) names.

Configuration files

The behavior of LVM is configured in **/etc/lvm/lvm.conf**. This file has settings to control locking behavior, device scanning, and naming of multipathed physical volumes, as well as other LVM features. The followin table lists some of the more common options:

/etc/lvm/lvm.conf Options	
dir	Which directory to scan for physical volume device nodes.
obtain_device_list_from_udev	If **udev** should be used to obtain a list of block devices eligible for scanning.
preferred_names	A list of regular expressions detailing which device names should have preference when displaying devices. If multiple device names (nodes) are found for the same PV, the LVM tools will give preference to those that appear earlier in this list.
filter	A list of regular expressions prefixed with **a** for *Add* or **r** for *Remove*. This list determines which device nodes will be scanned for the presence of a PV signature. The default is **["a/.*/"]**, which adds every single device. This option can be used to remove device that should never be scanned.
backup	This setting determines if a text-based backup of volume group metadata should be stored after every change. This backup can be used to restore a volume group if the on-disk metadata gets corrupted.

/etc/lvm/lvm.conf Options	
backup_dir	This setting specifies where the backup of volume group metadata should be stored.
archive	This setting determines if old volume group configurations/layouts should be archived so they can later be used to revert changes.
archive_dir	This setting specifies where archives of old configurations should be stored.
retain_min	This setting specifies the minimum number of archives to store.
retain_days	This setting specifies the minimum number of days for archive files to be kept.

Reverting LVM changes

In the course of routine storage administration, administrators may make a change to a volume group that they immediately regret, e.g., shrinking a logical volume before shrinking the file system on that logical volume. In this shrinking example, the administrator can try to extend the logical volume to the previous size, but there is no guarantee that the same blocks on the disk will be used as before, possibly resulting in corruption of the file system on the resized logical volume.

LVM's archive feature can be of tremendous help in this kind of situation. LVM can keep archived copies of volume group metadata. If the **archive** option is set to **1** in **/etc/lvm/lvm.conf**, the LVM tools will create an archived copy of the current volume group metadata *before* making any changes on disk. In a default configuration, these archives can be found in **/etc/lvm/archive**. The list of all archived metadata for a volume group can be displayed by examining the files in **/etc/lvm/archive** and paying special attention to the **description** lines. Alternatively, the same information can be displayed with the **vgcfgrestore -l *<vgname>*** command.

```
[root@demo ~]# vgcfgrestore -l vg_example
File:   /etc/lvm/archive/vg_example_00000-943759797.vg
VG name:      vg_example
Description:  Created *before* executing 'vgcreate vg_example /dev/sda5'
Backup Time:  Mon Jan 19 07:01:47 2016


File:   /etc/lvm/archive/vg_example_00001-1528176681.vg
VG name:      vg_example
Description:  Created *before* executing 'lvcreate -L 1G -n lv_example vg_example'
Backup Time:  Mon Jan 19 07:02:00 2016


File:   /etc/lvm/archive/vg_example_00002-1484695080.vg
VG name:      vg_example
Description:  Created *before* executing 'lvresize -L -256M /dev/vg_example/lv_example'
Backup Time:  Mon Jan 19 07:02:34 2016


File:   /etc/lvm/backup/vg_example
VG name:      vg_example
Description:  Created *after* executing 'lvresize -L -256M /dev/vg_example/lv_example'
Backup Time:  Mon Jan 19 07:02:34 2016
```

In the previous example, the output shows the sequence of recent actions performed on the **vg_example** volume group. First, the volume group **vg_example** is created, then a logical volume **lv_example** is created in the volume group. Lastly, the logical volume is reduced by **256** MiB. The last block in the output shows the backup of the current volume group metadata in **/etc/lvm/backup**, created *after* the logical volume had been shrunk.

To undo an LVM change, make sure that the logical volume is not currently in use (unmount any file systems that might have been created on the logical volume), and then use the **vgcfgrestore** command again, but this time with the **-f *<archive_file>*** option.

```
[root@demo ~]# vgcfgrestore -f /etc/lvm/archive/vg_example_00001-1528176681.vg
 vg_example
Restored volume group vg_example
```

In some cases, it might be necessary to deactivate and then reactivate the logical volume to make sure that all changes are committed in memory as well.

```
[root@demo ~]# lvchange -an /dev/vg_example/lv_example
[root@demo ~]# lvchange -ay /dev/vg_example/lv_example
```

References

Red Hat Storage Administration Guide

- Section 4: LVM

For more information, see the

> Red Hat Enterprise Linux 7 Logical Volume Manager Administration Guide
> https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Logical_Volume_Manager_Administration/

- Section 2: LVM Components
- Section 4.3.12: Backing Up Volume Group Metadata
- Section 6.4: Recovering Physical Volume Metadata

lvm.conf(5), **vgbackup**(8), and **vgcfgrestore**(8) manual pages

Guided Exercise: Recovering from LVM Accidents

In this lab, you will troubleshoot and resolve an LVM issue.

Resources	
Files	/mnt/lvm
Machines	**servera**

Outcome(s)

You should be able to identify and revert the cause of an LVM failure.

Before you begin

Prepare your system for this exercise by running the command **lab lvm setup** on your **workstation** machine.

```
[student@workstation ~#$ lab lvm setup
```

A request was recently received for an additional 20 MiB of storage to be allocated for use by the directory, **/mnt/lvm**, on **servera**. After the request was completed, the users report that the directory is no longer accessible. The users think that the critical files were somehow accidentally deleted by the administrator who fulfilled the request. The problem has been escalated and you are asked to investigate and identify the root cause of the problem. Once root cause analysis has been completed, attempt to restore the system to proper working order.

1. Look in the **/mnt/lvm** directory to confirm the problem reported by the user.

   ```
   [root@servera ~]# ll /mnt/lvm
   total 0
   ```

2. Check the contents of the **/etc/fstab** file to determine whether the directory contents are mounted from a separate file system.

   ```
   [root@servera ~]# grep /mnt/lvm /etc/fstab
   /dev/vg00/lvol0      /mnt/lvm     xfs     defaults     0 0
   ```

3. Verify that the file system on the logical volume is mounted at **/mnt/lvm**.

   ```
   [root@servera ~]# mount | grep /mnt/lvm
   ```

4. Since the file system is not mounted, attempt to mount it manually.

   ```
   [root@servera ~]# mount /mnt/lvm
   mount: /dev/mapper/vg00-lvol0: can't read superblock
   ```

5. Consult the list of metadata backup and archive files for the volume group on which the **/dev/vg00/lvol0** logical volume resides to determine what changes, if any, have been made to the logical volume configuration.

```
[root@servera ~]# vgcfgrestore -l vg00
 ... Output omitted ...

  File:         /etc/lvm/archive/vg00_00002-1999295985.vg
  VG name:      vg00
  Description:  Created *before* executing 'lvresize -L20M /dev/vg00/lvol0'
  Backup Time:  Wed Jan 20 02:43:40 2016


  File:         /etc/lvm/backup/vg00
  VG name:      vg00
  Description:  Created *after* executing 'lvresize -L20M /dev/vg00/lvol0'
  Backup Time:  Wed Jan 20 02:43:40 2016

 ... Output omitted ...
```

6. Analyze the descriptions of each archive log and identify the archive that existed prior to the space allocation on the logical volume. Determine whether the command recorded in the archive log caused the issue observed by the user.

```

 ... Output omitted ...

  File:         /etc/lvm/archive/vg00_00002-1999295985.vg
  VG name:      vg00
  Description:  Created *before* executing 'lvresize -L20M /dev/vg00/lvol0'
  Backup Time:  Wed Jan 20 02:43:40 2016

 ... Output omitted ...
```

Rather than allocating another 20 MiB of space to the logical volume with the **-L+20M** option, the administrator erroneously reduced the logical volume to 20 MiB.

7. Revert the incorrect action that was executed during the attempt to allocate space to the logical volume.

```
[root@servera ~]# vgcfgrestore -f /etc/lvm/archive/vg00_00002-1999295985.vg vg00
  Restored volume group vg00
```

8. Deactivate and reactivate the **/dev/vg00/lvol0** logical volume to ensure everything is updated.

```
[root@servera ~]# lvchange -an /dev/vg00/lvol0
[root@servera ~]# lvchange -ay /dev/vg00/lvol0
```

9. Mount the file system to determine if the issue has been resolved.

```
[root@servera ~]# mount /mnt/lvm
[root@servera ~]# ll /mnt/lvm
total 12
```

```
drwxr-xr-x. 94 root root     8192 Jan 19 22:26 etc
```

10. Correctly fulfill the user's request by allocating another 20 MiB of space to the **/dev/vg00/lvol0** file system.

```
[root@servera ~]# lvresize -L+20M /dev/vg00/lvol0
  Size of logical volume vg00/lvol0 changed from 40.00 MiB (10 extents) to 60.00 MiB
 (15 extents).
  Logical volume lvol0 successfully resized.
[root@servera ~]# xfs_growfs /dev/vg00/lvol0
meta-data=/dev/mapper/vg00-lvol0 isize=256    agcount=2, agsize=5120 blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=0        finobt=0
data     =                       bsize=4096   blocks=10240, imaxpct=25
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0 ftype=0
log      =internal               bsize=4096   blocks=853, version=2
         =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
data blocks changed from 10240 to 15360
```

11. Grade your work, then clean up your systems for the next exercise.

    11.1. Grade your work.

    ```
    [student@workstation ~]$ lab lvm grade
    ```

    11.2. Clean up your systems for the next exercise.

    ```
    [student@workstation ~]$ lab lvm reset
    ```

Dealing with LUKS Issues

Objectives

After completing this section, students should be able to recover data from encrypted file systems.

Using encrypted volumes

Linux Unified Key Setup (LUKS) is the standard format for device encryption in Red Hat Enterprise Linux. A LUKS-encrypted partition or volume must be decrypted before the file system in it can be mounted.

The persistent mounting of encrypted volume that takes place during system boot consists of the following steps.

1. The **`/etc/crypttab`** file is consulted for a list of devices to be unlocked during boot. The file lists one device per line, with the following space-separated fields:

   ```
   name  /dev/vdaN  /path/to/keyfile
   ```

   where **name** is the name the device mapper will use for the decrypted device (**`/dev/mapper/name`**, **`/dev/vdaN`** is the underlying "locked" device, and **`/path/to/keyfile`** is the key file to use to unlock the device. If the last field is left blank (or set to none), the user will be prompted for the decryption password during startup.

2. Using the passphrase supplied by the user or using a key contained in a file, the encrypted volume is decrypted and mapped to a device mapper device named /dev/mapper/**name**, using the **name** specified in **`/etc/crypttab`**.

3. If the encrypted volume is successfully decrypted, the decrypted mapped device will be created and its contents will then be available for access. The file system contained in the decrypted mapped device is mounted based on the relevant entry in **`/etc/fstab`**. These entries can refer to the contents on the decrypted volume either by the device mapper name or by its UUID.

   ```
   /dev/mapper/name  /secret  ext4  defaults  1 2
   ```

   ```
   UUID="2460ab6e-e869-4011-acae-31b2e8c05a3b"  /secret  ext4  defaults  1 2
   ```

Troubleshooting LUKS volumes

Misconfiguration of `/etc/crypttab`

A common source of decryption issues is the misconfiguration of the **`/etc/crypttab`** configuration file. The entries in this file determine how encrypted volumes are decrypted and also how they are to be mapped after decryption.

The first field identifies the name of the mapped device without the **`/dev/mapper`** prefix. The second field corresponds to the encrypted device. The third field of the entries can either be left blank for passphrase prompting or specify a key file to be used for decryption.

If decryption is failing, verify that the contents of the second and third fields are correct. An incorrect encrypted device name and an incorrect key file will both result in decryption failures.

If decryption is successful but the mounting of the mapped device is failing, verify that the contents of the first field for the relevant entry is correct. If there is a mismatch between the mapped name specified in **/etc/crypttab** and the corresponding device mapper name in **/etc/fstab**, the decrypted content will not be available at the configured mount point.

Decryption failure

Aside from the previously mentioned misconfigurations in **/etc/crypttab**, decryption of an encrypted device can also occur in the following situations.

- LUKS passphrase has been forgotten or has been changed to an unknown password. This is possibly the most frequently encountered scenario.

  LUKS offers a total of eight key slots for encrypted devices. If other keys or passphrases exist, they can be used to reset the forgotten or unknown password. Associated keys can be displayed using the **cryptsetup luksDump** command.

  ```
  [root@demo ~]# cryptsetup luksDump /dev/vdb1
  LUKS header information for /dev/vdb1

  Version:        1
  Cipher name:    aes
  Cipher mode:    xts-plain64
  Hash spec:      sha1
  Payload offset: 4096
  MK bits:        256
  MK digest:      78 1a 67 57 7f b6 4e a3 60 40 fb 0b 76 91 ee 96 21 2e 09 d4
  MK salt:        bd c2 45 9d 30 32 4e 69 aa 6b a8 90 33 c5 f1 6a
                  4a f4 c7 fc e8 7f 21 ae a4 3b a2 91 74 dc f8 9e
  MK iterations:  81000
  UUID:           3d1d164c-569c-4809-ba6e-9b3de75ed223

  Key Slot 0: ENABLED
          Iterations:             324049
          Salt:                   30 95 87 15 61 c5 96 87 af 55 cd 9d 4a d0 59 22
                                  61 12 89 9f 59 e5 80 1f ae f9 03 1d c2 01 46 44
          Key material offset:    8
          AF stripes:             4000
  Key Slot 1: DISABLED
  Key Slot 2: DISABLED
  Key Slot 3: DISABLED
  Key Slot 4: DISABLED
  Key Slot 5: DISABLED
  Key Slot 6: DISABLED
  Key Slot 7: DISABLED
  ```

  If a backup of the LUKS header exists, the issue can also be resolved by restoring the header from backup, which will allow decryption using the previously working password.

- LUKS header has been corrupted. LUKS stores a metadata header and key slots at the beginning of each encrypted device. Therefore, a corruption of the LUKS header can render the encrypted data inaccessible. If a backup of the corrupted LUKS header exists, the issue can be resolved by restoring the header from backup.

- The hash functions and ciphers needed to execute the decryption are not in the kernel. A list of installed cryptographic ciphers supported by the kernel on a system can be determined by examining the contents of **`/proc/crypto`**.

`/etc/fstab` misconfiguration
The decrypted mapped device is configured for mounting in **`/etc/fstab`**. An incorrectly specified device name will lead to mounting failure. A common mistake is to configure the encrypted device, rather than the mapped device, for mounting.

Restoring LUKS headers

For some commonly encountered LUKS issues, LUKS header backups can mean the difference between a simple administrative fix and permanently unrecoverable data. Therefore, administrators of LUKS-encrypted volumes should engage in the good practice of routinely backing up their headers. In addition, they should be familiar with the procedures for restoring the headers from backup should the need arise.

LUKS header backup

LUKS header backups are performed using the **`cryptsetup`** tool in conjunction with the **`luksHeaderBackup`** subcommand. The location of the backup file is specified with the **`--header-backup-file`** option.

```
[root@demo ~]# cryptsetup luksHeaderBackup /path/to/encrypted/device --header-backup-
file /path/to/backup/file
```

As with all system administration tasks, LUKS header backups should be made before each and every administrative task performed on a LUKS-encrypted volume.

Testing and recovering LUKS headers

If an encrypted volume's LUKS header has been backed up, the backups can be restored to the volume to overcome issues such as forgotten passwords or corrupted headers. Before performing a header restoration, as a good practice, administrators should back up the existing header even if the header may be corrupt or is associated with an unknown password.

If multiple backups exist for an encrypted volume, an administrator needs to identify the proper one to restore. Rather than restoring the header and then attempt a decryption, administrator should conduct trial decryption with a header file first. This can be accomplished using the following command.

```
[root@demo ~]# cryptsetup luksOpen /path/to/encrypted/device --header /path/to/backup/
file
Enter passphrase for /path/to/encrypted/device:
```

If the test succeeds, the header can then be restored using the **`cryptsetup`** command along with the **`luksHeaderRestore`** command as demonstrated in the following example.

```
[root@demo ~]# cryptsetup luksHeaderRestore /path/to/encrypted/device --header-backup-
file /path/to/backup/file

WARNING!
========
Device /path/to/encrypted/device already contains LUKS header. Replacing header will
 destroy existing keyslots.
```

```
Are you sure? (Type uppercase yes): YES
```

References

cryptsetup(8) man page

> Red Hat Enterprise Linux 7 Security Guide: Using LUKS Disk Encryption
> https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Security_Guide

> All about LUKS, cryptsetup, and dm-crypt
> https://access.redhat.com/articles/193443

> How to recover lost LUKS key or passphrase
> https://access.redhat.com/solutions/1543373

> Cryptsetup project page
> https://gitlab.com/groups/cryptsetup

Guided Exercise: Dealing with LUKS Issues

In this lab, you will troubleshoot and resolve problems with mounting an encrypted file system.

Resources	
Files	/root/luks_header_backup
Machines	• **servera**

Outcome(s)

You should be able to troubleshoot and recover from a failure to mount an encrypted volume.

Before you begin

Prepare your systems for this exercise by running the command **lab luks setup** on your **workstation** machine.

```
[student@workstation ~#$ lab luks setup
```

A LUKS-encrypted volume mounted at **/mnt/secure** on **servera** was protected by the password **RedHatR0cks!**. To comply with the company's password policy, a junior administrator changed the password to the new password, **JustMyLUK**. However, after the password change, the junior administrator reports that the new password is not working on the encrypted volume. Fortunately, a backup of the LUKS header for the volume was recently made and is available at **/root/luks_header_backup**. Assess the situation and restore the volume to working order. Use the LUKS header backup if necessary to restore the encrypted volume's function.

1. Determine the system's encrypted volume configuration.

   1.1. Consult **/etc/fstab** to determine the volume configured to mount at **/mnt/secure**.

   ```
   [root@servera ~]# grep /mnt/secure /etc/fstab
   /dev/mapper/secure     /mnt/secure     xfs     defaults     1 2
   ```

   1.2. Consult **/etc/crypttab** to determine how encrypted devices are set up on the system at boot time.

   ```
   [root@servera ~]# cat /etc/crypttab
   secure /dev/vdb1 none
   ```

2. Assess the current state of the encrypted volume. Use the **dmsetup** command to determine if a mapped device exists for the encrypted block device. Use the **--target crypt** to filter for encrypted block device only.

   ```
   [root@servera ~]# dmsetup ls --target crypt
   No devices found
   ```

3. Since the mapped device does not exist, use the **cryptsetup luksOpen** command to manually open the encrypted volume and create the associated mapped device specified in **/etc/crypttab**. Try decrypting the encrypted volume with the new password set by the junior administrator. If that does not work, try the old password.

```
[root@servera ~]# cryptsetup luksOpen /dev/vdb1 secure
Enter passphrase for /dev/vdb1: JustMyLUK
No key available with this passphrase.
Enter passphrase for /dev/vdb1: JustMyLUK
No key available with this passphrase.
Enter passphrase for /dev/vdb1: JustMyLUK
No key available with this passphrase.
```

4. Use the **cryptsetup luksDump** to determine if any other keys exist for the encrypted volume to determine if there are other possible passwords that can be tried.

```
[root@servera ~]# cryptsetup luksDump /dev/vdb1
LUKS header information for /dev/vdb1

Version:        1
Cipher name:    aes
Cipher mode:    xts-plain64
Hash spec:      sha1
Payload offset: 4096
MK bits:        256
MK digest:      d1 db 30 00 8d f4 f9 7a 4c 9f eb 7a 8c 2a 3a 0f 30 e0 75 d9
MK salt:        c8 ca d5 90 5a f7 cd 5d b9 35 4e c5 83 b1 b4 f7
                47 e3 4e b8 5b 36 bb f7 67 25 61 61 8e e5 9f de
MK iterations:  81250
UUID:           b4b499bb-be72-4912-b9f6-0d23970aba6b

Key Slot 0: DISABLED
Key Slot 1: ENABLED
        Iterations:             324049
        Salt:                   19 55 1a fa 00 ee 3f af fa 38 6d 18 90 c8 79 67
                                39 ed ef cd cd 3c a1 7a f0 3f 24 05 6e fa 26 66
        Key material offset:    264
        AF stripes:             4000
Key Slot 2: DISABLED
Key Slot 3: DISABLED
Key Slot 4: DISABLED
Key Slot 5: DISABLED
Key Slot 6: DISABLED
Key Slot 7: DISABLED
```

5. Since neither the new nor old passwords are able to decrypt the encrypted volume and there are no other keys, the only option is to reinstate the old password by restoring the LUKS header from the LUKS header backup file, **/root/luks_header_backup**.

   5.1. Since it is good practice to make administrative changes reversible, make a backup of the current LUKS header prior to replacing it with the backup LUKS header.

   ```
   [root@servera ~]# cryptsetup luksHeaderBackup /dev/vdb1 --header-backup-file /
   root/luks_header_new
   ```

   5.2. Restore the old LUKS header from the **/root/luks_header_backup** backup file to reinstate the old password.

   ```
   [root@servera ~]# cryptsetup luksHeaderRestore /dev/vdb1 --header-backup-file /
   root/luks_header_backup
   ```

```
WARNING!
========
Device /dev/vdb1 already contains LUKS header. Replacing header will destroy
 existing keyslots.

Are you sure? (Type uppercase yes): YES
```

6. Use the **cryptsetup luksOpen** command once again to attempt to manually open the encrypted volume and create the associated mapped device specified in **/etc/crypttab**. Try decrypting the encrypted volume with the old password.

```
[root@servera ~]# cryptsetup luksOpen /dev/vdb1 secure
Enter passphrase for /dev/vdb1: RedHatR0cks!
```

7. Use the **dmsetup** command to validate that the **secure** mapped device now exists for the encrypted block device. Use the **--target crypt** to filter for encrypted block device only.

```
[root@servera ~]# dmsetup ls --target crypt
secure  (252, 0)
```

8. Verify that the mapped device for the encrypted volume can now be successfully mounted at **/mnt/secure**.

```
[root@servera ~]# mount /mnt/secure
[root@servera ~]# mount | grep /mnt/secure
/dev/mapper/secure on /mnt/secure type xfs (rw)
```

9. Grade your work, then clean up your systems for the next exercise.

   9.1. Grade your work.

   ```
   [student@workstation ~]$ lab luks grade
   ```

   9.2. Clean up your systems for the next exercise.

   ```
   [student@workstation ~]$ lab luks reset
   ```

Resolving iSCSI Issues

Objectives

After completing this section, students should be able to identify and fix iSCSI issues.

Configuring iSCSI initiators

The Internet Small Computer System Interface (iSCSI) is a TCP/IP-based protocol for emulating a SCSI high-performance local storage bus over IP networks, providing data transfer and management to remote block storage devices. As a storage area network (SAN) protocol, iSCSI extends SANs across local and wide area networks (LANs, WANs, and the Internet), providing location-independent data storage retrieval with distributed servers and arrays.

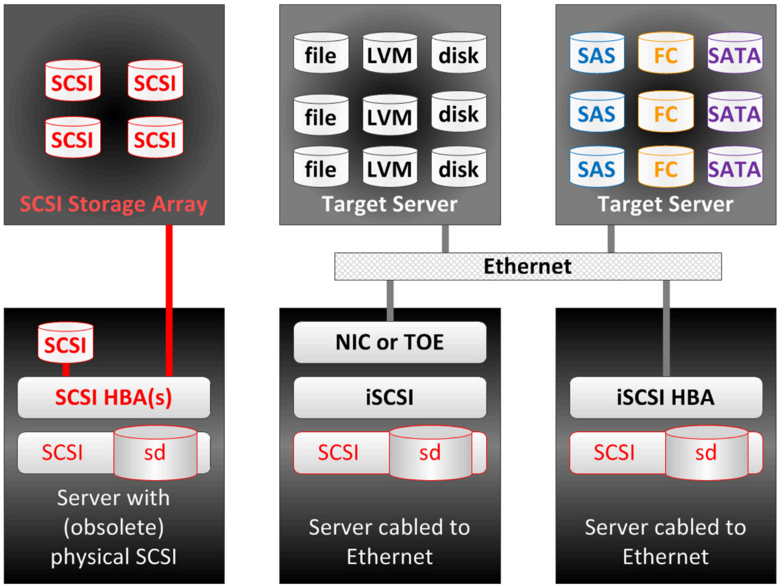


Figure 5.4: SCSI and iSCSI block storage topologies

iSCSI Component Terminology

Term	Description
`initiator`	An iSCSI client, typically available as software but also implemented as iSCSI HBAs. Initiators must be given unique names (see **`IQN`**).
`target`	An iSCSI storage resource, configured for connection from an iSCSI server. Targets must be given unique names (see **`IQN`**). A target provides one or more numbered block devices called logical units (see **`LUN`**). An iSCSI server can provide many targets concurrently.
`ACL`	An access control list (entry), an access restriction using the node IQN (commonly the iSCSI initiator Name) to validate access permissions for an initiator.
`discovery`	Querying a target server to list configured targets. Target use requires an additional access step (see **`login`**).
`IQN`	An iSCSI qualified name, a worldwide unique name used to identify both initiators and targets, in the mandated naming format:

Term	Description
	iqn.YYYY-MM.com.reversed.domain[:optional_string] **iqn**—Signifying that this name will use a domain as its identifier. **YYYY-MM**—The first month in which the domain name was owned. **com.reversed.domain**—The reversed domain name of the organization creating this iSCSI name. **optional_string**—An optional, colon-prefixed string assigned by the domain owner as desired while remaining worldwide unique. It may include colons to separate organization boundaries.
login	Authenticating to a target or LUN to begin client block device use.
LUN	A logical unit number, numbered block devices attached to and available through a target. One or more LUNs may be attached to a single target, although typically a target provides only one LUN.
node	Any iSCSI initiator or iSCSI target, identified by its IQN.
portal	An IP address and port on a target or initiator used to establish connections. Some iSCSI implementations use *portal* and *node* interchangeably.
TPG	Target portal group, the set of interface IP addresses and TCP ports to which a specific iSCSI target will listen. Target configuration (e.g., ACLs) can be added to the TPG to coordinate settings for multiple LUNs.

Configuring iSCSI initiators

The iSCSI protocol functions in a familiar client-server configuration. Client systems configure *initiator* software to send SCSI commands to remote server storage *targets*. Accessed iSCSI targets appear on the client system as local, unformatted SCSI block devices, identical to devices connected with SCSI cabling, FC direct attached, or FC switched fabric.

In Red Hat Enterprise Linux an iSCSI initiator is typically implemented in software, and functions similar to a hardware iSCSI HBA to access targets from a remote storage server. Using a software-based iSCSI initiator requires connecting to an existing Ethernet network of sufficient bandwidth to carry the expected storage traffic.

iSCSI can also be implemented using a hardware initiator that includes the required protocols in a dedicated host bus adapter. iSCSI HBAs and TCP offload engines (TOE), which include the TCP network stack on an Ethernet NIC, move the processing of iSCSI or TCP overhead and Ethernet interrupts to hardware, easing the load on system CPUs.

Configuring an iSCSI client initiator requires installing the *iscsi-initiator-utils* package, which includes the **iscsi** and **iscsid** services and the **/etc/iscsi/iscsid.conf** and **/etc/iscsi/initiatorname.iscsi** configuration files.

As an iSCSI node, the client requires a unique IQN. The default **/etc/iscsi/initiatorname.iscsi** file contains a generated IQN using Red Hat's domain. Administrators typically reset the IQN to their own domain and an appropriate client system string.

The **/etc/iscsi/iscsid.conf** file contains default settings for node records created during new target discovery. Settings include iSCSI timeouts, retry parameters, and authentication usernames and passwords. Changing this file requires restarting the **iscsi** service.

```
[root@serverX ~]# systemctl restart iscsi
```

Connecting to iSCSI targets

To be able to discover targets, the client must have functional network connectivity to the target host. The *iscsi-initiator-utils* package needs to be installed and the **`iscsi`** service needs to be started and enabled. Targets must be discovered before a device can be connected to and used. The discovery and logging in of targets are performed using the **`iscsiadm`** utility provided by the *iscsi-initiator-utils* package.

Authentication options

iSCSI uses ACLs to perform *LUN masking*, managing the accessibility of appropriate targets and LUNs to initiators. Access to targets may also be limited with CHAP authentication. iSCSI ACLs are similar to FC's use of device worldwide numbers (WWNs) for *soft zoning* management restrictions. Although FC switch-level compulsory port restriction (*hard zoning*) has no comparable iSCSI mechanism, Ethernet VLANs could provide similar isolation security.

Storage area network traffic is typically unencrypted, since physical server-to-storage cabling is normally enclosed within secure data centers. For WAN security, iSCSI and Fibre Channel over Ethernet (FCoE) can utilize Internet Protocol Security (IPsec), a protocol suite for securing IP network traffic.

iSCSI offers Challenge-Handshake Authentication Protocol (CHAP) usernames and passwords as an authentication mechanism to limit connectivity between chosen initiators and targets. By default, the iSCSI initiator in Red Hat Enterprise Linux is configured for no authentication. The port number can be omitted when the target server is configured on default port 3260. Therefore, if enhanced data security over the network is desired, CHAP authentication must be configured on the initiator.

If password authentication has been enabled and configured on the target, then matching authentication must be configured on the initiator. Settings are located in **`/etc/iscsi/iscsid.conf`**. For one-way authentication that verifies the initiator to the target server, use the **`username`** and **`password`** credentials. For two-way authentication that additionally verifies the target portal to the initiator, the **`username_in`** and **`password_in`** credentials are also required.

- For *discovery* authentication, use these parameters:
  discovery.sendtargets.auth.authmethod = CHAP
  discovery.sendtargets.auth.username = *username*
  discovery.sendtargets.auth.password = *password*
  discovery.sendtargets.auth.username_in = *username_in*
  discovery.sendtargets.auth.password_in = *password_in*

- For *normal* authentication, set these parameters:
  node.session.auth.authmethod = CHAP
  node.session.auth.username = *username*
  node.session.auth.password = *password*
  node.session.auth.username_in = *username_in*
  node.session.auth.password_in = *password_in*

- For *no* authentication, clear all credential values and recomment the **`authmethod`** lines.

Troubleshooting iSCSI initiator issues

Since iSCSI is a network-based protocol, iSCSI initiator issues can arise due to issues or misconfiguration at the network layer. A successful network connection to the iSCSI target host is required before its targets can be discovered by the iSCSI initiator. Therefore, troubleshooting of discovery issues should begin with a networking assessment.

Identifying and resolving network issues

Network troubleshooting should begin with testing basic network connectivity from the initiator to the target host. To ensure connections are being made to the proper target host, begin first by ensuring that the name resolution is working properly.

Validate that the destination of connections initiated by **iscsiadm** resolve correctly to the target host's IP address. If it does not, use **dig** to perform name resolution using DNS. If problems with DNS resolution has been eliminated, look in **/etc/hosts** and **/etc/nsswitch.conf** to see if their configurations are the source of the name resolution issues.

By default, target servers are configured to listen on port 3260. If name resolution is working properly, test making a connection to this port using a utility such as Netcat.

```
[root@serverX ~]# nc -v target_server_ip 3260
```

By default, target servers are configured to listen on port 3260. If the connection fails, confirm that the configurations on the target server are correct. Ensure that the target server is indeed configured to listen on the default port and that **firewalld** has been configured for remote connections to the port.

If the networking component is no longer the cause of iSCSI issues, an initiator should be able to perform a discovery of targets on the iSCSI server. Discovery of targets can be assessed with the following command. The port number can be omitted when the target server is configured on default port 3260.

```
[root@serverX ~]# iscsiadm -m discovery -t sendtargets -p target_server[:port]
172.25.X.11:3260,1 iqn.2016-01.com.example.lab:iscsistorage
```

To see if targets have already been successfully discovered, execute the following command. The command will report back a list of known nodes along with their associated portal address and port.

```
[root@serverX ~]# iscsiadm -m node
172.25.250.254:3260,1 iqn.2016-01.com.example.lab:iscsistorage
```

Identifying and resolving login issues

If discovery is successful but the initiator experiences an issue logging into the discovered target, then the problem is likely related to access control or authentication. Access control to iSCSI targets are granted on the iSCSI server by specifying the names of initiators that should be allowed access. The initiator name is configured on the client in **/etc/iscsi/initiatorname.iscsi**. A mismatch between the client's initiator name and that configured on the server will result in a target login failure. If the issue is due to an incorrect initiator name configuration on the client, be sure to restart the **iscsid** service after the initiator name is fixed in **/etc/iscsi/initiatorname.iscsi** in order for the change to take effect.

```
[root@serverX ~]# systemctl restart iscsid
```

If the initiator name is correctly configured, then login failures may be attributed to authentication misconfiguration. In order to authenticate successfully, the initiator must be configured to use the authentication method required by the target server. By default, no authentication method is configured on the server or the client. If the server is configured for CHAP authentication, then the initiator's CHAP configurations must be specified in **/etc/iscsi/iscsid.conf** as previously mentioned.

To assess the cause for login failures, it may be useful to execute the login in verbose mode. The **-d** option enables debug mode. Debug level can be set between 0 to 8. Level 8 is required in order to see authentication details.

```
[root@serverX ~]# iscsiadm -m node -T iqn.2016-01.com.example.lab:iscsistorage --login -d8
... Output omitted ...
iscsiadm: updated 'node.session.auth.authmethod', 'None' => 'CHAP'
... Output omitted ...
```

If the issue is due to an incorrect CHAP configuration on the client, be sure to restart the **iscsid** service after changes are made to **/etc/iscsi/iscsid.conf** in order for the changes to take effect.

```
[root@serverX ~]# systemctl restart iscsid
```

The discovery process stores target node information and settings in the **/var/lib/iscsi/nodes** directory, using defaults from **/etc/iscsi/iscsid.conf**. Since the same target can exist on multiple portals, node records are stored for each portal. Upon discovery, a node record is written to **/var/lib/iscsi/nodes** and used for subsequent logins.

When changes are made to **/etc/iscsi/iscsid.conf** after the initial login attempt, it is important to remember that the login process will continue to use settings saved in **/var/lib/iscsi/nodes**. In order to log in using newly configured settings, administrators can either modify existing settings stored under **/var/lib/iscsi/nodes** or remove all saved information from that directory.

Existing settings for a target can be displayed using the following command.

```
[root@serverX ~]# iscsiadm -m node -T iqn.2016-01.com.example.lab:iscsistorage
# BEGIN RECORD 6.2.0.873-30
node.name = iqn.2016-01.com.example.lab:iscsistorage
node.tpgt = 1
... Output omitted ...
node.conn[0].iscsi.HeaderDigest = None
node.conn[0].iscsi.IFMarker = No
node.conn[0].iscsi.OFMarker = No
```

To modify existing settings, use the **iscsiadm** command. Use the **-o** option to specify the **update** operation. Also use the **-n** option to specify the parameter to modify and use the **-v** option to specify the value for the parameter. For example, to change the authentication method to disable CHAP authentication, execute the following command.

```
[root@serverX ~]# iscsiadm -m node -T iqn.2016-01.com.example.lab:iscsistorage -o update
 -n node.session.auth.authmethod -v None [-p target_server[:port]]
```

Alternatively, administrators can also choose to purge all node information from cache. To purge existing node settings for a target, use the **iscsiadm** command along with the **-o** option to specify the **delete** operation.

```
[root@serverX ~]# iscsiadm -m node -T iqn.2016-01.com.example.lab:iscsistorage -o delete
 [-p target_server[:port]]
```

Settings for all known nodes can be purged by using the same command and not specifying a target.

```
[root@serverX ~]# iscsiadm -m node -o delete [-p target_server[:port]]
```

References

iscsiadm(8) man page

> Red Hat Enterprise Linux 7 Storage Administration Guide: Create an iSCSI Initiator
> https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Storage_Administration_Guide/index.html

Guided Exercise: Resolving iSCSI Issues

In this lab, you will troubleshoot and resolve issues with an iSCSI initiator.

Resources	
Machines	• **servera**

Outcome(s)

You should be able to identify and resolve a misconfiguration of an iSCSI initiator on a client system.

Before you begin

Prepare your systems for this exercise by running the command **lab iscsi setup** on your **workstation** machine.

```
[student@workstation ~#$ lab iscsi setup
```

The **workstation** system has been configured with the iSCSI target, **iqn.2016-01.com.example.lab:iscsistorage**. The target has been configured with an ACL that allows access for an initiator named **iqn.2016-01.com.example.lab:servera**. The target has been configured so that authentication by the initiator is not required.

A fellow administrator was tasked with configuring the **servera** to use the **iqn.2016-01.com.example.lab:iscsistorage** target on **workstation**. The administrator has reported that while she can discover the target, she is unable to log in successfully. Troubleshoot, identify, and correct the issue so that the initiator can successfully log into the target.

1. Begin by validating the reported issue.

   1.1. Validate whether the administrator successfully discovered the target. The following command will list all node records.

   ```
   [root@servera ~]# iscsiadm -m node
   172.25.250.254:3260,1 iqn.2016-01.com.example.lab:iscsistorage
   ```

   1.2. Validate whether any active sessions or connections exist. Since the administrator reported a failure to log into the target, no active sessions are expected.

   ```
   [root@servera ~]# iscsiadm -m session
   iscsiadm: No active sessions.
   ```

   1.3. Attempt a login to the target to verify what was reported by the administrator and also to see the nature of the errors generated.

   ```
   [root@servera ~]# iscsiadm -m node -T iqn.2016-01.com.example.lab:iscsistorage
    --login
   Logging in to [iface: default, target: iqn.2016-01.com.example.lab:iscsistorage,
    portal: 172.25.250.254,3260] (multiple)
   iscsiadm: Could not login to [iface: default, target:
    iqn.2016-01.com.example.lab:iscsistorage, portal: 172.25.250.254,3260].
   ```

```
iscsiadm: initiator reported error (24 - iSCSI login failed due to authorization
 failure)
iscsiadm: Could not log into all portals
```

2. Using the **-d** option with the **iscsiadm** command, attempt to log into the target again, but also generate verbose debug information. Since the error messages are related to authorization failure, look for debug messages that correspond to this.

```
[root@servera ~]# iscsiadm -m node -T iqn.2016-01.com.example.lab:iscsistorage --
login -d8
... Output omitted ...
iscsiadm: updated 'node.session.queue_depth', '32' => '32'
iscsiadm: updated 'node.session.nr_sessions', '1' => '1'
iscsiadm: updated 'node.session.auth.authmethod', 'None' => 'CHAP'
iscsiadm: updated 'node.session.timeo.replacement_timeout', '120' => '120'
iscsiadm: updated 'node.session.err_timeo.abort_timeout', '15' => '15'
... Output omitted ...
```

3. The debug messages indicate that the initiator is configured to use CHAP authentication. Since the target does not require authentication, the initiator's authorization method is misconfigured. Correct the misconfiguration in **/etc/iscsi/iscsid.conf** by commenting out the following line and then restart **iscsid** for the changes to be read.

```
# node.session.auth.authmethod = CHAP
```

```
[root@servera ~]# systemctl restart iscsid
```

4. Reattempt a login to the target to verify whether the login issue has been resolved.

```
[root@servera ~]# iscsiadm -m node -T iqn.2016-01.com.example.lab:iscsistorage --
login
Logging in to [iface: default, target: iqn.2016-01.com.example.lab:iscsistorage,
 portal: 172.25.250.254,3260] (multiple)
iscsiadm: Could not login to [iface: default, target:
 iqn.2016-01.com.example.lab:iscsistorage, portal: 172.25.250.254,3260].
iscsiadm: initiator reported error (24 - iSCSI login failed due to authorization
 failure)
iscsiadm: Could not log into all portals
```

5. Since the failure still persists, use the **-d** option with the **iscsiadm** command and attempt to log into the target again, but also generate verbose debug information. Since the error messages are related to authorization failure, look for debug messages that correspond to this.

```
[root@servera ~]# iscsiadm -m node -T iqn.2016-01.com.example.lab:iscsistorage --
login -d8
... Output omitted ...
iscsiadm: updated 'node.session.queue_depth', '32' => '32'
iscsiadm: updated 'node.session.nr_sessions', '1' => '1'
iscsiadm: updated 'node.session.auth.authmethod', 'None' => 'CHAP'
iscsiadm: updated 'node.session.timeo.replacement_timeout', '120' => '120'
iscsiadm: updated 'node.session.err_timeo.abort_timeout', '15' => '15'
... Output omitted ...
```

6. The debug messages indicate that the initiator is still using CHAP authentication to log into the target. This is due to the fact that the authorization method used in previous login attempts has been saved for the target. Use the following **iscsiadm** command to change the authorization method to **None**.

```
[root@servera ~]# iscsiadm -m node -T iqn.2016-01.com.example.lab:iscsistorage -o
 update -n node.session.auth.authmethod -v None
```

7. Reattempt a login to the target to verify whether the login issue has been resolved.

```
[root@servera ~]# iscsiadm -m node -T iqn.2016-01.com.example.lab:iscsistorage --
login
Logging in to [iface: default, target: iqn.2016-01.com.example.lab:iscsistorage,
 portal: 172.25.250.254,3260] (multiple)
iscsiadm: Could not login to [iface: default, target:
 iqn.2016-01.com.example.lab:iscsistorage, portal: 172.25.250.254,3260].
iscsiadm: initiator reported error (24 - iSCSI login failed due to authorization
 failure)
iscsiadm: Could not log into all portals
```

8. Since the failure still persists, use the **-d** option with the **iscsiadm** command and attempt to log into the target again, but also generate verbose debug information. Since the error messages are related to authorization failure, look for debug messages that correspond to this.

```
[root@servera ~]# iscsiadm -m node -T iqn.2016-01.com.example.lab:iscsistorage --
login -d8
... Output omitted ...
iscsiadm: updated 'node.session.queue_depth', '32' => '32'
iscsiadm: updated 'node.session.nr_sessions', '1' => '1'
iscsiadm: updated 'node.session.auth.authmethod', 'None' => 'None'
iscsiadm: updated 'node.session.timeo.replacement_timeout', '120' => '120'
iscsiadm: updated 'node.session.err_timeo.abort_timeout', '15' => '15'
... Output omitted ...
```

9. The debug messages indicate that the initiator is no longer using authentication to log into the target. With that issue removed, the failure to login to the target might be due to an ACL restriction. Check to see if the initiator name matches the **iqn.2016-01.com.example.lab:servera** name allowed in the ACL.

```
[root@servera ~]# cat /etc/iscsi/initiatorname.iscsi
InitiatorName=iqn.2016-01.lab.example.com:servera
```

10. The initiator name is incorrectly configured. Fix the name and then restart **iscsid** for the change to take effect.

```
[root@servera ~]# cat /etc/iscsi/initiatorname.iscsi
InitiatorName=iqn.2016-01.com.example.lab:servera
[root@servera ~]# systemctl restart iscsid
```

11. Reattempt a login to the target to verify whether the login issue has been resolved.

```
[root@servera ~]# iscsiadm -m node -T iqn.2016-01.com.example.lab:iscsistorage --
login
Logging in to [iface: default, target: iqn.2016-01.com.example.lab:iscsistorage,
 portal: 172.25.250.254,3260] (multiple)
Login to [iface: default, target: iqn.2016-01.com.example.lab:iscsistorage, portal:
 172.25.250.254,3260] successful.
```

12. Grade your work, then clean up your systems for the next exercise.

    12.1. Grade your work.

    ```
    [student@workstation ~]$ lab iscsi grade
    ```

13. Reset all of your systems to provide a clean enviropnment for the next exercises.

Lab: Troubleshooting Storage Issues

In this lab, you will troubleshoot and resolve storage issues related to encryption, file system, LVM, and iSCSI.

Resources	
Files	/mnt/save/luks/iscsistorage_luks_header

Outcome(s)

You should be able to troubleshoot and repair issues with corrupted file systems, LUKS headers, LVM administration, and iSCSI targets.

Before you begin

Prepare your systems for this exercise by running the command **lab storage setup** on your **workstation** machine.

```
[student@workstation ~#$ lab storage setup
```

The CFO has requested access to the company's financial data. The data resides in an encrypted volume on the iSCSI target, **iqn.2016-01.com.example.lab:iscsistorage**, provided by **workstation**. The target does not require authentication and has been configured with an ACL granting access to the **iqn.2016-01.com.example.lab:servera** initiator.

An administrator working on this request has run into an issue accessing the encrypted volume. Help him resolve this issue and then decrypt the volume using the last known password of **RedHatR0cks!**. Make the volume available as the **/dev/mapper/finance** mapped device mounted at the **/mnt/finance** mount point on **servera**.

If you have trouble decrypting the encrypted volume, fortunately, a backup of the encrypted volume's header was made and stored on the **/dev/save/old** LVM logical volume on **servera**. Mount the logical volume at **/mnt/save** on **servera** and the LUKS header backup file can be located at **/mnt/save/luks/iscsistorage_luks_header**. If, for whatever reason, the file is not accessible, troubleshoot the issue and then restore the file to that location.

1. On **servera**, assess the situation by generating a list of iSCSI active sessions and known nodes.

2. If no sessions or known nodes exist, attempt to perform a discovery of targets on **workstation**. If any issues are encountered, troubleshoot and resolve them so the discovery succeeds.

3. Once discovery of the iSCSI target on **workstation** succeeds, log into the target. If any issues are encountered, troubleshoot and resolve them so the login succeeds.

4. Once login to the target is successful, decrypt the encrypted volume using the last known password.

5. Since the decryption failed, mount the **/dev/save/old** LVM logical volume to **/mnt/save** to access the LUKS header dump file. If any issues are encountered with the logical volume, troubleshoot and resolve them so the logical volume is available.

6. Locate and verify the LUKS header backup file located at **/mnt/save/luks/iscsistorage_luks_header**. If any issues are encountered verifying the file, troubleshoot and resolve them so the file is available at the specified location.

7. Using the LUKS header backup located at **/mnt/save/luks/iscsistorage_luks_header**, restore the LUKS header to the encrypted volume located on the **/dev/sda1** partition.

8. Decrypt the encrypted volume using the last known password and map it to **/dev/mapper/finance**.

9. Make the contents of the decrypted volume accessible at the **/mnt/finance** mount point.

10. Grade your work, then clean up your systems for the next exercise.

    10.1. Grade your work.

    ```
    [student@workstation ~]$ lab storage grade
    ```

11. Reset all of your machines to provide a clean environment for the next exercises.

Solution

In this lab, you will troubleshoot and resolve storage issues related to encryption, file system, LVM, and iSCSI.

Resources	
Files	/mnt/save/luks/iscsistorage_luks_header

Outcome(s)

You should be able to troubleshoot and repair issues with corrupted file systems, LUKS headers, LVM administration, and iSCSI targets.

Before you begin

Prepare your systems for this exercise by running the command **lab storage setup** on your **workstation** machine.

```
[student@workstation ~#$ lab storage setup
```

The CFO has requested access to the company's financial data. The data resides in an encrypted volume on the iSCSI target, **iqn.2016-01.com.example.lab:iscsistorage**, provided by **workstation**. The target does not require authentication and has been configured with an ACL granting access to the **iqn.2016-01.com.example.lab:servera** initiator.

An administrator working on this request has run into an issue accessing the encrypted volume. Help him resolve this issue and then decrypt the volume using the last known password of **RedHatR0cks!**. Make the volume available as the **/dev/mapper/finance** mapped device mounted at the **/mnt/finance** mount point on **servera**.

If you have trouble decrypting the encrypted volume, fortunately, a backup of the encrypted volume's header was made and stored on the **/dev/save/old** LVM logical volume on **servera**. Mount the logical volume at **/mnt/save** on **servera** and the LUKS header backup file can be located at **/mnt/save/luks/iscsistorage_luks_header**. If, for whatever reason, the file is not accessible, troubleshoot the issue and then restore the file to that location.

1. On **servera**, assess the situation by generating a list of iSCSI active sessions and known nodes.

```
[root@servera ~]# iscsiadm -m session
iscsiadm: No active sessions.
[root@servera ~]# iscsiadm -m node
iscsiadm: No records found
```

2. If no sessions or known nodes exist, attempt to perform a discovery of targets on **workstation**. If any issues are encountered, troubleshoot and resolve them so the discovery succeeds.

   2.1. Perform a discovery of the targets on **workstation**.

```
[root@servera ~]# iscsiadm -m discovery -t st -p workstation.lab.example.com
iscsiadm: cannot make connection to 172.25.254.254: Connection refused
iscsiadm: cannot make connection to 172.25.254.254: Connection refused
iscsiadm: cannot make connection to 172.25.254.254: Connection refused
iscsiadm: cannot make connection to 172.25.254.254: Connection refused
iscsiadm: cannot make connection to 172.25.254.254: Connection refused
```

```
iscsiadm: cannot make connection to 172.25.254.254: Connection refused
iscsiadm: connection login retries (reopen_max) 5 exceeded
iscsiadm: Could not perform SendTargets discovery: encountered connection
 failure
```

2.2. Verify that the address resolved for **workstation.lab.example.com** during the discovery is correct.

```
[root@servera ~]# dig +short workstation.lab.example.com
172.25.250.254
```

2.3. Since the address does not match that provided by DNS name resolution, check **/etc/hosts** to determine if it is the source of the problem.

```
[root@servera ~]# grep workstation /etc/hosts
172.25.254.254  workstation.lab.example.com workstation
172.25.254.254  workstation.lab.example.com workstation
```

2.4. Fix the erroneous entries in **/etc/hosts** to correct the name resolution issue causing the connectivity issues to **workstation.lab.example.com**.

```
[root@servera ~]# grep workstation /etc/hosts
172.25.250.254  workstation.lab.example.com workstation
172.25.250.254  workstation.lab.example.com workstation
```

2.5. Reattempt a discovery of the targets on **workstation**.

```
[root@servera ~]# iscsiadm -m discovery -t st -p workstation.lab.example.com
172.25.250.254:3260,1 iqn.2016-01.com.example.lab:iscsistorage
```

3. Once discovery of the iSCSI target on **workstation** succeeds, log into the target. If any issues are encountered, troubleshoot and resolve them so the login succeeds.

3.1. Log into the target on **workstation**.

```
[root@servera ~]# iscsiadm -m node -T iqn.2016-01.com.example.lab:iscsistorage
 --login
Logging in to [iface: default, target: iqn.2016-01.com.example.lab:iscsistorage,
 portal: 172.25.250.254,3260] (multiple)
iscsiadm: Could not login to [iface: default, target:
 iqn.2016-01.com.example.lab:iscsistorage, portal: 172.25.250.254,3260].
iscsiadm: initiator reported error (24 - iSCSI login failed due to authorization
 failure)
iscsiadm: Could not log into all portals
```

3.2. Verify the authentication method configured on the initiator to ensure no authentication is configured.

```
[root@servera ~]# iscsiadm -m node -T iqn.2016-01.com.example.lab:iscsistorage |
 grep authmethod
node.session.auth.authmethod = None
```

3.3. Since the authentication method is correctly configured, verify that the authentication failure is not due to ACL restriction by checking that the initiator name matches that granted access by the target's ACL.

```
[root@servera ~]# cat /etc/iscsi/initiatorname.iscsi
InitiatorName=iqn.com.example.lab:servera
```

3.4. Fix the erroneous initiator name in **/etc/iscsi/initiatorname.iscsi** to correct the target login name resolution issue preventing the authentication to the target.

```
[root@servera ~]# cat /etc/iscsi/initiatorname.iscsi
InitiatorName=iqn.2016-01.com.example.lab:servera
```

3.5. Restart **iscsid** for the change to the initiator name to take effect.

```
[root@servera ~]# systemctl restart iscsid
```

3.6. Reattempt the login to the target.

```
[root@servera ~]# iscsiadm -m node -T iqn.2016-01.com.example.lab:iscsistorage
 --login
Logging in to [iface: default, target: iqn.2016-01.com.example.lab:iscsistorage,
 portal: 172.25.250.254,3260] (multiple)
Login to [iface: default, target: iqn.2016-01.com.example.lab:iscsistorage,
 portal: 172.25.250.254,3260] successful.
```

4. Once login to the target is successful, decrypt the encrypted volume using the last known password.

```
[root@servera ~]# cryptsetup luksOpen /dev/sda1 finance
Enter passphrase for /dev/sda1:
No key available with this passphrase.
```

5. Since the decryption failed, mount the **/dev/save/old** LVM logical volume to **/mnt/save** to access the LUKS header dump file. If any issues are encountered with the logical volume, troubleshoot and resolve them so the logical volume is available.

5.1. Mount the **/dev/save/old** logical volume.

```
[root@servera ~]# mkdir /mnt/save
[root@servera ~]# mount /dev/save/old /mnt/save
mount: special device /dev/save/old does not exist
```

5.2. Scan for the **/dev/save/old** logical volume.

```
[root@servera ~]# lvs
  LV   VG   Attr       LSize  Pool Origin Data%  Meta%  Move Log Cpy%Sync
 Convert
  new  save -wi-a----- 16.00m
```

5.3. Since the logical volume is missing, investigate the LVM archive log to determine the reason for its absence.

```
[root@servera ~]# vgcfgrestore -l save

  File:         /etc/lvm/archive/save_00000-1431778654.vg
  Couldn't find device with uuid eYfFIc-HUFc-NxnQ-w0xq-QmnE-j6DC-36y0UY.
  VG name:      save
  Description:  Created *before* executing 'vgcreate save /dev/vdb1'
  Backup Time:  Fri Jan 22 13:34:33 2016


  File:         /etc/lvm/archive/save_00001-884435814.vg
  VG name:      save
  Description:  Created *before* executing 'lvcreate -W y -L 15M -n old save'
  Backup Time:  Fri Jan 22 13:34:33 2016


  File:         /etc/lvm/archive/save_00002-527329952.vg
  VG name:      save
  Description:  Created *before* executing 'lvcreate -W y -L 15M -n new save'
  Backup Time:  Fri Jan 22 13:34:33 2016


  File:         /etc/lvm/archive/save_00003-600513272.vg
  VG name:      save
  Description:  Created *before* executing 'lvremove -f /dev/save/old'
  Backup Time:  Fri Jan 22 13:34:33 2016


  File:         /etc/lvm/backup/save
  VG name:      save
  Description:  Created *after* executing 'lvremove -f /dev/save/old'
  Backup Time:  Fri Jan 22 13:34:33 2016
```

5.4. Revert the removal of the logical volume and then mount the volume to **/mnt/save**.

```
[root@servera ~]# vgcfgrestore -f /etc/lvm/archive/save_00003-600513272.vg save
  Restored volume group save
[root@servera ~]# lvs
  LV   VG   Attr       LSize  Pool Origin Data%  Meta%  Move Log Cpy%Sync
 Convert
  new  save -wi-a----- 16.00m

  old  save -wi------- 16.00m
[root@servera ~]# lvchange -an /dev/save/old
[root@servera ~]# lvchange -ay /dev/save/old
[root@servera ~]# mount /dev/save/old /mnt/save
```

6. Locate and verify the LUKS header backup file located at **/mnt/save/luks/iscsistorage_luks_header**. If any issues are encountered verifying the file, troubleshoot and resolve them so the file is available at the specified location.

6.1. Look for and verify the **/mnt/save/luks/iscsistorage_luks_header** file.

```
[root@servera ~]# ls -la /mnt/save/luks
ls: cannot access /mnt/save/luks/iscsistorage_luks_header: Invalid argument
total 0
```

```
drwxr-xr-x. 2 root root 37 Jan 21 18:57 .
drwxr-xr-x. 5 root root 40 Jan 21 18:56 ..
??????????? ? ?    ?     ?            ? iscsistorage_luks_header
[root@servera ~]# file /mnt/save/luks/iscsistorage_luks_header
/mnt/save/luks/iscsistorage_luks_header: cannot open (Invalid argument)
```

6.2. Determine the type of file system formatted on the logical volume and run a file system check.

```
[root@servera ~]# blkid /dev/save/old
/dev/save/old: UUID="c878808f-3c8e-45a3-abc1-0559694e5410" TYPE="xfs"
[root@servera ~]# umount /dev/save/old
[root@servera ~]# xfs_repair -n /dev/save/old
Phase 1 - find and verify superblock...
Only one AG detected - cannot validate filesystem geometry.
Use the -o force_geometry option to proceed.
[root@servera ~]# xfs_repair -n -o force_geometry /dev/save/old
Phase 1 - find and verify superblock...
Phase 2 - using internal log
        - scan filesystem freespace and inode maps...
        - found root inode chunk
Phase 3 - for each AG...
        - scan (but don't clear) agi unlinked lists...
        - process known inodes and perform inode discovery...
        - agno = 0
entry "iscsistorage_luks_header" in shortform directory 13800 references invalid
 inode 1234567890
would have junked entry "iscsistorage_luks_header" in directory inode 13800
        - process newly discovered inodes...
Phase 4 - check for duplicate blocks...
        - setting up duplicate extent list...
        - check for inodes claiming duplicate blocks...
        - agno = 0
entry "iscsistorage_luks_header" in shortform directory 13800 references invalid
 inode 1234567890
would have junked entry "iscsistorage_luks_header" in directory inode 13800
No modify flag set, skipping phase 5
Phase 6 - check inode connectivity...
        - traversing filesystem ...
        - traversal finished ...
        - moving disconnected inodes to lost+found ...
disconnected inode 13801, would move to lost+found
Phase 7 - verify link counts...
No modify flag set, skipping filesystem flush and exiting.
```

6.3. Repair the XFS file system.

```
[root@servera ~]# xfs_repair -o force_geometry /dev/save/old
Phase 1 - find and verify superblock...
Phase 2 - using internal log
        - zero log...
        - scan filesystem freespace and inode maps...
        - found root inode chunk
Phase 3 - for each AG...
        - scan and clear agi unlinked lists...
        - process known inodes and perform inode discovery...
        - agno = 0
entry "iscsistorage_luks_header" in shortform directory 13800 references invalid
 inode 1234567890
junking entry "iscsistorage_luks_header" in directory inode 13800
        - process newly discovered inodes...
```

```
Phase 4 - check for duplicate blocks...
        - setting up duplicate extent list...
        - check for inodes claiming duplicate blocks...
        - agno = 0
Phase 5 - rebuild AG headers and trees...
        - reset superblock...
Phase 6 - check inode connectivity...
        - resetting contents of realtime bitmap and summary inodes
        - traversing filesystem ...
        - traversal finished ...
        - moving disconnected inodes to lost+found ...
disconnected inode 13801, moving to lost+found
Phase 7 - verify and correct link counts...
done
  Restored volume group save
```

6.4. Mount the repaired file system. Based on the file system repair report, we expect to find the **iscsistorage_luks_header** file in the **lost+found** directory. Restore it to the **/mnt/save/luks** directory.

```
[root@servera ~]# mount /dev/save/old /mnt/save
[root@servera ~]# ls -la /mnt/save/lost+found/
ls -la /mnt/save/lost+found/
total 1028
drwxr-xr-x. 2 root root      18 Jan 22 19:33 .
drwxr-xr-x. 6 root root      57 Jan 21 18:56 ..
-rw-r--r--. 1 root root 1052672 Jan 21 18:57 13801
[root@servera ~]# file /mnt/save/lost+found/13801
/mnt/save/lost+found/13801: LUKS encrypted file, ver 1 [aes, xts-plain64, sha1]
 UUID: b91a11a8-1bf1-4c9a-9f31-3cc2e8947476
[root@servera ~]# mv /mnt/save/lost+found/13801 /mnt/save/luks/
iscsistorage_luks_header
```

7. Using the LUKS header backup located at **/mnt/save/luks/iscsistorage_luks_header**, restore the LUKS header to the encrypted volume located on the **/dev/sda1** partition.

```
[root@servera ~]# cryptsetup luksHeaderRestore /dev/sda1 --header-backup-file /mnt/
save/luks/iscsistorage_luks_header

WARNING!
========
Device /dev/sda1 already contains LUKS header. Replacing header will destroy
 existing keyslots.

Are you sure? (Type uppercase yes): YES
```

8. Decrypt the encrypted volume using the last known password and map it to **/dev/mapper/finance**.

```
[root@servera ~]# cryptsetup luksOpen /dev/sda1 finance
Enter passphrase for /dev/sda1:
```

9. Make the contents of the decrypted volume accessible at the **/mnt/finance** mount point.

```
[root@servera ~]# mkdir /mnt/finance
```

```
[root@servera ~]# mount /dev/mapper/finance /mnt/finance/
[root@servera ~]# ls -la /mnt/finance
total 0
drwxr-xr-x. 2 root root 6 Jan 21 14:57 accounts
drwxr-xr-x. 2 root root 6 Jan 21 14:57 customers
drwxr-xr-x. 2 root root 6 Jan 21 14:58 employees
drwxr-xr-x. 2 root root 6 Jan 21 14:57 loans
drwxr-xr-x. 2 root root 6 Jan 21 14:58 management
drwxr-xr-x. 2 root root 6 Jan 21 14:57 shareholders
```

10. Grade your work, then clean up your systems for the next exercise.

    10.1. Grade your work.

    ```
    [student@workstation ~]$ lab storage grade
    ```

11. Reset all of your machines to provide a clean environment for the next exercises.

Summary

In this chapter, you learned:

- ext3 and ext4 file systems can be checked and repaired using the **`e2fsck`** command.
- XFS file systems can be checked and repaired using the **`xfs_repair`** command.
- LVM archives are kept under **`/etc/lvm/archives`**.
- LVM administration mistakes can be reverted using the **`vgcfgrestore`** command.
- LUKS volume headers can be backed up and restored using the **`cryptsetup`** command.
- LUKS volume decryption failures related to unknown passwords and corrupt headers can be resolved by restoring the LUKS header to the LUKS volume.
- Issues with iSCSI connections to the target server are typically caused by networking issues if discovery of targets does not succeed.
- Issues with iSCSI connections to the target server are typically caused by ACL or authentication issues if discovery of targets succeeds.

CHAPTER 6

TROUBLESHOOTING RPM ISSUES

Overview	
Goal	Identify and fix problems in, and using, the package management subsystem.
Objectives	• Resolve package management dependency issues manually. • Recover a corrupted Red Hat Package Management (RPM) database. • Identify and restore changed files using the package management subsystem. • Employ the Red Hat subscription management tools to receive updates.
Sections	• Resolving Dependency Issues (and Guided Exercise) • Recovering a Corrupt RPM Database (and Guided Exercise) • Identifying and Recovering Changed Files (and Guided Exercise) • Subscribing Systems to Red Hat Updates (and Quiz)
Lab	• Troubleshooting RPM Issues

Resolving Dependency Issues

Objectives

After completing this section, students should be able to identify and resolve package dependency issues.

Package dependencies

Package dependencies occur when an RPM package requires functionality provided by other RPM packages. Such package requirements are included in the metadata of each RPM package. The **rpm** command does not resolve these dependencies. The **yum** command resolves dependencies and uses **rpm** to install or erase a package.

Despite Yum's best attempts, sometimes there are RPM dependency issues that it will not resolve by default. One example of this can be seen when installing a package that requires a specific version of another package, and a different, incompatible version of the required package is already installed on the system.

When diagnosing RPM dependency issues, the system logs are not very helpful. In addition to the RPM database, the **/var/log/yum.log** log contains a history of installed and erased packages. There is not much detailed information from RPM to see. Invoking the **rpm** command with the **-v** option causes it to display more diagnostic information to standard error when it runs. Additional **-v** options will make **rpm** even more verbose.

Displaying package dependencies

The **yum deplist** command displays a list of dependencies of the package passed as an argument to the command. It also prints the name and the latest version of the package available for installation that satisfies each dependency.

```
[root@demo ~]# yum deplist yum
Loaded plugins: langpacks, search-disabled-repos, versionlock
package: yum.noarch 3.4.3-132.el7
  dependency: /bin/bash
   provider: bash.x86_64 4.2.46-19.el7
  dependency: /usr/bin/python
   provider: python.x86_64 2.7.5-34.el7
  dependency: cpio
   provider: cpio.x86_64 2.11-24.el7
  dependency: diffutils
   provider: diffutils.x86_64 3.3-4.el7
... Output omitted ...
```

The **rpm** command also can be used to display useful package dependency information. The **--requires** option, or **-R** for short, displays the requirements for the package passed as a parameter. It does not display as much information as **yum deplist**.

```
[root@demo ~]# rpm -q --requires yum
/bin/bash
/usr/bin/python
config(yum) = 3.4.3-132.el7
cpio
diffutils
pygpgme
```

```
pyliblzma
python >= 2.4
python(abi) = 2.7
```

The **rpm -q --provides** command displays the list of potential requirements that a package provides for other packages.

```
[root@demo ~]# rpm -q --provides yum
config(yum) = 3.4.3-132.el7
yum = 3.4.3-132.el7
yum-allow-downgrade = 1.1.20-0.yum
yum-basearchonly = 1.1.9.yum
yum-plugin-allow-downgrade = 1.1.22-0.yum
yum-plugin-basearchonly = 1.1.9.yum
yum-plugin-downloadonly = 3.4.3-44.yum
```

Resolving package dependencies

Yum has a **downgrade** subcommand that downgrades the package specified as the argument to the previous version of the currently installed version. It erases the newer version of the package and installs the previous version in its place. This command can downgrade to an earlier version of a package if the specific version was specified. It works for packages that only allow one version of a package to be installed on the system at a time. It does not work for "install only" packages, such as the Linux kernel.

> **Note**
>
> If the package being downgraded is a prerequisite for other installed packages on the system, the dependent packages may also be specified and downgraded as well.

RPM provides similar functionality with the **--oldpackage** option. This option is used with the **rpm -U** command. It tells RPM upgrade to install the older version of an installed package.

The **rpm -U --force** command is the same as using the **--oldpackage**, **--replacepkgs**, and **--replacefiles** options together. The **--replacepkgs** option tells RPM to install the specified packages, even if some of them are already installed on the system. The **--replacefiles** option will install the specified packages, even if they replace files from other, already installed, packages.

Using Yum to lock package versions

Yum has a plug-in that allows an administrator to lock down a package (or group of packages) to a specific version. The *yum-plugin-versionlock* package provides that enhanced functionality. Once this package is installed, the **yum** command will support a new **versionlock** subcommand.

Command	Function
yum versionlock list	Display the list of locked package versions.
yum versionlock add *WILDCARD*	Lock the current versions of the packages matched by the wildcard.
yum versionlock delete *WILDCARD*	Delete the locks matched by the wildcard.
yum versionlock clear	Clear all package version locks.

Once a specific version of a package is locked, Yum will not attempt to update the package when the **yum update** command runs.

The command **yum list --showduplicates** can be used to find all available versions of a package.

R

References

rpm(8), **yum**(8), and **yum-versionlock**(1) man pages

Guided Exercise: Resolving Dependency Issues

In this lab, you will resolve a package dependency issue, then lock the prerequisite package to the specific version needed.

Resources	
Machines	• **workstation** • **servera**

Outcome(s)

You should be able to resolve package dependency issues.

Before you begin

Set up your systems for this exercise by running the command **lab package-dependencies setup** on your **workstation** system.

```
[student@workstation ~#$ lab package-dependencies setup
```

The security team has updated their system requirements for hosts in the datacenter. Part of that update requires a custom package, *rht-main*, to be installed on every host.

One of your colleagues attempted to install *rht-main* on **servera** without success. You have been called in to solve this problem and get the package installed.

1. Use Yum to install the *rht-main* package. Check the error messages for helpful clues.

```
[root@servera ~]# yum -y install rht-main
Loaded plugins: langpacks, search-disabled-repos
Resolving Dependencies
--> Running transaction check
---> Package rht-main.noarch 0:0.1-1.el7 will be installed
--> Processing Dependency: rht-prereq = 0.1 for package: rht-main-0.1-1.el7.noarch
--> Finished Dependency Resolution
Error: Package: rht-main-0.1-1.el7.noarch (rht_lab)
           Requires: rht-prereq = 0.1
           Installed: rht-prereq-0.2-1.el7.noarch (@rht_lab)
               rht-prereq = 0.2-1.el7
           Available: rht-prereq-0.1-1.el7.noarch (rht_lab)
               rht-prereq = 0.1-1.el7
 You could try using --skip-broken to work around the problem
 You could try running: rpm -Va --nofiles --nodigest
```

2. The *rht-main* package requires a specific version of the *rht-prereq* package to be installed, specifically version 0.1. A newer version has already been installed on the system. Use Yum to downgrade the package to the previous version.

```
[root@servera ~]# yum -y downgrade rht-prereq
Loaded plugins: langpacks, search-disabled-repos
Resolving Dependencies
--> Running transaction check
---> Package rht-prereq.noarch 0:0.1-1.el7 will be a downgrade
---> Package rht-prereq.noarch 0:0.2-1.el7 will be erased
```

```
... Output omitted ...
```

3. Now that the prerequisite software is installed, use Yum to install the required *rht-main* package.

```
[root@servera ~]# yum install -y rht-main
Loaded plugins: langpacks, search-disabled-repos
Resolving Dependencies
--> Running transaction check
---> Package rht-main.noarch 0:0.1-1.el7 will be installed
--> Finished Dependency Resolution
... Output omitted ...
Installed:
  rht-main.noarch 0:0.1-1.el7

Complete!
```

4. Now that the *rht-main* package is installed, try to update the prerequisite package, *rht-prereq*. Yum will try to update the package, but because of dependencies, it will generate error messages and fail.

```
[root@servera ~]# yum update rht-prereq
Loaded plugins: langpacks, search-disabled-repos
Resolving Dependencies
--> Running transaction check
---> Package rht-prereq.noarch 0:0.1-1.el7 will be updated
--> Processing Dependency: rht-prereq = 0.1 for package: rht-main-0.1-1.el7.noarch
---> Package rht-prereq.noarch 0:0.2-1.el7 will be an update
--> Finished Dependency Resolution
Error: Package: rht-main-0.1-1.el7.noarch (@rht_lab)
           Requires: rht-prereq = 0.1
           Removing: rht-prereq-0.1-1.el7.noarch (@rht_lab)
               rht-prereq = 0.1-1.el7
           Updated By: rht-prereq-0.2-1.el7.noarch (rht_lab)
               rht-prereq = 0.2-1.el7
 You could try using --skip-broken to work around the problem
 You could try running: rpm -Va --nofiles --nodigest
```

5. Yum has a plug-in that allows an administrator to lock a package down to a specific version. Use Yum to generate a list of available Yum plug-in packages.

```
[root@servera ~]# yum list available yum-plugin*
Loaded plugins: langpacks, search-disabled-repos
Available Packages
yum-plugin-aliases.noarch                   1.1.31-34.el7              rhel_dvd
yum-plugin-changelog.noarch                 1.1.31-34.el7              rhel_dvd
yum-plugin-ovl.noarch                       1.1.31-34.el7              rhel_dvd
yum-plugin-tmprepo.noarch                   1.1.31-34.el7              rhel_dvd
yum-plugin-verify.noarch                    1.1.31-34.el7              rhel_dvd
yum-plugin-versionlock.noarch               1.1.31-34.el7              rhel_dvd
```

The *yum-plugin-versionlock* package will provide the needed functionality. Use Yum to install it.

```
[root@servera ~]# yum -y install yum-plugin-versionlock
... Output omitted ...
```

```
Installed:
  yum-plugin-versionlock.noarch 0:1.1.31-34.el7

Complete!
```

6. The *yum-plugin-versionlock* package adds a **`versionlock`** subcommand to the **`yum`** command. Running **`yum versionlock list`** lists the packages that are currently locked. Initially no package versions are locked.

   ```
   [root@servera ~]# yum versionlock list
   Loaded plugins: langpacks, search-disabled-repos, versionlock
   versionlock list done
   ```

   Specify the *rht-prereq* package name as an argument to **`yum versionlock add`**. This will tell Yum to lock the current version of the package.

   ```
   [root@servera ~]# yum versionlock add rht-prereq
   Loaded plugins: langpacks, search-disabled-repos, versionlock
   Adding versionlock on: 0:rht-prereq-0.1-1.el7
   versionlock added: 1
   [root@servera ~]# yum versionlock list
   Loaded plugins: langpacks, search-disabled-repos, versionlock
   0:rht-prereq-0.1-1.el7.*
   versionlock list done
   ```

7. Try to update the *rht-prereq* package.

   ```
   [root@servera ~]# yum update rht-prereq
   Loaded plugins: langpacks, search-disabled-repos, versionlock
   No packages marked for update
   ```

   Yum will ignore it and no longer print error messages.

8. Once you are done, verify your work by running the **`lab package-dependencies grade`** command from your **`workstation`** machine.

   ```
   [student@workstation ~]$ lab package-dependencies grade
   ```

9. After grading your systems, successfully reset your machines to the state they had before starting the lab, either by resetting your virtual machines or by running the following command on your **`workstation`** system.

   ```
   [student@workstation ~]$ lab package-dependencies reset
   ```

Recovering a Corrupt RPM Database

Objectives

After completing this section, students should be able to identify and rebuild a corrupt RPM database.

Corrupt RPM database

When **yum** or **rpm** are used to install or erase packages on a Red Hat Enterprise Linux system, local RPM databases are updated with information about the packages being managed. The RPM database consists of a number of files in the **/var/lib/rpm/** directory. These are all Berkeley database files, except for the **__db.*** index files.

Occasionally the RPM database can get corrupted for a variety of reasons. Use the following steps to repair a corrupt database. Pay close attention to any error messages that appear.

1. Remove stale lock files and avoid additional lock issues by removing the database indexes.

```
[root@demo ~]# lsof | grep /var/lib/rpm
[root@demo ~]# rm /var/lib/rpm/__db*
rm: remove regular file `/var/lib/rpm/__db.001'? y
rm: remove regular file `/var/lib/rpm/__db.002'? y
rm: remove regular file `/var/lib/rpm/__db.003'? y
```

2. Make a backup of the RPM database files.

```
[root@demo ~]# tar cjvf rpmdb-$(date +%Y%m%d-%H%M).tar.bz2 /var/lib/rpm
tar: Removing leading `/' from member names
/var/lib/rpm/
/var/lib/rpm/.dbenv.lock
/var/lib/rpm/Packages
/var/lib/rpm/Name
/var/lib/rpm/Basenames
/var/lib/rpm/Group
/var/lib/rpm/Requirename
/var/lib/rpm/Providename
/var/lib/rpm/Conflictname
/var/lib/rpm/Obsoletename
/var/lib/rpm/Triggername
/var/lib/rpm/Dirnames
/var/lib/rpm/Installtid
/var/lib/rpm/Sigmd5
/var/lib/rpm/Sha1header
/var/lib/rpm/.rpm.lock
```

3. Verify the RPM database integrity.

```
[root@demo ~]# cd /var/lib/rpm
[root@demo rpm]# /usr/lib/rpm/rpmdb_verify Packages
rpmdb_verify: BDB0682 Page 32: bad prev_pgno 0 on overflow page (should be 31)
rpmdb_verify: BDB0683 Page 32: overflow item incomplete
rpmdb_verify: Packages: BDB0090 DB_VERIFY_BAD: Database verification failed
BDB5105 Verification of Packages failed.
```

4. If necessary, make a dump of the corrupt database and use it to create an intact database file.

```
[root@demo rpm]# mv Packages Packages.bad
[root@demo rpm]# /usr/lib/rpm/rpmdb_dump Packages.bad |
> /usr/lib/rpm/rpmdb_load Packages
[root@demo rpm]# /usr/lib/rpm/rpmdb_verify Packages
BDB5105 Verification of Packages succeeded.
```

5. Rebuild the RPM database indexes.

```
[root@demo rpm]# rpm -v --rebuilddb
error: rpmdbNextIterator: skipping h#       4 region trailer: BAD, tag 0 type 0
 offset 0 count 0
```

At this point, additional RPM operations should not produce error messages.

If the restored RPM database has a small subset of the original packages, consult **/var/log/yum.log** to identify the missing packages. The **rpm --justdb** command can be used to update the RPM database without making changes to the host's file system.

References

rpm(8) man page

For more information, see the

> RPM Database Recovery document
> http://www.rpm.org/wiki/Docs/RpmRecovery

Guided Exercise: Recovering a Corrupt RPM Database

In this lab, you will repair a corrupted RPM package database.

Resources	
Machines	• **workstation** • **servera**

Outcome(s)

You should be able to diagnose and repair a corrupted RPM package database.

Before you begin

Set up your systems for this exercise by running the **lab package-database setup** command on your **workstation** system.

```
[student@workstation ~#$ lab package-database setup
```

One of the junior system administrators was tasked with installing an Apache web server on one of the servers in your organization. When he began to install the *httpd* package with Yum, an error message appeared that caused him concern, so he stopped the installation and called you to help him figure out what was wrong with the system.

1. First, see if you can reproduce the error that the junior administrator saw. Begin a Yum installation of the *httpd* package, but gracefully decline the installation before Yum makes any changes to the system.

```
[root@servera ~]# yum install httpd
Loaded plugins: langpacks, search-disabled-repos
Resolving Dependencies
--> Running transaction check
---> Package httpd.x86_64 0:2.4.6-40.el7 will be installed
--> Processing Dependency: httpd-tools = 2.4.6-40.el7 for package:
 httpd-2.4.6-40.el7.x86_64
error: rpmdbNextIterator: skipping h#       4 region trailer: BAD, tag 0 type 0
 offset 0 count 0
--> Processing Dependency: /etc/mime.types for package: httpd-2.4.6-40.el7.x86_64
--> Processing Dependency: libapr-1.so.0()(64bit) for package:
 httpd-2.4.6-40.el7.x86_64
--> Processing Dependency: libaprutil-1.so.0()(64bit) for package:
 httpd-2.4.6-40.el7.x86_64
--> Running transaction check
---> Package apr.x86_64 0:1.4.8-3.el7 will be installed
---> Package apr-util.x86_64 0:1.5.2-6.el7 will be installed
---> Package httpd-tools.x86_64 0:2.4.6-40.el7 will be installed
---> Package mailcap.noarch 0:2.1.41-2.el7 will be installed
error: rpmdbNextIterator: skipping h#       4 region trailer: BAD, tag 0 type 0
 offset 0 count 0
--> Finished Dependency Resolution
... Output omitted ...
Is this ok [y/d/N]: n
Exiting on user command
Your transaction was saved, rerun it with:
 yum load-transaction /tmp/yum_save_tx.2016-01-11.17-24.mjLElP.yumtx
```

2. Check the system logs for error messages related to this problem. **`/var/log/messages`** and **`/var/log/yum.log`** would be good places to look.

```
[root@servera ~]# tail /var/log/messages
... Output omitted ...
Jan 11 20:23:16 servera systemd: Started Session 16 of user root.
Jan 11 20:23:16 servera systemd: Starting Session 16 of user root.
Jan 11 20:23:16 servera systemd-logind: Removed session 16.
Jan 11 20:24:28 servera systemd: Started Session 17 of user root.
Jan 11 20:24:28 servera systemd-logind: New session 17 of user root.
Jan 11 20:24:28 servera systemd: Starting Session 17 of user root.
Jan 11 20:24:29 servera dbus[507]: [system] Activating service
 name='org.freedesktop.problems' (using servicehelper)
Jan 11 20:24:29 servera dbus-daemon: dbus[507]: [system] Activating service
 name='org.freedesktop.problems' (using servicehelper)
Jan 11 20:24:29 servera dbus[507]: [system] Successfully activated service
 'org.freedesktop.problems'
Jan 11 20:24:29 servera dbus-daemon: dbus[507]: [system] Successfully activated
 service 'org.freedesktop.problems'
[root@servera ~]# tail /var/log/yum.log
... Output omitted ...
Dec 15 15:25:26 Installed: cryptsetup-1.6.7-1.el7.x86_64
Dec 15 15:25:27 Installed: words-3.0-22.el7.noarch
```

They do not appear to provide useful information for debugging this type of problem.

3. Perform a simple RPM package query and see if it produces an error message.

```
[root@servera ~]# rpm -q redhat-release-server
redhat-release-server-7.2-9.el7.x86_64
```

No useful diagnostic information displays.

4. Perform a more extensive RPM package query.

```
[root@servera ~]# rpm -qa
... Output omitted ...
```

The **`rpm -qa`** command generated a useful error message, but it scrolled off the screen before it could be read. Redirecting the command's standard output will allow you to see only the error messages.

```
[root@servera ~]# rpm -qa > /dev/null
error: rpmdbNextIterator: skipping h#       4 region trailer: BAD, tag 0 type 0
 offset 0 count 0
```

5. Identify and remove any stale locks. Remove the RPM database indexes.

```
[root@servera ~]# lsof | grep /var/lib/rpm
[root@servera ~]# rm /var/lib/rpm/__db*
rm: remove regular file ‘/var/lib/rpm/__db.001’? y
rm: remove regular file ‘/var/lib/rpm/__db.002’? y
rm: remove regular file ‘/var/lib/rpm/__db.003’? y
```

6. Back up the RPM database files before making any further changes.

```
[root@servera ~]# tar cjvf rpmdb-$(date +%Y%m%d-%H%M).tar.bz2 /var/lib/rpm
tar: Removing leading `/' from member names
/var/lib/rpm/
... Output omitted ...
```

7. Use **rpmdb_verify** to verify the contents of the **Packages** RPM database.

```
[root@servera ~]# cd /var/lib/rpm
[root@servera rpm]# /usr/lib/rpm/rpmdb_verify Packages
rpmdb_verify: BDB0682 Page 32: bad prev_pgno 0 on overflow page (should be 31)
rpmdb_verify: BDB0683 Page 32: overflow item incomplete
rpmdb_verify: Packages: BDB0090 DB_VERIFY_BAD: Database verification failed
BDB5105 Verification of Packages failed.
```

8. Back up the **Packages** RPM database. Use **rpmdb_dump** and **rpmdb_load** to create an intact **Packages** RPM database file.

```
[root@host rpm]# mv Packages Packages.broken
[root@host rpm]# /usr/lib/rpm/rpmdb_dump Packages.broken | \
> /usr/lib/rpm/rpmdb_load Packages
[root@host rpm]# /usr/lib/rpm/rpmdb_verify Packages
BDB5105 Verification of Packages succeeded.
```

9. Use **rpm --rebuilddb** to create new RPM database indexes.

```
[root@servera rpm]# rpm --rebuilddb
error: rpmdbNextIterator: skipping h#       4 region trailer: BAD, tag 0 type 0
 offset 0 count 0
```

10. Run the **rpm -qa** command again and display only standard error messages.

```
[root@servera rpm]# rpm -qa > /dev/null
```

    No error messages displayed. The corrupted database is fixed.

11. Once you are done, verify your work by running the **lab package-dependencies grade** command from your **workstation** machine.

```
[student@workstation ~]$ lab package-database grade
```

12. After grading your systems, successfully reset your machines to the state they had before starting the lab, either by resetting your virtual machines or by running the following command on your **workstation** system.

```
[student@workstation ~]$ lab package-database reset
```

Identifying and Recovering Changed Files

Objectives

After completing this section, students should be able to identify and restore changed files owned by package management.

Verifying installed packages

One major advantage of RPM package management is the local RPM database. Each time an RPM package is installed, information about each of the files it provides gets recorded in the RPM database. This information includes the file sizes, creation timestamps, content checksums, permissions, and user/group ownerships. Files provided by RPM packages can be verified because of the data in the RPM database.

Verifying packages with rpm

Verifying an installed package compares the attributes of the files it provides against the RPM database. Any inconsistencies will be reported. The **rpm -V** command verifies the packages specified as arguments to the command. **rpm -Va** will verify the files of all the packages installed on the system. Normally the command is silent, unless it finds discrepencies between the file system and the RPM database. When a difference is found, **rpm** prints the name of the file and a string that indicates what attributes of the file are different. Consider the following sample output.

```
[root@demo ~]# rpm -Va
missing     /var/run/wpa_supplicant
SM5....T.  c /etc/ssh/sshd_config
....L....  c /etc/pam.d/fingerprint-auth
....L....  c /etc/pam.d/password-auth
....L....  c /etc/pam.d/postlogin
....L....  c /etc/pam.d/smartcard-auth
....L....  c /etc/pam.d/system-auth
....L....  c /etc/rc.d/rc.local
S.5....T.  c /etc/systemd/logind.conf
.M.......    /var/lib/nfs/rpc_pipefs
.M.......    /run/cloud-init
... Output omitted ...
```

The first string of characters is a mask of the file's attributes. Periods are displayed for attributes that agree with the database. The following table lists the most common fields of file attributes.

Letter	File attribute
S	File size
M	Mode (permissions, including file type)
5	Contents (digest, formerly the MD5 checksum)
L	A symbolic link points to a different location
U	User ownership
G	Group ownership
T	Modification time

Optionally, **rpm -V** may display a single character before the file name. This character represents the RPM file type that the package builder specified. The following table lists the most common RPM file types specified.

Letter	File type
c	**%config** configuration file
d	**%doc** documentation file
l	**%license** license file
r	**%readme** readme file

Verifying packages with Yum

Yum can also verify installed packages. This functionality is not available by default. The *yum-plugin-verify* package adds the **verify**, **verify-rpm**, and **verify-all** subcommands to the **yum** command.

The **yum verify** command displays detailed information about file differences, but it excludes configuration files without other options. This assumption is based on the fact that configuration file changes are normal. The **verify-rpm** subcommand includes configuration files when they differ from their original state.

```
[root@demo ~]# rpm -V openssh-server
SM5....T.  c /etc/ssh/sshd_config
[root@demo ~]# yum verify openssh-server
Loaded plugins: langpacks, search-disabled-repos, verify, versionlock
verify done
[root@demo ~]# yum verify-rpm openssh-server
Loaded plugins: langpacks, search-disabled-repos, verify, versionlock
==================== Installed Packages ====================
openssh-server.x86_64 : An open source SSH server daemon
    File: /etc/ssh/sshd_config
    Tags: configuration
        Problem:  checksum does not match
        Current:  sha256:0b94e3b6ef4f76f8d0c2b324bd5ba5b93aa778...
        Original: sha256:bbd30bfaf820eeda7ab43b61a66d48a4d4fc67...
                                 --------
        Problem:  size does not match
        Current:          4386 B
        Original:         4361 B
                                 --------
        Problem:  mode does not match
        Current:  user:wr-, group:-r-, other:-r-
        Original: user:wr-, group:---, other:---
                                 --------
        Problem:  mtime does not match
        Current:  Tue Dec 15 12:26:40 2015 (81 days, 11:44:07 later)
        Original: Fri Sep 25 00:42:33 2015
verify-rpm done
```

Note that the **yum verify** commands produce more readable output. In addition to printing the file differences, the commands print the comparable values from the file system and the RPM database.

Recovering changed files

The **rpm** command has a **--setperms** option that will restore the file permissions of files installed by the packages specified as arguments to the command. The following command sequence shows how **rpm --setperms** resets the permissions of files in a package.

```
[root@demo ~]# rpm -V setup
.M....G..  c /etc/motd
[root@demo ~]# rpm --setperms setup
[root@demo ~]# rpm -V setup
......G..  c /etc/motd
```

Note that the group ownership of the example file still does not agree with its original ownership in the RPM database. The **rpm --setugids** command restores the user and/or group ownerships of the files in a package to their original values.

```
[root@demo ~]# rpm --setugids setup
[root@demo ~]# rpm -V setup
```

The **yum reinstall** command can be used to recover modified files that belong to a package. This command reinstalls the identical version of a package that is currently installed. It works for packages that only allow one version of a package to be installed on the system at a time. It does not work for "install only" packages, such as the Linux kernel.

The following sample output shows **yum reinstall** restoring a modified executable. The first **rpm -V** command confirms that the size, contents, and timestamps of **/usr/sbin/vsftpd** differ from their original values in the RPM database. The second **rpm -V** command does not display any output because the modified file has been reinstalled on the file system.

```
[root@demo ~]# rpm -V vsftpd
S.5....T.    /usr/sbin/vsftpd
[root@demo ~]# yum -y reinstall vsftpd
... Output omitted ...
Transaction test succeeded
Running transaction
Installing : vsftpd-3.0.2-10.el7.x86_64                                   1/1
Verifying  : vsftpd-3.0.2-10.el7.x86_64                                   1/1

Installed:
vsftpd.x86_64 0:3.0.2-10.el7

Complete!
[root@demo ~]# rpm -V vsftpd
```

References

rpm(8), **yum**(8), and **yum-verify**(1) man pages

Guided Exercise: Identifying and Recovering Changed Files

In this lab, you will diagnose and correct file permissions and corrupted files that belong to packages.

Resources	
Machines	• **workstation** • **servera**

Outcome(s)

You should be able to diagnose and correct file permission issues and corrupted files using **rpm** and **yum**.

Before you begin

Set up your systems for this exercise by running the **lab broken-commands setup** command on your **workstation** system.

```
[student@workstation ~#$ lab broken-commands setup
```

Reports are coming in that some of the commands on **servera** are broken. Users cannot list files with **ls** and the **sudo** command is behaving strangely. You have been called in to diagnose and correct these problems.

1. Log in as **root** on **servera**. An error message appears as you log in that gives you a clue about the first broken command that needs to be fixed.

```
[student@workstation ~]$ ssh root@servera
Last login: Tue Jan  5 20:25:01 2016 from workstation.lab.example.com
-bash: /usr/bin/ls: Permission denied
[root@servera ~]#
```

2. Use **rpm** to identify the package the command comes from and verify the package contents.

```
[root@servera ~]# rpm -qf /usr/bin/ls
coreutils-8.22-15.el7.x86_64
[root@servera ~]# rpm -V coreutils
.M.......    /usr/bin/ls
```

The **M** in the output indicates the permissions, or mode, of **/usr/bin/ls** has changed. Every other attribute about the file is the same as when it was installed.

3. Use **rpm** to restore the permissions of the files in the *coreutils* package.

```
[root@servera ~]# rpm --setperms coreutils
```

4. Confirm that the command has been fixed. Reverify the *coreutils* package, then see if **ls** is working again.

```
[root@servera ~]# rpm -V coreutils
[root@servera ~]# ls
anaconda-ks.cfg
```

5. Now it is time to solve the issue with **sudo**. Log in as **student** on **servera** and try to use **sudo** to grant you access to count the words in **/etc/ssh/sshd_config**.

```
[student@servera ~]$ sudo wc -l /etc/ssh/sshd_config
bash: /usr/bin/sudo: Permission denied
```

6. Determine the package that provides the **sudo** command.

```
[student@servera ~]$ which sudo
/usr/bin/sudo
[student@servera ~]$ rpm -qf /usr/bin/sudo
sudo-1.8.6p7-16.el7.x86_64
```

7. Log in as **root** on **servera**. Use **rpm -V** to verify the package that provides the **sudo** command.

```
[root@servera ~]# rpm -V sudo
S.5....T.  c /etc/sudoers
..5....T.    /usr/bin/sudo
[root@servera ~]# ls -l /usr/bin/sudo
---s--x--x. 1 root root 130712 Jan  5 20:52 /usr/bin/sudo
```

It indicates contents and timestamp of the executable have changed.

8. Use the **yum reinstall** command to repair the installed package. Verify the package when the command finishes to confirm the executable has been restored.

```
[root@servera ~]# yum -y reinstall sudo
... Output omitted ...
Installed:
  sudo.x86_64 0:1.8.6p7-16.el7

Complete!
[root@servera ~]# rpm -V sudo
S.5....T.  c /etc/sudoers
```

The executable does not appear in the **rpm -V** output, so it has been restored.

9. Log back in as **student** on **servera** and confirm that **sudo** is working.

```
[student@servera ~]$ sudo wc -l /etc/ssh/sshd_config
[sudo] password for student: student
152 /etc/ssh/sshd_config
```

10. Once you are done, verify your work by running the **lab broken-commands grade** command from your **workstation** machine.

```
[student@workstation ~]$ lab broken-commands grade
```

11. After grading your systems, successfully reset your machines to the state they had before starting the lab, either by resetting your virtual machines or by running the following command on your **workstation** system.

```
[student@workstation ~]$ lab broken-commands reset
```

Subscribing Systems to Red Hat Updates

Objectives

After completing this section, students should be able to register a system with Red Hat and manage Red Hat subscriptions.

Red Hat Enterprise Linux systems must be entitled to receive software packages. Red Hat Subscription Manager (RHSM) applications are used to manage a host's software entitlements. Red Hat Subscription Manager uses X.509 certificates on a host to keep track of its subscription status. The entitlement process is a three-step process.

1. Products are added to a system. Product certificates on the system identify which products it has installed. This step is performed when the system is installed and when additional software is installed.

2. Register the system with Red Hat directly or a Red Hat Satellite server. This step is taken once and an identity certificate is placed on the system.

3. Attach subscriptions to the system. Entitlement certificates are stored locally once this step is completed.

The X.509 certificates have expiration dates and they can be refreshed when new certificates become available. Yum has a **subscription-manager** plug-in that causes it to work with the Red Hat Subscription Management system. The **rhsmcertd** service periodically checks system configuration to confirm entitlement certificates are current and identify certificate updates.

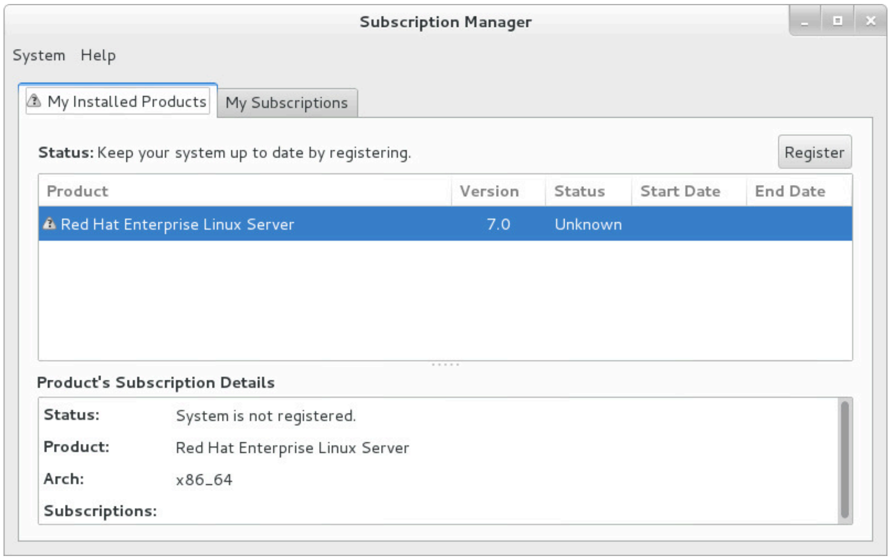
There are troubleshooting benefits because of the use of certificates. This approach allows an administrator to determine which products are installed on a system and which subscriptions are consumed. The focus of Red Hat Subscription Manager is subscription management, not content or system management.

Red Hat Subscription Management client systems must be able to reach the subscription management service and content delivery network. Both network services are accessed through port 443 by default. The default Internet configuration defines the RHSM server as **subscription.rhn.redhat.com**. The host name for the content delivery server is **cdn.redhat.com**.

subscription-manager-gui

On graphical machines where X is installed, **subscription-manager-gui** can be used to configure and manage Red Hat software subscriptions. **subscription-manager-gui** can be launched from a shell, or it can be launched by selecting the **Applications** > **System Tools** > **Red Hat Subscription Manager** menu item from the desktop.

When **subscription-manager-gui** first launches, a window will appear that displays the current system's RHSM status. The **My Installed Products** tab will be selected. A list of products already installed on the system will display in the frame at the top of the screen. Additional information about the currently selected product will display in the bottom window.



Status: Keep your system up to date by registering.

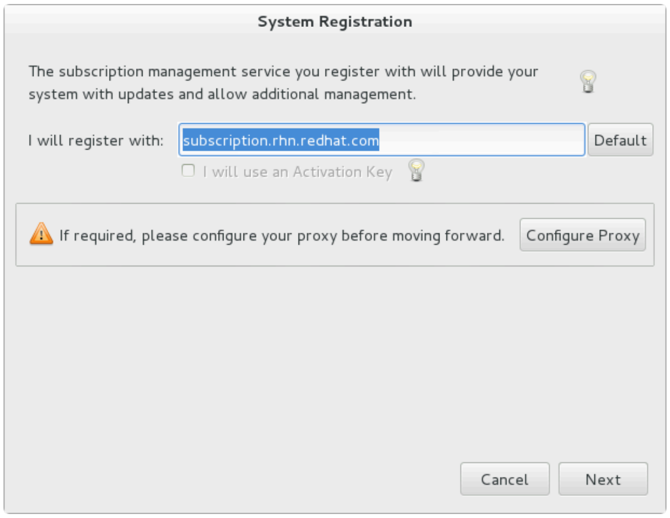
Product	Version	Status	Start Date	End Date
Red Hat Enterprise Linux Server	7.0	Unknown		

Product's Subscription Details

Status:	System is not registered.
Product:	Red Hat Enterprise Linux Server
Arch:	x86_64
Subscriptions:	

Figure 6.1: subscription-manager-gui My Installed Products tab

Clicking the **Register** button on this screen will start the RHSM registration process. The **System Registration** dialog box will appear.



The subscription management service you register with will provide your system with updates and allow additional management.

I will register with: subscription.rhn.redhat.com

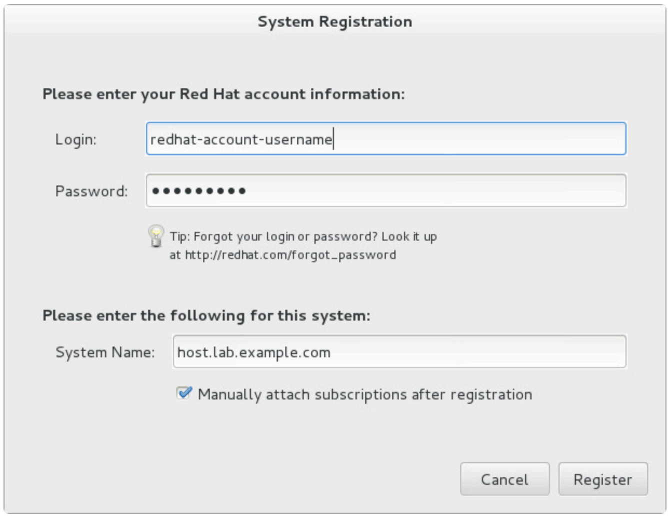
☐ I will use an Activation Key

If required, please configure your proxy before moving forward.

Figure 6.2: Registration host selection

The **`subscription.rhn.redhat.com`** host is selected by default, but an administrator can provide the host name for a Red Hat Satellite server instead. If the machine being registered has to use a proxy to access the RHSM host, **Configure Proxy** must be selected and proxy connection information provided.

Once the registration host has been selected, clicking **Next** will cause the next **System Registration** screen to appear. This will prompt for login credentials for the Red Hat Customer Portal, or a Red Hat Satellite account if a Satellite server was selected.



System Registration

Please enter your Red Hat account information:

Login: redhat-account-username

Password: •••••••••

Tip: Forgot your login or password? Look it up at http://redhat.com/forgot_password

Please enter the following for this system:

System Name: host.lab.example.com

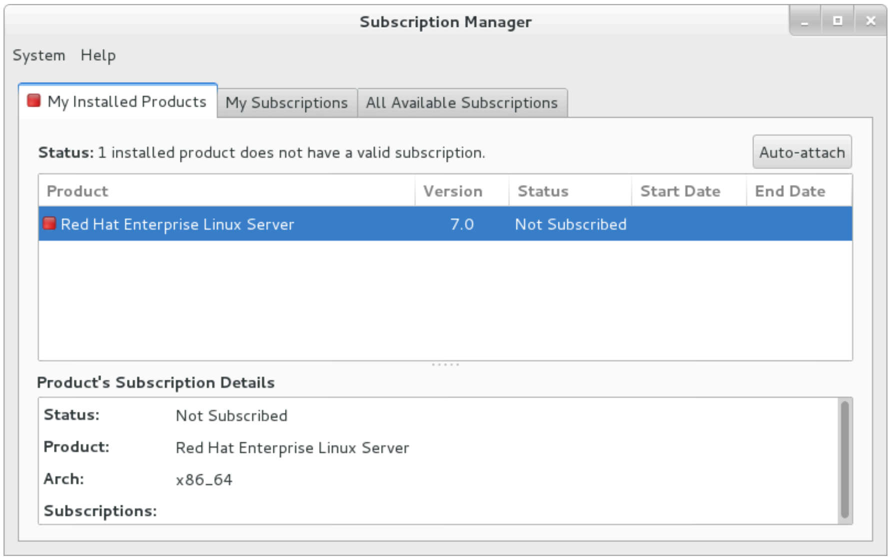
☑ Manually attach subscriptions after registration

Figure 6.3: Red Hat Customer Portal authentication

Note

If the **I will use an Activate Key** checkbox was selected on the previous screen, the administrator can provide the name of an activation key and the Satellite organization it is defined in.

Once the system is registered with the RHSM, then the **My Installed Products** tab will display, but the product status will change from **Unknown** to **Not Subscribed**.



Subscription Manager

Status: 1 installed product does not have a valid subscription.

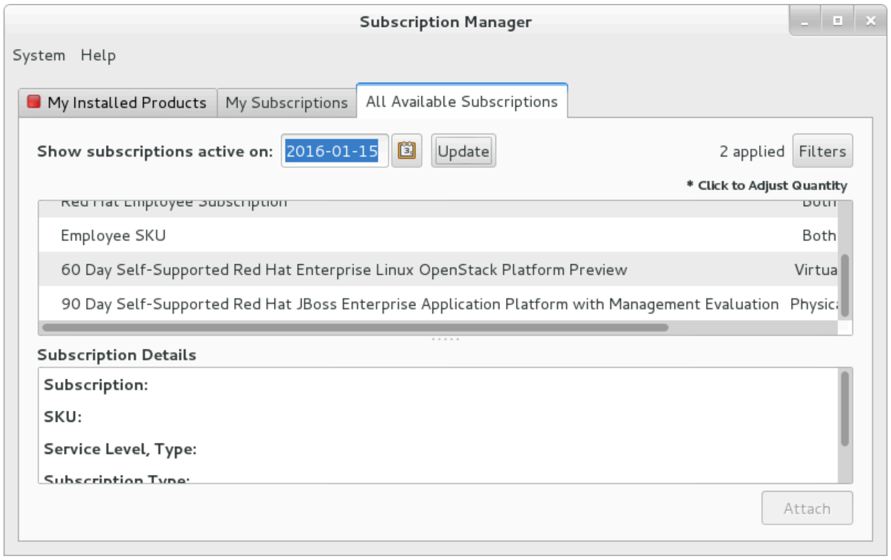
Product	Version	Status	Start Date	End Date
🟥 Red Hat Enterprise Linux Server	7.0	Not Subscribed		

Product's Subscription Details

Status:	Not Subscribed
Product:	Red Hat Enterprise Linux Server
Arch:	x86_64
Subscriptions:	

Figure 6.4: Host registered, but not subscribed

Another tab will appear, **All Available Subscriptions.** Click this tab to see the RHSM subscriptions that can be attached to the system.



Show subscriptions active on: 2016-01-15

2 applied

* Click to Adjust Quantity

Red Hat Employee Subscription	Both
Employee SKU	Both
60 Day Self-Supported Red Hat Enterprise Linux OpenStack Platform Preview	Virtua
90 Day Self-Supported Red Hat JBoss Enterprise Application Platform with Management Evaluation	Physic

Subscription Details

Subscription:

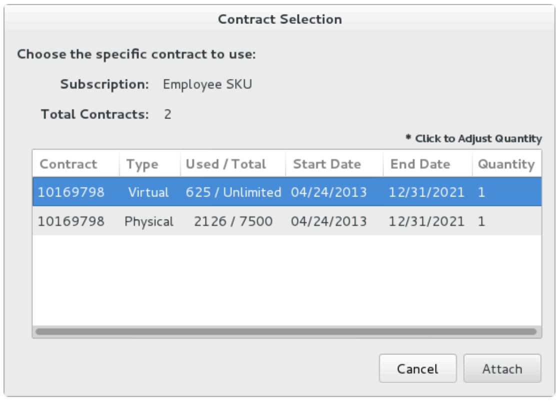
SKU:

Service Level, Type:

Subscription Type:

Figure 6.5: Listing available subscriptions

Once the desired subscription is selected, the **Attach** button is clicked. Either the subscription will be attached to the system, or an additional **Contract Selection** dialog box will appear if the subscription has multiple types of contracts associated with it.



Choose the specific contract to use:

Subscription: Employee SKU

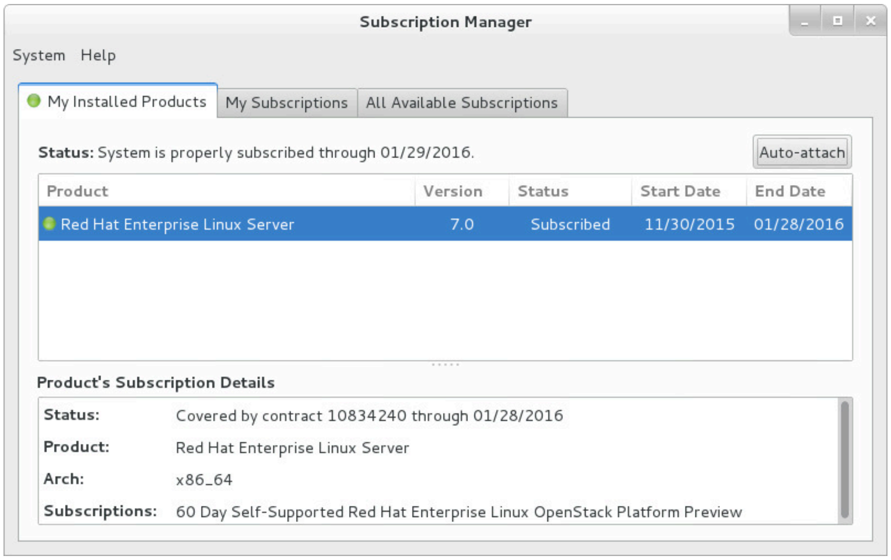
Total Contracts: 2

* Click to Adjust Quantity

Contract	Type	Used / Total	Start Date	End Date	Quantity
10169798	Virtual	625 / Unlimited	04/24/2013	12/31/2021	1
10169798	Physical	2126 / 7500	04/24/2013	12/31/2021	1

Figure 6.6: Choosing specific contract

If the host has multiple products installed, then it may be necessary to attach multiple subscriptions to cover all installed products. As each subscription requirement is met, the status of the products will change to **Subscribed**.



Status: System is properly subscribed through 01/29/2016.

Product	Version	Status	Start Date	End Date
Red Hat Enterprise Linux Server	7.0	Subscribed	11/30/2015	01/28/2016

Product's Subscription Details

Status:	Covered by contract 10834240 through 01/28/2016
Product:	Red Hat Enterprise Linux Server
Arch:	x86_64
Subscriptions:	60 Day Self-Supported Red Hat Enterprise Linux OpenStack Platform Preview

Figure 6.7: Host successfully subscribed

subscription-manager

Red Hat Subscription Management also has a text-based utility for managing system subscriptions, **subscription-manager**. **subscription-manager** provides all of the functionality of **subscription-manager-gui**, but command-line options are used to override default values. For example, the **--serverurl=HOSTNAME** option can be specified to point to a different RHSM host than **subscription.rhn.redhat.com**.

The **subscription-manager register** command registers a host with RHSM. The Red Hat Customer Portal username and password can be specified with the **--username** and **--password** options. If they are not specified, **subscription-manager** will prompt for the username and password. Proxy host and authentication credentials can also be specified with options.

```
[root@demo ~]# subscription-manager register
Username: rhn-test-user
Password: SecretPassw0rd!
The system has been registered with ID: 89062c77-0deb-471b-92f2-586a60829017
```

After registration, the **subscription-manager status** command will display a list of the installed products and their current statuses.

```
[root@demo ~]# subscription-manager status
+-------------------------------------------+
   System Status Details
+-------------------------------------------+
Overall Status: Invalid

Red Hat Enterprise Linux Server:
- Not supported by a valid subscription.
```

Now subscriptions can be installed to the installed products. The **--auto-attach** option could have installed necessary subscriptions during system registration, but that may have an unintended side effect of consuming high support service-level subscriptions on a nonproduction machine. Instead, specific subscriptions can be manually attached. The following **subscription-manager list** command lists the available subscriptions that can be attached to this system. The **Provides** entries display which products each subscription will support.

```
[root@demo ~]# subscription-manager list --available
+-------------------------------------------+
    Available Subscriptions
+-------------------------------------------+
Subscription Name: 60 Day Self-Supported Red Hat Enterprise Linux OpenStack
                   Platform Preview
Provides:          Red Hat OpenStack
                   Red Hat Enterprise Linux Server
                   Red Hat Software Collections (for RHEL Server)
                   Oracle Java (for RHEL Server)
                   Red Hat Enterprise MRG Messaging
                   Red Hat Enterprise Linux High Availability (for RHEL Server)
                   Red Hat Enterprise Linux Load Balancer (for RHEL Server)
SKU:               SER0123
Contract:          98765432
Pool ID:           0123456789abcdef0123456789abcdef
Available:         Unlimited
Suggested:         1
Service Level:     Self-Support
Service Type:      L1-L3
Subscription Type: Standard
Ends:              01/28/2016
System Type:       Virtual

... Output omitted ...
```

The value of the **Pool ID** field of the desired subscription is passed to **subscription-manager attach** with the **--pool** option.

```
[root@demo ~]# subscription-manager attach --pool 0123456789abcdef0123456789abcdef
Successfully attached a subscription for: 60 Day Self-Supported Red Hat
 Enterprise Linux OpenStack Platform Preview
[root@demo ~]# subscription-manager status
+-------------------------------------------+
   System Status Details
+-------------------------------------------+
Overall Status: Current
```

The **subscription-manager repos** command manages the software repositories made available by subscribed products. The **--list** option displays all of the available software repositories with their current state. **--enable *REPO_ID*** enables the specified repository. The **--disable** option disables the specified repository. Wildcards can be specified to match multiple repository IDs.

```
[root@demo ~]# subscription-manager repos --list
+----------------------------------------------------------+
    Available Repositories in /etc/yum.repos.d/redhat.repo
+----------------------------------------------------------+
Repo ID:   rhel-7-server-v2vwin-1-debug-rpms
Repo Name: Red Hat Virt V2V Tool for RHEL 7 (Debug RPMs)
Repo URL:  https://cdn.redhat.com/content/dist/rhel/server/7/$releasever/$basear
           ch/v2vwin/debug
Enabled:   0

... Output omitted ...
```

```
Repo ID:   rhel-7-server-rpms
Repo Name: Red Hat Enterprise Linux 7 Server (RPMs)
Repo URL:  https://cdn.redhat.com/content/dist/rhel/server/7/$releasever/$basear
           ch/os
Enabled:   1

... Output omitted ...
```

The **`subscription-manager remove`** command removes attached subscriptions. There are a couple of different ways to specify which subscriptions to remove. The **`--all`** command will match all attached subscriptions. Subscriptions can also be removed by specifying their serial numbers. The list of attached subscriptions is displayed by the **`subscription-manager list --consumed`** command, then the serial number is passed to **`subscription-manager remove`** using the **`--serial`** option.

```
[root@demo ~]# subscription-manager list --consumed
+-------------------------------------------+
   Consumed Subscriptions
+-------------------------------------------+
Subscription Name: 60 Day Self-Supported Red Hat Enterprise Linux OpenStack
                   Platform Preview
Provides:          Red Hat OpenStack
                   Red Hat Enterprise Linux Server
                   Red Hat Software Collections (for RHEL Server)
                   Oracle Java (for RHEL Server)
                   Red Hat Enterprise MRG Messaging
                   Red Hat Enterprise Linux High Availability (for RHEL Server)
                   Red Hat Enterprise Linux Load Balancer (for RHEL Server)
SKU:               SER0123
Contract:          98765432
Account:           543540
Serial:            7868709839063398548
Pool ID:           0123456789abcdef0123456789abcdef
Active:            True
Quantity Used:     1
Service Level:     Self-Support
Service Type:      L1-L3
Status Details:
Subscription Type: Standard
Starts:            11/30/2015
Ends:              01/28/2016
System Type:       Virtual

[root@demo ~]# subscription-manager remove --serial 7868709839063398548
Serial numbers successfully removed at the server:
   7868709839063398548
1 local certificate has been deleted.
```

The **`subscription-manager unregister`** command removes the current host from RHSM. It removes all of the subscriptions attached to the system and locally deletes system ID and subscription certificates.

Troubleshooting Red Hat Subscription Management

Examining log files

Checking log files for errors is typically the first step for effective troubleshooting. The Red Hat Subscription Manager applications have two primary log files, both located in the **`/var/log/rhsm`** directory: **`rhsm.log`** and **`rhsmcertd.log`**. Log messages generated by

subscription-manager and **subscription-manager-gui** executions are written to **rhsm.log**. **rhsmcertd.log** contains messages pertaining to the **rhsmcertd** daemon.

Troubleshooting configuration files

The primary configuration file for Red Hat Subscription Manager commands is **/etc/rhsm/rhsm.conf**. **/etc/rhsm/pluginconf.d** may contain additional configuration files for RHSM plug-ins. The following snippets from **rhsm.conf** show the key variables that define how RHSM utilities behave.

The file starts with the **[server]** section. It includes the variables that point to the RHSM server. The **proxy_*** variables defines the proxy server and proxy authentication when a proxy must be used to access the Internet.

```
[server]
# Server hostname:
hostname = subscription.rhn.redhat.com
# Server prefix:
prefix = /subscription
# Server port:
port = 443

... Output omitted ...

# an http proxy server to use
proxy_hostname =
# port for http proxy server
proxy_port =
# user name for authenticating to an http proxy, if needed
proxy_user =
# password for basic http proxy auth, if needed
proxy_password =
```

The **rhsm** section defines the URL for the content delivery network service, **baseurl**. The directories where the X.509 certificates are kept are also defined in this section.

```
[rhsm]
# Content base URL:
baseurl= https://cdn.redhat.com

... Output omitted ...

# Server CA certificate location:
ca_cert_dir = /etc/rhsm/ca/

... Output omitted ...

# Where the certificates should be stored
productCertDir = /etc/pki/product
entitlementCertDir = /etc/pki/entitlement
consumerCertDir = /etc/pki/consumer

# Manage generation of yum repositories for subscribed content:
manage_repos = 1

... Output omitted ...

# The directory to search for subscription manager plugins
pluginDir = /usr/share/rhsm-plugins
# The directory to search for plugin configuration files
```

```
pluginConfDir = /etc/rhsm/pluginconf.d
```

The **rhsmcertd** section defines the runtime intervals for the **rhsmcertd** daemon.

```
[rhsmcertd]
# Interval to run cert check (in minutes):
certCheckInterval = 240
# Interval to run auto-attach (in minutes):
autoAttachInterval = 1440
```

The other configuration file that impacts RHSM behavior is the Yum plug-in configuration file, **/etc/yum/pluginconf.d/subscription-manager.conf**. This determines whether Yum uses RHSM repositories. The **enabled** setting should be 1.

```
[main]
enabled=1
```

Inspecting certificates

All X.509 certificates are stored in subdirectories in the **/etc/pki** directory. The **consumer** subdirectory stores the server identity key and certificate.

```
[root@demo ~]# ls /etc/pki/consumer
cert.pem  key.pem
```

The **product** and **product-default** subdirectories contains the certificates of the installed products. The certificates of the subscribed products are stored in the **entitlement** subdirectory.

```
[root@demo ~]# ls /etc/pki/product-default
69.pem
[root@demo ~]# ls /etc/pki/entitlement
7868709839063398548-key.pem  7868709839063398548.pem
```

The **rct cat-cert** command displays the fields and values embedded in the RHSM X.509 certificates. It can be used to compare the ID of entitlement certificates with the ID of corresponding product certificates.

```
[root@demo ~]# rct cat-cert /etc/pki/product-default/69.pem | grep 69
        Path: /etc/pki/product/69.pem
        ID: 69
[root@demo ~]# rct cat-cert /etc/pki/entitlement/7868709839063398548.pem |
> egrep 'ID|Serial'
        Serial: 7868709839063398548
        Pool ID: 0123456789abcdef0123456789abcdef
        ID: 180
        ID: 191
        ID: 201
        ID: 205
        ID: 220
        ID: 239
        ID: 240
        ID: 69
        ID: 83
        ID: 85
```

Diagnosing network connectivity

If **subscription-manager** hangs and is unresponsive, there may be a problem with network connectivity. The **curl** command can be used to communicate with Red Hat's servers. If it is unable to connect, then the network issues must be diagnosed and corrected before continuing with **subscription-manager**.

The following example shows a successful connection to Red Hat. The URL passed as an argument is a **baseurl** copied from **/etc/yum.repos.d/redhat.repo**, except the **$releasever/$basearch** part of the URL was expanded.

```
[root@demo ~]# curl --head
> --key /etc/pki/entitlement/7868709839063398548-key.pem
> --cert /etc/pki/entitlement/7868709839063398548.pem
> --cacert /etc/rhsm/ca/redhat-uep.pem
> https://cdn.redhat.com/content/dist/rhel/server/7/7Server/x86_64/os
HTTP/1.1 403 Forbidden
Server: AkamaiGHost
Mime-Version: 1.0
Content-Type: text/html
Content-Length: 340
Expires: Mon, 18 Jan 2016 16:40:13 GMT
Date: Mon, 18 Jan 2016 16:40:13 GMT
X-Cache: TCP_DENIED from a128-241-218-165.deploy.akamaitechnologies.com
 (AkamaiGHost/7.4.0.1.1-16456689) (-)
Connection: keep-alive
EJ-HOST: Not_Set
X-Akamai-Request-ID: 111e1374
```

Disregard the 403 HTTP error. Network connectivity is what is being checked for, and **curl** was successful to get an HTTP response from Red Hat's servers.

References

subscription-manager(8), **rct**(8), and **rhsmcertd**(8) man pages

Quiz: Subscribing Systems to Red Hat Updates

The steps to troubleshoot Red Hat Subscription Manager issues are shown as follows. Indicate the order in which the steps should be taken.

__ a. Check the **`/etc/rhsm/rhsm.conf`** configuration file for correct values.

__ b. Use **`curl`** to identify possible network connectivity issues with **`subscription.rhn.redhat.com`**.

__ c. Use the **`rct cat-cert`** command to inspect the X.509 certificates used by RHSM.

__ d. Examine the **`/var/log/rhsm/rhsm.log`** file for helpful error messages.

Solution

The steps to troubleshoot Red Hat Subscription Manager issues are shown as follows. Indicate the order in which the steps should be taken.

2 a. Check the **/etc/rhsm/rhsm.conf** configuration file for correct values.

4 b. Use **curl** to identify possible network connectivity issues with **subscription.rhn.redhat.com**.

3 c. Use the **rct cat-cert** command to inspect the X.509 certificates used by RHSM.

1 d. Examine the **/var/log/rhsm/rhsm.log** file for helpful error messages.

Lab: Troubleshooting RPM Issues

In this lab, you will fix a software package issue that is causing a recently installed network service to fail.

Resources	
Application URL	`http://servera.lab.example.com`
Machines	• **workstation** • **servera**

Outcome(s)

You should be able to diagnose and repair package issues that cause a service to fail.

Before you begin

Prepare your systems for this exercise by running the **lab package-issues setup** command on your **workstation** system. This will install and deploy a web server on your **servera** system.

```
[student@workstation ~]$ lab package-issues setup
```

Someone on your team installed a web server on the **servera** server. Instead of displaying content, the following error message displays when it is accessed.

```
[student@workstation ~]$ curl -4 http://servera.lab.example.com
curl: (7) Failed connect to servera.lab.example.com:80; Connection refused
```

You have been assigned the task of making **servera** publish web content.

1. Connect to your **servera** machine and assess the problem.
2. Permanently fix the issue.
3. Grade your work by running the **lab package-issues grade** command from your **workstation** machine.

   ```
   [student@workstation ~]$ lab package-issues grade
   ```

4. Reset all of your machines to provide a clean environment for the next exercises.

Solution

In this lab, you will fix a software package issue that is causing a recently installed network service to fail.

Resources	
Application URL	`http://servera.lab.example.com`
Machines	• **workstation** • **servera**

Outcome(s)

You should be able to diagnose and repair package issues that cause a service to fail.

Before you begin

Prepare your systems for this exercise by running the **lab package-issues setup** command on your **workstation** system. This will install and deploy a web server on your **servera** system.

```
[student@workstation ~]$ lab package-issues setup
```

Someone on your team installed a web server on the **servera** server. Instead of displaying content, the following error message displays when it is accessed.

```
[student@workstation ~]$ curl -4 http://servera.lab.example.com
curl: (7) Failed connect to servera.lab.example.com:80; Connection refused
```

You have been assigned the task of making **servera** publish web content.

1. Connect to your **servera** machine and assess the problem.

    1.1. First, check the status of the **httpd** service to see if it is running.

```
[root@servera ~]# systemctl status httpd
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; vendor
 preset: disabled)
   Active: failed (Result: exit-code) since Fri 2016-01-08 10:05:34 PST; 1min
 43s ago
     Docs: man:httpd(8)
           man:apachectl(8)
  Process: 32644 ExecStop=/bin/kill -WINCH ${MAINPID} (code=exited, status=1/
FAILURE)
  Process: 32642 ExecStart=/usr/sbin/httpd $OPTIONS -DFOREGROUND (code=exited,
 status=203/EXEC)
 Main PID: 32642 (code=exited, status=203/EXEC)

Jan 08 10:05:34 servera.lab.example.com systemd[1]: Starting The Apache HTTP ...
Jan 08 10:05:34 servera.lab.example.com systemd[32642]: Failed at step EXEC s...
Jan 08 10:05:34 servera.lab.example.com systemd[1]: httpd.service: main proce...
Jan 08 10:05:34 servera.lab.example.com kill[32644]: kill: cannot find proces...
Jan 08 10:05:34 servera.lab.example.com systemd[1]: httpd.service: control pr...
Jan 08 10:05:34 servera.lab.example.com systemd[1]: Failed to start The Apach...
Jan 08 10:05:34 servera.lab.example.com systemd[1]: Unit httpd.service entere...
Jan 08 10:05:34 servera.lab.example.com systemd[1]: httpd.service failed.
Hint: Some lines were ellipsized, use -l to show in full.
```

The service is down, and error messages suggest it was not able to be started.

1.2. Since the service was recently installed, check the Yum log file to identify recently installed packages. Your output may be different, but assume the following packages were recently installed on your system.

```
[root@servera ~]# tail /var/log/yum.log
... Output omitted ...
Dec 15 15:25:26 Installed: cryptsetup-1.6.7-1.el7.x86_64
Dec 15 15:25:27 Installed: words-3.0-22.el7.noarch
Jan 08 10:05:32 Installed: apr-1.4.8-3.el7.x86_64
Jan 08 10:05:32 Installed: apr-util-1.5.2-6.el7.x86_64
Jan 08 10:05:33 Installed: httpd-tools-2.4.6-40.el7.x86_64
Jan 08 10:05:33 Installed: mailcap-2.1.41-2.el7.noarch
Jan 08 10:05:33 Installed: httpd-2.4.6-40.el7.x86_64
```

Verify the recently installed packages to see if they are intact.

```
[root@servera ~]# rpm -V apr apr-util httpd-tools mailcap httpd
..5....T.    /usr/sbin/httpd
```

The **rpm -V** command indicates the contents and timestamp of **/usr/sbin/httpd** has changed since it was installed. Identify the specific package that provides the corrupted file.

```
[root@servera ~]# rpm -qf /usr/sbin/httpd
httpd-2.4.6-40.el7.x86_64
```

2. Permanently fix the issue.

2.1. Use Yum to reinstall the affected package.

```
[root@servera ~]# yum -y reinstall httpd
```

2.2. Restart the **httpd** service.

```
[root@servera ~]# systemctl restart httpd
```

Confirm that it successfully started.

```
[root@servera ~]# systemctl status httpd
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; vendor
 preset: disabled)
   Active: active (running) since Fri 2016-01-08 10:11:54 PST; 1s ago
     Docs: man:httpd(8)
           man:apachectl(8)
  Process: 2376 ExecStop=/bin/kill -WINCH ${MAINPID} (code=exited, status=1/
FAILURE)
 Main PID: 2394 (httpd)
   Status: "Processing requests..."
   CGroup: /system.slice/httpd.service
           ├─2394 /usr/sbin/httpd -DFOREGROUND
           ├─2395 /usr/sbin/httpd -DFOREGROUND
... Output omitted ...
Jan 08 10:11:54 servera.lab.example.com systemd[1]: Starting The Apache HTTP ...
Jan 08 10:11:54 servera.lab.example.com systemd[1]: Started The Apache HTTP S...
Hint: Some lines were ellipsized, use -l to show in full.
```

2.3. Confirm from **workstation** that **servera** is providing web content.

```
[student@workstation ~]$ curl -4 http://servera.lab.example.com
Lab is fixed.
```

3. Grade your work by running the **lab package-issues grade** command from your **workstation** machine.

```
[student@workstation ~]$ lab package-issues grade
```

4. Reset all of your machines to provide a clean environment for the next exercises.

Summary

In this chapter, you learned:

- Package dependency information can be displayed with the **yum deplist**, **rpm -q --requires**, and **rpm -q --provides** commands.
- The **yum downgrade** and **rpm -U --oldpackage** commands can be used to install older versions of installed packages.
- The *yum-plugin-versionlock* package extends Yum so that specific versions of packages cannot be upgraded.
- The **rpmdb_verify**, **rpmdb_dump**, and **rpmdb_load** utilities can be used to repair a mildly corrupted RPM database. **rpm --rebuilddb** will rebuild the RPM database **__db.*** index files.
- The **rpm -V** command will verify installed packages. The *yum-plugin-verify* package extends Yum so that it can also verify installed packages.
- The **rpm --setperms**, **rpm --setugids**, and **yum reinstall** commands can recover changed files.
- Software subscriptions can be managed with either the **subscription-manager-gui** or **subscription-manager** commands.
- The **rct cat-cert** command can be used to display the contents of RHSM identity, product, and entitlement certificates.
- **curl** can be used to confirm network connectivity with the Red Hat Subscription Manager host.

CHAPTER 7

TROUBLESHOOTING NETWORK ISSUES

Overview	
Goal	Identify and resolve network connectivity issues.
Objectives	• Verify network connectivity using standard tools. • Identify and fix network connectivity issues. • Inspect network traffic to aid troubleshooting.
Sections	• Testing Connectivity (and Guided Exercise) • Resolving Connectivity Issues (and Guided Exercise) • Inspecting Network Traffic (and Guided Exercise)
Lab	• Troubleshooting Network Issues

Testing Connectivity

Objectives

After completing this section, students should be able to verify network connectivity using standard tools.

Sending IMCP echo requests

The Internet Control Message Protocol (ICMP) is a low-level protocol used to test host availability and send error messages. One of the first steps for testing connectivity is sending ICMP echo requests to the remote host. Hosts by default are configured to send an ICMP echo reply to indicate that they are present and running. This is accomplished with the **ping** command.

The **ping** command takes the host name, or IP address, of the host of interest as an argument. When the **-b** option is used, the command argument specified is a broadcast address. By default, **ping** will continuously send ICMP echo requests every second. All responses that it receives are displayed with their packet sequence number and latency time. When the user interrupts the command with a **Ctrl**+**C**, **ping** displays a summary.

```
[user@demo ~]$ ping content.example.com
PING content.example.com (172.25.254.254) 56(84) bytes of data.
64 bytes from classroom.example.com (172.25.254.254): icmp_seq=1 ttl=63 time=0.426 ms
64 bytes from classroom.example.com (172.25.254.254): icmp_seq=2 ttl=63 time=0.545 ms
Ctrl+C
--- content.example.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 999ms
rtt min/avg/max/mdev = 0.426/0.485/0.545/0.063 ms
```

There are a couple of options that make **ping** useful in shell programs. The **-c *COUNT*** option specifies a count that limits the number of echo requests to send out. The **-W *TIMEOUT*** option specifies the number of seconds to wait for replies before timing out. The following **ping** command sends a single echo request and waits three seconds for a reply.

```
[user@demo ~]$ ping -c 1 -W 3 172.25.250.254
PING 172.25.250.254 (172.25.250.254) 56(84) bytes of data.
64 bytes from 172.25.250.254: icmp_seq=1 ttl=64 time=0.217 ms

--- 172.25.250.254 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.217/0.217/0.217/0.000 ms
[user@demo ~]$ echo $?
0
```

ping returns a zero exit status when the target host responds, and a non-zero exit status when no reply is received.

The **ping6** command is the IPv6 version of **ping**. Since multiple interfaces can have an IPv6 link local address (fe80::), the **-I *INTERFACE*** option specifies the interface to send the echo requests out.

```
[user@demo ~]$ ping6 -I eth0 fe80::5054:ff:fe00:fa09
PING fe80::5054:ff:fe00:fa09(fe80::5054:ff:fe00:fa09) from fe80::5054:ff:fe00:fa0a eth0:
 56 data bytes
```

```
64 bytes from fe80::5054:ff:fe00:fa09: icmp_seq=1 ttl=64 time=0.267 ms
64 bytes from fe80::5054:ff:fe00:fa09: icmp_seq=2 ttl=64 time=0.332 ms
Ctrl+C
--- fe80::5054:ff:fe00:fa09 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 999ms
rtt min/avg/max/mdev = 0.267/0.299/0.332/0.036 ms
```

The **-I** option is not needed when a routable IPv6 address is used.

```
[user@demo ~]$ ping6 -n remote.example.com
PING remote.example.com(fd2a:6552:6448:6174::abc) 56 data bytes
64 bytes from fd2a:6552:6448:6174::abc: icmp_seq=1 ttl=64 time=0.372 ms
64 bytes from fd2a:6552:6448:6174::abc: icmp_seq=2 ttl=64 time=0.305 ms
Ctrl+C
--- remote.example.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1000ms
rtt min/avg/max/mdev = 0.305/0.338/0.372/0.038 ms
```

The following table lists the most commonly used **ping** and **ping6** options.

Option	Description
-b	Broadcast to the network specified as an argument
-n	Display host information numerically
-i *INTERVAL*	Specify the echo request interval in seconds (default is 1)
-I *INTERFACE*	Send echo requests out *INTERFACE*
-c *COUNT*	Send only *COUNT* echo requests
-W *TIMEOUT*	Wait *TIMEOUT* seconds before quitting

Scanning network ports

Nmap is an open source port scanner that is provided by the Red Hat Enterprise Linux distribution. It is a tool that administrators use to rapidly scan large networks, but it can also do more intensive port scans on individual hosts. According to the **nmap**(1) man page,

> Nmap uses raw IP packets in novel ways to determine what hosts are available on the network, what services (application name and version) those hosts are offering, what operating systems (and OS versions) they are running, what type of packet filters/firewalls are in use, and dozens of other characteristics.

The *nmap* package provides the **nmap** executable.

The following example shows Nmap scanning the 192.168.37.0/24 private network. The **-n** option tells it to display host information numerically, not using DNS. As Nmap discovers each host, it scans privileged TCP ports looking for services. It displays the MAC address, with the corresponding network adapter manufacturer, of each host.

```
[root@demo-64 ~]# nmap -n 192.168.37.0/24

Starting Nmap 6.40 ( http://nmap.org ) at 2016-01-28 15:05 PST
Nmap scan report for 192.168.37.1
Host is up (0.00030s latency).
Not shown: 996 filtered ports
PORT    STATE  SERVICE
22/tcp  open   ssh
```

```
53/tcp  open   domain
80/tcp  closed http
631/tcp closed ipp
MAC Address: 52:54:00:D1:C3:F6 (QEMU Virtual NIC)

Nmap scan report for 192.168.37.11
Host is up (0.0000050s latency).
Not shown: 998 closed ports
PORT    STATE SERVICE
22/tcp  open  ssh
111/tcp open  rpcbind

Nmap done: 256 IP addresses (2 hosts up) scanned in 7.10 seconds
```

Port scans can be disabled with the **-sn** option. Administrators use this option when they want to quickly see which hosts are running on a network.

```
[root@demo ~]# nmap -n -sn 192.168.37.0/24

Starting Nmap 6.40 ( http://nmap.org ) at 2016-01-28 15:02 PST
Nmap scan report for 192.168.37.1
Host is up (0.00030s latency).
MAC Address: 52:54:00:D1:C3:F6 (QEMU Virtual NIC)
Nmap scan report for 192.168.37.11
Host is up.
Nmap done: 256 IP addresses (2 hosts up) scanned in 1.97 seconds
```

The **-sU** option tells **nmap** to perform a UDP port scan. This scan takes substantially more time to run than the default TCP port scan. It is useful when an administrator wants a more complete picture of what services a host exposes to the network.

```
[root@demo ~]# nmap -n -sU 192.168.37.1

Starting Nmap 6.40 ( http://nmap.org ) at 2016-01-28 15:23 PST
Nmap scan report for 192.168.37.1
Host is up (0.00042s latency).
Not shown: 997 filtered ports
PORT    STATE         SERVICE
53/udp  open|filtered domain
67/udp  open|filtered dhcps
631/udp open|filtered ipp
MAC Address: 52:54:00:D1:C3:F6 (QEMU Virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 1076.23 seconds
```

Nmap has many other useful options that are beyond the scope of this course.

Communicating with a remote service

A troubleshooting tool that allows an administrator to communicate with a service directly is Ncat. Ncat can use either TCP and UDP to communicate with a network service, and it also supports SSL communication. Ncat is invoked as either the **nc** or **ncat** commands. The *nmap-ncat* package of Red Hat Enterprise Linux provides the Ncat utility.

Ncat has two modes of operation. It acts as a network client by default, also known as connect mode. The host name and port to connect to are specified as arguments to the command.

```
[root@demo ~]# nc mailserver 25
```

```
220 mail.example.com ESMTP Postfix
EHLO host.example.com
250-mail.example.com
250-PIPELINING
250-SIZE 10240000
250-VRFY
250-ETRN
250-ENHANCEDSTATUSCODES
250-8BITMIME
250 DSN
QUIT
221 2.0.0 Bye
Ctrl+C
[root@demo ~]#
```

Ncat acts as a server when it is invoked with the **-l** option. This is known as listen mode. The **-k** option causes Ncat to keep the port open to listen for more connections when used in this mode.

```
[root@demo ~]# nc -l 2510
line of text 1
line of text 2
... Output omitted ...
```

The default behavior of Ncat in listen mode is to display any text that it receives over the network to the screen. The **-e *COMMAND*** option causes Ncat to pass incoming network traffic to the command specified with the option. The following command launches Ncat in listen mode on port 2510. It will pass all network traffic to **/bin/bash**.

```
[root@remote ~]# nc -l 2510 -e /bin/bash
```

The following output shows what happens when a Ncat connect mode session is used to communicate with the preceding listener.

```
[root@demo ~]# nc remote.example.com 2510
ls
anaconda-ks.cfg
pwd
/root
uptime
 05:30:49 up 4 days, 13:50,  1 user,  load average: 0.02, 0.02, 0.05
hostname
remote.example.com
Ctrl+D
```

Each line of text sent to the server is executed by **/bin/bash**. The resulting output of the commands is sent back to the network client.

Important

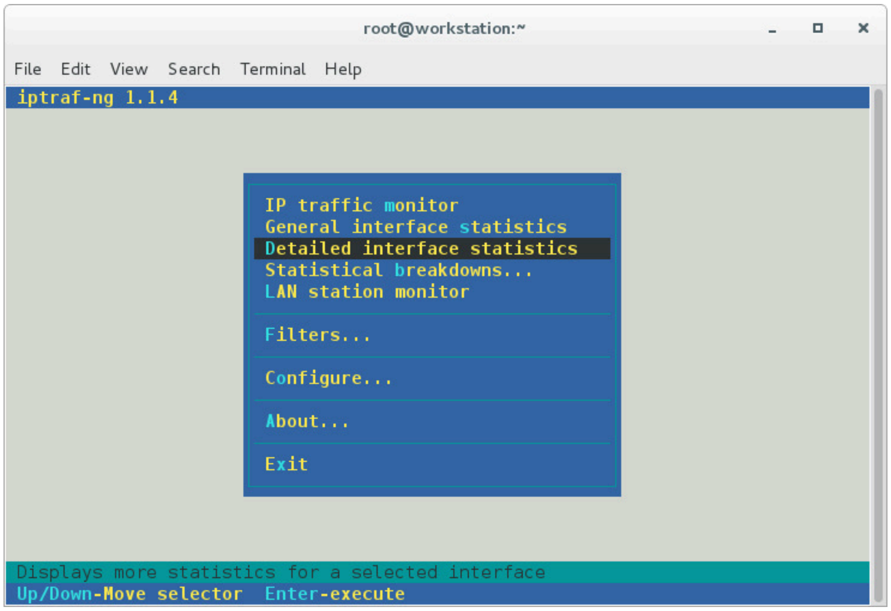
A shell should never be directly connected to a network port for obvious security reasons. The previous example was given to demonstrate the functionality of Ncat using the **-e** option.

Ncat supports both IPv4 and IPv6. The **-4** and **-6** options force Ncat to use either IPv4 or IPv6 respectively.

Monitoring network traffic

IPTraf was a network monitor software, originally developed in the mid-1990s. Red Hat Enterprise Linux includes the next-generation version of IPTraf, provided by the *iptraf-ng* package.

The **iptraf-ng** command launches the application. The program requires superuser access, so it must be executed by **root**. IPtraf-ng has a curses menu interface. The following screenshot shows the main menu that is displayed when the tool is first launched.



```
iptraf-ng 1.1.4

                    IP traffic monitor
                    General interface statistics
                    Detailed interface statistics
                    Statistical breakdowns...
                    LAN station monitor

                    Filters...

                    Configure...

                    About...

                    Exit

Displays more statistics for a selected interface
Up/Down-Move selector  Enter-execute
```

Figure 7.1: Main menu of iptraf-ng network monitor

Selections are made with the up and down arrow keys, then **Enter** executes the current selection. Alternatively, typing the highlighted single character can also execute a command.

IPtraf-ng can monitor current network connections. It can display detailed or summary counts about the network interfaces on the local system. It can display UDP and ICMP packet information.

The **Filters...** main menu selection allows an administrator to create filters to include (or exclude) specific types of network traffic. Each filter is a collection of rules that can select packets based on source, or destination, address, port, and IP protocol type.

R

References

icmp(7), **iptraf-ng**(8), **ncat**(1), **nmap**(1), and **ping**(8) man pages

Guided Exercise: Testing Connectivity

In this lab, you will use some software tools to test network connectivity.

Resources	
Application URL	**`http://content.example.com`**
Machines	• **`servera`** • **`serverb`**

Outcome(s)

You should be able to use software tools to test network connectivity.

Before you begin

Set up your systems for this exercise by running the **`lab network-testing setup`** command on your **`workstation`** system. This will install software, open firewall ports, and configure a web server on serverb.

```
[student@workstation ~#$ lab network-testing setup
```

1. The first step for testing network connectivity is to **`ping`** a remote host. Log in as **`root`** on **`servera`**. Use the **`ping`** command to confirm that **`serverb`** is reachable.

```
[root@servera ~]# ping serverb.lab.example.com
PING serverb.lab.example.com (172.25.250.11) 56(84) bytes of data.
64 bytes from serverb.lab.example.com (172.25.250.11): icmp_seq=1 ttl=64 time=0.455
 ms
64 bytes from serverb.lab.example.com (172.25.250.11): icmp_seq=2 ttl=64 time=0.263
 ms
64 bytes from serverb.lab.example.com (172.25.250.11): icmp_seq=3 ttl=64 time=0.265
 ms
Ctrl+C
--- serverb.lab.example.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2000ms
rtt min/avg/max/mdev = 0.263/0.327/0.455/0.092 ms
```

2. The **`ping6`** command tests IPv6 connectivity. Use the **`ping6`** command to see if **`serverb`** is reachable. The **`-n`** option will display host address information numerically.

```
[root@servera ~]# ping6 -n serverb.lab.example.com
[root@servera ~]#
PING serverb.lab.example.com(fd37:5265:6448:6174::b) 56 data bytes
64 bytes from fd37:5265:6448:6174::b: icmp_seq=1 ttl=64 time=0.285 ms
64 bytes from fd37:5265:6448:6174::b: icmp_seq=2 ttl=64 time=0.311 ms
64 bytes from fd37:5265:6448:6174::b: icmp_seq=3 ttl=64 time=0.314 ms
Ctrl+C
--- serverb.lab.example.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2000ms
rtt min/avg/max/mdev = 0.285/0.303/0.314/0.019 ms
```

3. The *nmap* package is typically not installed by default. Use Yum to install it on your **`servera`** host.

```
[root@servera ~]# yum -y install nmap
```

4. Scan the **lab.example.com** network, 172.25.250.0/24, and see what machines are reachable.

```
[root@servera ~]# nmap -sn 172.25.250.0/24

Starting Nmap 6.40 ( http://nmap.org ) at 2016-01-24 20:10 PST
Nmap scan report for serverb.lab.example.com (172.25.250.11)
Host is up (0.00016s latency).
MAC Address: 52:54:00:00:FA:0B (QEMU Virtual NIC)
Nmap scan report for 172.25.250.250
Host is up (0.00015s latency).
MAC Address: 52:54:00:CB:EE:60 (QEMU Virtual NIC)
Nmap scan report for workstation.lab.example.com (172.25.250.254)
Host is up (0.00021s latency).
MAC Address: 52:54:00:00:FA:09 (QEMU Virtual NIC)
Nmap scan report for servera.lab.example.com (172.25.250.10)
Host is up.
Nmap done: 256 IP addresses (4 hosts up) scanned in 1.94 seconds
```

The **-sn** option tells **nmap** to perform a ping scan. No port scans are performed.

5. Perform a port scan of **serverb**.

```
[root@servera ~]# nmap serverb.lab.example.com

Starting Nmap 6.40 ( http://nmap.org ) at 2016-01-24 20:12 PST
Nmap scan report for serverb.lab.example.com (172.25.250.11)
Host is up (0.00023s latency).
Not shown: 997 filtered ports
PORT    STATE SERVICE
22/tcp  open  ssh
80/tcp  open  http
443/tcp open  https
MAC Address: 52:54:00:00:FA:0B (QEMU Virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 5.08 seconds
```

By default, **nmap** performs an IPv4 port scan. The **-6** option causes it to scan ports of the IPv6 address.

```
[root@servera ~]# nmap -6 serverb.lab.example.com

Starting Nmap 6.40 ( http://nmap.org ) at 2016-01-24 20:15 PST
Nmap scan report for serverb.lab.example.com (fd37:5265:6448:6174::b)
Host is up (0.00020s latency).
Not shown: 997 filtered ports
PORT    STATE SERVICE
22/tcp  open  ssh
80/tcp  open  http
443/tcp open  https
MAC Address: 52:54:00:01:FA:0B (QEMU Virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 4.90 seconds
```

6. Another useful network testing tool is Netcat, **nc** or **ncat** for short. Install the *nmap-ncat* package on **servera**.

```
[root@servera ~]# yum -y install nmap-ncat
```

7. Use Netcat as an IPv4 client of the web server running on **serverb**. Connect to port 80, then issue the **GET** HTTP command. Netcat will display what is sent from the server, and the web server does not issue a prompt.

```
[root@servera ~]# nc serverb 80
GET
Main web page for serverb.
Ctrl+C
```

   Connect to port 80 of **serverb** with Netcat using IPv6. The **-6** option tells Netcat to use IPv6 instead of IPv4.

```
[root@servera ~]# nc -6 serverb.lab.example.com 80
GET
Main web page for serverb.
Ctrl+C
```

8. Netcat can also be used as a network service that displays whatever content it receives.

   8.1. Log in as **root** on **serverb**. TCP port 4231 was opened by the lab setup script, so launch Netcat as a server that will listen on port 4231. It should keep listening for multiple connections.

```
[root@serverb ~]# nc -l -k 4231
```

   8.2. From **servera**, use **nmap** to perform a port scan of **serverb**.

```
[root@servera ~]# nmap -p4000-4999 serverb

Starting Nmap 6.40 ( http://nmap.org ) at 2016-01-24 20:32 PST
Nmap scan report for serverb (172.25.250.11)
Host is up (0.00019s latency).
rDNS record for 172.25.250.11: serverb.lab.example.com
Not shown: 999 filtered ports
PORT     STATE SERVICE
4231/tcp open  unknown
MAC Address: 52:54:00:00:FA:0B (QEMU Virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 5.08 seconds
```

   8.3. From **servera**, use the **nc** client to connect to port 4231 of **serverb**. Type a line of text, then type **Ctrl**+**C** to exit.

```
[root@servera ~]# nc serverb 4231
this is a test
Ctrl+C
```

The text that you typed should be displayed on the screen of **serverb**.

8.4. By default, the Netcat server listens to the TCP port on both IPv4 and IPv6. Use the Netcat client from **servera** to connect to the Netcat server listening on port 4231 of **serverb** using IPv6.

```
[root@servera ~]# nc -6 serverb.lab.example.com 4231
another test
Ctrl+C
```

Again, the text that you typed should be displayed on the screen of **serverb**.

9. The *iptraf-ng* package is typically not installed by default. Use Yum to install it on your **servera** host.

```
[root@servera ~]# yum -y install iptraf-ng
```

10. Log in as **root** on the **servera** console (do not SSH into the server). Launch the **iptraf-ng** application.

```
[root@servera ~]# iptraf-ng
```

Select **IP traffic monitor** in the main menu. Use arrow keys to navigate, then hit **Enter** to choose the highlighted menu item.

When the **Select Interface** dialog appears, select **All interfaces** and hit **Enter** to start monitoring network traffic on all interfaces. The top frame shows all TCP connections. The bottom frame shows other types of packets. From **workstation**, SSH as **root** into **servera**.

```
[student@workstation ~]$ ssh root@servera
```

Information about the new connection will appear in the top frame. One of the addresses will be 172.25.250.10:22; this is the SSH service port of **servera**. The other address, 172.25.250.254, is **workstation**'s address and will have a random port assignment.

11. In the **servera** SSH session, use Yum to install the *lftp* package. Watch the **iptraf-ng** windows as the package is installed.

Another connection will appear in the top frame. One of the addresses in that connection will be 172.25.254.254:80. This is the connection used by Yum to download the *lftp* RPM package. Once the package is installed, the connection flag will change to **CLOS**.

Log out from the SSH session. The other TCP connection flag will change to **CLOS**.

12. From **workstation**, ping **servera**.

```
[student@workstation ~]$ ping servera.lab.example.com
```

Ping requests and replies will appear in the lower frame. You may see additional UDP traffic in the lower frame. This will probably be DNS resolution packets and NTP time coordination.

13. SSH from **serverb** to **servera** using IPv6. You will see the TCP connection to fd37:5265:6448:6174::a port 22 in the top frame.

```
[student@serverb ~]$ ssh -6 servera.lab.example.com
```

14. Instead of monitoring network traffic, display network interface statistics. Typing **X** will exit the current window in **iftraf-ng**. Type **X** to exit the monitor window, then select **General interface statistics**, or type **S** to display interface statistics.

    In the SSH window, generate some network traffic.

```
[student@servera ~]$ ls -R /usr
```

    Watch the traffic counters change. Total packets, IPv6 packets, will increase. Activity for **eth1** will rise, then go back to 0.00 kbps once the **ls** command completes.

    Type **X** to exit the window, then type **X** to exit the application and log out of the console.

15. Create another SSH session into **servera**. Log in as **root** and SSH in from **workstation** using IPv4. Launch **iptraf-ng** in that window.

```
[root@servera ~]# iptraf-ng
```

    Select **IP traffic monitor**, then select **All interfaces** from the **Select Interface** dialog.

    Observe how the counters increase continuously in the SSH TCP session in the upper frame. Network traffic generates a status update, which generates more network traffic.

16. Create a filter to exclude SSH traffic, but monitor all other network traffic.

    16.1. Type **X** to exit the network traffic monitor screen.

    16.2.Select **Filters...**, then select **IP...**. This screen allows you to create and manage IP network filters.

    16.3.Select **Define new filter...**. Enter “Exclude SSH” for the description, then hit **Enter**. An empty list of filtering rules will display.

    16.4.First, create a rule that will delete all incoming SSH traffic, TCP traffic destined for port 22.

    Type **A** to add a new rule to the list of rules for this filter. Enter “22” in the field for the **Destination** port. Type **Y** in the **TCP protocol** field. Type **E** in the **Include/Exclude** field

    Hit **Enter** to accept your changes.

    16.5.Create another rule that will include all other network traffic.

    Type **A** to add another rule to the list of rules for this filter. Enter **Y** in the **All IP** protocol field. Hit **Enter** to accept your changes.

    16.6.Type **X** to exit the rule definition screen.

17. Apply the new filter that has been created. Select **Apply filter...**, select **Exclude SSH** filter, then hit **Enter**.

    Type **X** to exit the **Filters** menu. In the **Filter Status** frame, **IP filter active** confirms the filter is in effect.

18. See the filter in action. Type **X** to exit the **Filter Status** menu. Select **IP traffic monitor**, then select **All interfaces** for the interface.

    The SSH TCP connection will not appear in the top frame. The **Packets captured** counter, displayed at the bottom of the screen, will still increase.

19. In the other SSH window to **servera**, use **lftp** to connect to **content.example.com**.

```
[student@servera ~]$ lftp http://content.example.com
cd ok, cwd=/
lftp content.example.com:/>
```

A temporary connection to 172.25.254.254:80 will appear in the top frame. This TCP connection was used for the HTTP connection. Since it did not match the first rule in the active filter, it displays.

20. In the **iptraf-ng** window, detach the **Exclude SSH** filter.

    20.1. Type **X** to go to the main menu.

    20.2Select **Filter...**, select **IP...**, then select **Detach filter**. **iptraf-ng** will report that the filter is deactivated. Hit any key to continue.

    20.3Type **X** to exit. The **Filter status** has changed to **No IP filter active**.

    20.4Type **X** to return to the main menu.

    20.5Select **IP traffic monitor**, then select **All interfaces**. Monitoring now shows the SSH connection running up because of the network traffic caused by the monitoring updates.

    20.6Type **X** to return to the main menu, then type **X** again to exit the application.

21. When you finish testing network connectivity, reset your machines to the state they had before starting the lab. Either reset your virtual machines, or run the following command on your **workstation** system.

```
[student@workstation ~]$ lab network-testing reset
```

Resolving Connectivity Issues

Objectives

After completing this section, students should be able to identify and fix network connectivity issues.

Resolving issues with network device naming

Red Hat Enterprise Linux 7 supports device names based on hardware attributes. This device naming approach is implemented on physical systems. It requires that the *biosdevname* package has been installed and the **biosdevname=1** kernel parameter has been specified. It is enabled by default on Dell systems.

On-board network devices are assigned names similar to **eno1**. **en** is the Ethernet prefix, and **o** stands for on-board, followed by a device number. PCI Express hotplug slot devices have names similar to **ens1**; **s** stands for slot. Other network devices have names based on physical location of the hardware connector, **enp2s0**. Traditional, unpredictable naming, such as **eth0**, is used if previous methods did not work.

> ### Note
>
> Network devices of virtual guests use traditional network device naming.

The **/etc/udev/rules.d/80-net-name-slot.rules** udev rules file implements the persistent naming. This file will be symbolically linked to **/dev/null** when traditional device naming is desired. System administrators can also write their own udev device naming rules in **/etc/udev/rules.d/70-persistent-net.rules**. These rules will override the default network device naming performed by **systemd**.

Log files provide useful information that helps troubleshoot network device naming issues. The **/var/log/messages** file has device detection information that is performed at boot time. It also records the messages that NetworkManager produces when it tries to activate the device. The following example shows **eth1** being successfully configured and activated.

```
[root@demo ~]# grep -n eth1 /var/log/messages
3904:Feb  3 05:29:25 host NetworkManager[610]: <info>  (eth1): new
 Ethernet device (carrier: OFF, driver: 'virtio_net', ifindex: 3)
3905:Feb  3 05:29:25 host NetworkManager[610]: <info>  (eth1): device
 state change: unmanaged -> unavailable (reason 'managed') [10 20 2]
3906:Feb  3 05:29:25 host kernel: IPv6: ADDRCONF(NETDEV_UP): eth1:
 link is not ready
3907:Feb  3 05:29:25 host NetworkManager[610]: <info>  (eth1): link
 connected
3914:Feb  3 05:29:25 host NetworkManager[610]: <info>  (eth1): device
 state change: unavailable -> disconnected (reason 'none') [20 30 0]
... Output omitted ...
3942:Feb  3 05:29:25 host NetworkManager[610]: <info>  (eth1): Activation:
 successful, device activated.
3954:Feb  3 05:29:25 host nm-dispatcher: Dispatching action 'up' for eth1
```

Correcting network device naming issues

A network interface device name can be assigned by setting values in the **ifcfg-*** configuration file. The MAC address must be matched with the **HWADDR** variable. The value of the **DEVICE** variable determines the name of the network interface device corresponding to the specified MAC address. The following example demonstrates how to change the name of the device that was originally called **eth1** to **test123**.

```
[root@demo ~]# cat /etc/sysconfig/network-scripts/ifcfg-eth1
DEVICE=test123
HWADDR=52:54:00:01:fa:0a
BOOTPROTO=static
ONBOOT=yes
TYPE=Ethernet
USERCTL=yes
PEERDNS=no
IPV6INIT=no
IPADDR=172.24.250.10
NETMASK=255.255.255.0
[root@demo ~]# ip addr show dev test123
3: test123: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
 state UP qlen 1000
    link/ether 52:54:00:01:fa:0a brd ff:ff:ff:ff:ff:ff
    inet 172.24.250.10/24 brd 172.24.250.255 scope global test123
       valid_lft forever preferred_lft forever
    inet6 fe80::5054:ff:fe01:fa0a/64 scope link
       valid_lft forever preferred_lft forever
```

Note that the file name needed to start with **ifcfg-**, but it did not have to match the specified device name, although that is a recommended practice. Once the changes were made to the configuration file, the system was rebooted to implement them.

Troubleshooting interface configuration problems

Once network device naming issues have been resolved, there are other network interface configuration settings that can impair network connectivity. Log files do not provide detailed information about the current network configuration. The **ip addr** command displays the active network settings for network interfaces.

```
[root@demo ~]# ip addr show dev eth1
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
 state UP qlen 1000
    link/ether 52:54:00:01:fa:0a brd ff:ff:ff:ff:ff:ff
    inet 172.24.250.10/24 brd 172.24.250.255 scope global eth1
       valid_lft forever preferred_lft forever
    inet6 fe80::5054:ff:fe01:fa0a/64 scope link
       valid_lft forever preferred_lft forever
```

The **ip route** command displays the current routing table. The **default** route should be checked to ensure it is the correct gateway address for this subnet.

```
[root@demo ~]# ip route
default via 172.25.250.254 dev eth0  proto static  metric 100
172.24.250.0/24 dev eth1  proto kernel  scope link  src 172.24.250.10
 metric 100
172.25.250.0/24 dev eth0  proto kernel  scope link  src 172.25.250.10
 metric 100
```

If Dynamic Host Configuration Protocol (DHCP) is used to configure a network interface, then NetworkManager launches **dhclient** to handle that interface's configuration. Interface network settings, such as IP address, network mask, the default gateway, and DNS servers, are determined by the values provided by the DHCP server. When a network interface uses static addressing, the system administrator must correctly specify these values.

Inspecting network interface configuration files

Static IPv4 and IPv6 network configuration settings are defined in the **/etc/sysconfig/network-scripts/ifcfg-*** files. When NetworkManager is not used, the values in these files determine the persistent settings for network interfaces. In this case, network configuration changes must be made to these files, then the network should be restarted to apply the changes.

```
[root@demo ~]# cat /etc/sysconfig/network-scripts/ifcfg-eth1
DEVICE=eth1
BOOTPROTO=static
ONBOOT=yes
TYPE=Ethernet
USERCTL=yes
PEERDNS=no
IPV6INIT=yes
IPV6_AUTOCONF=no
IPV6ADDR=fd37:5265:6448:6174::2931/64
IPADDR=192.168.25.31
NETMASK=255.255.255.0
```

The **BOOTPROTO** variable determines what type of IPv4 configuration is performed on the network interface. A value of **none**, or **static**, indicates IPv4 networking will be statically configured. The **dhcp** or **bootp** values cause **dhclient** to be used to manage the interface configuration. The variables to set for static IPv4 configuration include **IPADDR0** and **PREFIX0**. For backwards compatibility the older **IPADDR** and **GATEWAY** settings are also supported. When the network interface connects to the external network, the **GATEWAY** variable should have the IP address of the default gateway.

The **IPV6INIT** variable must have the value **yes** for any IPv6 configuration to take place on the network interface. **IPV6ADDR** specifies the primary static IPv6 address and netmask. Additional IPv6 addresses can be assigned with the **IPV6_SECONDARIES** variable. The **IPV6_AUTOCONF** variable determines if IPv6 autoconfiguration is performed on the network interface.

Managing NetworkManager configuration settings

When NetworkManager is used to manage network interfaces on a system, the **nmcli** command can be used to display and change network settings. **nmcli conn show '*CONNECTION_NAME*'** displays the current network configuration for a NetworkManager connection. Values that start with **ipv4** or **ipv6** correspond to the IPv4 or the IPv6 configuration of an interface respectively.

```
[root@demo ~]# nmcli conn show 'System eth0' | grep ipv
ipv4.method:                            manual
ipv4.dns:                               172.25.250.254
ipv4.dns-search:                        example.com
ipv4.addresses:                         172.25.250.11/24
ipv4.gateway:                           172.25.250.254
ipv4.routes:
ipv4.route-metric:                      -1
ipv4.ignore-auto-routes:                no
ipv4.ignore-auto-dns:                   no
ipv4.dhcp-client-id:                    --
ipv4.dhcp-send-hostname:                yes
```

```
ipv4.dhcp-hostname:                     --
ipv4.never-default:                     no
ipv4.may-fail:                          yes
ipv6.method:                            manual
ipv6.dns:
ipv6.dns-search:
ipv6.addresses:                         fd37:5265:6448:6174::b/64
ipv6.gateway:                           --
ipv6.routes:
ipv6.route-metric:                      -1
ipv6.ignore-auto-routes:                no
ipv6.ignore-auto-dns:                   no
ipv6.never-default:                     no
ipv6.may-fail:                          yes
ipv6.ip6-privacy:                       -1 (unknown)
ipv6.dhcp-send-hostname:                yes
ipv6.dhcp-hostname:                     --
```

The **ipv4.method** and **ipv6.method** settings define how IPv4 and IPv6 addressing is configured on the interface. A value of **auto** indicates DHCP will be used for network configuration. When **manual** is specified for an IP protocol, then IP information must be specified with **addresses** and **gateway** settings. NetworkManager updates the network interface configuration files when **nmcli** is used to change and assign values to these settings. Another approach would be to change the configuration files in **/etc/sysconfig/network-scripts**, then use **nmcli conn reload** to have NetworkManager read them for configuration changes.

NetworkManager network interface configuration changes are not applied to an interface if it is already active. In this case, the interface must be bounced, brought down and back up, to make the configuration changes go into effect.

```
[root@demo ~]# nmcli conn down 'System eth1' ; nmcli conn up 'System eth1'
Connection 'System eth1' successfully deactivated (D-Bus active path:
 /org/freedesktop/NetworkManager/ActiveConnection/2)
Connection successfully activated (D-Bus active path:
 /org/freedesktop/NetworkManager/ActiveConnection/3)
```

Restarting affected network services

Network services can still be unavailable when network interface configuration changes have been made. The affected services should be restarted so they listen to the new address on a changed network interface.

```
[root@demo ~]# systemctl restart httpd
```

Resolving DNS resolution problems

Client-side DNS resolution problems can often cause services to be sluggish or unavailable. For example, the SSH daemon has a **UseDNS** SSH setting. It determines whether **sshd** should check if the host name matches the IP address of incoming connections. This feature is enabled by default in Red Hat Enterprise Linux.

The main configuration file for DNS resolution is **/etc/resolv.conf**. Displaying this file identifies the current DNS servers. The nameservers listed can be explicitly queried directly with the **host** command to confirm they are reachable and working.

```
[user@demo ~]$ cat /etc/resolv.conf
# Generated by NetworkManager
```

```
search example.com
nameserver 10.1.1.2
nameserver 10.1.1.3
[user@demo ~]$ host puppet.example.com 10.1.1.2
Using domain server:
Name: 10.1.1.2
Address: 10.1.1.2#53
Aliases:

puppet.example.com has address 192.168.2.43
```

Although **/etc/resolv.conf** is a good starting point for DNS troubleshooting, this file is often managed by other scripts. The **DNS** variables defined in the network interface files, **/etc/sysconfig/network-scripts/ifcfg-***, can define the **nameserver** entries in **/etc/resolv.conf**. The **PEERDNS** variable must be set to **yes** for these settings to be applied.

The following network configuration settings illustrate how DNS values are assigned on this host. Since **PEERDNS** is set to **yes** in **ifcfg-eth0**, the IP addresses defined by **DNS1** and **DNS2** define the **nameserver** entries in **/etc/resolv.conf**. The configuration file for **eth1** does not affect DNS configuration in any way.

```
[user@demo ~]$ grep -i dns /etc/sysconfig/network-scripts/ifcfg-*
/etc/sysconfig/network-scripts/ifcfg-eth0:PEERDNS=yes
/etc/sysconfig/network-scripts/ifcfg-eth0:DNS1=10.1.1.2
/etc/sysconfig/network-scripts/ifcfg-eth0:DNS2=10.1.1.3
/etc/sysconfig/network-scripts/ifcfg-eth1:PEERDNS=no
```

Note

The **PEERDNS=yes** setting also configures **dhclient** to modify **/etc/resolv.conf** when using DHCP to assign network values.

When NetworkManager is active, it makes changes to the network interface configuration files in **/etc/sysconfig/network-scripts**. In these cases, DNS configuration changes need to be made to NetworkManager's configuration information. This is accomplished using the **nmcli** command.

```
[root@demo ~]# nmcli conn show 'System eth0' | grep dns
ipv4.dns:                               10.1.1.2,10.1.1.3
ipv4.dns-search:                        example.com
ipv4.ignore-auto-dns:                   no
ipv6.dns:
ipv6.dns-search:
ipv6.ignore-auto-dns:                   no
[root@demo ~]# nmcli conn mod 'System eth0' ipv4.dns '10.1.1.2'
[root@demo ~]# nmcli conn show 'System eth0' | grep dns
ipv4.dns:                               10.1.1.2
ipv4.dns-search:                        example.com
ipv4.ignore-auto-dns:                   no
ipv6.dns:
ipv6.dns-search:
ipv6.ignore-auto-dns:                   no
```

Although the configuration change has been made to NetworkManager, **/etc/resolv.conf** still has its original configuration. The **nmcli** command must be used to bounce the network connection to apply the configuration changes.

```
[root@demo ~]# cat /etc/resolv.conf
# Generated by NetworkManager
search example.com
nameserver 10.1.1.2
nameserver 10.1.1.3
[root@demo ~]# nmcli conn down 'System eth0' ; nmcli conn up 'System eth0'
Connection 'System eth0' successfully deactivated (D-Bus active path:
 /org/freedesktop/NetworkManager/ActiveConnection/2)
Connection successfully activated (D-Bus active path:
 /org/freedesktop/NetworkManager/ActiveConnection/8)
[root@demo ~]# cat /etc/resolv.conf
# Generated by NetworkManager
search example.com
nameserver 10.1.1.2
```

Troubleshooting firewall issues

Once basic network connectivity has been confirmed, a network service may be unavailable because of firewall problems. The firewall rules on the server hosting the network service may have blocked the ports used to access the service. Less frequently, firewall issues can sometimes be a problem on the client machines.

The **`firewall-cmd`** command can be used to display and adjust current firewall settings. Invoking the command with the **`--list-all-zones`** option gives a complete overview of **`firewalld`** configuration.

```
[root@demo ~]# firewall-cmd --list-all-zones
... Output omitted ...
internal
  interfaces:
  sources:
  services: dhcpv6-client ipp-client mdns samba-client ssh
  ports:
  masquerade: no
  forward-ports:
  icmp-blocks:
  rich rules:

public (default, active)
  interfaces: eth0 eth1
  sources:
  services: dhcpv6-client http https ssh
  ports:
  masquerade: no
  forward-ports:
  icmp-blocks:
  rich rules:
... Output omitted ...
```

Using the **`--permanent`** option with **`firewall-cmd`** displays the persistent settings. Comparing active **`firewalld`** settings with persistent settings can sometimes identify potential problems. The preceding sample output shows that the ports for the **`http`** and **`https`** services are not blocked. The following output demonstrates that their current configuration is not persistent.

```
[root@demo ~]# firewall-cmd --list-all-zones --permanent
... Output omitted ...
internal
  interfaces:
```

```
  sources:
  services: dhcpv6-client ipp-client mdns samba-client ssh
  ports:
  masquerade: no
  forward-ports:
  icmp-blocks:
  rich rules:

public (default)
  interfaces:
  sources:
  services: dhcpv6-client ssh
  ports:
  masquerade: no
  forward-ports:
  icmp-blocks:
  rich rules:
... Output omitted ...
```

A quick way to convert runtime rules into permanenet rules is with the command **firewall-cmd --runtime-to-permanent**

```
[root@demo ~]# firewall-cmd --runtime-to-permanent
success
```

Firewalld is built on top of Netfilter functionality. This means **iptables** can be used to get firewall information at a lower level than **firewall-cmd**.

```
[root@demo ~]# iptables -L -t filter -n
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0            ctstate
 RELATED,ESTABLISHED
ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0
INPUT_direct  all  --  0.0.0.0/0            0.0.0.0/0
INPUT_ZONES_SOURCE  all  --  0.0.0.0/0            0.0.0.0/0
INPUT_ZONES  all  --  0.0.0.0/0            0.0.0.0/0
ACCEPT     icmp --  0.0.0.0/0            0.0.0.0/0
REJECT     all  --  0.0.0.0/0            0.0.0.0/0            reject-with
 icmp-host-prohibited

... Output omitted ...

Chain INPUT_ZONES (1 references)
target     prot opt source               destination
IN_public  all  --  0.0.0.0/0            0.0.0.0/0           [goto]
IN_public  all  --  0.0.0.0/0            0.0.0.0/0           [goto]
IN_public  all  --  0.0.0.0/0            0.0.0.0/0           [goto]

... Output omitted ...

Chain IN_public (3 references)
target     prot opt source               destination
IN_public_log  all  --  0.0.0.0/0            0.0.0.0/0
IN_public_deny  all  --  0.0.0.0/0            0.0.0.0/0
IN_public_allow  all  --  0.0.0.0/0            0.0.0.0/0

Chain IN_public_allow (1 references)
target     prot opt source               destination
ACCEPT     tcp  --  0.0.0.0/0            0.0.0.0/0            tcp dpt:22
 ctstate NEW
```

```
ACCEPT     tcp  --  0.0.0.0/0            0.0.0.0/0            tcp dpt:80
 ctstate NEW
ACCEPT     tcp  --  0.0.0.0/0            0.0.0.0/0            tcp dpt:443
 ctstate NEW

... Output omitted ...
```

Similarly, the **ip6tables** command displays and manages the IPv6 firewall tables.

References

For more information, see the

> Red Hat Enterprise Linux 7 Networking Guide - Consistent Network Device Naming
> https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Networking_Guide/ch-Consistent_Network_Device_Naming.html

biosdevname(1), **firewall-cmd**(1), **iptables**(8), **nmcli**(1), **nmcli-examples**(5), and **resolv.conf**(5) man pages

Guided Exercise: Resolving Connectivity Issues

In this lab, you will diagnose and resolve network connectivity issues.

Resources	
Machines	• **servera** • **serverb**

Outcome(s)

You should be able to diagnose and resolve network connectivity issues on a Red Hat Enterprise Linux host.

Before you begin

Set up your systems for this exercise by running the **lab network-fix setup** command on your **workstation** system. This will configure IPv6 on both **servera** and **serverb** and break some of the network connectivity of **servera**.

```
[student@workstation ~#$ lab network-fix setup
```

A second network interface has been added to **servera**, but it does not work properly. It is connected to a private network and it should have the following IP address assignments:

- IPv4 address = 172.24.250.10/24
- IPv6 address = fd37:5265:6448:6174::a/64

DNS stopped working when the second interface was configured. 172.24.250.254 is the IP address of a working DNS server.

Your task is to correct the network configuration issues on **servera**. The network settings should persist when the machine is rebooted. When the network is functioning properly, **servera** should be able to ping the IPv6 address of **serverb.lab.example.com** using its host name.

```
[root@servera ~]# ping6 serverb.lab.example.com
```

1. Log into **servera** as **root**. Try to ping **serverb** over IPv6 using its host name to confirm it does not work.

```
[root@servera ~]# ping6 serverb.lab.example.com
unknown host
```

2. Gather more system information.

   2.1. Use the **ip** command to display the current IP settings for **servera**.

```
[root@servera ~]# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
 qlen 1000
    link/ether 52:54:00:00:fa:0a brd ff:ff:ff:ff:ff:ff
    inet 172.25.250.10/24 brd 172.25.250.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::5054:ff:fe00:fa0a/64 scope link
       valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
 qlen 1000
    link/ether 52:54:00:01:fa:0a brd ff:ff:ff:ff:ff:ff
```

You observe that the **eth0** interface is up with an assigned IPv4 address and a link-local IPv6 address, but **eth1** does not have any IP addresses assigned to it.

2.2. Use **nmcli** to display NetworkManager's network configurations. Display both connection and device information.

```
[root@servera ~]# nmcli conn
NAME           UUID                                  TYPE            DEVICE
System enp2s0  8c6fd7b1-ab62-a383-5b96-46e083e04bb1  802-3-ethernet  --
System eth0    5fb06bd0-0bb0-7ffb-45f1-d6edd65f3e03  802-3-ethernet  eth0
[root@servera ~]# nmcli dev
DEVICE  TYPE      STATE         CONNECTION
eth0    ethernet  connected     System eth0
eth1    ethernet  disconnected  --
lo      loopback  unmanaged     --
```

NetworkManager has two connections: one assigned to **eth0**, and the other is called **System enp2s0** without a device assigned to it. It appears that there is a device/configuration naming issue.

2.3. Check the boot messages in **/var/log/messages** to see if there are other clues. Jump to the bottom of the file, then scroll up until you see the network initialization messages.

```
Feb  1 09:39:34 servera systemd: Starting Network Manager...
Feb  1 09:39:34 servera NetworkManager[1549]: <info>  NetworkManager
 (version 1.0.6-27.el7) is starting...
Feb  1 09:39:34 servera NetworkManager[1549]: <info>  Read config:
 /etc/NetworkManager/NetworkManager.conf and conf.d: 00-server.conf,
 10-ibft-plugin.conf
Feb  1 09:39:35 servera systemd: Started Network Manager.
Feb  1 09:39:35 servera systemd: Starting Network Manager Wait Online...
Feb  1 09:39:35 servera NetworkManager[1549]: <info>  Loaded
 settings plugin ifcfg-rh: (c) 2007 - 2015 Red Hat, Inc.
 To report bugs please use the NetworkManager mailing
 list. (/usr/lib64/NetworkManager/libnm-settings-plugin-ifcfg-rh.so)
Feb  1 09:39:35 servera NetworkManager[1549]: <info>
 Loaded settings plugin iBFT: (c) 2014 Red Hat, Inc.
 To report bugs please use the NetworkManager mailing list.
 (/usr/lib64/NetworkManager/libnm-settings-plugin-ibft.so)
Feb  1 09:39:35 servera NetworkManager[1549]: <info>  Loaded settings
 plugin keyfile: (c) 2007 - 2015 Red Hat, Inc.  To report bugs please
 use the NetworkManager mailing list.
Feb  1 09:39:35 servera NetworkManager[1549]: <info>  ifcfg-rh:
 new connection /etc/sysconfig/network-scripts/ifcfg-enp2s0
 (8c6fd7b1-ab62-a383-5b96-46e083e04bb1,"System enp2s0")
Feb  1 09:39:35 servera NetworkManager[1549]: <info>  ifcfg-rh:
 new connection /etc/sysconfig/network-scripts/ifcfg-eth0
```

```
  (5fb06bd0-0bb0-7ffb-45f1-d6edd65f3e03,"System eth0")
```

You have found more useful evidence. There is no device that matches the **enp2s0** interface.

3. Check out the current status of the network interface configuration files.

   3.1. Change to the **/etc/sysconfig/network-scripts** directory where the network interface configuration files are kept. List the directory to see what configuration files exist.

   ```
   [root@servera ~]# cd /etc/sysconfig/network-scripts
   [root@servera network-scripts]# ls ifcfg-*
   ifcfg-enp2s0  ifcfg-eth0  ifcfg-lo
   ```

   3.2. Display the contents of the **enp2s0** network interface.

   ```
   [root@servera network-scripts]# cat ifcfg-enp2s0
   DEVICE=en2ps0
   BOOTPROTO=static
   ONBOOT=yes
   TYPE=Ethernet
   USERCTL=yes
   PEERDNS=no
   IPV6INIT=yes
   IPV6_AUTOCONF=no
   IPV6ADDR=fd37:5265:6448:6174::a/64
   IPADDR=172.24.250.10
   NETMASK=255.255.255.0
   ```

   The network settings look correct, but the file defines them for an interface that does not exist. The second network interface on this system is known as **eth1**, not **en2ps0**.

4. Adjust the network configuration files to reflect the network devices on the system.

   4.1. Make a copy of the original **ifcfg-enp2s0** file before making changes. The copy cannot be in the **/etc/sysconfig/network-scripts** directory, otherwise it will stay in effect.

   ```
   [root@servera network-scripts]# cp ifcfg-enp2s0 ~
   ```

   4.2. Change the name of the configuration file so its name references the **eth1** network interface.

   ```
   [root@servera network-scripts]# mv ifcfg-enp2s0 ifcfg-eth1
   ```

   4.3. Change the contents of the file so the **DEVICE** variable references the correct network interface, **eth1**. The resulting **ifcfg-eth1** file should have the following contents.

   ```
   DEVICE=eth1
   BOOTPROTO=static
   ONBOOT=yes
   TYPE=Ethernet
   USERCTL=yes
   ```

```
PEERDNS=no
IPV6INIT=yes
IPV6_AUTOCONF=no
IPV6ADDR=fd37:5265:6448:6174::a/64
IPADDR=172.24.250.10
NETMASK=255.255.255.0
```

5. Use the **nmcli** command to have NetworkManager reread the network interface configuration files.

```
[root@servera network-scripts]# nmcli conn reload
[root@servera network-scripts]# nmcli conn
NAME         UUID                                  TYPE            DEVICE
System eth1  9c92fad9-6ecb-3e6c-eb4d-8a47c6f50c04  802-3-ethernet  eth1
System eth0  5fb06bd0-0bb0-7ffb-45f1-d6edd65f3e03  802-3-ethernet  eth0
```

The network connections appear to be correct. Use the **ip** command to confirm the network settings are in effect.

```
[root@servera network-scripts]# ip addr show dev eth1
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen
 1000
    link/ether 52:54:00:01:fa:0a brd ff:ff:ff:ff:ff:ff
    inet 172.24.250.10/24 brd 172.24.250.255 scope global eth1
       valid_lft forever preferred_lft forever
    inet6 fd37:5265:6448:6174::a/64 scope global
       valid_lft forever preferred_lft forever
    inet6 fe80::5054:ff:fe01:fa0a/64 scope link
       valid_lft forever preferred_lft forever
```

6. Ping **serverb** using its IPv6 address. This works, so no further changes will be done here. Change back to your home directory.

```
[root@servera network-scripts]# ping6 -c 1 fd37:5265:6448:6174::b
PING fd37:5265:6448:6174::b(fd37:5265:6448:6174::b) 56 data bytes
64 bytes from fd37:5265:6448:6174::b: icmp_seq=1 ttl=64 time=0.272 ms

--- fd37:5265:6448:6174::b ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.272/0.272/0.272/0.000 ms
[root@servera network-scripts]# cd
```

7. Try to ping **serverb** using its host name. Name resolution is still broken, so gather system information that will help you troubleshoot the problem.

```
[root@servera ~]# ping6 serverb.lab.example.com
unknown host
```

7.1. Display the contents of **/etc/resolv.conf**. It defines how client-side DNS behaves.

```
[root@servera ~]# cat /etc/resolv.conf
# Generated by NetworkManager
search lab.example.com example.com
nameserver 172.25.250.11
nameserver 172.25.250.10
nameserver 127.0.0.1
# NOTE: the libc resolver may not support more than 3 nameservers.
# The nameservers listed below may not be recognized.
nameserver 172.25.250.254
```

7.2. Use the **host** command to query the hosts listed in **/etc/resolv.conf**.

```
[root@servera ~]# host -v -t aaaa serverb.lab.example.com 172.25.250.11
Trying "serverb.lab.example.com"
;; connection timed out; trying next origin
Trying "serverb.lab.example.com.lab.example.com"
;; connection timed out; trying next origin
Trying "serverb.lab.example.com.example.com"
;; connection timed out; no servers could be reached
[root@servera ~]# host -v -t aaaa serverb.lab.example.com 172.25.250.254
Trying "serverb.lab.example.com"
Using domain server:
Name: 172.25.250.254
Address: 172.25.250.254#53
Aliases:

;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 64858
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;serverb.lab.example.com. IN AAAA

;; ANSWER SECTION:
serverb.lab.example.com. 3600 IN AAAA fd37:5265:6448:6174::b

Received 69 bytes from 172.25.250.254#53 in 0 ms
```

All of them fail, except the last one in the file. Notice that NetworkManager included a comment in the file that suggests the fourth entry might be ignored.

7.3. Use the **nmcli** command to check how DNS is configured. Since **eth0** is the external network interface for **servera**, check its configuration.

```
[root@servera ~]# nmcli conn show 'System eth0' | grep -i dns
ipv4.dns:
 172.25.250.11,172.25.250.10,127.0.0.1,172.25.250.254
ipv4.dns-search:                        lab.example.com,example.com
ipv4.ignore-auto-dns:                   no
ipv6.dns:
ipv6.dns-search:
ipv6.ignore-auto-dns:                   no
IP4.DNS[1]:                             172.25.250.11
IP4.DNS[2]:                             172.25.250.10
IP4.DNS[3]:                             127.0.0.1
IP4.DNS[4]:                             172.25.250.254
```

Observe that the **ipv4.dns** setting agrees with the DNS **nameserver** entries defined in **/etc/resolv.conf**.

8. Correct the DNS settings defined in NetworkManager. Any changes made to **/etc/resolv.conf** will be temporary, since NetworkManager manages that file.

```
[root@servera ~]# nmcli conn mod 'System eth0' ipv4.dns '172.25.250.254'
[root@servera ~]# nmcli conn show 'System eth0' | grep -i dns
ipv4.dns:                               172.25.250.254
ipv4.dns-search:                        lab.example.com,example.com
ipv4.ignore-auto-dns:                   no
ipv6.dns:
ipv6.dns-search:
ipv6.ignore-auto-dns:                   no
IP4.DNS[1]:                             172.25.250.11
IP4.DNS[2]:                             172.25.250.10
IP4.DNS[3]:                             127.0.0.1
IP4.DNS[4]:                             172.25.250.254
```

NetworkManager updated the network interface configuration file for **eth0** with the change.

```
[root@servera ~]# grep -i dns /etc/sysconfig/network-scripts/ifcfg-eth0
DNS1=172.25.250.254
```

9. Try to ping **serverb** using its host name. DNS is still not functioning.

```
[root@servera ~]# ping6 serverb.lab.example.com
unknown host
```

Despite the NetworkManager changes, **/etc/resolv.conf** still references the original nameservers.

```
[root@servera ~]# cat /etc/resolv.conf
# Generated by NetworkManager
search lab.example.com example.com
nameserver 172.25.250.11
nameserver 172.25.250.10
nameserver 127.0.0.1
# NOTE: the libc resolver may not support more than 3 nameservers.
# The nameservers listed below may not be recognized.
nameserver 172.25.250.254
```

10. Bounce the NetworkManager connection for **System eth0**. This will apply the changes to the connection. If you used SSH to connect to the host, be sure to bring the connection up with the same command line.

```
[root@servera ~]# nmcli conn down 'System eth0' ; nmcli conn up 'System eth0'
Connection 'System eth0' successfully deactivated (D-Bus active path:
 /org/freedesktop/NetworkManager/ActiveConnection/2)
Connection successfully activated (D-Bus active path:
 /org/freedesktop/NetworkManager/ActiveConnection/3)
```

Display **/etc/resolv.conf** and confirm the DNS nameserver has been changed.

```
[root@servera ~]# cat /etc/resolv.conf
# Generated by NetworkManager
search lab.example.com example.com
nameserver 172.25.250.254
```

11. Use **ping6** to ping **serverb** using its host name.

```
[root@servera ~]# ping6 -c 1 serverb.lab.example.com
PING serverb.lab.example.com(serverb.lab.example.com) 56 data bytes
64 bytes from serverb.lab.example.com: icmp_seq=1 ttl=64 time=0.353 ms

--- serverb.lab.example.com ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.353/0.353/0.353/0.000 ms
```

Reboot **servera** and confirm the network settings are persistent.

```
[root@servera ~]# reboot
```

Wait for the reboot to finish before moving on to the grading step.

12. Once you are done, verify your work by running the **lab network-fix grade** command from your **workstation** machine.

```
[student@workstation ~]$ lab network-fix grade
```

13. After grading your systems, successfully reset your machines to the state they had before starting the lab, either by resetting your virtual machines or by running the following command on your **workstation** system.

```
[student@workstation ~]$ lab network-fix reset
```

Inspecting Network Traffic

Objectives

After completing this section, students should be able to inspect network traffic to aid troubleshooting.

Inspecting network traffic with Wireshark

Wireshark is an open source, graphical application for capturing, filtering, and inspecting network packets. It was formerly called Ethereal, but because of trademark issues, the project name was changed. Wireshark can perform promiscuous packet sniffing when network interface controllers support it. Packets are colorized for easy identification.

Protocol	Color
HTTP	Light green
TCP	Grey
UDP	Light blue
ARP	Light yellow
ICMP	Pink
Errors	Black

Red Hat Enterprise Linux 7 includes the *wireshark-gnome* package. This provides Wireshark functionality on a system installed with X.

```
[root@demo ~]# yum -y install wireshark-gnome
```

Once Wireshark is installed, it can be launched by selecting **Applications** > **Internet** > **Wireshark Network Analyzer** from the GNOME desktop. It can also be launched directly from a shell.

```
[root@demo ~]# wireshark
```

Capturing packets with Wireshark

Wireshark can capture network packets. It must be executed by **root** to capture packets, because direct access to the network interfaces requires **root** privilege. The **Capture** top-level menu permits the user to start and stop Wireshark from capturing packets. It also allows the administrator to select the interfaces to capture packets on. The **any** interface matches all of the network interfaces.

Once packet capturing has been stopped, the captured network packets can be written to a file for sharing, or later analysis. The **File** > **Save** or **File** > **Save as...** menu items allow the user to specify the file to save the packets into. Wireshark supports a variety of file formats.

Wireshark can also be used to analyze captured packets previously saved in files. Analyzing packets from existing capture files does not require **root** privilege. The **File** > **Open...** menu item selects the file to load, or the user can specify the file as an option when launching **wireshark** from the command line.

```
[user@demo ~]$ wireshark -r yesterday-eth0,pcapng &
```

Inspecting packets with Wireshark

The **Filter:** box allows the user to enter expressions to limit which packets are displayed in Wireshark. Wireshark recognizes a variety of network protocols, both low and high level. Entering "http" will filter the captured packets so only HTTP TCP packets will be displayed. Typing "ip" or "ipv6" will filter packets so only IPv4 or IPv6 packets respectively will be displayed.

The **Expression...** button takes users to a dialog box that helps them create more robust filtering expressions. The expression **`ip.src == 192.168.10.23`** will filter the packets so only packets originating from 192.168.10.23 will be displayed.

The frames in Wireshark's main display allows the user to inspect packet contents. The top frame displays the list of captured packet headers that have been selected with the current filter. The highlighted packet is the one currently being displayed in the middle and bottom frames. The bottom frame displays the whole packet, both headers and data, in hexadecimal and ASCII format.

The middle frame displays the packet as it has been parsed by Wireshark. Each network layer header is displayed in a brief human-readable format. Expanding each header allows the user to look at more detailed information about that network layer data. Each line starts with the header field name followed by its value. Wireshark translates values to strings when it can. For example, when 22 is the value of a TCP port address, it is displayed as "ssh" with the raw value of 22 in parentheses. As each field is selected, the corresponding raw data is highlighted in the bottom frame.

Wireshark can also display related packets between a protocol client and server in a much more readable format. This is accomplished by right-clicking a packet, then selecting **Follow TCP Stream**. Messages from the client display in one color and responses from the server display in another color.

Capturing packets with tcpdump

The **`tcpdump`** command can be used to capture and display packets in a nongraphical environment. It is provided by the *tcpdump* package and is installed on Red Hat Enterprise Linux 7 Server systems by default.

When invoked without arguments, **`tcpdump`** captures and displays brief information about all of the packets departing and arriving on the primary network interface. Execution continues until the user interrupts the program by typing **`Ctrl`**+**`C`**. The **`-c`** ***`COUNT`*** option causes **`tcpdump`** to exit after *COUNT* packets have been captured. When the program exits, it displays a summary of the number of packets captured by the program.

```
[root@demo ~]# tcpdump -c 3
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
10:17:58.499620 IP servera.lab.example.com.ssh > 172.25.250.250.56573:
 Flags [P.], seq 861940776:861940972, ack 498126305, win 325, options
 [nop,nop,TS val 56643816 ecr 1185210527], length 196
10:17:58.499680 IP 172.25.250.250.56573 > servera.lab.example.com.ssh:
 Flags [.], ack 196, win 322, options [nop,nop,TS val 1185210632 ecr
 56643816], length 0
10:17:58.499985 IP servera.lab.example.com.42442 >
 workstation.lab.example.com.domain: 670+ PTR?
 250.250.25.172.in-addr.arpa. (45)
3 packets captured
4 packets received by filter
0 packets dropped by kernel
```

tcpdump displays packet information to standard output, unless the **-w** ***FILE*** option is used. The **-w** option causes the packets to be written to the file specified as raw data for later analysis. It is a good practice to name packet capture files with a **.pcap** extension, although **tcpdump** does not require it. **tcpdump** can be invoked with a **-r** ***FILE*** option to read from a capture file, instead of listening to network interfaces. Using **tcpdump** to process existing capture files does not require **root** privileges.

```
[user@demo ~]$ tcpdump -r http-test.pcap
reading from file http-test.pcap, link-type EN10MB (Ethernet)
11:52:54.699448 IP servera.lab.example.com.51091 >
 serverb.lab.example.com.http: Flags [S], seq 981393578, win 29200,
 options [mss 1460,sackOK,TS val 103035086 ecr 0,nop,wscale 7], length 0
11:52:54.699499 IP serverb.lab.example.com.http >
... Output omitted ...
```

tcpdump can filter the packets that it captures based on an expression passed as the argument. The following command only captures packets coming to or from the host named **ntpserver**.

```
[root@demo ~]# tcpdump 'host ntpserver'
```

The following **tcpdump** command captures ICMP packets coming to and from 192.168.32.7.

```
[root@demo ~]# tcpdump 'icmp and host 192.168.32.7'
```

More complex expressions can be used with logical operators. The following **tcpdump** command filters all IP packets between the **matrix** host and any other host, except **human175**.

```
[root@demo ~]# tcpdump 'ip host matrix and not human175'
```

More information about the syntax for **tcpdump** filter expressions can be found in the **pcap-filter**(7) man page.

The **tcpdump** command also has options that control what information it displays about each captured packet. The **-x** option causes **tcpdump** to display all of the packet header and data information as hexadecimal values. The **-X** option displays the data as hexadecimal and ASCII values. This is useful if the higher-level network protocol includes text commands. The packet capture files created by **tcpdump** can also be read and parsed by Wireshark for further parsing and analysis.

Note

Wireshark also has a text-based version, called **tshark**. This program is provided by the *wireshark* package.

References

pcap-filter(7), **tcpdump**(8), **tcpslice**(8), **tshark**(1), and **wireshark-filter**(4) man pages

Guided Exercise: Inspecting Network Traffic

In this lab, you will install Wireshark and use it to capture, filter, and inspect network packets. You will also use **tcpdump** to perform similar functions in a textual environment.

Resources	
Application URL	• **http://materials.example.com** • **http://materials.example.com/rh342-practice.tcpdump** • **http://serverb.lab.example.com**
Files	• **http-test.tcpdump**. • **practice.pcapng** • **rh342-practice.tcpdump**
Machines	• **servera** • **serverb** • **workstation**

Outcome(s)
You should be able to capture, filter, and inspect network packets using Wireshark and **tcpdump**.

Before you begin
Set up your systems for this exercise by running the **lab network-traffic setup** command on your **workstation** system. This will install and configure a web server on **serverb**.

```
[student@workstation ~#$ lab network-traffic setup
```

1. Log in as **student** on **workstation**. Open a terminal window as **root** and install the packages that provide Wireshark.

```
[root@workstation ~]# yum -y install wireshark-gnome
```

2. Launch Wireshark as **root** and configure it to capture packets on all network interfaces.

   2.1. As **root**, use the shell to launch Wireshark.

```
[root@workstation ~]# wireshark &
```

   2.2. Select **Capture** > **Interfaces...** to open the **Wireshark: Capture Interfaces** dialog. Click the checkbox next to the **any** interface, then click the **Start** button to start capturing packets.

3. Generate some network traffic to **servera** from **workstation**.

   3.1. Send a few ICMP requests.

```
[root@workstation ~]# ping servera.lab.example.com
```

3.2. Generate some NTP traffic. Use **`ntpdate`** to update the system clock.

```
[root@workstation ~]# ntpdate classroom.example.com
```

3.3. Generate some HTTP network traffic. Launch **Firefox** and browse **`http://materials.example.com`**.

4. Stop Wireshark from capturing more network packets. Select **Capture** > **Stop**, or type **`Ctrl`**+**`E`**.

5. Perform some simple filtering and inspection on the captured network traffic.

5.1. Filter the packets so only the ICMP traffic is visible. Enter **`icmp`** in the **Filter:** box.
Use the middle frame to inspect the different header and data values for some of the packets displayed.

5.2. Filter the packets so only the NTP traffic is visible. Enter **`ntp`** in the **Filter:** box. Inspect the headers and data of some of the packets.

5.3. Filter so only the HTTP packets display. Enter **`http`** in the **Filter:** box.

Right-click any of the HTTP packets, then select **Follow TCP Stream**. The stream content will be displayed in a much more readable format. Messages from the HTTP client display in one color (red), responses from the HTTP server display in another color (blue).

6. Save the captured packet data for later analysis. Select **File** > **Save**, or type **`Ctrl`**+**`S`** to open the **Save Capture File** dialog. Select **`root`**'s home directory for the **Save in folder:** field. Specify **`practice`** in the **Name:** field. Leave **Wireshark/... - pcapng** as the selected type in the **File type:** field. Click the **Save** button to save the captured data.

Confirm the captured data file has been created.

```
[root@workstation ~]# ls practice*
practice.pcapng
[root@workstation ~]# file practice*
practice.pcapng: pcap-ng capture file - version 1.0
```

7. Log in as **`student`** on **`workstation`**. Download an existing capture file and open it with Wireshark. The file can be downloaded from the following URL: **`http://materials.example.com/rh342-practice.tcpdump`**.

7.1. Download the **`rh342-practice.tcpdump`** capture file.

```
[student@workstation ~]$ curl -O http://materials.example.com/rh342-
practice.tcpdump
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  126k  100  126k    0     0  11.0M      0 --:--:-- --:--:-- --:--:-- 12.3M
[student@workstation ~]$ ls -l rh342-practice.tcpdump
```

```
-rw-r--r--. 1 student student 129205 Feb  4 11:10 rh342-practice.tcpdump
```

7.2. Launch Wireshark from the GNOME desktop. Select **Applications** > **Internet** > **Wireshark Network Analyzer**.

7.3. Open the capture file. Select **File** > **Open...**, or type **Ctrl+O** to open the **Open Capture File** dialog. Browse and select the **rh342-practice.tcpdump** file. Click the **Open** button.

8. Filter the captured data so only SMTP packets display. Enter **smtp** in the **Filter:** box.

9. Inspect the content of the network packets. Click the plus icons in the middle frame to expand different parts of the network packet. See if you can find the text of the email message.

10. Right-click any packet in the SMTP exchange, then select **Follow TCP Stream**. The stream content will be displayed in a much more readable format. Messages from the SMTP server display in one color (blue), messages from the SMTP client display in another color (red). The "GOLD RING HAS BEEN CAUGHT" string is the first line of the body of the mail message.

11. The **tcpdump** utility can also be used to capture and display network traffic. It is useful in nongraphical environments.

11.1. Log in as **root** on **serverb** and confirm **tcpdump** is installed on the system.

```
[root@serverb ~]# rpm -q tcpdump
tcpdump-4.5.1-3.el7.x86_64
```

11.2. Launch **tcpdump** on **serverb** so that it captures network traffic related to **servera**. Capturing packets from the network interfaces requires **root** privilege.

```
[root@serverb ~]# tcpdump 'ip host servera'
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
```

11.3. Log in as **student** on **servera** and **ping serverb**.

```
[student@servera ~]$ ping -c 3 serverb
PING serverb.lab.example.com (172.25.250.11) 56(84) bytes of data.
64 bytes from serverb.lab.example.com (172.25.250.11): icmp_seq=1 ttl=64
 time=0.266 ms
64 bytes from serverb.lab.example.com (172.25.250.11): icmp_seq=2 ttl=64
 time=0.303 ms
64 bytes from serverb.lab.example.com (172.25.250.11): icmp_seq=3 ttl=64
 time=0.229 ms

--- serverb.lab.example.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1999ms
rtt min/avg/max/mdev = 0.229/0.266/0.303/0.030 ms
```

11.4. Observe the **tcpdump** output displayed on **serverb**.

```
11:37:54.013541 IP servera.lab.example.com > serverb.lab.example.com:
 ICMP echo request, id 3657, seq 1, length 64
```

```
11:37:54.013598 IP serverb.lab.example.com > servera.lab.example.com:
 ICMP echo reply, id 3657, seq 1, length 64
11:37:55.013249 IP servera.lab.example.com > serverb.lab.example.com:
 ICMP echo request, id 3657, seq 2, length 64
11:37:55.013284 IP serverb.lab.example.com > servera.lab.example.com:
 ICMP echo reply, id 3657, seq 2, length 64
11:37:56.013223 IP servera.lab.example.com > serverb.lab.example.com:
 ICMP echo request, id 3657, seq 3, length 64
11:37:56.013257 IP serverb.lab.example.com > servera.lab.example.com:
 ICMP echo reply, id 3657, seq 3, length 64
```

11.5. Break out of **`tcpdump`** on **`serverb`**.

```
Ctrl+C
6 packets captured
6 packets received by filter
0 packets dropped by kernel
```

12. Use **`tcpdump`** to capture network traffic coming into a specific port. Save the captured packets to a file for later analysis.

12.1. Launch **`tcpdump`** on **`serverb`** so that it captures HTTP traffic. It should save the captured network packets to a file called **`http-test.tcpdump`**.

```
[root@serverb ~]# tcpdump -w http-test.tcpdump 'port 80'
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 65535
 bytes
```

12.2. From **`servera`**, display the web content served by **`serverb`**.

```
[student@servera ~]$ curl http://serverb.lab.example.com
serverb is providing web content.
```

12.3. Terminate **`tcpdump`** to finish the capture.

```
Ctrl+C
10 packets captured
10 packets received by filter
0 packets dropped by kernel
```

13. Use **`tcpdump`** to display the captured network traffic.

13.1. The **`-r`** ***`FILENAME`*** option causes **`tcpdump`** to read network data from a file instead of from the network interfaces. Without other options, only packet header information is displayed.

```
[root@serverb ~]# tcpdump -r http-test.tcpdump
reading from file http-test.tcpdump, link-type EN10MB (Ethernet)
11:52:54.699448 IP servera.lab.example.com.51091 >
 serverb.lab.example.com.http: Flags [S], seq 981393578, win 29200,
 options [mss 1460,sackOK,TS val 103035086 ecr 0,nop,wscale 7], length 0
11:52:54.699499 IP serverb.lab.example.com.http >
 servera.lab.example.com.51091: Flags [S.], seq 3953903445, ack
 981393579, win 28960, options [mss 1460,sackOK,TS val 223556028 ecr
 103035086,nop,wscale 7], length 0
```

```
11:52:54.699656 IP servera.lab.example.com.51091 >
 serverb.lab.example.com.http: Flags [.], ack 1, win 229, options
 [nop,nop,TS val 103035086 ecr 223556028], length 0
11:52:54.699689 IP servera.lab.example.com.51091 >
 serverb.lab.example.com.http: Flags [P.], seq 1:72, ack 1, win 229,
 options [nop,nop,TS val 103035086 ecr 223556028], length 71
... Output omitted ...
```

13.2. Display only the incoming packets that came from **servera**. Expressions can be used to filter the packets that are displayed. The **-x** option causes packet data to be displayed in hexadecimal.

```
[root@serverb ~]# tcpdump -x -r http-test.tcpdump 'src servera'
reading from file http-test.tcpdump, link-type EN10MB (Ethernet)
11:52:54.699448 IP servera.lab.example.com.51091 >
 serverb.lab.example.com.http: Flags [S], seq 981393578, win 29200,
 options [mss 1460,sackOK,TS val 103035086 ecr 0,nop,wscale 7], length 0
        0x0000:  4500 003c d064 4000 4006 1e0e ac19 fa0a
        0x0010:  ac19 fa0b c793 0050 3a7e e0aa 0000 0000
        0x0020:  a002 7210 4c78 0000 0204 05b4 0402 080a
        0x0030:  0624 30ce 0000 0000 0103 0307
11:52:54.699656 IP servera.lab.example.com.51091 >
 serverb.lab.example.com.http: Flags [.], ack 3953903446, win 229, options
 [nop,nop,TS val 103035086 ecr 223556028], length 0
        0x0000:  4500 0034 d065 4000 4006 1e15 ac19 fa0a
        0x0010:  ac19 fa0b c793 0050 3a7e e0ab ebab c756
        0x0020:  8010 00e5 4c70 0000 0101 080a 0624 30ce
        0x0030:  0d53 31bc
... Output omitted ...
```

13.3. The **-X** option causes packet data to be displayed in hexadecimal and ASCII. See if you can recognize some of the HTTP requests that came from **servera**.

```

[root@serverb ~]# tcpdump -X -r http-test.tcpdump 'src servera'
reading from file http-test.tcpdump, link-type EN10MB (Ethernet)
11:52:54.699448 IP servera.lab.example.com.51091 >
 serverb.lab.example.com.http: Flags [S], seq 981393578, win 29200,
 options [mss 1460,sackOK,TS val 103035086 ecr 0,nop,wscale 7], length 0
        0x0000:  4500 003c d064 4000 4006 1e0e ac19 fa0a  E..<.d@.@.......
        0x0010:  ac19 fa0b c793 0050 3a7e e0aa 0000 0000  .......P:~......
        0x0020:  a002 7210 4c78 0000 0204 05b4 0402 080a  ..r.Lx..........
        0x0030:  0624 30ce 0000 0000 0103 0307            .$0.........
11:52:54.699656 IP servera.lab.example.com.51091 >
 serverb.lab.example.com.http: Flags [.], ack 3953903446, win 229, options
 [nop,nop,TS val 103035086 ecr 223556028], length 0
        0x0000:  4500 0034 d065 4000 4006 1e15 ac19 fa0a  E..4.e@.@.......
        0x0010:  ac19 fa0b c793 0050 3a7e e0ab ebab c756  .......P:~.....V
        0x0020:  8010 00e5 4c70 0000 0101 080a 0624 30ce  ....Lp.......$0.
        0x0030:  0d53 31bc                                .S1.
11:52:54.699689 IP servera.lab.example.com.51091 >
 serverb.lab.example.com.http: Flags [P.], seq 0:71, ack 1, win 229,
 options [nop,nop,TS val 103035086 ecr 223556028], length 71
        0x0000:  4500 007b d066 4000 4006 1dcd ac19 fa0a  E..{.f@.@.......
        0x0010:  ac19 fa0b c793 0050 3a7e e0ab ebab c756  .......P:~.....V
        0x0020:  8018 00e5 4cb7 0000 0101 080a 0624 30ce  ....L........$0.
        0x0030:  0d53 31bc 4745 5420 2f20 4854 5450 2f31  .S1.GET./.HTTP/1
        0x0040:  2e31 0d0a 5573 6572 2d41 6765 6e74 3a20  .1..User-Agent:.
        0x0050:  6375 726c 2f37 2e32 392e 300d 0a48 6f73  curl/7.29.0..Hos
        0x0060:  743a 2073 6572 7665 7262 0d0a 4163 6365  t:.serverb..Acce
        0x0070:  7074 3a20 2a2f 2a0d 0a0d 0a              pt:.*/*....
... Output omitted ...
```

14. Copy the captured HTTP data to **workstation** for further analysis.

```
[root@serverb ~]# scp http-test.tcpdump student@workstation:
student@workstation's password: student
http-test.tcpdump                             100% 1224     1.2KB/s   00:00
```

Use Wireshark on **workstation** to view the captured HTTP network traffic. Be sure to "follow" the TCP stream to view the HTTP exchange in a much more readable format.

Lab: Troubleshooting Network Issues

In this lab, you will identify and correct the misconfigured network settings of a server.

Resources	
Application URL	`http://serverb.lab.example.com`
Machines	• `servera` • `serverb`

Outcome(s)

You should be able to diagnose and repair network configuration issues that cause a network service to be unavailable.

Before you begin

Prepare your systems for this exercise by running the **`lab network-lab setup`** command on your **`workstation`** system. This will deploy a web server and adjust the network configuration of **`serverb`**.

```
[student@workstation ~]$ lab network-lab setup
```

There is a web server running on **`serverb`** that publishes content. The content should be viewable at the following URL: **`http://serverb.lab.example.com`**.

The junior administrator who configured **`serverb`** lost the paperwork that had the network specifications for the server. The network team has properly configured the DNS entry for **`serverb`** on their servers. Diagnose and fix **`serverb`**'s network configuration so the web content is available. Make sure the network fixes persist.

1. Determine what **`serverb`**'s public IP network settings are supposed to be.

2. Log into **`serverb`**. Diagnose and troubleshoot its network issues so its web content will be available.

3. Confirm the web content is available from **`serverb`**.

4. Grade your work by running the **`lab network-lab grade`** command from your **`workstation`** machine.

   ```
   [student@workstation ~]$ lab network-lab grade
   ```

5. Reset all of your machines to provide a clean environment for the next exercises.

Solution

In this lab, you will identify and correct the misconfigured network settings of a server.

Resources	
Application URL	**http://serverb.lab.example.com**
Machines	• **servera** • **serverb**

Outcome(s)

You should be able to diagnose and repair network configuration issues that cause a network service to be unavailable.

Before you begin

Prepare your systems for this exercise by running the **lab network-lab setup** command on your **workstation** system. This will deploy a web server and adjust the network configuration of **serverb**.

```
[student@workstation ~]$ lab network-lab setup
```

There is a web server running on **serverb** that publishes content. The content should be viewable at the following URL: **http://serverb.lab.example.com**.

The junior administrator who configured **serverb** lost the paperwork that had the network specifications for the server. The network team has properly configured the DNS entry for **serverb** on their servers. Diagnose and fix **serverb**'s network configuration so the web content is available. Make sure the network fixes persist.

1. Determine what **serverb**'s public IP network settings are supposed to be.

   1.1. Identify **serverb**'s public IPv4 address.

   ```
   [student@servera ~]$ host serverb.lab.example.com
   serverb.lab.example.com has address 172.25.250.11
   serverb.lab.example.com has IPv6 address fd37:5265:6448:6174::b
   ```

   1.2. Identify the IPv4 netmask and gateway. (Hint: Both **servera** and **workstation** are configured for that network.)

   ```
   [student@servera ~]$ ip addr show dev eth0
   2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
    state UP qlen 1000
       link/ether 52:54:00:00:fa:0a brd ff:ff:ff:ff:ff:ff
       inet 172.25.250.10/24 brd 172.25.250.255 scope global eth0
          valid_lft forever preferred_lft forever
       inet6 fe80::5054:ff:fe00:fa0a/64 scope link
          valid_lft forever preferred_lft forever
   [student@servera ~]$ ip route
   default via 172.25.250.254 dev eth0  proto static  metric 100
   172.24.250.0/24 dev eth1  proto kernel  scope link  src 172.24.250.10
    metric 100
   172.25.250.0/24 dev eth0  proto kernel  scope link  src 172.25.250.10
    metric 100
   ```

Network is /24 with a default gateway of 172.25.250.254.

1.3. Determine a DNS name server **serverb** can use.

```
[student@servera ~]$ cat /etc/resolv.conf
# Generated by NetworkManager
search lab.example.com example.com
nameserver 172.25.250.254
```

2. Log into **serverb**. Diagnose and troubleshoot its network issues so its web content will be available.

2.1. Display the current network settings. This will identify the number of current network interfaces and show their initial configuration.

```
[root@serverb ~]# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
 state UP qlen 1000
    link/ether 52:54:00:00:fa:0b brd ff:ff:ff:ff:ff:ff
    inet 172.25.251.11/24 brd 172.25.251.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::5054:ff:fe00:fa0b/64 scope link
       valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
 state UP qlen 1000
    link/ether 52:54:00:01:fa:0b brd ff:ff:ff:ff:ff:ff
    inet 172.24.250.11/24 brd 172.24.250.255 scope global eth1
       valid_lft forever preferred_lft forever
    inet6 fd37:5265:6448:6174::b/64 scope global
       valid_lft forever preferred_lft forever
    inet6 fe80::5054:ff:fe01:fa0b/64 scope link
       valid_lft forever preferred_lft forever
```

2.2. There are two physical network interfaces. Determine their NetworkManager connection names.

```
[root@serverb ~]# nmcli conn
NAME         UUID                                  TYPE            DEVICE
System eth1  9c92fad9-6ecb-3e6c-eb4d-8a47c6f50c04  802-3-ethernet  eth1
System eth0  5fb06bd0-0bb0-7ffb-45f1-d6edd65f3e03  802-3-ethernet  eth0
```

A third connection may reference **eth1**, but have no device associated with it in the last column. It will disappear when **serverb** is rebooted and can be safely ignored. The interface associated with **eth0** is connected to the external network, so it deserves more attention.

2.3. Display the IPv4 NetworkManager settings for **eth0**.

```
[root@serverb ~]# nmcli conn show 'System eth0' | grep ipv4
ipv4.method:                            manual
```

```
ipv4.dns:                               172.25.250.254
ipv4.dns-search:                        lab.example.com,example.com
ipv4.addresses:                         172.25.251.11/24
ipv4.gateway:                           172.25.251.254
ipv4.routes:
ipv4.route-metric:                      -1
ipv4.ignore-auto-routes:                no
ipv4.ignore-auto-dns:                   no
ipv4.dhcp-client-id:                    --
ipv4.dhcp-send-hostname:                yes
ipv4.dhcp-hostname:                     --
ipv4.never-default:                     no
ipv4.may-fail:                          yes
```

The DNS server settings look correct, but the network interface is configured for the wrong subnet. It should be configured for the **172.25.250.0/24** network. The junior administrator must have mistyped the third octet of the IP address.

2.4. Back up the interface configuration files before making changes.

```
[root@serverb ~]# cp /etc/sysconfig/network-scripts/ifcfg-* .
```

2.5. Adjust the network connection configuration that corresponds to the **eth0** device.

```
[root@serverb ~]# nmcli conn mod 'System eth0' \
> ipv4.addresses '172.25.250.11/24' ipv4.gateway 172.25.250.254
```

2.6. Confirm the new settings.

```

[root@serverb ~]# nmcli conn show 'System eth0' | grep ipv4
ipv4.method:                            manual
ipv4.dns:                               172.25.250.254
ipv4.dns-search:                        lab.example.com,example.com
ipv4.addresses:                         172.25.250.11/24
ipv4.gateway:                           172.25.250.254
ipv4.routes:
ipv4.route-metric:                      -1
ipv4.ignore-auto-routes:                no
ipv4.ignore-auto-dns:                   no
ipv4.dhcp-client-id:                    --
ipv4.dhcp-send-hostname:                yes
ipv4.dhcp-hostname:                     --
ipv4.never-default:                     no
ipv4.may-fail:                          yes
```

2.7. Bounce the connection to activate the new settings.

```
[root@serverb ~]# nmcli conn down 'System eth0'
Connection 'System eth0' successfully deactivated (D-Bus active path:
 /org/freedesktop/NetworkManager/ActiveConnection/1)
[root@serverb ~]# nmcli conn up 'System eth0'
Connection successfully activated (D-Bus active path:
 /org/freedesktop/NetworkManager/ActiveConnection/2)
[root@serverb ~]# ip addr show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
 state UP qlen 1000
```

```
    link/ether 52:54:00:00:fa:0b brd ff:ff:ff:ff:ff:ff
    inet 172.25.250.11/24 brd 172.25.250.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::5054:ff:fe00:fa0b/64 scope link
       valid_lft forever preferred_lft forever
```

2.8. Confirm the HTTP firewall ports are open.

```
[root@serverb ~]# firewall-cmd --list-ports
443/tcp 80/tcp
```

2.9. Restart the service so it binds to the new address.

```
[root@serverb ~]# systemctl restart httpd
```

3. Confirm the web content is available from **serverb**.

```
[student@servera ~]$ curl http://serverb.lab.example.com
serverb is providing web content.
```

4. Grade your work by running the **lab network-lab grade** command from your **workstation** machine.

```
[student@workstation ~]$ lab network-lab grade
```

5. Reset all of your machines to provide a clean environment for the next exercises.

Summary

In this chapter, you learned:

- **ping** and **ping6** are useful tools for sending ICMP requests.
- **nmap** can quickly scan a network looking for hosts, or it can perform more exhaustive port scans that require much more time.
- The **ncat**, or **nc**, command can be used as a simple network client, or it can act as a simple server for testing network connectivity.
- **iptraf-ng** is a curses-based tool for monitoring network traffic.
- Incorrect network device names can be fixed by adjusting the **DEVICE** and **HWADDR** variables in the appropriate network interface configuration file, **ifcfg-****, then rebooting.
- The **nmcli** command can be used to diagnose and correct network configuration problems with NetworkManager. Active interfaces need to be bounced to make new settings active.
- Wireshark is an effective tool for capturing, filtering, and inspecting network packets.
- The **tcpdump** command can capture packets, and optionally display them, in nongraphical environments.

CHAPTER 8

TROUBLESHOOTING APPLICATION ISSUES

Overview	
Goal	Debug application issues.
Objectives	• Identify library dependencies for third-party software. • Identify if an application suffers from memory leaks. • Debug an application using standard tools.
Sections	• Resolving Library Dependencies (and Guided Exercise) • Debugging Memory Leaks (and Guided Exercise) • Debugging Application Execution (and Guided Exercise)
Lab	• Troubleshooting Application Issues

Resolving Library Dependencies

Objectives

After completing this section, students should be able to identify library dependencies for third-party software.

Using shared libraries with applications

Most Linux applications are dynamically linked against the shared libraries they use. Shared libraries are built in a way that allows their functions to be mapped into an application's memory when the application is executed. Since the required code is provided by the library at runtime, it is not copied into the application itself.

Dynamic linking applications with shared libraries has many benefits:

- Application binary executables are smaller, since the library code is not copied into them.
- The library can be shared with multiple, simultaneously executing programs. This takes up less system memory.
- Shared libraries make it easier to fix bugs in the library code. Most fixes to a shared library do not require every application linked against it to be rebuilt.

Linking against shared libraries

When an application is built, it is linked against the shared libraries that provide the functions that it uses. The compiler checks the libraries for required symbols and confirms they exist.

Identifying information about each required library is embedded in the executable. This can be the absolute path name of the library, but it is more commonly the **DT_SONAME** field of the shared library. The **DT_SONAME** includes the name and version of the shared library, so that the application will use the correct version at runtime. This field is set when the shared library is created with the **-soname** option to the **ln** linker. The **objdump** command includes this field when it displays information about a shared library.

```
[user@demo ~]$ objdump -p /usr/lib64/libpthread-2.17.so | grep SONAME
  SONAME               libpthread.so.0
```

The shared library has a symbolic link that points to it that matches its **DT_SONAME** field.

```
[user@demo ~]$ ls -l /usr/lib64/libpthread*
-rwxr-xr-x. 1 root root 142304 Aug 14  2015 /usr/lib64/libpthread-2.17.so
lrwxrwxrwx. 1 root root     18 Nov  2 11:16 /usr/lib64/libpthread.so.0 ->
 libpthread-2.17.so
```

When an application executes, the runtime linker identifies the required shared libraries and maps them into the program's memory space. On Red Hat Enterprise Linux 7, **/lib64/ld-linux-x86-64.so*** is the default, 64-bit version of the runtime linker. It is provided by the *glibc.x86_64* package. The 32-bit version of the **glibc** library provides the 32-bit version of the runtime linker when it is installed.

```
[user@demo ~]$ rpm -qlp glibc-2.17-105.el7.i686.rpm | grep ld-linux
```

```
/lib/ld-linux.so.2
```

The runtime linker uses the **DT_SONAME** names embedded in the application to determine which versions of the libraries to use. It then searches the system for those libraries, using the following steps in order:

- It searches the directories specified in the **LD_LIBRARY_PATH** environment variable. This step is ignored for setUID and setGID applications. This environment variable is often used when developing or testing an application, and development libraries are to be used.

- The **/etc/ld.so.cache** cache file is consulted. It contains a compiled list of candidate libraries previously found. The **ldconfig -p** command prints the list of libraries mapped in **/etc/ld.so.cache**.

```
[user@demo ~]$ ldconfig -p
373 libs found in cache `/etc/ld.so.cache'
        p11-kit-trust.so (libc6,x86-64) => /lib64/p11-kit-trust.so
        libz.so.1 (libc6,x86-64) => /lib64/libz.so.1
        libyaml-0.so.2 (libc6,x86-64) => /lib64/libyaml-0.so.2
        libyajl.so.2 (libc6,x86-64) => /lib64/libyajl.so.2
        libxtables.so.10 (libc6,x86-64) => /lib64/libxtables.so.10
        libxslt.so.1 (libc6,x86-64) => /lib64/libxslt.so.1
... Output omitted ...
```

- Finally it searches the directories in the default library paths. The 64-bit shared library loader points to 64-bit versions of the default locations, **/lib64**, then **/usr/lib64**. **/lib** then **/usr/lib** are the directories searched by the 32-bit runtime linker.

```
[user@demo ~]$ strings /lib64/ld-linux-x86-64.so.2 | grep '^/'
/t6E
/~5D9
/var/tmp
/var/profile
/lib64/
/usr/lib64/
/etc/suid-debug
/%x %s
/proc/self/exe
/etc/ld.so.cache
/proc/sys/kernel/osrelease
/dev/full
/dev/null
/etc/ld.so.preload
```

The **ldd** command displays the shared libraries that are required by the specified application at runtime. Each library's **DT_SONAME** name is displayed, followed by the path name to the library on the system.

```
[user@demo ~]$ ldd /usr/sbin/httpd
        linux-vdso.so.1 =>  (0x00007ffc4d377000)
        libpcre.so.1 => /lib64/libpcre.so.1 (0x00007fa706100000)
        libselinux.so.1 => /lib64/libselinux.so.1 (0x00007fa705edb000)
        libaprutil-1.so.0 => /lib64/libaprutil-1.so.0 (0x00007fa705cb1000)
        libcrypt.so.1 => /lib64/libcrypt.so.1 (0x00007fa705a7a000)
        libexpat.so.1 => /lib64/libexpat.so.1 (0x00007fa705850000)
        libdb-5.3.so => /lib64/libdb-5.3.so (0x00007fa705491000)
        libapr-1.so.0 => /lib64/libapr-1.so.0 (0x00007fa705262000)
        libpthread.so.0 => /lib64/libpthread.so.0 (0x00007fa705046000)
        libdl.so.2 => /lib64/libdl.so.2 (0x00007fa704e41000)
        libc.so.6 => /lib64/libc.so.6 (0x00007fa704a80000)
        liblzma.so.5 => /lib64/liblzma.so.5 (0x00007fa70485b000)
        /lib64/ld-linux-x86-64.so.2 (0x00007fa7065f1000)
        libuuid.so.1 => /lib64/libuuid.so.1 (0x00007fa704655000)
        libfreebl3.so => /lib64/libfreebl3.so (0x00007fa704452000)
```

Note

The absolute path name of a library is embedded in a program when the shared library does not have a **DT_SONAME** field. In the previous example, the **/lib64/ld-linux-x86-64.so.2** reference points to the runtime linker itself.

Troubleshooting library dependency issues

Library dependency problems can occur on a Red Hat Enterprise Linux system when third-party software is installed without using Yum or RPM. Also, using the **--force** or **--nodeps** options with **rpm** can also cause library dependency issues to arise. Library dependency issues are easy to identify. When an application references a shared library that is not available, the runtime linker displays a distinctive error message. It displays the name of the first library it cannot find, then exits the application with an exit status of 127.

```
[user@demo ~]$ application
application: error while loading shared libraries: library.so.X: cannot
 open shared object file: No such file or directory
[user@demo ~]$ echo $?
127
```

It is possible, and quite likely, that there are other libraries the runtime linker cannot resolve. The **ldd** command can be used to display the shared libraries used by an application. Libraries that cannot be resolved at runtime display as "not found".

```
[user@demo ~]$ which application
/usr/bin/application
[student@demo ~]$ ldd /usr/bin/application
        linux-vdso.so.1 =>  (0x00007ffcbeba9000)
... Output omitted ...
        libpthread.so.0 => /lib64/libpthread.so.0 (0x00007fcbd3e7d000)
        library1.so.2 => not found
        library2.so.0 => not found
        libbz2.so.1 => /lib64/libbz2.so.1 (0x00007fcbd409a000)
        /lib64/ld-linux-x86-64.so.2 (0x00007fcbd4aa7000)
```

Once the missing shared libraries are identified, they need to be installed on the system. This is a relatively easy task when they are provided by an RPM package. RPM packages include shared library dependency information within the package metadata.

```
[user@demo ~]$ rpm -q --requires httpd | grep pthread
libpthread.so.0()(64bit)
libpthread.so.0(GLIBC_2.2.5)(64bit)
```

The **yum whatprovides** command identifies the package that provides the specified shared library.

```
[user@demo ~]$ yum whatprovides '*lib/libpthread.so.0'
```

```
Loaded plugins: langpacks, search-disabled-repos
glibc-2.17-105.el7.i686 : The GNU libc libraries
Repo        : rhel_dvd
Matched from:
Filename    : /lib/libpthread.so.0
```

If the shared library is not packaged and distributed by Red Hat, the third-party vendor will have to be called to see if they can provide it. When a shared library is installed or removed from the system, the **ldconfig** command should be run without arguments. **ldconfig** updates the runtime linker cache, **/etc/ld.so.cache**.

```
[user@demo ~]$ rpm -q --scripts libnl
postinstall program: /sbin/ldconfig
postuninstall program: /sbin/ldconfig
```

References

ld(1), **ldconfig**(8), **ld.so**(8), **ldd**(1), **nm**(1), and **objdump**(1) man pages

Guided Exercise: Resolving Library Dependencies

In this lab, you will install third-party software and resolve library dependencies.

Resources	
Machine	• **servera**

Outcome(s)

You should be able to identify which libraries an application requires to run on a system.

Before you begin

Prepare for this exercise by running the **lab app-libdeps setup** command on your **workstation** system. This will download and install the **genisoimage** application on **servera**.

```
[student@workstation ~#$ lab app-libdeps setup
```

Another system administrator installed the **genisoimage** application and has been unable to get it working. You have the task of getting it to work.

1. Log in as **student** on **servera**. Try to execute the newly installed program to observe how it behaves.

```
[student@servera ~]$ genisoimage
genisoimage: error while loading shared libraries: libusal.so.0: cannot
 open shared object file: No such file or directory
[student@servera ~]$ echo $?
127
```

A helpful error message is displayed. When the system cannot resolve a shared library, the program generates the same exit status as the shell does when it cannot find a command.

2. Find out what shared libraries the program requires. The error message makes it clear that **libusal.so.0** is required, but there may be other required libraries missing.

```
[student@servera ~]$ which genisoimage
/usr/bin/genisoimage
[student@servera ~]$ ldd /usr/bin/genisoimage
        linux-vdso.so.1 =>  (0x00007ffcbeba9000)
... Output omitted ...
        libbz2.so.1 => /lib64/libbz2.so.1 (0x00007fcbd409a000)
        libusal.so.0 => not found
        librols.so.0 => not found
        libpthread.so.0 => /lib64/libpthread.so.0 (0x00007fcbd3e7d000)
        /lib64/ld-linux-x86-64.so.2 (0x00007fcbd4aa7000)
```

In the previous output, all of the libraries can be resolved except for two: **libusal.so.0** and **librols.so.0**.

3. Use Yum to determine if packages can be installed that provide those libraries.

```
[student@servera ~]$ yum whatprovides libusal.so.0
Loaded plugins: langpacks, search-disabled-repos
libusal-1.1.11-23.el7.i686 : Library to communicate with SCSI devices
Repo        : rhel_dvd
Matched from:
Provides    : libusal.so.0
... Output omitted ...
[student@servera ~]$ yum whatprovides librols.so.0
Loaded plugins: langpacks, search-disabled-repos
libusal-1.1.11-23.el7.i686 : Library to communicate with SCSI devices
Repo        : rhel_dvd
Matched from:
Provides    : librols.so.0
... Output omitted ...
```

4. Both libraries are provided by the *libusal* package. Log in as **root** on **servera** and install it.

```
[root@servera ~]# yum -y install libusal
```

5. As the **student** user, try running the program again.

```
[student@servera ~]$ genisoimage
genisoimage: Missing pathspec.
Usage: genisoimage [options] -o file directory ...

Use genisoimage -help
to get a list of valid options.

Report problems to debburn-devel@lists.alioth.debian.org.
```

The executable works. It printed a usage message because of invalid arguments.

6. Red Hat Enterprise Linux provides a package that provides a **genisoimage** command. Use Yum to determine which RPM provides that functionality.

```
[student@servera ~]$ yum whatprovides '*bin/genisoimage'
Loaded plugins: langpacks, search-disabled-repos
genisoimage-1.1.11-23.el7.x86_64 : Creates an image of an ISO9660 file-system
Repo        : rhel_dvd
Matched from:
Filename    : /usr/bin/genisoimage
```

7. Use the **yumdownloader** command to download the package into **student**'s home directory. Do not attempt to install it.

```
[student@servera ~]$ yumdownloader genisoimage
Loaded plugins: langpacks
genisoimage-1.1.11-23.el7.x86_64.rpm                       | 298 kB   00:00
```

8. Using RPM package management simplifies system administration. The library dependency would have been detected when the software was originally installed.

```
[student@servera ~]$ rpm -q --requires -p genisoimage-*.rpm
```

```
warning: genisoimage-1.1.11-23.el7.x86_64.rpm: Header V3 RSA/SHA256 Signature, key
 ID fd431d51: NOKEY
... Output omitted ...
libbz2.so.1()(64bit)
libc.so.6()(64bit)
libc.so.6(GLIBC_2.14)(64bit)
libc.so.6(GLIBC_2.2.5)(64bit)
libc.so.6(GLIBC_2.3)(64bit)
libc.so.6(GLIBC_2.3.2)(64bit)
libc.so.6(GLIBC_2.3.4)(64bit)
libc.so.6(GLIBC_2.4)(64bit)
libc.so.6(GLIBC_2.7)(64bit)
libmagic.so.1()(64bit)
libpthread.so.0()(64bit)
libpthread.so.0(GLIBC_2.2.5)(64bit)
librols.so.0()(64bit)
libusal = 1.1.11-23.el7
libusal.so.0()(64bit)
libz.so.1()(64bit)
... Output omitted ...
```

Runtime libraries are identified and stored in the package metadata when a package is built.

9. Once you are done, verify your work by running the **lab app-libdeps grade** command from your **workstation** machine.

```
[student@workstation ~]$ lab app-libdeps grade
```

10. Optional: Remove the package that provides the needed libraries.

10.1. As **root** on **servera**, use **yum** to remove the *libusal* package.

```
[root@servera ~]# yum -y erase libusal
```

Yum did not complain when asked to remove the required libraries.

10.2. Try to execute the application.

```
[root@servera ~]# genisoimage
genisoimage: error while loading shared libraries: libusal.so.0: cannot
 open shared object file: No such file or directory
```

The program is broken again. If the application was provided by a package, **rpm** and **yum** would have been aware of the library dependency.

11. After grading your system, successfully reset your machine to the state it had before starting the lab, either by resetting your virtual machine or by running the following command on your **workstation** system.

```
[student@workstation ~]$ lab app-libdeps reset
```

Debugging Memory Leaks

Objectives

After completing this section, students should be able to identify if an application suffers from memory leaks.

Diagnosing memory leaks

Sometimes a process does not correctly free memory after it is done using it. Since the kernel frees all the memory of a process when it exits, this is not a major issue when the process is a short-lived one, like **ls** or **netstat**. The problem can become quite severe when long-running processes leak memory. Imagine a web server that leaks 8 KiB of memory for every single HTTP request it handles. Such a process would consume most of the available memory in a system in just a few hours. Other than killing and restarting a process, there is not much that a system administrator can do to fix a memory leak.

Identifying memory leaks is one of the responsibilities of a system administrator. Generic tools like **ps**, **top**, **free**, **sar -r**, and **sar -R** can be used to identify a memory leak. There are also dedicated tools, like the **memcheck** tool from **valgrind**, that will identify if an application has a memory leak. The following command will launch a process using the **memcheck** tool.

```
[root@demo ~]# valgrind --tool=memcheck program [program arguments]
```

The **--leak-check=full** option will produce more detailed information on which function inside the program is leaking memory.

```
[root@demo ~]# valgrind --tool=memcheck --leak-check=full program [program arguments]
```

There are two different types of memory leak to watch for. In the first case, a program requests memory with a system call like **malloc**, but it does not actually use the requested memory. This causes the virtual size of the program to go up (**VIRT** in **top**). The **Committed_AS** line in **/proc/meminfo** will also increase, but no actual physical memory is used. The resident size (**RSS** in **top**) stays (almost) the same.

When a program actually uses the memory it allocates, this causes the resident size to go up in step with the virtual size, causing an actual memory shortage. While leaking virtual memory is not a nice thing, leaking resident memory will impact the system much more.

Once a memory leak is detected, contact the software developer that wrote the application. The source code will require further analysis, corrections will have to be made, and the application will have to be rebuilt and tested to solve this type of problem.

References

sar(1) and **valgrind**(1) man pages

For more information, see the section on **Valgrind** in the **Red Hat Enterprise Linux 7 Developer Guide**, available at **https://access.redhat.com/docs**

For more information, see the section on **Valgrind** in the **Red Hat Enterprise Linux 7 Performance Tuning Guide**, available at **https://access.redhat.com/docs**

For more information, see the section on **Profiling application memory usage with Valgrind** in the **Red Hat Enterprise Linux 7 Performance Tuning Guide**, available at **https://access.redhat.com/docs**

Guided Exercise: Debugging Memory Leaks

In this lab, you will identify memory leaks.

Resources	
Machine	• **servera**

Outcome(s)

You should be able to use **valgrind** to identify a virtual memory leak and a resident memory leak.

Before you begin

Prepare for this exercise by running the **lab app-memleak setup** command on your **workstation** system. This command installs the *bigmem* package on **servera**.

```
[student@workstation ~#$ lab app-memleak setup
```

The **bigmem** program is notorious for leaking memory (by design). To verify this, run it under the **memcheck** tool of **valgrind**.

1. Install *valgrind* on **servera**.

```
[root@servera ~]# yum -y install valgrind
```

2. Run the **bigmem** command under **valgrind**, asking it to allocate 256 MiB of resident memory.

```
[root@servera ~]# valgrind --tool=memcheck bigmem 256
==3182== Memcheck, a memory error detector
==3182== Copyright (C) 2002-2013, and GNU GPL'd, by Julian Seward et al.
==3182== Using Valgrind-3.10.0 and LibVEX; rerun with -h for copyright info
==3182== Command: bigmem 256
==3182==
Attempting to allocate 256 Mebibytes of resident memory...
Press <Enter> to exit Enter
==3182==
==3182== HEAP SUMMARY:
==3182==     in use at exit: 268,435,456 bytes in 256 blocks
==3182==   total heap usage: 256 allocs, 0 frees, 268,435,456 bytes allocated
==3182==
==3182== LEAK SUMMARY:
==3182==    definitely lost: 265,289,728 bytes in 253 blocks
==3182==    indirectly lost: 0 bytes in 0 blocks
==3182==      possibly lost: 3,145,728 bytes in 3 blocks
==3182==    still reachable: 0 bytes in 0 blocks
==3182==         suppressed: 0 bytes in 0 blocks
==3182== Rerun with --leak-check=full to see details of leaked memory
==3182==
==3182== For counts of detected and suppressed errors, rerun with: -v
==3182== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 1 from 1)
```

The **valgrind** summary indicates a memory leak was detected.

3. Repeat this task with **bigmem**, now asking for 256 MiB of virtual memory.

```
[root@servera ~]# valgrind --tool=memcheck bigmem -v 256
[root@servera student]# valgrind --tool=memcheck bigmem -v 256
==3183== Memcheck, a memory error detector
==3183== Copyright (C) 2002-2013, and GNU GPL'd, by Julian Seward et al.
==3183== Using Valgrind-3.10.0 and LibVEX; rerun with -h for copyright info
==3183== Command: bigmem -v 256
==3183==
Attempting to allocate 256 MebiBytes of virtual memory...
Press <Enter> to exit Enter
==3183==
==3183== HEAP SUMMARY:
==3183==     in use at exit: 268,435,456 bytes in 256 blocks
==3183==   total heap usage: 256 allocs, 0 frees, 268,435,456 bytes allocated
==3183==
==3183== LEAK SUMMARY:
==3183==    definitely lost: 265,289,728 bytes in 253 blocks
==3183==    indirectly lost: 0 bytes in 0 blocks
==3183==      possibly lost: 3,145,728 bytes in 3 blocks
==3183==    still reachable: 0 bytes in 0 blocks
==3183==         suppressed: 0 bytes in 0 blocks
==3183== Rerun with --leak-check=full to see details of leaked memory
==3183==
==3183== For counts of detected and suppressed errors, rerun with: -v
==3183== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 1 from 1)
```

The **valgrind** output is the same as the previous run.

4. **valgrind** detected the memory leak, but it is not clear what type of memory leak **bigmem** has.

   4.1. In a separate terminal, run the following command:

   ```
   [root@servera ~]# watch -d -n1 'free -m; grep -i commit /proc/meminfo'
   ```

   This will give an overview of the allocated and committed memory.

   4.2. Now that you are sure that **bigmem** leaks memory, run the two commands again, but this time without **valgrind**. Pay close attention to the output of the **watch** command.

   ```
   [root@servera ~]# bigmem 256
   [root@servera ~]# bigmem -v 256
   ```

   Can you explain what you are seeing?

   Since **bigmem** actually uses the memory it requests when run without the **-v** option, both committed memory and resident memory go up by the requested amount.

   When run with the **-v** option, the committed memory goes up with the requested amount, but resident memory does not go up by as much. (It still goes up a little, since there needs to be memory used for the executable itself, its page tables, etc.)

   4.3. Use **Ctrl**+**C** to interrupt the **watch** command. Close that terminal session.

5. Reset your machine to the state it had before starting the lab, either by resetting your virtual machines or by running the following command on your **workstation** system.

```
[student@workstation ~]$ lab app-memleak reset
```

Debugging Application Execution

Objectives

After completing this section, students should be able to debug an application using standard Red Hat Enterprise Linux software tools.

Troubleshooting application execution

Some applications have problems when they execute. Programs do not always produce helpful error messages that provide detailed information that help diagnose their problems. Other applications silently hang. They could be stuck in a loop, waiting for an external event to occur, such as the creation of a file or a connection from a network client. To effectively troubleshoot these types of applications, more information about their current state is needed.

The **strace** and **ltrace** commands display current information about a program's execution. These utilities intercept system calls of a running application, signals, or shared library calls in the case of **ltrace**. They then display detailed information about the application's execution to standard error.

Displaying system calls with strace

strace displays information about system calls and signals related to an application. This utility is provided by the *strace* RPM. The simplest way to use **strace** is to use it to launch an application. The application, with its options and arguments, is passed as arguments to **strace**. Consider the following **strace** command.

```
[user@demo ~]$ strace pwd
execve("/bin/pwd", ["pwd"], [/* 17 vars */]) = 0
brk(0)                                  = 0x11a1000
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
 0x7f2e6ad66000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
open("/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=28767, ...}) = 0
mmap(NULL, 28767, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f2e6ad5e000
close(3)                                = 0
open("/lib64/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0 \34\2\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=2107800, ...}) = 0
mmap(NULL, 3932736, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
 0x7f2e6a785000
mprotect(0x7f2e6a93b000, 2097152, PROT_NONE) = 0
mmap(0x7f2e6ab3b000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
 3, 0x1b6000) = 0x7f2e6ab3b000
mmap(0x7f2e6ab41000, 16960, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS,
 -1, 0) = 0x7f2e6ab41000
close(3)                                = 0
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
 0x7f2e6ad5d000
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
 0x7f2e6ad5b000
arch_prctl(ARCH_SET_FS, 0x7f2e6ad5b740) = 0
mprotect(0x7f2e6ab3b000, 16384, PROT_READ) = 0
mprotect(0x606000, 4096, PROT_READ)     = 0
mprotect(0x7f2e6ad67000, 4096, PROT_READ) = 0
munmap(0x7f2e6ad5e000, 28767)           = 0
```

```
brk(0)                                  = 0x11a1000
brk(0x11c2000)                          = 0x11c2000
brk(0)                                  = 0x11c2000
open("/usr/lib/locale/locale-archive", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=106065056, ...}) = 0
mmap(NULL, 106065056, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f2e6425e000
close(3)                                = 0
getcwd("/home/user", 4096)           = 14
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(136, 0), ...}) = 0
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
 0x7f2e6ad65000
write(1, "/home/user\n", 14/home/user
)         = 14
close(1)                                = 0
munmap(0x7f2e6ad65000, 4096)            = 0
close(2)                                = 0
exit_group(0)                           = ?
+++ exited with 0 +++
```

strace starts by executing the **pwd** command. This is accomplished by the **execve** command. The following **access**, **open**, **fstat**, **mmap**, and **close** system calls show the runtime linker mapping the shared libraries **pwd** needs into its memory space. The **getcwd** system call returns the current working directory to the application, then **write** displays it to the screen. Each system call is displayed with the arguments passed in, or returned, with the return value displayed. When a system call fails, the numeric and symbolic value of the **errno** variable is displayed also. This can help an administrator understand why the system call failed.

Normally **strace** will only display the information about the process that is launched. If it is desirable to follow the execution of child processes spawned by the application, the **-f** option must be specified.

strace can also trace a process that is already executing. Invoking **strace** with a **-p *PID*** option attaches it to the running process and outputs details of the system calls made by that process. Consider the following example.

```
[user@demo ~]$ ps -ef | grep httpd
user      1010 10243  0 09:28 pts/4    00:00:00 grep --color=auto httpd
apache    1552  1729  0 Feb22 ?        00:00:00 /usr/sbin/httpd -DFOREGROUND
apache    1553  1729  0 Feb22 ?        00:00:00 /usr/sbin/httpd -DFOREGROUND
apache    1554  1729  0 Feb22 ?        00:00:00 /usr/sbin/httpd -DFOREGROUND
apache    1555  1729  0 Feb22 ?        00:00:00 /usr/sbin/httpd -DFOREGROUND
apache    1556  1729  0 Feb22 ?        00:00:00 /usr/sbin/httpd -DFOREGROUND
root      1729     1  0 Feb10 ?        00:00:48 /usr/sbin/httpd -DFOREGROUND
[user@demo ~]$ strace -p 1552
strace: attach: ptrace(PTRACE_ATTACH, ...): Operation not permitted
```

The previous use of **strace** failed. Only **root**, or the user who owns a process, can use **strace** to attach to that process.

```
[root@demo ~]# strace -p 1552
Process 1552 attached
accept4(4, Ctrl+CProcess 1552 detached
 <detached ...>
```

Process 1552 is suspended executing the **accept** system call. It is probably waiting for an incoming network client connection to be made. Interrupting **strace** connected to a process with **Ctrl**+**C** terminates the trace, but the application continues its execution.

```
[root@demo ~]# strace -p 1729
Process 1729 attached
select(0, NULL, NULL, NULL, {0, 161417}) = 0 (Timeout)
wait4(-1, 0x7ffc71ec05dc, WNOHANG|WSTOPPED, NULL) = 0
select(0, NULL, NULL, NULL, {1, 0})     = 0 (Timeout)
wait4(-1, 0x7ffc71ec05dc, WNOHANG|WSTOPPED, NULL) = 0
select(0, NULL, NULL, NULL, {1, 0})     = 0 (Timeout)
wait4(-1, 0x7ffc71ec05dc, WNOHANG|WSTOPPED, NULL) = 0
select(0, NULL, NULL, NULL, {1, 0})     = 0 (Timeout)
wait4(-1, 0x7ffc71ec05dc, WNOHANG|WSTOPPED, NULL) = 0
select(0, NULL, NULL, NULL, {1, 0}Ctrl+CProcess 1729 detached
 <detached ...>
```

Process 1729 is in a loop that executes the **select** system call, causing the process to sleep for one second. Each time it wakes up, it executes the **wait4** system call to see if any of its child processes has exited.

The output of **strace** can also be saved to a separate file with the **-o *FILENAME*** option. This allows a user to interact normally with the application without the output being interspersed with **strace** output.

To only trace specific system calls the **-e** option can be used. For example, to only trace the **open** and **stat** system calls, while sending the output to the file **/tmp/mytrace**, the following command can be used:

```
[root@demo ~]# strace -o /tmp/mytrace -e open,stat mycommand
```

Displaying library calls with ltrace

ltrace is similar to **strace**, except it traces calls to shared library functions instead of system calls. It will additionally display system calls when used with the **-S** option. Consider the following **ltrace** output. It shows the library functions used by the **pwd** command. Compare the following output with the comparable **strace** output earlier in this section.

```
[root@demo ~]# ltrace pwd
__libc_start_main(0x401760, 1, 0x7ffd1251f0a8, 0x404a00 <unfinished ...>
getenv("POSIXLY_CORRECT")                        = nil
strrchr("pwd", '/')                              = nil
setlocale(LC_ALL, "")                            = "en_US.UTF-8"
bindtextdomain("coreutils", "/usr/share/locale") = "/usr/share/locale"
textdomain("coreutils")                          = "coreutils"
__cxa_atexit(0x4022f0, 0, 0, 0x736c6974756572)   = 0
getopt_long(1, 0x7ffd1251f0a8, "LP", 0x606d00, nil) = -1
getcwd(nil, 0)                                   = ""
puts("/root"/root
)                                    = 6
free(0x21ba030)                                  = <void>
exit(0 <unfinished ...>
__fpending(0x7f70c03c7400, 0, 64, 0x7f70c03c7eb0) = 0
fileno(0x7f70c03c7400)                           = 1
__freading(0x7f70c03c7400, 0, 64, 0x7f70c03c7eb0) = 0
__freading(0x7f70c03c7400, 0, 2052, 0x7f70c03c7eb0) = 0
fflush(0x7f70c03c7400)                           = 0
fclose(0x7f70c03c7400)                           = 0
__fpending(0x7f70c03c71c0, 0, 0x7f70c03c8a00, 0xfbad000c) = 0
fileno(0x7f70c03c71c0)                           = 2
__freading(0x7f70c03c71c0, 0, 0x7f70c03c8a00, 0xfbad000c) = 0
__freading(0x7f70c03c71c0, 0, 4, 0xfbad000c)     = 0
fflush(0x7f70c03c71c0)                           = 0
```

```
fclose(0x7f70c03c71c0)                           = 0
+++ exited (status 0) +++
```

ltrace can also be attached to currently executing processes.

```
[root@demo ~]# ltrace -p 1729
apr_proc_wait_all_procs(0x7ffc71ec06a0, 0x7ffc71ec069c, 0x7ffc71ec0698, 1) = 0x11176
apr_sleep(0xf4240, 0x7ffc71ec05dc, 3, 0)         = 0
apr_proc_wait_all_procs(0x7ffc71ec06a0, 0x7ffc71ec069c, 0x7ffc71ec0698, 1) = 0x11176
apr_sleep(0xf4240, 0x7ffc71ec05dc, 3, 0Ctrl+C
```

Some binary programs are only executable, not readable, by unprivileged users. **ltrace** requires read access to the executable to work.

```
[user@demo ~]$ ls -l /usr/bin/testprog
-rwx--x--x. 1 root root 57903 Feb 15 15:01 /usr/bin/testprog
[user@demo ~]$ ltrace testprog
Can't open /bin/testprog: Permission denied
```

strace will function when the executable is executable and not readable, but its output provides limited useful information. In the following example, **testprog** is trying to open a file for read access. The file exists, but the program is unable to open the file for read access. Because **strace** cannot read the executable, it cannot display the name of the file of interest.

```
[user@demo ~]$ strace testprog
execve("/bin/testprog", ["testprog"], [/* 17 vars */]) = 0
brk(0)                                  = 0xb51000
... Output omitted ...
mprotect(0x604000, 4096, PROT_READ)     = 0
mprotect(0x7f0f60157000, 4096, PROT_READ) = 0
munmap(0x7f0f6014e000, 28826)           = 0
open(0x404515, O_RDONLY)                = -1 EACCES (Permission denied)
write(2, 0x404528, 35Unable to open configuration file.
)                  = 35
exit_group(1)                           = ?
+++ exited with 1 +++
```

Auditing program execution

The Linux audit daemon, **auditd**, can also be used to troubleshoot application runtime problems. The **auditctl** command manages the rules that cause the audit daemon to record information about system events in **/var/log/audit/audit.log**. These rules can trigger based on failed system calls, the user executing an application, or on the specific application.

The following session shows how to use **auditd** to troubleshoot an application that fails at runtime.

```
[root@demo ~]# example
Error: configuration file not found
[root@demo ~]# auditctl -a always,exit -F arch=b64 -S open -F success=0
[root@demo ~]# example
Error: configuration file not found
[root@demo ~]# tail /var/log/audit/audit.log
... Output omitted ...
type=CONFIG_CHANGE msg=audit(1456381239.150:916): auid=0 ses=2
 subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 op="add_rule" key=(null)
 list=4 res=1
```

```
type=SYSCALL msg=audit(1456381246.155:917): arch=c000003e syscall=2 success=no exit=-2
 a0=404515 a1=0 a2=d a3=7fff048aae20 items=1 ppid=1484 pid=29841 auid=0 uid=0 gid=0
 euid=0 suid=0 fsuid=0 egid=0 sgid=0 fsgid=0 tty=pts0 ses=2 comm="example" exe="/usr/
bin/example" subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 key=(null)
type=CWD msg=audit(1456381246.155:917):  cwd="/root"
type=PATH msg=audit(1456381246.155:917): item=0 name="/etc/example.conf" objtype=UNKNOWN
```

The error message suggests there is a failure opening a file, so **auditctl** is used to activate a rule that audits **open()** system calls that fail (**-F success=0**). With the new rule in place, the application failure generates a useful error message in the **audit.log** file. The application was unable to open a file called **/etc/example.conf**.

Once the needed information is collected, the **auditctl -d** command can be used to remove the **auditd** rule.

```
[root@demo ~]# auditctl -d always,exit -F arch=b64 -S open -F success=0
```

References

auditctl(8), **auditd**(8), **ausearch**(8), **ltrace**(1), and **strace**(1) man pages

Guided Exercise: Debugging Application Execution

In this lab, you will troubleshoot an application that has runtime problems.

Resources	
Machine	• **servera**

Outcome(s)

You should be able to use **ltrace** and **strace** to diagnose runtime application issues.

Before you begin

Set up your systems for this exercise by running the **lab app-runtime setup** command on your **workstation** system. This will download and install the **progress** application on **servera**.

```
[student@workstation ~#$ lab app-runtime setup
```

Your manager has heard about a useful utility called **progress**. After some searching on the Internet, a **tar** archive has been found that provides this program for Linux. You have been assigned the task to install the program, with its documentation, on **servera** and test it.

1. Run the **progress** executable. It will display an error message that is not very helpful.

```
[root@servera ~]# progress
Unable to open configuration file.
```

2. Look in **/var/log/messages** for AVC errors. If AVC errors are found that relate to this program's errors, then querying **auditd** could provide more helpful information.

```
[root@servera ~]# tail -n 5 /var/log/messages
Feb 15 19:36:06 servera systemd: Starting Session 2 of user root.
Feb 15 19:36:06 servera dbus-daemon: dbus[515]: [system] Activating
 service name='org.freedesktop.problems' (using servicehelper)
Feb 15 19:36:06 servera dbus[515]: [system] Activating service
 name='org.freedesktop.problems' (using servicehelper)
Feb 15 19:36:06 servera dbus[515]: [system] Successfully activated service
 'org.freedesktop.problems'
Feb 15 19:36:06 servera dbus-daemon: dbus[515]: [system] Successfully
 activated service 'org.freedesktop.problems'
```

The **progress** command did not produce any AVC errors, so you must look elsewhere for useful information.

3. Use the **strace** command to display the system calls **progress** makes.

```
[root@servera ~]# strace progress
execve("/usr/bin/progress", ["progress"], [/* 25 vars */]) = 0
brk(0)                                  = 0x1722000
... Output omitted ...
mprotect(0x604000, 4096, PROT_READ)     = 0
mprotect(0x7fa05801b000, 4096, PROT_READ) = 0
munmap(0x7fa058012000, 28767)           = 0
open("/etc/ntp.conf", O_RDONLY)         = -1 ENOENT (No such file or directory)
write(2, "Unable to open configuration fil"..., 35Unable to open configuration file.
) = 35
exit_group(1)                           = ?
+++ exited with 1 +++
```

The program tried to open **/etc/ntp.conf** for read access. The file was not found; that is why the program exited.

4. The version of **progress** installed on **servera** requires access to **/etc/ntp.conf**. Take steps to make this file available.

   4.1. Use the **yum whatprovides** command to determine if **/etc/ntp.conf** is provided by a package.

```
[root@servera ~]# yum whatprovides /etc/ntp.conf
Loaded plugins: langpacks, search-disabled-repos
rhel_dvd                                                 | 4.1 kB     00:00
(1/2): rhel_dvd/group_gz                                   | 136 kB   00:00
(2/2): rhel_dvd/primary_db                                 | 3.6 MB   00:00
ntp-4.2.6p5-22.el7.x86_64 : The NTP daemon and utilities
Repo        : rhel_dvd
Matched from:
Filename    : /etc/ntp.conf
```

   4.2. It is provided by the *ntp* package. Install that package on **servera**.

```
[root@servera ~]# yum -y install ntp
```

5. Execute the **progress** utility again.

```
[root@servera ~]# progress
No command currently running: cp, mv, dd, tar, cat, rsync, grep, fgrep,
 egrep, cut, sort, md5sum, sha1sum, sha224sum, sha256sum, sha384sum,
 sha512sum, adb, gzip, gunzip, bzip2, bunzip2, xz, unxz, lzma, unlzma,
 zcat, bzcat, lzcat, or wrong permissions.
[root@servera ~]# echo $?
0
```

It successfully runs and returns a zero exit status.

6. Observe that happens when **/etc/ntp.conf** is present, but unreadable by an unprivileged user.

   6.1. Change the permissions of the file to make it unreadable.

```
[root@servera ~]# ls -l /etc/ntp.conf
-rw-r--r--. 1 root root 1992 Oct 16 01:46 /etc/ntp.conf
[root@servera ~]# chmod 600 /etc/ntp.conf
[root@servera ~]# ls -l /etc/ntp.conf
-rw-------. 1 root root 1992 Oct 16 01:46 /etc/ntp.conf
```

   6.2. Log in as **student** on **servera** and execute the **progress** command.

```
[student@servera ~]$ progress
Unable to open configuration file.
[student@servera ~]$ strace progress
execve("/bin/progress", ["progress"], [/* 17 vars */]) = 0
brk(0)                                  = 0x144f000
... Output omitted ...
mprotect(0x604000, 4096, PROT_READ)     = 0
mprotect(0x7f4fbe256000, 4096, PROT_READ) = 0
munmap(0x7f4fbe24d000, 28826)           = 0
open(0x404515, O_RDONLY)                = -1 EACCES (Permission denied)
write(2, 0x404528, 35Unable to open configuration file.
)                  = 35
exit_group(1)                           = ?
+++ exited with 1 +++
```

Although it would produce the same unhelpful error message, **strace** displays the **errno** value indicating why the **open** system call failed.

7. As **root** on **servera**, install the *ltrace* package.

```
[root@servera ~]# yum -y install ltrace
```

8. As **student**, use **ltrace** to run **progress**.

```
[student@servera ~]$ ltrace progress
__libc_start_main(0x403554, 1, 0x7ffcc9485118, 0x403b40 <unfinished ...>
getenv("PROGRESS_ARGS")                          = nil
open64("/etc/ntp.conf", 0, 015)                  = -1
fwrite("Unable to open configuration fil"..., 1, 35, 0x7fc6e8d571c0Unable
 to open configuration file.
) = 35
exit(1 <no return ...>
+++ exited (status 1) +++
```

ltrace produces diagnostic information similar to **strace**. It would have helped identify the problem with **progress**.

9. Reset your machine to the state it had before starting the lab, either by resetting your virtual machines or by running the following command on your **workstation** system.

```
[student@workstation ~]$ lab app-runtime reset
```

Lab: Troubleshooting Application Issues

In this lab, you will debug a third-party application.

Resources	
Machine	• **servera**

Outcome(s)

You should be able to diagnose and troubleshoot library dependency and application runtime issues.

Before you begin

Set up your systems for this exercise by running the **lab app-lab setup** command on your **workstation** system. This will download and install the **gdisk** application on **servera**.

```
[student@workstation ~#$ lab app-lab setup
```

Another system administrator installed the **gdisk** application and has been unable to get it working. You have the task of getting it to work.

1. Execute the newly installed program to observe how it behaves.

2. Identify the shared libraries the program needs. Install packages that will satisfy the shared library requirements.

3. Resolve any additional runtime issues that prevent **gdisk** from executing.

4. Once you are done, verify your work by running the **lab app-lab grade** command from your **workstation** machine.

   ```
   [student@workstation ~]$ lab app-lab grade
   ```

5. After successfully grading your work, reset all three of your machines.

Solution

In this lab, you will debug a third-party application.

Resources	
Machine	• **servera**

Outcome(s)

You should be able to diagnose and troubleshoot library dependency and application runtime issues.

Before you begin

Set up your systems for this exercise by running the **lab app-lab setup** command on your **workstation** system. This will download and install the **gdisk** application on **servera**.

```
[student@workstation ~#$ lab app-lab setup
```

Another system administrator installed the **gdisk** application and has been unable to get it working. You have the task of getting it to work.

1. Execute the newly installed program to observe how it behaves.

   1.1. Log in as **root** on **servera** and run the **gdisk** executable.

   ```
   [root@servera ~]# gdisk
   gdisk: error while loading shared libraries: libicuio.so.50: cannot open
    shared object file: No such file or directory
   ```

   1.2. Note the missing library dependency, **libicuio.so.50**. There are possibly more libraries that are needed to make this program run.

2. Identify the shared libraries the program needs. Install packages that will satisfy the shared library requirements.

   2.1. Identify the shared libraries that the **gdisk** command uses that cannot be resolved.

   ```
   [root@servera ~]# which gdisk
   /usr/sbin/gdisk
   [root@servera ~]# ldd /usr/sbin/gdisk
           linux-vdso.so.1 =>  (0x00007ffe7d94b000)
           libicuio.so.50 => not found
           libicuuc.so.50 => not found
           libuuid.so.1 => /lib64/libuuid.so.1 (0x00007fedd9a5f000)
   ... Output omitted ...
   ```

   All of the shared libraries resolved except for two: **libicuio.so.50** and **libicuuc.so.50**.

   2.2. Use Yum to determine if packages can be installed that provide those libraries.

   ```
   [root@servera ~]# yum whatprovides libicuio.so.50
   Loaded plugins: langpacks, search-disabled-repos
   libicu-50.1.2-15.el7.i686 : International Components for Unicode - libraries
   Repo        : rhel_dvd
   ```

```
Matched from:
Provides    : libicuio.so.50
... Output omitted ...
[root@servera ~]# yum whatprovides libicuuc.so.50
Loaded plugins: langpacks, search-disabled-repos
libicu-50.1.2-15.el7.i686 : International Components for Unicode - libraries
Repo        : rhel_dvd
Matched from:
Provides    : libicuuc.so.50
... Output omitted ...
```

2.3. Both libraries are provided by the *libicu* package. Install it.

```
[root@servera ~]# yum -y install libicu
```

2.4. **gdisk** successfully executes, but now it displays its own error message.

```
[root@servera ~]# gdisk
GPT fdisk (gdisk) version 0.8.6

Fatal error: license not found.
```

3. Resolve any additional runtime issues that prevent **gdisk** from executing.

3.1. The **gdisk** command produces an obscure error message: "Fatal error: license not found." Look in **/var/log/messages** for AVC errors.

```
[root@servera ~]# tail -n 5 /var/log/messages
Feb 18 12:01:01 servera systemd: Started Session 464 of user root.
Feb 18 12:01:01 servera systemd: Starting Session 464 of user root.
Feb 18 12:10:01 servera systemd: Started Session 465 of user root.
Feb 18 12:10:01 servera systemd: Starting Session 465 of user root.
Feb 18 12:14:15 servera yum[2758]: Installed: libicu-50.1.2-15.el7.x86_64
```

There are no AVC errors caused by the program, so there is no need to query **auditd**.

3.2. Use **strace** or **ltrace** to display the system calls **gdisk** makes. This might shed some light on the problem.

Note

You may have to install the *ltrace* package on **servera**, if you choose to use it.

```
[root@servera ~]# strace gdisk
... Output omitted ...
write(1, "GPT fdisk (gdisk) version 0.8.6\n"..., 33GPT fdisk (gdisk)
 version 0.8.6
) = 33
open("/usr/share/doc/gdisk-0.8.6/GPL", O_RDONLY) = -1 ENOENT (No such
 file or directory)
write(2, "Fatal error: license not found.\n", 32Fatal error: license
 not found.
) = 32
```

```
exit_group(1)                           = ?
+++ exited with 1 +++
```

3.3. **gdisk** tried to open a file called **GPL** in its documentation directory in **/usr/share/doc**. Create the file it is looking for and see if that solves the problem.

```
[root@servera ~]# touch /usr/share/doc/gdisk-0.8.6/GPL
[root@servera ~]# gdisk
GPT fdisk (gdisk) version 0.8.6

Type device filename, or press <Enter> to exit: Enter
```

4. Once you are done, verify your work by running the **lab app-lab grade** command from your **workstation** machine.

```
[student@workstation ~]$ lab app-lab grade
```

5. After successfully grading your work, reset all three of your machines.

Summary

In this chapter, you learned:

- The **objdump -p** command can be used to detect the embedded **SONAME** of a shared library.
- **ldconfig** is used to update and print the contents of the **/etc/ld.so.cache** runtime linker cache.
- The **ldd *APPLICATION*** command displays the shared libraries required by an application.
- The **valgrind --tool=memcheck** command can identify if a program has a memory leak.
- There are two kinds of memory leaks: those that steal from the address space for a process and those that consume physical memory that affect the whole system.
- The **strace** command can trace the system calls made by an application.
- **ltrace** can display the shared library calls an application makes.
- Audit rules can be added to cause the audit daemon to log useful diagnostic information in **/var/log/audit/audit.log**.

CHAPTER 9

DEALING WITH SECURITY ISSUES

Overview	
Goal	Identify and fix issues related to security subsystems.
Objectives	• Identify and fix issues related to SELinux. • Identify and fix issues in user authentication and authorization. • Identify and fix issues related to LDAP and Kerberos identity management.
Sections	• Fixing SELinux Issues (and Guided Exercise) • Handling Authentication Issues (and Guided Exercise) • Resolving Kerberos and LDAP Issues (and Guided Exercise)
Lab	• Dealing With Security Issues

Fixing SELinux Issues

Objectives

After completing this section, students should be able to identify and fix issues related to SELinux.

SELinux logging

When SELinux blocks an action from happening, for example, when a web server tries to write to **/etc/shadow**, this action is logged using the **auditd** daemon. These audit logs can be viewed in **/var/log/audit/audit.log**, or they can be searched using the **ausearch** command. Using **ausearch** allows administrators to focus on exactly the messages they are interested in.

```
[root@demo ~]# ausearch -m avc❶ -ts recent❷
----
time->Mon Feb 22 10:55:49 2016

type=SYSCALL❸ msg=audit(1456134949.107:5677): arch=c000003e syscall=2 success=no
 exit=-13 a0=7f62712ebd40 a1=80000 a2=0 a3=7ffc267d3650 items=0 ppid=26656
 pid=26658 auid=4294967295 uid=48 gid=48 euid=48 suid=48 fsuid=48 egid=48
 sgid=48 fsgid=48 tty=(none) ses=4294967295 comm="httpd" exe="/usr/sbin/httpd"
 subj=system_u:system_r:httpd_t:s0 key=(null)

type=AVC❹ msg=audit(1456134949.107:5677)❺: avc:  denied  { open }❻ for
 pid=26658 comm="httpd" path="/var/www/html/index.html" dev="vda1" ino=18259175
 scontext=system_u:system_r:httpd_t:s0 tcontext=unconfined_u:object_r:user_tmp_t:s0
 tclass=file
```

❶ The **-m avc** option tells **ausearch** it should only display *Access Vector Control* (AVC) messages, the type of message associated with SELinux denials.

❷ The **-ts recent** option specifies that messages should be shown starting from 10 minutes ago. Other indications, like **today**, **yesterday**, and **this-week** can also be used, as well as others, and actual times.

❸ Typically two lines are shown for every SELinux denial; the action that caused the denial, typically a system call, and the actual denial itself. The **type=SYSCALL** line will have useful information, like the actual, original, and effective user ID of the calling process.

❹ The actual denial will have **type=AVC** at the beginning.

❺ The **audit(1456134949.107:5677)** part lists the time when this message was passed to the audit subsystem. This time is in seconds since the epoch (UNIX time); to convert to a local time zone, the command **date --date=@*timestamp*** can be used.

❻ This show the list of denied actions. Common actions include **read** for reading, **open** for opening, and **getattr** for requesting extended file information.

Further fields on the **type=AVC** line identify the name of the process for which an action was denied, the file system path (not the absolute path, but rather the path on an individual file system), the device that houses the file system, the inode of the *target* file, and the full SELinux contexts of both the process trying to access something, and the target, the file, process, or network port being accessed.

dontaudit rules

There are some actions that are performed so often, yet are always denied, that the SELinux policy has a special **dontaudit** rule for them. This means that the action will still be blocked, but it will not be logged.

A full list of all active **dontaudit** rules can be viewed with the command **sesearch -D** from the *setools-console* package.

When an administrator suspects that a **dontaudit** rule is the cause of an issue, all **dontaudit** rules can be disabled temporarily using the command **semanage dontaudit off**. This will disable all **dontaudit** rules until the command **semanage dontaudit on** is run.

Important

dontaudit rules exist for a reason. These are typically actions that are performed frequently, should always be blocked, and do not impact system behavior. There are, however, many occurrences of these actions. Keeping **dontaudit** disabled will fill the audit log rapidly, and make it harder to focus on the denials that do matter.

SELinux troubleshooting tools

While many simpler SELinux denials, like incorrectly labeled files in **/var/www/html**, can be diagnosed and fixed by looking at **/var/log/audit/audit.log** and running **restorecon** on the relevant files and directories, some problems can be harder to diagnose correctly and fix. In those cases, a bit more assistance can be had by installing *setroubleshoot-server*.

The *setroubleshoot-server* package provides two main tools, the **sealert** command, and the **sedispatch** plug-in for auditd.

Whenever the **auditd** daemon receives a message, it will pass it to the **sedispatch** plug-in. This plug-in will then forward any **AVC** message to **setroubleshootd** using the **dbus** protocol.

Note

The **auditd** daemon may need to be restarted before the **sedispatch** plug-in is activated. Unlike other services, this still requires the use of the **service** command: **service auditd restart**.

When **setroubleshootd** receives a denial message, it will parse it using a number of plug-ins, then log the denial, and a list of possible causes and fixes, to the system log.

Important

setroubleshootd will sometimes suggest building a policy module using the command **audit2allow**. In most cases, this will not be the preferred solution. Building custom policy modules should only be done when the impact of said module is fully understood. In most cases, a simpler, and more secure, solution can be found.

The **sealert** can be used in two ways: by passing it the UUID of a denial, as found in the system logs, e.g., **sealert -l fdfe9d90-557b-4ad0-a1a3-a6d09e065523**, or by having it parse all denial messages out of a file, e.g., **sealert -a /var/log/audit/audit.log**.

In both cases, the same output will be produced as written to the system logs by **setroubleshootd**.

Common SELinux issues

SELinux issues can be broadly classified into four categories:

Using nonstandard locations for service data

The **targeted** policy has a large list of default file contexts for the services shipped with Red Hat Enterprise Linux, even including secondary locations for most data. For example, the default location for web server data **/var/www**, and secondary locations in **/srv/*/www**, will all get the correct SELinux contexts applied when **restorecon** is run.

These lists of mappings between file names and standard contexts can be inspected, and modified, using the command **semanage fcontext**.

When a nonstandard location is used, the system will need to be told about this. For example, to use **/mysites/sitea/www** as a document root for a web server, the following steps will need to be performed:

1. Add the new location to the list of standard file contexts using **semanage**.

```
[root@demo ~]# semanage fcontext -a -t httpd_sys_content_t '/mysites/sitea/
www(/.*)?'
```

2. Apply the new file contexts.

```
[root@demo ~]# restorecon -Rv /mysites/sitea/www
```

Switching from **disabled** to **enforcing** mode.

Whenever a system is running with SELinux, disabled file contexts on newly created files will not be set according to the policy. This also includes files that are edited with most text editors, tools like **sed**, and even **NetworkManager**, as most tools do not write files in place, but create a temporary file and then *move* that file over the original one.

Normally, a file called **/.autorelabel** is created automatically when switching to disabled mode, but an administrator, or an automated tool, might have removed that file. Without that file present, the system will not perform an automatic relabel of all file systems when switching back to enforcing mode, causing the file to appear with the **unlabeled_t** type. While processes, such as a root shell, running as **unconfined_t** can still access these files, confined services cannot. This will result in denials that can even affect the ability of a system to boot.

To remedy these situations, create an (empty) file called **/.autorelabel** and reboot. This will cause the system to perform a full file system relabel on the next reboot, and then reboot again.

Booleans not set correctly

Many of the confined services on Red Hat Enterprise Linux 7 are limited in what they can do, but have toggles (Booleans) that allow them more access when a setup needs it. For example, by default the **httpd** services are not allowed to create outgoing network connections, but can be allowed by setting one or more Booleans.

In the case of **httpd**, specific Booleans exist for use cases such as connecting to various databases, **memcached**, FTP servers, LDAP, and more. There is also a Boolean to blanketallow all network connections, in case a specific use case is not covered by one of the other Booleans.

A list of Booleans, along with current state, default state, and description, can be queried with the command **semanage boolean --list**. Individual Booleans can be queried and set with the **getsebool** and **setsebool** command respectively.

> Important
>
> Unless the **-P** flag is used when executing **setsebool**, Boolean values will only be updated in memory, not persistently.

Using nonstandard network ports for services

By default, the **targeted** policy only allows confined services to listen on a predefined set of network ports. For example, the **httpd** daemon is only allowed to bind to ports labeled as either **http_port_t** and **http_cache_port_t**.

To label an (unlabeled) port with a new label, the **semanage** command can be used.

```
[root@demo ~]# semanage port -a -t http_port_t -p tcp 8001
```

In case a port should be used that is already labeled with a conflicting type, for example, making **sshd** listen on port **443/tcp**, a new policy module will need to be written to allow this bind to happen. This is typically not recommended.

Understanding SElinux rules

The **targeted** SELinux policy that is used on Red Hat Enterprise Linux 7 defines many types, rules, and booleans. To better udnerstand which types and rules are used, and what rules are enabled by booleans, the package *setools-console* can be installed.

This package provides a number of executables to help in SELinux troubleshooting, two of which will be discussed here: **seinfo** and **sesearch**.

The commands **seinfo -t** and **seinfo -b** can be used to list all types and booleans respectively. When the **-t** option is passed the name of a type it will show that type, and all configured aliases for that type.

```
[root@demo ~]# seinfo -thttpd_sys_content_t
   httpd_sys_content_t
   Aliases
      httpd_fastcgi_content_t
      httpd_httpd_sys_script_ro_t
      httpd_fastcgi_script_ro_t
```

The **seinfo** command can also be used to determine what port types are associated with a specific network port:

```
[root@demo ~]# seinfo --portcon=443 --protocol=tcp
        portcon tcp 443 system_u:object_r:http_port_t:s0
        portcon tcp 1-511 system_u:object_r:reserved_port_t:s0
```

The command **sesearch** can be used to search within the rules defined in the policy. This can be useful to see what rules a certain boolean enables. For example, to view all **allow** rules enabled by the **httpd_can_connect_ldap** boolean the following command can be used:

```
[root@demo ~]# sesearch --allow -b httpd_can_connect_ldap
Found 1 semantic av rules:
   allow httpd_t ldap_port_t : tcp_socket name_connect ;
```

This rule lets processes of the type **httpd_t** connect to TCP sockets of the type **ldap_port_t**.

References

For more information, see the SELinux User's and Administrator's Guide available on **https://access.redhat.com/docs**.

semanage(8), **sesearch**(1), **getsebool**(8), **setsebool**(8), **sealert**(8), **setroubleshootd**(8), **seinfo**(1), and **sesearch**(1) man pages.

Guided Exercise: Fixing SELinux Issues

In this lab, you will resolve a permissions issue for a website.

Resources	
Machines	• `servera`

Outcome(s)
You should be able to resolve SELinux permission issues.

Before you begin
From your **`workstation`** system, run the command **`lab selinuxts setup`**. This will set up this exercise on **`servera`**. As part of the setup process, your **`servera`** system will reboot twice.

```
[student@workstation ~]$ lab selinuxts setup
```

One of your coworkers recently performed some emergency maintenance and troubleshooting on your **`servera`** system. While the original problem is solved, the **`/var/log/audit/audit.log`** file on **`servera`** is now growing rapidly; this was spotted because the **chronyd** daemon now fails to start.

Contrary to your company's policies, your coworker did not document any of the steps performed. When asked, your coworker cannot remember exactly what was done to the system as well.

Investigate, and fix, this issue.

1. Check for the SELinux messages logged today. Is there a pattern?

    1.1. Check for the SELinux messages logged today.

    ```
    [root@servera ~]# ausearch -m avc -ts today | less
    ```

    1.2. Is there a pattern?

    There are a lot of denials for **`/etc/resolv.conf`**, along with various other denials. All of these denials include a target context type of **`unlabeled_t`**.

2. What could have caused so many unlabeled files? What are the possible fixes? Perform the most comprehensive fix.

    2.1. What could have caused so many unlabeled files?

    Unlabeled files are created when the system is run with SELinux disabled. This can happen when the system is rescued using the **anaconda** rescue mode, or when an administrator has (temporarily) run a system with SELinux disabled. In both cases, the automatically created file **`/.autorelabel`** will have to have been deleted, as the presence of this file would trigger an automatic full system relabel the next time it was started with SELinux in enforcing or permissive mode.

2.2. What are the possible fixes?

The system can be recovered using a **restorecon -Rv /**, or by creating an (empty) file **/.autorelabel** and rebooting.

The second method is more comprehensive, and also ensures that any services that might have had startup issues due to the unlabeled files are restarted.

2.3. Perform the most comprehensive fix.

```
[root@servera ~]# touch /.autorelabel
[root@servera ~]# reboot
```

3. Grade your work by running the command **lab selinuxts grade** on your **workstation** system.

```
[student@workstation ~]$ lab selinuxts grade
```

4. *Important*: If you did not successfully complete this exercise, but wish to proceed, clean up your systems by either resetting your **servera** system, or by running the command **lab selinuxts reset** on your **workstation** system.

```
[student@workstation ~]$ lab selinuxts reset
```

Handling Authentication Issues

Objectives

After completing this section, students should be able to identify and fix issues in user authentication and authorization.

What is PAM

Red Hat Enterprise Linux 7 uses a set of libraries called *Pluggable Authentication Modules* (PAM) to provide authentication, authorization, session handling, and password-changing services to applications and services.

Applications that use PAM perform a library call to PAM when they need to perform an action, then interact with the user based on callback functions (for example, to request a password), then receive a yea or nay from the PAM libraries.

PAM provides these services through a set of extension libraries to handle various cases and subsystems; for example, tying into the local system accounts, or always saying no, and a set of configuration files.

PAM can provide services for four distinct use cases:

- **account**: Checks whether a user exists, and is permitted to access a certain service.
- **auth**: Authenticate user credentials. This can be as simple as verifying a password, or as complex as requiring two-factor authentication using a hardware token and a passphrase, or even biometric verification.
- **password**: Updating passwords, typically tied strongly to the **auth** section for a service.
- **session**: Session management. This includes setting SELinux contexts, audit trails, and more.

Configuring PAM

PAM can be configured separately per service that uses it. This is done in files named after the service in **/etc/pam.d/**. For example, the **vsftpd** service will be configured in the file **/etc/pam.d/vsftpd**. If no appropriate configuration is found, the file **/etc/pam.d/other** is used, which blocks all access.

Configuration files consist of lines following the following format:

```
type control module-path [module-arguments]
```

Where **type** is one of **account**, **auth**, **password**, and **session**.

control can be specified in *historical* syntax, which is one of **required**, **requisite**, **sufficient**, **optional**, **include**, or **substack**, or in the modern format of **[value1=action1 value2=action2 ...]**. For most uses, the older historical notation is still used, unless the extra freedom of expression of the newer syntax is needed.

module-path can be an absolute pathname, or a path relative to **/lib/security/** or **/lib64/security/**. This specifies which module should be used to perform this check.

module-arguments can be a space-separated list of arguments for the specified module. Some modules require the use of some arguments, while others do not provide any arguments at all.

When a service is queried for a specific type, all lines matching that type will be parsed in the order that they are found in the configuration file. While it is allowed to mix service types throughout a configuration file, it is recommended to keep four separate blocks for these types for ease of maintenance.

An example configuration file, for **vsftpd**:

```
#%PAM-1.0
auth       required pam_listfile.so item=user sense=deny file=/etc/vsftpd/ftpusers
 onerr=succeed
auth       required pam_shells.so
auth       include password-auth

account    include password-auth

session    optional     pam_keyinit.so    force revoke
session    required     pam_loginuid.so
session    include password-auth
```

This configuration file relies heavily on another service: **password-auth**, and foregoes a **password** block altogether. For authentication, a **required** check is made against the **pam_listfile.so** module. Failing a required check means the entire block will fail, but will still execute to prevent against timing attacks. In this case, the **pam_listfile.so** module will give a fail for every user listed in the file **/etc/vsftpd/ftpusers**, and an okay for everyone else. The **pam_shells.so** module will give a fail for every user whose login shell is not listed in **/etc/shells**, and an okay for those whose does.

The included **password-auth** service is defined in its own file, **/etc/pam.d/password-auth**. This file is in turn a symbolic link to the file **/etc/pam.d/password-auth-ac**, which is a file written by the **authconfig** family of tools. A similar setup is done for the **system-auth** and **system-auth-ac** files. This allows an administrator to decouple authentication from the systemwide authentication if necessary, or make manual changes to a configuration without fear of having an automated tool overwrite it later, simply by breaking the symbolic link.

Warning

In most cases, PAM should not be configured by hand, but by using the **authconfig** family of tools.

Troubleshooting PAM

Some of the more common errors in PAM configuration and operation stem from manual configuration files. Spotting these errors can be done by examining the log files for the service that uses PAM, or by looking at **/var/log/secure**. In **journalctl**, PAM messages are logged as part of the service journal, while for **syslog**, PAM messages are sent to **/var/log/secure**.

Common errors include using the wrong control type, for example, **requisite** instead of **optional**; incorrect ordering of rules, for example, listing **pam_deny.so** before anything else; and attempts to use modules that do not support the current type, for example, attempting to use **pam_pwhistory.so** inside the **auth** block.

Individual PAM modules can provide their documentation, which, among other information, states their supported types in both man pages and as files under **`/usr/share/doc/pam-*/txts/`**. This documentation can also include explanations of error messages, and explanations on how to enable more verbose debugging for individual modules.

Restoring authconfig configurations

When authentication is configured using any of the tools in the **`authconfig`** family (**`authconfig`**, **`authconfig-tui`**, **`authconfig-gtk`**), all applied settings are stored in the file **`/etc/sysconfig/authconfig`**. This file consists of **`name=value`** pairings for all settings that can be made with **`authconfig`**.

To restore the previous **`authconfig`** settings after (unwanted) manual changes have been applied, the command **`authconfig --updateall`** can be used. This will apply the configuration as stored in **`/etc/sysconfig/authconfig`** to the system, overwriting any manual changes.

References

The chapter on using Pluggable Authentication Modules from the System-Level Authentication Guide available on **`https://access.redhat.com/docs`**.

`pam`(8), **`pam.conf`**(5), and **`authconfig`**(8) man pages.

Guided Exercise: Handling Authentication Issues

In this lab, you will diagnose and correct an authentication-related issue.

Resources	
Machines	• **`workstation`** • **`serverb`**

Outcome(s)
You should be able to resolve authentication-related issues.

Before you begin
On your **`workstation`** system, run the command **`lab auth setup`**; this will prepare your **`workstation`** and **`serverb`** systems for this exercise.

```
[student@workstation ~]$ lab auth setup
```

Even though you know it is insecure, your company provides authenticated FTP access to home directories on **`serverb`**.

After reading a bit too much on various Internet forums, one of your colleagues decided to try and increase the security on **`serverb`**. The attempt failed, but so did your former colleague's attempt at cleaning up afterwards. You have been called in to restore authenticated FTP access to home directories on **`serverb`**. To test, you have been given a user account named **`ftpuser`**, with the password **`redhat`**.

1. Begin by trying to use authenticated FTP as **`ftpuser`**.

   1.1. From your **`workstation`** system, use **`lftp`** to log into the **`ftpuser`** account on **`serverb`**. Remember that **`lftp`** only tries to log in *after* the first command has been sent.

   ```
   [student@workstation ~]$ lftp ftpuser@serverb.lab.example.com
   Password: redhat
   lftp ftpuser@serverb.lab.example.com:~> ls
   ls: Login failed: 530 Login incorrect.
   ```

2. Gather more information on the issue; in this case, the system journal on **`serverb`** might provide useful information.

   2.1. View the logs for **vsftpd** on **`serverb`**.

   ```
   [root@serverb ~]# journalctl -u vsftpd.service
   ....
   Fev 17 13:06:57 serverb.lab.example.com vsftpd[1640]: PAM unable to resolve
    symbol pam_sm_acct_mgmt
   ```

   This message seems to indicate a problem with the PAM configuration for **`vsftpd`**.

2.2. Use **rpm** to check if any files belonging to the *vsftpd* package have changed since installation.

```
[root@serverb ~]# rpm -V vsftpd
S.5....T.  c /etc/pam.d/vsftpd
```

This output seems to indicate that the PAM configuration for **vsftpd** has been changed.

3. Restore the PAM configuration for **vsftpd**, making sure you keep a backup of the current, broken file.

3.1. Move the current broken file out of the way.

```
[root@serverb ~]# mv /etc/pam.d/vsftpd{,.broken}
```

3.2. Reinstall the *vsftpd* package.

```
[root@serverb ~]# yum reinstall vsftpd
```

Note

The reason we *moved*, and not copied, the broken file in the previous step is that **yum** by default does not overwrite configuration files when reinstalling, so the broken file would have remained if we had copied.

4. Test if authenticated FTP access has been restored, then perform an analysis on why the modified PAM configuration file did not work.

4.1. Test if authenticated FTP access now works.

```
[student@workstation ~]$ lftp ftpuser@serverb.lab.example.com
Password: redhat
lftp ftpuser@serverb.lab.example.com:~> ls
-rw-r--r--    1 0        0          12 Feb 17 12:05 README.txt
```

4.2. Compare **/etc/pam.d/vsftpd** with **/etc/pam.d/vsftpd.broken**; what are the differences?

```
[root@serverb ~]# diff -u /etc/pam.d/vsftpd{,.broken}
...
+account    required    pam_ftp.so
...
```

It seems that the broken file had an extra requirement on account checks using the **pam_ftp.so** module.

4.3. Read the documentation for **pam_ftp** to find out what it is supposed to do, and why it failed in this case.

```
[root@serverb ~]# man pam_ftp
```

According to the man page, the **pam_ftp.so** module is used to provide anonymous FTP-like mapping of usernames to an anonymous account, but it can only be used inside an **auth** block, not an **account** block.

5. Grade your work by running the command **lab auth grade** from your **workstation** machine.

```
[student@workstation ~]$ lab auth grade
```

6. Clean up your machines by running the command **lab auth reset** from your **workstation** machine, or alternatively reset your **workstation** and **serverb** machines.

```
[student@workstation ~]$ lab auth reset
```

Resolving Kerberos and LDAP Issues

Objectives

After completing this section, students should be able to identify and fix issues related to LDAP and Kerberos identity management.

What is LDAP?

Lightweight Directory Access Protocol (LDAP) is a protocol for accessing, querying, and updating directory services. It evolved out of the need to simplify the X.509 protocol and related services.

LDAP organizes objects, such as user accounts, and their related attributes, such as passwords, full name, etc., in a tree. Which attributes can be used on an object depends on the *object classes* associated with that object. In a normal directory, all objects that represent the same thing, such as a user or a computer, will have the same object classes and attributes associated with them.

LDAP is an essential part of various identity management solutions, such as Red Hat Identity Management, Red Hat Directory Server, Microsoft Active Directory, and many more.

Various sets of tools exist to work with LDAP servers from the command line. One of the more popular is the OpenLDAP set of tools provided in the *openldap-clients* package.

The following command searches the LDAP server(s) configured in **/etc/openldap/ldap.conf** using simple authentication (**-x**), enforcing TLS encryption (**-ZZ**) for all objects under the search base defined in **/etc/openldap/ldap.conf** that have a **uid** attribute that exactly matches **ldapuser**, and displays only the **cn** and **homeDirectory** attributes. The **-LL** option disables extraneous comments in the output.

```
[user@demo ~]$ ldapsearch -x -ZZ -LL '(uid=ldapuser)' cn homeDirectory
version: 1

dn:uid=ldapuser,ou=People,dc=lab,dc=example,dc=com
cn: LDAP Test User
homeDirectory: /home/guests/ldapuser
```

What is Kerberos?

Kerberos is a method of providing authentication services and *Single Sign-On* (SSO) service to both users and services. Kerberos assumes that individual machines on the network can be trusted, but that the network itself is insecure, hence passwords will never be transmitted across the network.

A typical Kerberos transaction will involve three parties (hence the naming after the three-headed dog that guards the entrance to Hades):

- The *Kerberos Key Distribution Center* (KDC). This is the server that knows all the passwords for all users and services. Users and services are identified by a *principal*, a name, and a password.
- The user that wants to authenticate. This user should know his or her password.
- The service the user wishes to authenticate against. Since services cannot interactively type their own passwords, the service password will be stored in a file called a *keytab*.

Initial authentication

When a user first signs into a session using Kerberos, he or she will need to obtain a *Ticket Granting Ticket* (TGT). This TGT will then allow the user to obtain tickets for other services without having to type their password again for as long as the TGT is valid (typically a few hours). This ticket can be obtained automatically when interactively logging into a machine that uses Kerberos, or from the command line using the **`kinit`** tool.

Obtaining a TGT follows these steps:

1. The system the user is logging into sends a request for a TGT for the user to the KDC.
2. The KDC responds with a new TGT, encrypted with the user's password.
3. If the TGT can be decrypted, the login succeeds, and the TGT is stored.

Authenticating with a ticket

Once a TGT has been obtained, a user can connect to other Kerberos-enabled services without having to type a password. Authentication happens according to these steps:

1. The client sends a request to the KDC for a service ticket for the principal of the service he or she wishes to authenticate to.
2. The KDC responds with two copies of the same service ticket: one encrypted with the user's TGT, the other encrypted with the password for the service.
3. The client decrypts the version encrypted with the TGT, and uses the decrypted ticket to encrypt a timestamp.
4. The client sends the encrypted timestamp, and the ticket encrypted with the service password, to the service.
5. The service decrypts the service ticket, and uses it to decrypt the timestamp. If that succeeds, and the timestamp is less than two minutes old, the user is authenticated.

A note on principals

In its most basic form, a principal name looks something like **`username@REALM`**; for example, **`vimes@EXAMPLE.COM`**. The username will typically match that of a user defined in LDAP, or even locally on a system, while the realm is fixed across an organization.

Computer and service principals take a slightly different form: **`service/hostname@REALM`**, for example: **`host/demo.example.com@EXAMPLE.COM`**, **`nfs/demo.example.com@EXAMPLE.COM`**, or **`http/www.example.com@EXAMPLE.COM`**. The name of the service is determined by the type of service; authentication services like SSH, telnet, etc., use **`host/`**, NFS services use **`nfs/`**, and web services use **`http/`**. Other services can, and will, use different prefixes.

Troubleshooting LDAP issues

LDAP issues can fall into a number of different categories.

Network connectivity

LDAP works on TCP port 389 for both clear text and TLS-encrypted connections. Older versions of the protocol also support TCP port 636 for SSL-encrypted connections. If a firewall is blocking access to these ports, or if an older client attempts to connect to port

636/TCP while the server only supports the modern STARTTLS version on port 389/TCP, connection issues will occur.

Mismatched security settings

LDAP servers can put requirements on how clients connect and authenticate. Some servers might refuse to talk to clients that are not using a form of TLS encryption, and others might require a form of authentication, either simple or SASL, before responding to queries. Combinations of these requirements can also be set on individual actions; for example, a server can allow only TLS-encrypted connections for queries, but also require authentication for updates.

TLS certificate mismatches

When TLS is used to encrypt connections, clients need a way of verifying the certificate used by the server. It does this by checking if a trusted CA certificate has signed the server certificate. If these CA certificates are not downloaded to **/etc/openldap/cacerts**, and the directory hashed with **cacertdir_rehash**, clients will refuse to trust the server, and thus refuse to communicate with it.

Wrong search base

Clients typically do not request searches on the entire LDAP tree, but rather on the part of that tree that houses the information needed at the time. For example, a login program should typically query the part of the tree that has user accounts in it, and not the part with computer accounts. If the search base is configured incorrectly, searches might be slow, or not return the needed information.

Administrators can verify these settings by using the **ldapsearch** command. For troubleshooting user information issues, the **getent passwd** command can also be used.

Note

When **sssd** is used for user information and authentication, the **getent passwd** command will normally not list all available users, but only local users. For testing, the name of a known network account can be added to test if network users are resolved correctly as well; for example: **getent passwd ldapuser**.

Troubleshooting Kerberos issues

Kerberos issues can show up in two ways: Users that cannot obtain a TGT (and thus log in), and users or services that cannot authenticate to services.

To troubleshoot login issues, an administrator can check a few things:

- Are the times between all parties synchronized? A time drift of more than two minutes will cause Kerberos to fail.

- Does a **kinit *username*** for that user work from the same machine? If that does work, there is a problem with the (PAM) configuration for authentication, which can typically be fixed with the **authconfig** family of tools. If that does not work, there might be an issue with the KDC itself, or with the Kerberos configuration files on the machine.

- Do the **default_realm** setting, and the various host name to machine mappings in the **domain_realm** section of **/etc/krb5.conf**, match reality?

- Are the DNS **SRV** records for Kerberos (if used) correct?
- If **sssd** is used, do the **krb5_realm** and **krb5_server** settings in **/etc/sssd/sssd.conf** match reality?
- **/var/log/secure** and **journalctl** will have messages regarding failed logins; do they point to a likely cause?

When users can login, but cannot authenticate to certain services, or service-to-service authentication like NFSv4 fails, a number of extra things need to be checked.

- Are the services able to read their keytab? (**/etc/krb5.keytab** for services like SSH and NFS, various other files for other services).
- Is the keytab using the most recent version of the password? Keytabs can be inspected with the **klist -ek *<FILENAME>*** command. The **KVNO** column shows the version of the password stored. Whenever a principal is added to a keytab, a new, random password is generated automatically.

  Users with access to the Kerberos KDC management tool **kadmin** (or **kadmin.local** for **root** on the KDC) can use the command **getprinc *<PRINCIPAL>*** from within **kadmin** to view the password version as known by the KDC.
- Are the correct principals present in the keytab?
- Do services agree on how Kerberos should be used? For example, an NFS export that specifies **sec=krb5p** can only be mounted when the client specifies the same security level as well.
- Are all needed helper services running? For example, an NFS server needs the **nfs-secure-server.service** service, and an NFS client needs the **nfs-secure.service** service when using Kerberos-enabled NFSv4.

R

References

For more information see the Linux Domain Identity, Authentication, and Policy Guide, and the System-Level Authentication Guide at **https://access.redhat.com/docs**.

krb5.conf(5), **sssd.conf**(5), and **kadmin**(1) manual pages.

Guided Exercise: Resolving Kerberos and LDAP Issues

In this lab, you will resolve an issue related to LDAP user information or Kerberos authentication.

Resources	
Machines	• **workstation** • **servera**

Outcome(s)

You should be able to diagnose and resolve issues related to LDAP and Kerberos user information and authentication.

Before you begin

On your **workstation** system, run the command **lab krb5ldap setup**. This will prepare both your **workstation** and **servera** systems for this exercise.

```
[student@workstation ~#$ lab krb5ldap setup
```

Your **workstation** system is set up to act as both a LDAP server to provide user information (STARTTLS only), and as a Kerberos KDC for user authentication. Your **servera** system is configured to act as a client for both these services.

Your **workstation** system is also configured to act as a Kerberos-enabled NFS server, serving out the home directories of the users configured in LDAP and Kerberos.

After a recent maintenance window, in which Kerberos keytabs were updated across all the machines in your realm, and all clients were switched to **autofs** instead of static mounts, the user **ldapuser** reports that his home directory is no longer automatically mounted when he logs into **servera**. Home directories on **servera** for LDAP users are automounted using **autofs**.

The password for **ldapuser** is **kerberos**. For the purpose of troubleshooting, your security administrator has made keytabs for all your systems available via FTP on **ftp://workstation.lab.example.com/pub/**. The CA certificate to authenticate LDAP connections is also available here, as **example-ca.crt**.

> **Note**
>
> By default, **ssh** on your systems is configured to automatically allow any login from the **student** account on **workstation** to any account on **servera**. In order to log in with a password, to obtain a Kerberos ticket, the command **ssh -o PreferredAuthentications=keyboard-interactive,password *USER@HOST*** can be used. A wrapper that executes this command is installed on your **workstation** system as **passhwd**.

1. Begin by recreating the issue and collecting information.

   1.1. As **student** on **workstation**, use **ssh** to log into the **ldapuser** account on **servera** using password authentication to obtain a Kerberos *Ticket Granting Ticket* (TGT).

```
[student@workstation ~]$ ssh -o PreferredAuthentications=keyboard-
interactive,password ldapuser@servera
ldapuser@servera's password: kerberos
Last login: Wed Feb 17 09:40:55 2016 from workstation.lab.example.com
Could not chdir to home directory /home/guests/ldapuser: No such file or
 directory
```

1.2. Check if you received a TGT as **ldapuser**.

```
-bash-4.2$ klist
Ticker cache: KEYRING:persistent:1701:krb_cache_sRRVnEH
Default principal: ldapuser@LAB.EXAMPLE.COM

Valid starting       Expires              Service principal
02/16/2016 16:37:03  02/17/2016 16:37:03  krbtgt/LAB.EXAMPLE.COM@LAB.EXAMPLE.COM
```

2. The previous step indicated that both LDAP and Kerberos are functioning normally, so the problem may be in the actual NFS configuration. Inspect both the server export on **workstation** and the client configuration on **servera** to check for any irregularities.

2.1. Inspect the export on **workstation**.

```
[root@workstation ~]# cat /etc/exports /etc/exports.d/*
/home/guests *.lab.example.com(sec=krb5p,rw)
```

2.2. Verify that the **nfs-server** service is running on **workstation**.

```
[root@workstation ~]# systemctl is-active nfs-server
active
```

2.3. Verify that **autofs** is running on **servera**.

```
[root@servera ~]# systemctl status autofs
```

2.4. View the configuration for **autofs** related to the home directory for **ldapuser**.

```
[root@servera ~]# cat /etc/auto.master.d/guests.autofs
/home/guests /etc/auto.guests
```

```
[root@servera ~]# cat /etc/auto.guests
* -rw,sec=krb5p workstation.lab.example.com:/home/guests/&
```

3. Both the NFS server and **autofs** seem to be properly configured, so the problem might be in the security configuration. The export specifies **sec=krb5p**, which means that both the server and client need **host/** and/or **nfs/** keytab entries. The server must also run the **nfs-secure-server** service, and the client the **nfs-secure** service, both to exchange key information.

3.1. Check the status for the **nfs-secure** service on **servera**.

```
[root@servera ~]# systemctl status nfs-secure
...
Feb 17 10:18:32 servera.lab.example.com rpc.gssd[1596]: ERROR: No credentials
 found for connection to server workstation.lab.example.com
...
```

This message seems to indicate a problem with the keytab.

3.2. Check the contents of the **`/etc/krb5.keytab`** file on **`servera`**.

```
[root@servera ~]# klist -ek /etc/krb5.keytab
KVNO Principal
---- ----------------------------------------------------------------
   2 host/servera.lab.example.com@LAB.EXAMPLE.COM (aes-256-cts-hmac-sha1-96)
....
   2 nfs/servera.lab.example.com@LAB.EXAMPLE.COM (aes-256-cts-hmac-sha1-96)
....
```

3.3. Check the contents of the most recent keytab provided to you by your security administrator.

```
[root@servera ~]# wget ftp://workstation.lab.example.com/pub/servera.keytab
[root@servera ~]# klist -ek servera.keytab
KVNO Principal
---- ----------------------------------------------------------------
   3 host/servera.lab.example.com@LAB.EXAMPLE.COM (aes-256-cts-hmac-sha1-96)
....
   3 nfs/servera.lab.example.com@LAB.EXAMPLE.COM (aes-256-cts-hmac-sha1-96)
...
```

It appears that **`servera`** is not using the most recent keytab, which would explain the **`No credentials`** log message we saw earlier.

4. Update **`/etc/krb5.keytab`** on **`servera`**, then restart the **`autofs`** and **`nfs-secure`** services and try again.

4.1. Update the **`/etc/krb5.keytab`** keytab on **`servera`**.

```
[root@servera ~]# cp servera.keytab /etc/krb5.keytab
```

4.2. Restart the **`nfs-secure`** and **`autofs`** services on **`servera`**.

```
[root@servera ~]# systemctl restart nfs-secure autofs
```

4.3. Try again.

```
[student@workstation ~]$ ssh -o PreferredAuthentications=keyboard-
interactive,password ldapuser@servera
ldapuser@servera's password: kerberos
Last login: Wed Feb 17 10:18:55 2016 from workstation.lab.example.com
ldapuser@servera ~]$
```

5. Grade your work by running the command **lab krb5ldap grade** from your **workstation** machine.

```
[student@workstation ~]$ lab krb5ldap grade
```

6. *Important*: Clean up your work by resetting your **workstation** and **servera** systems.

Lab: Dealing with Security Issues

In this lab, you will resolve issues dealing with one or more security subsystems.

Resources	
Machines	• **workstation** • **servera**

Outcome(s)
You should be able to diagnose and fix issues related to security subsystems.

Before you begin
On your **workstation** system, run the command **lab security setup**; this will prepare your **workstation** and **servera** systems for this exercise.

```
[student@workstation ~#$ lab security setup
```

A new system has recently been added to your fleet of servers: **servera**. Unfortunately, this system has been configured by hand and not by a configuration management system, and as a result, there are some final bits of troubleshooting that need to be performed.

The two major open issues are:

- Users relying on LDAP + Kerberos for user information and authentication cannot log in. These users normally rely on the LDAP server on **workstation** for user information (out of the **dc=lab,dc=example,dc=com** tree), and on the Kerberos KDC providing services for the **LAB.EXAMPLE.COM** realm running on **workstation** for their authentication and single sign-on needs. These users also get their home directories automatically mounted from **workstation** over NFSv4.

  You have been given a testing account called **ldapuser**, with the password **kerberos**.

- The NFSv4 share **workstation.lab.example.com:/exports/importantfiles** should be mounted automatically on **/mnt/importantfiles**, but it isn't.

> **Note**
>
> Your classroom machines are set up so that the **student** account on **workstation** can **ssh** into any account on **servera** without a password. For testing purposes, you will need to log in with password authentication to test the Kerberos setup. To do this, you can use the command **ssh -o PreferredAuthentications=keyboard-interactive,password**. This command has been wrapped for your convenience in a utility script called **passhwd**.

1. Tackle the authentication first. Recreate the issue, gather information, form a hypothesis, then test and fix.

2. Investigate the mount issue for **/mnt/importantfiles**, gather information, form a hypothesis, then fix the issue.

3. Grade your work by running the command **lab security grade** on your **workstation** system.

   ```
   [student@workstation ~]$ lab security grade
   ```

4. Reset all of your machines to provide a clean environment for the next exercises.

Solution

In this lab, you will resolve issues dealing with one or more security subsystems.

Resources	
Machines	• **workstation** • **servera**

Outcome(s)

You should be able to diagnose and fix issues related to security subsystems.

Before you begin

On your **workstation** system, run the command **lab security setup**; this will prepare your **workstation** and **servera** systems for this exercise.

```
[student@workstation ~#$ lab security setup
```

A new system has recently been added to your fleet of servers: **servera**. Unfortunately, this system has been configured by hand and not by a configuration management system, and as a result, there are some final bits of troubleshooting that need to be performed.

The two major open issues are:

- Users relying on LDAP + Kerberos for user information and authentication cannot log in. These users normally rely on the LDAP server on **workstation** for user information (out of the **dc=lab,dc=example,dc=com** tree), and on the Kerberos KDC providing services for the **LAB.EXAMPLE.COM** realm running on **workstation** for their authentication and single sign-on needs. These users also get their home directories automatically mounted from **workstation** over NFSv4.

  You have been given a testing account called **ldapuser**, with the password **kerberos**.

- The NFSv4 share **workstation.lab.example.com:/exports/importantfiles** should be mounted automatically on **/mnt/importantfiles**, but it isn't.

> ## Note
>
> Your classroom machines are set up so that the **student** account on **workstation** can **ssh** into any account on **servera** without a password. For testing purposes, you will need to log in with password authentication to test the Kerberos setup. To do this, you can use the command **ssh -o PreferredAuthentications=keyboard-interactive,password**. This command has been wrapped for your convenience in a utility script called **passhwd**.

1. Tackle the authentication first. Recreate the issue, gather information, form a hypothesis, then test and fix.

   1.1. Recreate the problem by trying to log into **servera** as the user **ldapuser**, with the password **kerberos**.

   ```
   [student@workstation ~]$ passhwd ldapuser@servera.lab.example.com
   ```

```
Warning: Permanently added 'servera.lab.example.com' (ECDSA) to the list of
 known hosts.
ldapuser@servera.lab.example.com's password: kerberos
Permission denied, please try again.
```

1.2. Check the logs for **sshd.service** on **servera** to see why **ldapuser** was refused a login.

```
[root@servera ~]# journalctl -lu sshd.service
...
Feb 18 11:15:13 servera.lab.example.com sshd[1744]: Failed password for ldapuser
 from 172.25.250.254 port 54303 ssh2
...
```

1.3. Verify that user information is available for the **ldapuser** user.

```
[root@servera ~]# getent passwd ldapuser
ldapuser:*:1701:1701:LDAP Test User:/home/guests/ldapuser:/bin/bash
```

1.4. Try to obtain a Kerberos ticket for the **ldapuser** user.

```
[root@servera ~]# kinit ldapuser
kinit: Client 'ldapuser@LAB.EXMAPLE.COM' not found in Kerberos database while
 getting initial credentials
```

It appears that the Kerberos realm has been mistyped while setting up this system.

1.5. Fix all occurrences of **LAB.EXMAPLE.COM** in both **/etc/krb5.conf** and **/etc/sssd/sssd.conf** to **LAB.EXAMPLE.COM**, then restart the **sssd** daemon.

```
[root@servera ~]# sed -i 's/LAB.EXMAPLE.COM/LAB.EXAMPLE.COM/g' /etc/krb5.conf /
etc/sssd/sssd.conf
[root@servera ~]# systemctl restart sssd.service
```

1.6. Try to log in again from **workstation** using **ssh**.

```
[student@workstation ~]$ passhwd ldapuser@servera.lab.example.com
ldapuser@servera.lab.example.com's password: kerberos
```

Logins now work, and the home directory is mounted automatically as well.

2. Investigate the mount issue for **/mnt/importantfiles**, gather information, form a hypothesis, then fix the issue.

2.1. Recreate the issue by attempting to mount **/mnt/importantfiles** on **servera**.

```
[root@servera ~]# mount /mnt/importantfiles
mount.nfs: access denied by server while mounting workstation.lab.example.com:/
exports/importantfiles
```

This message seems to indicate that export options on **workstation** are denying access.

2.2. View the export options on **workstation** for the **/exports/importantfiles** file system.

```
[root@workstation ~]# exportfs -sv
/home/guests *.lab.example.com(rw,wdelay,root_squash,no_subtree_check,sec=krb5p,
rw,secure,root_squash,no_all_squash)
/exports/importantfiles(rw,wdelay,root_squash,no_subtree_check,sec=krb5p,rw,secu
re,root_squash,no_all_squash)
```

2.3. View the mount options defined in **/etc/fstab** on **servera** for **/mnt/importantfiles**.

```
[root@servera ~]# grep importantfiles /etc/fstab
workstation.lab.example.com:/exports/importantfiles /mnt/importantfiles nfs
 sec=krb5i 0 0
```

2.4. Do the mount options found in the previous step align with the export options found? If not, what is wrong?

The client attempts to mount the file system using Kerberos integrity checking, while the server requires a minimum of integrity checking plus encryption.

2.5. Change **sec=krb5i** to **sec=krb5p** in **/etc/fstab** for the **/mnt/importantfiles** mount, then attempt to mount again.

```
[root@servera ~]# sed -i /importantfiles/s/krb5i/krb5p/ /etc/fstab
[root@servera ~]# mount -a
```

3. Grade your work by running the command **lab security grade** on your **workstation** system.

```
[student@workstation ~]$ lab security grade
```

4. Reset all of your machines to provide a clean environment for the next exercises.

Summary

In this chapter, you learned:

- SELinux denials are logged to **/var/log/audit/audit.log**
- Installing *setroubleshoot-server* gives access to the **sealert** command, and automatic troubleshooting messages in the system log.
- The **semanage** tool can be used to update file context mappings and network port context mappings.
- PAM is configured per service with files in **/etc/pam.d**.
- PAM errors and messages are logged to the service that uses PAM.
- The **ldapsearch** command can be used to aid in LDAP troubleshooting.
- The OpenLDAP clients and libraries expect CA certificates in **/etc/openldap/cacerts**.
- Kerberos clients are configured in **/etc/krb5.conf**, including hostname to realm mappings.
- Service passwords are updated with a new, random, passowrd every time they are added to a keytab.
- The **klist -ek** can be used to view what, and what versions, of service principals are in a keytab.

CHAPTER 10

TROUBLESHOOTING KERNEL ISSUES

Overview	
Goal	Identify kernel issues and assist Red Hat Support in resolving kernel issues.
Objectives	• Create kernel crash dumps to assist Red Hat Support in resolving kernel issues. • Compile and execute SystemTap modules to assist Red Hat Support in resolving kernel issues.
Sections	• Create Kernel Crash Dumps (and Guided Exercise) • Kernel Debugging with SystemTap (and Guided Exercise)
Lab	• Troubleshooting Kernel Issues

Kernel Crash Dumps

Objectives

After completing this section, students should be able to configure systems to generate crash dumps.

Kernel crash dumps

When an application terminates abnormally, a dump of the process's memory image is captured into a file known as a *core dump*. The core file is then sent to the application vendor for analysis. Since the core file contains the contents of the application's memory space at the time of the fault, it can provide clues regarding the cause of the application's abnormal termination.

Similarly, when an operating system experiences a system hang or crash, it can also write the contents of the kernel's memory image to a file known as a *crash dump*. Since the kernel crash dump contains the contents of the kernel's memory space at the time of the system fault, its analysis can provide insight into its cause. By configuring their systems to capture crash dumps, administrators can provide their operating system vendor with the artifacts necessary to successfully determine and resolve the root cause of system hangs and crashes.

kdump and kexec

Red Hat Enterprise Linux offers the **kdump** software for the capture of kernel crash dumps. The **kdump** software works by using the **kexec** utility on a running system to boot a secondary Linux kernel without going through a system reset.

kexec is a kernel-to-kernel boot loader which provides the functionality of booting the secondary kernel and bypassing the BIOS. The secondary kernel, known as the *capture* kernel, boots up from a reserved memory area of the primary, running kernel to provide a platform for capturing the contents of the primary kernel's memory space to a crash dump file.

Other kernel crash mechanisms, such as *Linux Kernel Crash Dump (LKCD)*, perform crash dump in the context of the crashing, and consequently unreliable, kernel. In contrast, **kdump** captures crash dumps using a freshly booted kernel. The use of a separate capture kernel makes **kdump** an extremely reliable crash dump mechanism.

Configuring kdump

The **kdump** software is installed by default on Red Hat Enterprise Linux 7 through the installation of the **kexec-tools** package. The package provides the files and command line utilities necessary for administering **kdump** from the command-line. The *system-config-kdump* can also be installed to provide a graphical configuration tool for **kdump**.

Memory reservation

As previously mentioned, the **kdump** mechanism relies on **kexec** to boot a secondary capture kernel from a reserved memory area of the primary, running kernel. The size requirement of this reserved memory varies based on a system's hardware architecture and the total amount of physical memory installed.

For the x86_64 machine architecture, the minimum amount of reserved memory required is 160 MB. In addition, an additional 2 bits of reserved memory is required for every 4 KB of physical RAM installed on the system. As a example, a system with 512 GB of physical memory would require a total reserved memory of 192 MB: 160 MB baseline memory plus 32 MB additional memory due to the amount of physical RAM installed.

On most systems, the amount of memory required is properly calculated and automatically reserved by **kdump**. This behavior is configured by the **crashkernel=auto** setting in the **GRUB_CMDLINE_LINUX** parameter in the **/etc/default/grub** configuration file.

```
GRUB_CMDLINE_LINUX="console=tty0 crashkernel=auto no_timer_check net.ifnames=0
 console=ttyS0,115200n8"
```

In cases where the amount of reserved memory needs to be manually defined, customize the memory amount specified for the **crashkernel** parameter in **/etc/default/grub**. Once the changes to the file are complete, regenerate the GRUB2 configuration.

Use the following command for systems utilizing BIOS firmware.

```
[root@demo ~]# grub2-mkconfig -o /boot/grub2/grub.cfg
```

For systems utilizing UEFI firmware, use the following command.

```
[root@demo ~]# grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg
```

Systems then must be rebooted for the amount of reserved memory specified to take effect.

The kdump service

The **kdump** crash dump mechanism is provided through the **kdump** service. Administrators interested in enabling the collection of kernel crash dumps on their systems must ensure that the **kdump** service is enabled and started on each system.

```
[root@demo ~]# systemctl enable kdump
Created symlink from /etc/systemd/system/multi-user.target.wants/kdump.service to /usr/
lib/systemd/system/kdump.service.
[root@demo ~]# systemctl start kdump
[root@demo ~]# systemctl status kdump
● kdump.service - Crash recovery kernel arming
   Loaded: loaded (/usr/lib/systemd/system/kdump.service; enabled; vendor preset:
 enabled)
   Active: active (exited) since Thu 2016-02-18 00:40:38 EST; 4s ago
  Process: 479 ExecStart=/usr/bin/kdumpctl start (code=exited, status=0/SUCCESS)
 Main PID: 479 (code=exited, status=0/SUCCESS)

Feb 18 00:40:38 serverX.lab.example.com systemd[1]: Starting Crash recovery kernel
 arming...
Feb 18 00:40:38 serverX.lab.example.com kdumpctl[479]: kexec: loaded kdump kernel
Feb 18 00:40:38 serverX.lab.example.com kdumpctl[479]: Starting kdump: [OK]
Feb 18 00:40:38 serverX.lab.example.com systemd[1]: Started Crash recovery kernel
 arming.
```

With the **kdump** service enabled and started, kernel crash dumps will begin to be generated during system hangs and crashes. The behavior of the kernel crash dump and collection can be modified in various ways by editing the **/etc/kdump.conf** configuration file.

Designating crash dump targets

By default, **kdump** captures crash dumps locally to crash dump files located in subdirectories under the **/var/crash** path.

```
[root@demo ~]# ls -la /var/crash
total 4
drwxr-xr-x.  3 root root   42 Feb 17 01:08 .
drwxr-xr-x. 20 root root 4096 Feb 17 01:07 ..
drwxr-xr-x.  2 root root   42 Feb 18 01:28 127.0.0.1-2016-02-17-01:07:02
[root@demo ~]# ls -la /var/crash/127.0.0.1-2016-02-17-01\:07\:02/
total 47468
drwxr-xr-x. 2 root root       42 Feb 18 01:28 .
drwxr-xr-x. 3 root root       42 Feb 17 01:08 ..
-rw-------. 1 root root 48563396 Feb 17 01:07 vmcore
-rw-r--r--. 1 root root    39849 Feb 17 01:07 vmcore-dmesg.txt
```

The **vmcore** file contains the crash dump while the **vmcore-dmesg.txt** file contains the kernel log at the time of the crash. Since the crash dump file contains the memory dump from the system kernel, its size can be quite large, especially as the size of physical memory installations continues to increase in today's systems.

Due to its large size, the delivery of this file to Red Hat Support can take some time. To expedite the crash dump analysis, the much smaller **vmcore-dmesg.txt** file can be sent first to facilitate a preliminary assessment.

kdump's default behavior of writing crash dumps to the **/var/crash** directory, is configured with the following dump target option in **/etc/kdump.conf**.

```
path /var/crash
```

Besides writing to a local file system path, **kdump** also offers other types of crash dump targets with use of the following options in **/etc/kdump.conf**.

/etc/kdump.conf Options for Configuring Dump Target

Option	Description
raw *[partition]*	Use **dd** to copy the crash dump to the specified partition.
nfs *[nfs share]*	Mount and copy the crash dump to the location specified by the **path** option on the NFS share.
ssh *[user@server]*	Use **scp** to transfer the crash dump to the location specified by the **path** option on the specified remote server using the specified user account for authentication.
sshkey*[sshkeypath]*	Used in conjunction with the **ssh** crash dump type to specify the location of the SSH key to use for authentication.
[fs type] [partition]	Mount the specified partition containing the specified file system type to **/mnt** and then write the crash dump to the path location specified by the **path** option.
path *[path]*	Specifies the path to which the crash dump will be saved on the dump target. If no dump target is specified, then the path is assumed to be from the root of the local file system.

Core collection

By default, the collection of the crash dump is performed using the **makedumpfile** utility. The **core_collector** option is used in **/etc/kdump.conf** to specify how collection of the crash dump is to be performed.

```
core_collector makedumpfile -l --message-level 1 -d 31
```

The previous setting specifies that crash dump collection will be performed using the **makedumpfile** utility.

The **makedumpfile** utility offers the following options for the compression of crash dumps. The previous example uses the **lzo** compression algorithm. The following table lists the compression algorithms supported by **makedumpfile**.

makedumpfile Compression Options

Option	Description
-c	Use **zlib** for crash dump data compression.
-l	Use **lzo** for crash dump data compression.
-p	Use **snappy** for crash dump data compression.

The **makedumpfile** utility also offers options for customizing the message types that are included in the crash output. The previous example uses message level **1** which only includes a progress indicator in the crash output message. The following table lists some of the message levels offered.

Note

The full list of message levels offered can be referenced in the **makedumpfile(8)** man page.

makedumpfile Message Levels

Message level	Description
0	Do not include any messages.
1	Only include progress indicator.
4	Only include error messages.
31	Include all messages.

The **makedumpfile** utility also offers options for filtering the type of pages included in the crash dump. Administrators can choose to filter out zero pages, cached pages, and user data pages, as well as free pages, by specifying a dump level. This filtering can help to tremendously decrease the size of the crash dump.

The previous example uses dump level **31**, which excludes zero pages, cached pages, user data pages, and free pages. This dump level would offer the smallest crash dump.

Note

The full list of dump levels offered can be referenced in the **makedumpfile(8)** man page.

`makedumpfile` Dump Levels

Dump level	Description
0	Include all page types.
1	Do not include zero pages.
31	Exclude zero pages, cached pages, user data pages, and free pages.

For collection of crash dumps using SSH dump targets, the **scp** utility needs to be specified in place of **makedumpfile**.

```
core_collector scp
```

Using kdumpctl

The *kexec-tools* package includes a command-line utility, **kdumpctl**, which can perform some common kdump administration tasks.

```
[root@demo ~]# kdumpctl -h
Usage: /bin/kdumpctl {start|stop|status|restart|propagate}
```

The **kdumpctl** utility can be used to verify the status of the **kdump** service by using the **status** subcommand.

```
[root@demo ~]# kdumpctl status
Kdump is operational
```

With the **kdumpctl showmem** command, administrators can display the current amount of reserved memory configured for the capture kernel.

```
[root@demo ~]# kdumpctl showmem
Reserved 161MB memory for crash kernel
```

Lastly, the **kdumpctl propagate** command can be used to simplify the setup of SSH key-based authentication when **kdump** is configured to deliver crash dumps to a remote server using SSH. It consults the **sshkey** parameter in **/etc/kdump.conf** to determine the location of the SSH keys to use. If the key does not exist, **kdumpctl** will automatically create the key pair in the configured file. It then executes **ssh-copy-id** to transfer and install the public key on the target server specified by the **ssh** parameter.

```
[root@demo ~]# kdumpctl propagate
WARNING: '/root/.ssh/kdump_id_rsa' doesn't exist, using default value '/root/.ssh/
kdump_id_rsa'
Generating new ssh keys... done.
The authenticity of host 'serverY.lab.example.com (172.25.250.11)' can't be established.
ECDSA key fingerprint is 62:88:d6:2a:57:b1:3b:cd:9e:3c:52:e6:e3:94:f9:59.
Are you sure you want to continue connecting (yes/no)? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any
 that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now
 it is to install the new keys
root@serverY.lab.example.com's password: redhat
```

```
Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'root@serverY.lab.example.com'"
and check to make sure that only the key(s) you wanted were added.

/root/.ssh/kdump_id_rsa has been added to ~root/.ssh/authorized_keys on
 serverY.lab.example.com
```

Note

To simplify **kdump** configuration, Red Hat Labs offers the Kdump Helper online tool. This tool is designed to autogenerate a script based on users' responses regarding the intended use of their crash dumps.

The script configures **kdump** for the desired target, as well as message and dump level. In addition, it can also include **sysctl** configurations to trigger crash dumps for specific conditions such as soft lockups or out-of-memory events.

This online tool is located at *https://access.redhat.com/labs/kdumphelper/*.

Kernel crash dump triggers

When **kdump** is configured, a system will generate a crash dump when its kernel encounters an unrecoverable software or hardware error. Operating system failures that do not result in an unrecoverable error will not generate a crash dump. If a crash dump would be useful for troubleshooting these system failures, administrators can configure the operating system to trigger crash dumps when they occur.

OOM events

When a system runs out of memory, the Out of Memory (OOM) killer can be invoked to kill off processes in order to free up system memory and keep the system operational. A system can be configured to call a panic routine when OOM-killer events occur. The panic routine will in turn trigger the generation of a crash dump. The contents of the crash dump can provide insight into the system's memory state at the moment when the OOM-killer was invoked.

Administrators can temporarily configure a system to panic on OOM-killer events with the use of the following command:

```
[root@demo ~]# echo 1 > /proc/sys/vm/panic_on_oom
```

To make the configuration permanent, administrators should issue the following commands:

```
[root@demo ~]# echo "vm.panic_on_oom=1" >> /etc/sysctl.conf
[root@demo ~]# sysctl -p
```

Hung process

Systems can also be configured to trigger a crash dump when a process is deemed to be in a hung state. This condition occurs when a process is hung for a certain period of time, which by default is 120 seconds. The hung task timeout is configurable and its current setting can be viewed by displaying the contents of the **/proc/sys/kernel/hung_task_timeout_secs** file.

```
[root@demo ~]# cat /proc/sys/kernel/hung_task_timeout_secs
120
```

Administrators can temporarily configure a system to panic when processes are hung longer than the timeout value with the use of the following command:

```
[root@demo ~]# echo 1 > /proc/sys/kernel/hung_task_panic
```

To make the configuration permanent, administrators should issue the following commands:

```
[root@demo ~]# echo "kernel.hung_task_panic=1" >> /etc/sysctl.conf
[root@demo ~]# sysctl -p
```

Soft lockup

Soft lockups are the result of a software bug which causes the kernel to loop in kernel mode for an unreasonable amount of time. A kernel crash dump can also be configured to generate when soft lockups occur.

Administrators can temporarily configure a system to panic when soft lockups occur by using the following command:

```
[root@demo ~]# echo 1 > /proc/sys/kernel/softlockup_panic
```

To make the configuration permanent, administrators should issue the following commands:

```
[root@demo ~]# echo "kernel.softlockup_panic=1" >> /etc/sysctl.conf
[root@demo ~]# sysctl -p
```

Non-maskable interrupt

When a system encounters a nonrecoverable hardware failure, a *non-maskable interrupt (NMI)* can be used to trigger a kernel panic and generate a crash dump. The NMI can be generated automatically by the NMI Watchdog if it is enabled. It can also be generated manually by the administrator either by pressing the physical NMI button on the system hardware or a virtual NMI button from the system's out-of-band management interface, such as HP's iILO or Dell's iIDRAC.

Administrators can temporarily configure a system to panic when NMIs are detected using the following command.

```
[root@demo ~]# echo 1 > /proc/sys/kernel/panic_on_io_nmi
```

To make the configuration permanent, administrators should issue the following commands:

```
[root@demo ~]# echo "kernel.panic_on_io_nmi=1" >> /etc/sysctl.conf
[root@demo ~]# sysctl -p
```

Magic SysRq

The "Magic" SysRq key is a key sequence which can be used on a unresponsive system to diagnose kernel-related issues. This feature is disabled by default and can be enabled temporarily with the following command.

```
[root@demo ~]# echo 1 > /proc/sys/kernel/sysrq
```

To make the configuration permanent, administrators should issue the following commands:

```
[root@demo ~]# echo "kernel.sysrq=1" >> /etc/sysctl.conf
[root@demo ~]# sysctl -p
```

Once the SysRq facility has been enabled, the system can be triggered to execute SysRq events with the input of specific SysRq commands. These commands can be entered with the **Alt+PrintScreen+*[CommandKey]*** key sequence. The following table summarizes the SysRq commands and their associated events.

SysRq Commands and Associated Events

SysRq command	Event
m	Dump information about memory allocation.
t	Dump thread state information.
p	Dump CPU registers and flags.
c	Crash the system.
s	Sync mounted file systems.
u	Remount file systems read-only.
b	Initiate system reboot.
9	Power off the system.
f	Start OOM killer.
w	Dump hung processes.

An alternative to the previously mentioned method of issuing SysRq commands as a key combination is to issue SysRq commands by writing their associated SysRq key characters to **/proc/sysrq-trigger**. For example, the following command will initiate a system crash.

```
[root@demo ~]# echo 'c' > /proc/sysrq-trigger
```

Testing kdump

Once **kdump** has been configured, the kdump service must be restarted for the configuration settings to take effect. As previously mentioned, the **kdumpctl** can be used to verify the status of the **kdump** service.

To test other aspects of the **kdump** mechanism, a crash dump can be initiated through the SysRq facility. By enabling the SysRq facility and issuing the **c** command, a kernel crash sequence can be initiated.

```
[root@demo ~]# echo 1 > /proc/sys/kernel/sysrq
[root@demo ~]# echo c > /proc/sysrq-trigger
[200167.888304] SysRq : Trigger a crash
[200167.888508] BUG: unable to handle kernel NULL pointer dereference at
 (null)
[200167.888887] IP: [<ffffffff813b9716>] sysrq_handle_crash+0x16/0x20
[200167.889183] PGD 795be067 PUD 78575067 PMD 0
[200167.889405] Oops: 0002 [#1] SMP
```

```
... Output omitted ...
```

Analyzing kernel crash dumps

While analyzing kernel crash dumps can be highly specialized work, requiring detailed knowledge of Linux kernel internals, there are some basic analysis steps that can be performed by system administrators without a kernel hacking background. This includes looking at the list of open files at the time of the crash, running processes at the time of the crash, and more.

Preparing a system for crash dump analysis

Before a crash dump can be analyzed some packages will need to be installed:

- The *kernel-debuginfo* packages that matches the version of the kernel the dump was made on. This information can be found in the **vmcore-dmesg.txt** file that is stored alongside the kernel crash dump, or by running the **strings** command against the **vmcore** file itself.

  ```
  [root@demo 127.0.0.1-2016-03-15-08:28:06]# strings vmcore | head
  KDUMP
  Linux
  demo.example.com
  3.10.0-327.el7.x86_64
  #1 SMP Thu Oct 29 17:29:29 EDT 2015
  ...
  ```

- The *crash* package itself.

Analyzing crash dumps with crash

The **crash** command needs two parameters, the “debug” version of the kernel image (installed by the *kernel-debuginfo* package), and the kernel crash dump **vmcore** file. If the **vmcore** file omitted the **crash** session will run against the currently running kernel.

```
[root@demo ~]# crash \
> /usr/lib/debug/modules/3.10.0-327.el7.x86_64/vmlinux \
> /var/crash/127.0.0.1-2016-03-15-08:28:06
```

The resulting output will already show some useful information, including the reason for the kernel panic.

Inside the **crash** environment a number of useful commands are available. Detailed help can always be obtained by using the **help *[COMMAND]*** command. To exit, simply type **exit** at the prompt.

The follwoing commands can be useful for a system administrator investigating a kernel crash dump:

- **files *<PID>***: This command will show th open files for the specified process. The variation **foreach files** will show all open files for each process that was running at the time of the crash.

- **ps**: this command will show a listing of all porcesses that were running at the time of the crash. Various arguments can be used to alter the output, see **help ps**.

- **fuser *<PATHNAME>***: This command works like the regular **fuser** command, showing what processes were using a certain file or directory, and in what way. As an alternative to a pathname an inode number can specified in hexadecimal.

References

> Red Hat Enterprise Linux 7 Kernel Crash Dump Guide
> https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Kernel_Crash_Dump_Guide/

kdump(8), **kexec(8)**, **grub-mkconfig(1)**, **kdump.conf**(5), and **makedumpfile**(8) manual pages

Guided Exercise: Creating Kernel Crash Dumps

In this lab, you will configure and test kernel crash dumps.

Resources	
Files	• **/etc/kdump.conf** • **/var/kdump**
Machines	• **servera** • **serverb**

Outcome(s)

You should be able to configure **kdump** to dump to a remote server during a kernel crash.

Before you begin

Log into **servera** and **serverb** via the console as the **root** user.

You have been tasked to configure your servers to generate crash dumps during a system crash. For simpler administration and more efficient use of storage, you have decided to consolidate all crash dumps to a central server, **servera**. Configure **servera** so that crash dumps can be written by remote systems to the **/var/kdump** directory.

Use **serverb** to test your **kdump** configuration before deploying it to all of your remaining servers. Kernel crash dumps should be generated to the **/var/kdump** directory on **servera** over an SSH session using the **scp** command.

1. On **servera**, create the **/var/kdump** directory to make space available for crash dumps of remote systems.

```
[root@servera ~]# mkdir /var/kdump
```

2. On **serverb**, verify whether the **kdump** service is enabled and active. If not, enable and start the **kdump** service.

   2.1. Verify the status of the **kdump** service to see if the service has already been enabled and started. This service needs to be running in order for the kernel crash dumping mechanism to be triggered during a system crash.

```
[root@serverb ~]# systemctl is-enabled kdump
disabled
[root@serverb ~]# systemctl is-active kdump
inactive
```

   2.2. Since the service is not enabled and started, start and enable the service to enable kernel crash dumps and then verify the status of the service to make sure it started successfully.

```
[root@serverb ~]# systemctl enable kdump
Created symlink from /etc/systemd/system/multi-user.target.wants/kdump.service
 to /usr/lib/systemd/system/kdump.service.
```

```
[root@serverb ~]# systemctl start kdump
[root@serverb ~]# systemctl status kdump
● kdump.service - Crash recovery kernel arming
   Loaded: loaded (/usr/lib/systemd/system/kdump.service; enabled; vendor
 preset: enabled)
   Active: active (exited) since Wed 2016-02-10 22:40:28 EST; 6s ago
  Process: 1517 ExecStart=/usr/bin/kdumpctl start (code=exited, status=0/
SUCCESS)
 Main PID: 1517 (code=exited, status=0/SUCCESS)

Feb 10 22:40:27 serverb.lab.example.com systemd[1]: Starting Crash recovery k...
Feb 10 22:40:28 serverb.lab.example.com kdumpctl[1517]: kexec: loaded kdump k...
Feb 10 22:40:28 serverb.lab.example.com kdumpctl[1517]: Starting kdump: [OK]
Feb 10 22:40:28 serverb.lab.example.com systemd[1]: Started Crash recovery ke...
Hint: Some lines were ellipsized, use -l to show in full.
```

3. Edit **/etc/kdump.conf** to configure **kdump** so that crash dumps are deposited in the **/var/kdump** directory on **servera**.

   3.1. Enable **servera.lab.example.com** as an SSH target for **kdump** by adding the following entry.

   ```
   ssh root@servera.lab.example.com
   ```

   3.2. Specify **/root/.ssh/kdump_id_rsa** as the SSH key to use for authenticating to **servera** by adding the following entry.

   ```
   sshkey /root/.ssh/kdump_id_rsa
   ```

   3.3. Modify the **core_collector** entry so that **scp** will be used to deposit the crash dump on **servera**. This line should *replace* any existing **core_collector** lines.

   ```
   core_collector scp
   ```

   3.4. Modify the **path** entry so that crash dumps will be deposited in **/var/kdump** on **servera**.

   ```
   path /var/kdump
   ```

4. Use the **kdumpctl** to generate and remotely install the SSH key to be used for transferring the crash dump to **servera**.

```
[root@serverb ~]# kdumpctl propagate
WARNING: '/root/.ssh/kdump_id_rsa' doesn't exist, using default value '/root/.ssh/
kdump_id_rsa'
Generating new ssh keys... done.
The authenticity of host 'servera.lab.example.com (172.25.250.11)' can't be
 established.
ECDSA key fingerprint is 62:88:d6:2a:57:b1:3b:cd:9e:3c:52:e6:e3:94:f9:59.
Are you sure you want to continue connecting (yes/no)? yes
/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any
 that are already installed
/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now
 it is to install the new keys
```

```
root@servera.lab.example.com's password:redhat

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'root@servera.lab.example.com'"
and check to make sure that only the key(s) you wanted were added.

/root/.ssh/kdump_id_rsa has been added to ~root/.ssh/authorized_keys on
 servera.lab.example.com
```

5. Restart the **kdump** service so the changes made to its configuration take effect.

```
[root@serverb ~]# systemctl restart kdump
[root@serverb ~]# systemctl status kdump
● kdump.service - Crash recovery kernel arming
   Loaded: loaded (/usr/lib/systemd/system/kdump.service; enabled; vendor preset:
 enabled)
   Active: active (exited) since Wed 2016-02-10 23:33:16 EST; 18s ago
  Process: 3109 ExecStart=/usr/bin/kdumpctl start (code=exited, status=0/SUCCESS)
 Main PID: 3109 (code=exited, status=0/SUCCESS)

Feb 10 23:33:15 serverb.lab.example.com dracut[5229]: -rw-r--r--   1 root    ...
Feb 10 23:33:15 serverb.lab.example.com dracut[5229]: drwxr-xr-x   3 root    ...
Feb 10 23:33:15 serverb.lab.example.com dracut[5229]: drwxr-xr-x   2 root    ...
Feb 10 23:33:15 serverb.lab.example.com dracut[5229]: -rw-r--r--   1 root    ...
Feb 10 23:33:15 serverb.lab.example.com dracut[5229]: lrwxrwxrwx   1 root    ...
Feb 10 23:33:15 serverb.lab.example.com dracut[5229]: lrwxrwxrwx   1 root    ...
Feb 10 23:33:15 serverb.lab.example.com dracut[5229]: =======================...
Feb 10 23:33:16 serverb.lab.example.com kdumpctl[3109]: kexec: loaded kdump k...
Feb 10 23:33:16 serverb.lab.example.com kdumpctl[3109]: Starting kdump: [OK]
Feb 10 23:33:16 serverb.lab.example.com systemd[1]: Started Crash recovery ke...
Hint: Some lines were ellipsized, use -l to show in full.
```

6. Test the **kdump** configuration on **serverb** by triggering a system crash.

   6.1. Enable all kernel SysRq functions by setting the value within the **/proc/sys/kernel/sysrq** file to be **1**.

```
[root@serverb ~]# echo 1 > /proc/sys/kernel/sysrq
```

   6.2. Trigger a system crash by setting the value within the **/proc/sysrq-trigger** file to be **c**.

```
[root@serverb ~]# echo c > /proc/sysrq-trigger
```

7. Validate that the kernel crash dump was successfully generated to **servera** by looking for the presence of a new directory under **/var/kdump** containing the crash dump from **serverb**.

```
[root@servera ~]# ll /var/kdump
total 0
drwxr-xr-x. 2 root root 42 Feb 10 23:55 172.25.250.10-2016-02-10-23:55:28
[root@servera ~]# ll /var/kdump/172.25.250.10-2016-02-10-23\:55\:28/
total 1951732
-r--------. 1 root root 1998532608 Feb 10 23:55 vmcore
-rw-r--r--. 1 root root      39932 Feb 10 23:55 vmcore-dmesg.txt
```

8. Grade your work, then clean up your systems for the next exercise.

   8.1. Grade your work.

```
[student@workstation ~]$ lab kdump grade
```

   8.2. Clean up your systems for the next exercise.

```
[student@workstation ~]$ lab kdump reset
```

Kernel Debugging with SystemTap

Objectives

After completing this section, students should be able to compile SystemTap scripts into kernel modules for deployment on remote systems.

Introduction to SystemTap

The SystemTap framework allows easy probing and instrumentation of almost any component within the kernel. It provides administrators with a flexible scripting language and library by leveraging the **kprobes** facility within the Linux kernel.

Using **kprobes**, kernel programmers can attach instrumentation code to the start and/or end of any kernel function. SystemTap scripts specify where to attach probes and what data to collect when the probe executes. The **stap** command parses the script into dynamically generated C source code for a custom kernel module, compiles the instrumentation module, then calls **staprun** to insert the module, and when appropriate, remove the module from the kernel.

Installing SystemTap

Because SystemTap requires symbolic naming for instructions within the kernel, it depends on the following packages not usually found on production systems. These packages will pull in any required dependencies. Notably, *gcc* is a dependency of the *systemtap* package.

> ### Note
>
> The kernel-related package versions must match the target kernel's version number.

- *kernel-debuginfo*
- *kernel-devel*
- *systemtap*

Most of these packages and their dependencies are included as part of the Red Hat Enterprise Linux base distribution. However, *kernel-debuginfo* and its dependency *kernel-debuginfo-common* are provided by the **rhel-*MAJOR_VERS-SPIN*-debug-rpms** repository. An administrator will need to entitle the machine to this repository before installing these RPMs using **subscription-manager**. An example of how an administrator could use the **subscription-manager** command follows:

```
[root@demo ~]# subscription-manager repos --enable rhel-7-server-debug-rpms
```

In class, these RPMs are provided as part of the course content **yum** repository.

Note

One of the commands included in the *systemtap-client* package, a dependency of *systemtap*, is **stap-prep**. **stap-prep** is useful after an administrator has installed the *systemtap* package and registered the machine to the correct repos. When executed, **stap-prep** will look at the currently running kernel and install the matching *kernel-devel* and *kernel-debuginfo* packages.

The command **debuginfo-install** from the *yum-utils* package can also be used to install **-debuginfo* packages. This command will automatically enable any debuginfo repositories configured and install the **-debuginfo** packages for any packages requested.

Using **stap** to run SystemTap scripts

The *systemtap* package provides a variety of sample scripts that administrators may find useful for gathering data on their systems. These scripts are stored in **/usr/share/doc/systemtap-client-*/examples**. The scripts are further divided into several different subdirectories based on what type of information they have been written to collect. SystemTap scripts have an extension of **.stp**.

To compile and run these example scripts, or any SystemTap script for that matter, administrators use the **stap** command. The **stap** command will go through these five passes:

1. Parse the SystemTap script to validate its syntax.

2. Expand the script to include any probes and function calls.

3. Translate the script into C.

4. Compile the script into a kernel module.

5. Call the **staprun** command to load the kernel module, start the execution of the program, and unload the module at the end of the program run.

```
[root@demo ~]# stap /usr/share/doc/systemtap-client-*/examples/process/
syscalls_by_proc.stp
Collecting data... Type Ctrl-C to exit and display results
^C
#SysCalls  Process Name
2208       pgrep
1477       stapio
842        ksmtuned
776        Xorg
584        tuned
... Output Truncated ...
```

Running **stap** with the **-v** option will display the stages that **stap** executes to run the program.

```
[root@demo ~]# stap -v /usr/share/doc/systemtap-client-*/examples/process/
syscalls_by_proc.stp
Pass 1: parsed user script and 101 library script(s) using
 213896virt/31184res/2976shr/28836data kb, in 280usr/60sys/339real ms.
```

```
Pass 2: analyzed script: 396 probe(s), 18 function(s), 26 embed(s), 1 global(s) using
 488244virt/161312res/4352shr/157544data kb, in 5490usr/2210sys/8161real ms.
Pass 3: translated to C into "/tmp/stap7MUG27/
stap_820fd86a21675106576989f6c98eed6b_110418_src.c" using
 485776virt/164200res/7320shr/157544data kb, in 50usr/100sys/151real ms.
Pass 4: compiled C into "stap_820fd86a21675106576989f6c98eed6b_110418.ko" in
 14490usr/4680sys/19286real ms.
Pass 5: starting run.
Collecting data... Type Ctrl-C to exit and display results
#SysCalls  Process Name
283        stapio
108        tuned
100        pool
42         gnome-shell
... Output Truncated ...
Pass 5: run completed in 20usr/26040sys/54119real ms.
```

In the previous **stap** output, **stap** explicitly displays the stages it uses to progress from a text-based script to loading and running the kernel module. Notice **Pass 2**; during this pass, the script is expanded and translated to C. In **Pass 3**, the new C program is compiled from the temporary cache directory into a kernel module. In **Pass 4**, the kernel module is loaded from the cache directory. **Pass 5** handles the running of the SystemTap instrumentation and, when it is terminated, the unloading of the kernel module.

Compiling SystemTap programs to portable kernel modules

As part of a typical SystemTap compilation, the **stap** program will compile a kernel module. By default, this is stored in a cache directory in the home directory of the user running **stap**. However, administrators may find it useful to have **stap** halt its program run after it has built this kernel module, and further, store the module with a specific name in an alternate directory location.

Writing the kernel object out with a specific name and to a directory outside of the normal cache directory will facilitate other users, with the correct permissions, to be able to load and use these modules. Further, it will allow these modules to be easily copied to other systems for use, as long as the other target systems are running the same version of the kernel as the system where the SystemTap module was compiled.

```
[root@demo ~]# stap -p 4 -v -m syscalls_by_proc \
> /usr/share/doc/systemtap-client-*/examples/process/syscalls_by_proc.stp
```

The previous example will stop the **stap** run after **Pass 4** (**-p 4**). The resulting kernel module will be placed in **syscalls_by_proc.ko** (**-m syscalls_by_proc**) in the current working directory.

Note

The name used for the kernel module may contain only the following characters:

- ASCII lower-case alphabetic characters (a-z)
- Digits (0-9)
- Underscores (_)

Running SystemTap programs as a nonroot user

The SystemTap framework provides the ability for nonroot users to run SystemTap programs that have been compiled into kernel modules. Users must be added into either the **stapusr** or the **stapdev** group. Depending on which group the user belongs to, the level of privilege with SystemTap applications and the instrumentation kernel modules will be adjusted.

SystemTap group	Description
stapusr	User may run SystemTap instrumentation modules, but only if they exist in the **/lib/modules/$(uname -r)/systemtap** directory. Users with this group may not compile or otherwise interact with SystemTap. Further, the **/lib/modules/$(uname -r)/systemtap** directory must be owned by the root user and have write permissions for root only. User may only use **staprun** for loading and running SystemTap kernel modules. Users with this group are not permitted access to **stap**.
stapdev	Users with a **stapdev** group membership may compile their own SystemTap instrumentation kernel modules using **stap**. If the user also has **stapusr** membership, they may use **staprun** to load a module, even if it does not reside in the **/lib/modules/$(uname -r)/systemtap** directory.

Suppose an administrator had compiled a SystemTap instrumentation module called **/root/syscalls_by_proc.ko**. To make this module available for users of the **stapusr** group, they would perform the following:

```
[root@demo ~]# mkdir /lib/modules/$(uname -r)/systemtap
[root@demo ~]# ls -ld /lib/modules/$(uname -r)/systemtap
drwxr-xr-x. 2 root root 6 Jan 26 17:26 /lib/modules/3.10.0-123.el7.x86_64/systemtap
[root@demo ~]# cp /root/syscalls_by_proc.ko /lib/modules/$(uname -r)/systemtap
```

For their part, a user who belongs to **stapusr** could now run this staged instrumentation module.

```
[student@demo ~]$ staprun syscalls_by_proc
Collecting data... Type Ctrl-C to exit and display results
... Output Truncated ...
```

Alternately, **staprun** can be provided with the full path to the target module, but when called with a bare module name, it will look for the matching **.ko** file in the currently running kernel's **/lib/modules/*KERNELVERS*/systemtap** directory.

A user with both the **stapusr** and **stapdev** group memberships may run the **stap** command just as **root** can.

```
[student@demo ~]$ stap -v /usr/share/doc/systemtap-client-*/examples/process/
syscalls_by_proc.stp
```

Note

Because of this very special **root** privilege to compile and load kernel modules into the running kernel, it is recommended that administrators grant this permission sparingly.

Preparing a machine with SystemTap runtime environment

SystemTap instrumentation modules are used by Red Hat Support and customers to help gather data to determine the cause of issues. They are very useful, but having to install the software needed to compile SystemTap modules may not be a desirable package set to run on all machines. Considering file-system utilization, additional package updates, the system resources consumed to compile SystemTap programs, etc., administrators may be wary of deploying this technology. However, it is very common to only have the full SystemTap compilation suite installed on a single machine. On this machine, the instrumentation modules are compiled into kernel modules. After the modules are compiled, they are copied to the target systems, which have only the SystemTap runtime environment installed. The SystemTap runtime environment provides **staprun**, which will load and run the module.

For the SystemTap runtime environment, a machine needs a single package:

- *systemtap-runtime*

On Red Hat Enterprise Linux 7, *systemtap-runtime* is part of the **Base** package group, so it should be installed on most systems by default.

Deploying a SystemTap instrumentation module

After copying the compiled SystemTap instrumentation module to the target system, the only application available for loading and executing it is **staprun**. As **root**, the module may be executed from any directory location. As seen in a previous section, nonroot users may be privileged with group memberships in **stapusr** and/or **stapdev** to permit them to load and run the module as well.

A user with **stapdev** and **stapusr** membership on the runtime-only environment machine will be able to run the module from anywhere on the file system.

A user with membership in only **stapusr** will only be able to load and execute modules stored in the **/lib/modules/$(uname -r)/systemtap** directory.

References

Red Hat Enterprise Linux 7 SystemTap Beginners Guide

> Knowledgebase: "How to download debuginfo packages?"
> https://access.redhat.com/solutions/9907

stap(1) and **staprun**(8) man pages

Guided Exercise: Kernel Debugging with SystemTap

In this lab, you will compile and distribute a System Tap module.

Resources	
Files:	• **/usr/share/doc/systemtap-client-2.8/examples/io/iotop.stp** • **/lib/modules/3.10.0-327.el7.x86_64/systemtap**
Machines:	• **servera** • **serverb**

Outcome(s)

You should be able to compile a SystemTap program and distribute it for execution by a user on another system.

Before you begin

Prepare your systems for this exercise by running the command **lab systemtap setup** on your **workstation** machine.

```
[student@workstation ~#$ lab systemtap setup
```

A developer working on **serverb** complains about poor system performance. A junior administrator informs you that he has already tried to perform some troubleshooting and has determined that the poor performance is due to high system disk I/O activities. He has confirm this with the **iostat** utility which reports that the heavy I/O activity is occurring on the system's secondary disk, **/dev/vdb**.

The junior admin is seeking your assistance because he is unable to determine what process is responsible for the heavy I/O activities on that disk. Create and install a SystemTap program which can help him determine the cause of the issue. Also provide the developer the ability to execute the program using his **student** account so that he can troubleshoot himself if the issue arises in the future.

1. Install the *systemtap* software.

   1.1. Verify if the *kernel-debuginfo* and *kernel-devel* packages needed by *systemtap* are installed. If not, install the packages.

   ```
   [root@servera ~]# rpm -q kernel-debuginfo kernel-devel
   package kernel-debuginfo is not installed
   package kernel-devel is not installed
   [root@servera ~]# yum install -y kernel-debuginfo kernel-devel
   ```

   1.2. Install the *systemtap* package.

   ```
   [root@servera ~]# yum install -y systemtap
   ```

2. Determine if a suitable SystemTap script exists for determining disk I/O usage by process.

   2.1. Look through the I/O example scripts included in the *systemtap* package to see if any are suitable.

```
[root@servera ~]# ls -la /usr/share/doc/systemtap-client-2.8/examples/io
total 176
-rw-r--r--. 1 root root  448 Sep  2 13:06 deviceseeks.meta
-rwxr-xr-x. 1 root root  771 Sep  2 13:06 deviceseeks.stp
... Output omitted ...
-rw-r--r--. 1 root root  418 Sep  2 13:06 iotop.meta
-rwxr-xr-x. 1 root root  656 Sep  2 13:06 iotop.stp
... Output omitted ...
-rwxr-xr-x. 1 root root 1681 Sep  2 13:06 ttyspy.stp
-rw-r--r--. 1 root root  455 Sep  2 13:06 ttyspy.txt
```

   2.2. The **iotop.stp** script looks promising since its contents indicate that it will report on the top 10 IO processes.

```
[root@servera ~]# cat /usr/share/doc/systemtap-client-2.8/examples/io/iotop.stp
... Output omitted ...
# print top 10 IO processes every 5 seconds
probe timer.s(5) {
    foreach (name in writes)
        total_io[name] += writes[name]
... Output omitted ...
```

3. Compile the **iotop.stp** into a portable kernel module called **iotop**.

```
[root@servera ~]# cd /root; stap -v -p 4 -m iotop /usr/share/doc/systemtap-
client-2.8/examples/io/iotop.stp
Pass 1: parsed user script and 110 library script(s) using
 219940virt/37556res/3052shr/34748data kb, in 80usr/10sys/96real ms.
Pass 2: analyzed script: 5 probe(s), 7 function(s), 10 embed(s), 3 global(s) using
 347196virt/165800res/4216shr/162004data kb, in 1
290usr/340sys/1635real ms.
Pass 3: translated to C into "/tmp/stapZRfkvF/iotop_src.c" using
 347196virt/166144res/4560shr/162004data kb, in 10usr/30sys/41real
ms.
iotop.ko
Pass 4: compiled C into "iotop.ko" in 3660usr/680sys/4789real ms.
```

4. Distribute the **iotop** kernel module to **serverb**.

   4.1. On **serverb**, verify if the SystemTap kernel module directory already exists. If not, create the directory for the SystemTap module to reside in. Name the module directory after the version of the kernel running on **serverb**.

```
[root@serverb ~]# ls -la /lib/modules/$(uname -r)/systemtap
ls: cannot access /lib/modules/3.10.0-327.el7.x86_64/systemtap: No such file or
 directory
[root@serverb ~]# mkdir -p /lib/modules/$(uname -r)/systemtap
```

   4.2. Copy the **iotop.ko** kernel module from **servera** to the SystemTap kernel module directory on **serverb**.

```
[root@serverb ~]# rsync -avz -e ssh root@servera:/root/iotop.ko /lib/modules/
$(uname -r)/systemtap/
root@servera's password: redhat
receiving incremental file list
iotop.ko

sent 30 bytes  received 34918 bytes  13979.20 bytes/sec
total size is 115531  speedup is 3.31
```

5. On **serverb**, configure the **student** user account so that it can execute the newly installed SystemTap kernel module. Since the user account will only need to run SystemTap kernel modules, it only needs to be added to the **stapusr** group.

```
[root@serverb ~]# usermod -a -G stapusr student
```

6. On **serverb**, verify that the **student** user can successfully execute the newly installed SystemTap program. After several iterations of output, end the program by entering **Ctrl-C**.

```
[root@serverb ~]# su - student
[student@serverb ~]$ id
uid=1000(student) gid=1000(student) groups=1000(student),10(wheel),156(stapusr)
 context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[student@serverb ~]$ staprun iotop
         Process           KB Read      KB Written
              dd             41758             885
    cleandisk.sh              8831               0
   systemd-udevd                 0               0
          stapio                 0               0

         Process           KB Read      KB Written
              dd             42094             892
    cleandisk.sh              8901               0
      irqbalance                 3               0
            sshd                 0               0
          stapio                 0               0

         Process           KB Read      KB Written
              dd             42060             891
    cleandisk.sh              8895               0
            sshd                 0               0
          stapio                 0               0
          master                 0               0
          pickup                 0               0
Ctrl-C
```

7. Use the output of the SystemTap program to determine the cause of the heavy disk I/O and track down its parent process.

```
[root@serverb ~]# ps -lef | grep cleandisk.sh
0 S root     12663 12662 18  82   2 - 28312 wait   22:22 ?        00:00:59 /bin/
bash /usr/local/bin/cleandisk.sh
[root@serverb ~]# cat /usr/local/bin/cleandisk.sh
#!/bin/bash

while true; do
```

```
  dd if=/dev/zero of=/dev/vdb size=1024 count=1000000
done
```

8. Grade your work, then cleanup your systems for the next exercise.

   8.1. Grade your work.

   ```
   [student@workstation ~]$ lab systemtap grade
   ```

   8.2. Cleanup your systems for the next exercise.

   ```
   [student@workstation ~]$ lab systemtap reset
   ```

Lab: Troubleshooting Kernel Issues

In this lab, you will compile a SystemTap module and configure a system for crash dump.

Resources	
Files	• **/usr/share/doc/systemtap-client-2.8/examples/network/nettop.stp** • **/etc/kdump.conf** • **/var/crash**
Machines	• **servera** • **serverb**

Outcome(s)

You should be able to configure a system to collect a crash dump during a kernel crash and compile a SystemTap program as a portable kernel module.

Before you begin

Prepare your system for this exercise by running the command **lab kernel setup** on your **workstation** machine.

```
[student@workstation ~#$ lab kernel setup
```

A fellow administrator is experiencing issues on **serverb** during times of heavy network activity. The system's performance becomes poor and then eventually the system hangs. To try to determine the cause of the issue, he would like to be able to determine the amount of network activity associated with each network daemon when the system performance degrades.

The administrator noted that the SystemTap script located at **/usr/share/doc/systemtap-client-2.8/examples/network/nettop.stp** would provide the desired output. He has installed the *kernel-debuginfo*, *kernel-devel*, and *systemtap* packages on **servera**. He wanted to compile the script on **servera** as a portable kernel module. He is seeking your assistance because he is unable to compile the **nettop.stp** SystemTap script. Resolve the SystemTap compilation issue on **servera**.

To help the administrator identify the source of the problem on **serverb**, configure it to generate a crash dump when the system hangs. The dumps should be written locally to the **/var/crash** directory. To reduce the size of the crash dumps, compress them using the **lzo** algorithm and only include user data pages. Provide the maximum amount of diagnostic information by having the output generated during the crash dump include all message types.

1. On **servera**, attempt to compile the **nettop.stp** SystemTap script into a portable kernel module to see what errors were encountered by your fellow administrator.

2. Analyze the error messages and use them to troubleshoot and resolve the issue with the compilation.

3. Compile the SystemTap script into a portable kernel module. Store the compiled modules as **/root/nettop.ko** on **servera**.

4. On **serverb**, verify whether the **kdump** service is enabled and active. If not, enable and start the **kdump** service.

5. Configure **kdump** so that crash dumps are deposited in the **/var/crash** directory using the **lzo** compression algorithm and generating all message types.

6. Log into **serverb** via the console and test the **kdump** configuration by triggering a system crash.

7. Validate that the kernel crash dump was successfully generated by looking for the presence of a new directory under **/var/crash** containing the crash dump generated.

8. Grade your work, then clean up your systems for the next exercise.

   8.1. Grade your work.

   ```
   [student@workstation ~]$ lab kernel grade
   ```

   8.2. Reset all of your machines to provide a clean environment for the next exercises.

Solution

In this lab, you will compile a SystemTap module and configure a system for crash dump.

Resources	
Files	• **/usr/share/doc/systemtap-client-2.8/examples/network/nettop.stp** • **/etc/kdump.conf** • **/var/crash**
Machines	• **servera** • **serverb**

Outcome(s)

You should be able to configure a system to collect a crash dump during a kernel crash and compile a SystemTap program as a portable kernel module.

Before you begin

Prepare your system for this exercise by running the command **lab kernel setup** on your **workstation** machine.

```
[student@workstation ~#$ lab kernel setup
```

A fellow administrator is experiencing issues on **serverb** during times of heavy network activity. The system's performance becomes poor and then eventually the system hangs. To try to determine the cause of the issue, he would like to be able to determine the amount of network activity associated with each network daemon when the system performance degrades.

The administrator noted that the SystemTap script located at **/usr/share/doc/systemtap-client-2.8/examples/network/nettop.stp** would provide the desired output. He has installed the *kernel-debuginfo*, *kernel-devel*, and *systemtap* packages on **servera**. He wanted to compile the script on **servera** as a portable kernel module. He is seeking your assistance because he is unable to compile the **nettop.stp** SystemTap script. Resolve the SystemTap compilation issue on **servera**.

To help the administrator identify the source of the problem on **serverb**, configure it to generate a crash dump when the system hangs. The dumps should be written locally to the **/var/crash** directory. To reduce the size of the crash dumps, compress them using the **lzo** algorithm and only include user data pages. Provide the maximum amount of diagnostic information by having the output generated during the crash dump include all message types.

1. On **servera**, attempt to compile the **nettop.stp** SystemTap script into a portable kernel module to see what errors were encountered by your fellow administrator.

```
[root@servera ~]# stap -v -p 4 -m nettop /usr/share/doc/systemtap-client-2.8/
examples/network/nettop.stp
Pass 1: parsed user script and 110 library script(s) using
 219940virt/37560res/3052shr/34748data kb, in 90usr/10sys/97real ms.
semantic error: while resolving probe point: identifier 'kernel' at /usr/share/
systemtap/tapset/linux/networking.stp:80:5
        source:         =  kernel.function("dev_queue_xmit")
                           ^
```

```
semantic error: missing x86_64 kernel/module debuginfo [man warning::debuginfo]
 under '/lib/modules/3.10.0-327.el7.x86_64/build'

semantic error: while resolving probe point: identifier 'netdev' at /usr/share/doc/
systemtap-client-2.8/examples/network/nettop.stp
:6:7
        source: probe netdev.transmit
                      ^
... Output omitted ...
```

2. Analyze the error messages and use them to troubleshoot and resolve the issue with the compilation.

   2.1. Since the error messages report that the necessary debuginfo dependencies are missing, verify that the correct kernel packages are installed.

```
[root@servera ~]# rpm -qa | grep ^kernel
kernel-3.10.0-327.el7.x86_64
kernel-tools-3.10.0-327.el7.x86_64
kernel-debuginfo-common-x86_64-3.10.0-327.4.5.el7.x86_64
kernel-debuginfo-3.10.0-327.4.5.el7.x86_64
kernel-devel-3.10.0-327.el7.x86_64
kernel-tools-libs-3.10.0-327.el7.x86_64
```

   2.2. Uninstall the incorrect *kernel-debuginfo* and *kernel-debuginfo-common-x86_64* packages and install the version of the package which matches the installed kernel version.

```
[root@servera ~]# yum erase -y kernel-debuginfo kernel-debuginfo-common-x86_64
Loaded plugins: langpacks, search-disabled-repos
Resolving Dependencies
--> Running transaction check
... Output omitted ...
Removed:
  kernel-debuginfo.x86_64 0:3.10.0-327.4.5.el7
  kernel-debuginfo-common-x86_64.x86_64 0:3.10.0-327.4.5.el7

Complete!
```

```
[root@servera ~]# yum list kernel-debuginfo-$(uname -r) kernel-debuginfo-common-
x86_64-$(uname -r)
Loaded plugins: langpacks, search-disabled-repos
Available Packages
kernel-debuginfo.x86_64
 3.10.0-327.el7                                   rhel_debug
kernel-debuginfo-common-x86_64.x86_64
 3.10.0-327.el7                                   rhel_debug
```

```
[root@servera ~]# yum install -y kernel-debuginfo-3.10.0-327.el7.x86_64 kernel-
debuginfo-common-x86_64-3.10.0-327.el7.x86_64
Loaded plugins: langpacks, search-disabled-repos
Resolving Dependencies
--> Running transaction check
... Output omitted ...
Installed:
  kernel-debuginfo.x86_64 0:3.10.0-327.el7                  kernel-debuginfo-
common-x86_64.x86_64 0:3.10.0-327.el7
```

```
Complete!
```

3. Compile the SystemTap script into a portable kernel module. Store the compiled modules as **`/root/nettop.ko`** on **`servera`**.

```
[root@servera ~]# stap -v -p 4 -m nettop /usr/share/doc/systemtap-client-2.8/
examples/network/nettop.stp
Pass 1: parsed user script and 110 library script(s) using
 219940virt/37560res/3052shr/34748data kb, in 90usr/10sys/98real ms.
Pass 2: analyzed script: 5 probe(s), 7 function(s), 3 embed(s), 3 global(s) using
 344104virt/162788res/4220shr/158912data kb, in 12
10usr/300sys/1524real ms.
Pass 3: translated to C into "/tmp/stapdQMqAs/nettop_src.c" using
 344104virt/163132res/4564shr/158912data kb, in 10usr/30sys/41real
 ms.
nettop.ko
Pass 4: compiled C into "nettop.ko" in 3760usr/670sys/4390real ms.
```

4. On **`serverb`**, verify whether the **`kdump`** service is enabled and active. If not, enable and start the **`kdump`** service.

   4.1. Verify the status of the **`kdump`** service to see if the service has already been enabled and started. This service needs to be running in order for the kernel crash dumping mechanism to be triggered during a system crash.

```
[root@serverb ~]# systemctl is-enabled kdump
disabled
[root@serverb ~]# systemctl is-active kdump
inactive
```

   4.2. Since the service is not enabled and started, start and enable the service to enable kernel crash dumps and then verify the status of the service to make sure it started successfully.

```
[root@serverb ~]# systemctl enable kdump
Created symlink from /etc/systemd/system/multi-user.target.wants/kdump.service
 to /usr/lib/systemd/system/kdump.service.
[root@serverb ~]# systemctl start kdump
[root@serverb ~]# systemctl status kdump
● kdump.service - Crash recovery kernel arming
   Loaded: loaded (/usr/lib/systemd/system/kdump.service; enabled; vendor
 preset: enabled)
   Active: active (exited) since Wed 2016-02-10 22:40:28 EST; 6s ago
  Process: 1517 ExecStart=/usr/bin/kdumpctl start (code=exited, status=0/
SUCCESS)
 Main PID: 1517 (code=exited, status=0/SUCCESS)

Feb 10 22:40:27 serverb.lab.example.com systemd[1]: Starting Crash recovery k...
Feb 10 22:40:28 serverb.lab.example.com kdumpctl[1517]: kexec: loaded kdump k...
Feb 10 22:40:28 serverb.lab.example.com kdumpctl[1517]: Starting kdump: [OK]
Feb 10 22:40:28 serverb.lab.example.com systemd[1]: Started Crash recovery ke...
Hint: Some lines were ellipsized, use -l to show in full.
```

5. Configure **`kdump`** so that crash dumps are deposited in the **`/var/crash`** directory using the **`lzo`** compression algorithm and generating all message types.

5.1. Edit **/etc/kdump.conf** and modify the **core_collector** entry so that **lzo** compression will be used and all message types are included in the output. Also ensure that only user data pages are included in the dump.

```
core_collector makedumpfile -l --message-level 31 -d 23
```

5.2. Restart the **kdump** service so the changes made to its configuration take effect.

```
[root@serverb ~]# systemctl restart kdump
[root@serverb ~]# systemctl status kdump
● kdump.service - Crash recovery kernel arming
   Loaded: loaded (/usr/lib/systemd/system/kdump.service; enabled; vendor
 preset: enabled)
   Active: active (exited) since Tue 2016-02-16 02:48:58 EST; 6s ago
  Process: 14158 ExecStop=/usr/bin/kdumpctl stop (code=exited, status=0/SUCCESS)
  Process: 14167 ExecStart=/usr/bin/kdumpctl start (code=exited, status=0/
SUCCESS)
 Main PID: 14167 (code=exited, status=0/SUCCESS)

Feb 16 02:48:58 serverb.lab.example.com dracut[16307]: drwxr-xr-x   3 root
 root            0 Feb 16 02:48 usr/share/zoneinfo
Feb 16 02:48:58 serverb.lab.example.com dracut[16307]: drwxr-xr-x   2 root
 root            0 Feb 16 02:48 usr/share/zon...erica
Feb 16 02:48:58 serverb.lab.example.com dracut[16307]: -rw-r--r--   1 root
 root         3519 Oct  1 19:53 usr/share/zon..._York
Feb 16 02:48:58 serverb.lab.example.com dracut[16307]: drwxr-xr-x   2 root
 root            0 Feb 16 02:48 var
Feb 16 02:48:58 serverb.lab.example.com dracut[16307]: lrwxrwxrwx   1 root
 root           11 Feb 16 02:48 var/lock -> ..../lock
Feb 16 02:48:58 serverb.lab.example.com dracut[16307]: lrwxrwxrwx   1 root
 root            6 Feb 16 02:48 var/run -> ../run
Feb 16 02:48:58 serverb.lab.example.com dracut[16307]:
 ========================================================================
Feb 16 02:48:58 serverb.lab.example.com kdumpctl[14167]: kexec: loaded kdump
 kernel
Feb 16 02:48:58 serverb.lab.example.com kdumpctl[14167]: Starting kdump: [OK]
Feb 16 02:48:58 serverb.lab.example.com systemd[1]: Started Crash recovery
 kernel arming.
Hint: Some lines were ellipsized, use -l to show in full.
```

6. Log into **serverb** via the console and test the **kdump** configuration by triggering a system crash.

6.1. Enable all kernel SysRq functions by setting the value within the **/proc/sys/kernel/sysrq** file to be **1**.

```
[root@serverb ~]# echo 1 > /proc/sys/kernel/sysrq
```

6.2. Trigger a system crash by setting the value within the **/proc/sysrq-trigger** file to be **c**.

```
[root@serverb ~]# echo c > /proc/sysrq-trigger
```

7. Validate that the kernel crash dump was successfully generated by looking for the presence of a new directory under **/var/crash** containing the crash dump generated.

```
[root@serverb ~]# ll /var/crash
total 0
drwxr-xr-x. 2 root root 42 Feb 10 23:55 127.0.0.1-2016-02-10-23:55:28
[root@serverb ~]# ll /var/crash/127.0.0.1-2016-02-10-23\:55\:28/
total 1951732
-r--------. 1 root root 1998532608 Feb 10 23:55 vmcore
-rw-r--r--. 1 root root      39932 Feb 10 23:55 vmcore-dmesg.txt
```

8. Grade your work, then clean up your systems for the next exercise.

   8.1. Grade your work.

   ```
   [student@workstation ~]$ lab kernel grade
   ```

   8.2. Reset all of your machines to provide a clean environment for the next exercises.

Summary

In this chapter, you learned:

- The amount of memory reserved for the crash kernel can be displayed with the **kdumpctl showmem** command.
- The amount of memory reserved for the crash kernel can be customized in the **/etc/default/grub** configuration file.
- The behavior of the **kdump** service is configured in **/etc/kdump.conf**.
- A kernel crash dump can be configured to be triggered by OOM, hung process, soft lookup, and NMI events using the **sysctl** utility.
- SystemTap requires the installation of the *kernel-debuginfo* package.
- The **stap** utility can be used to run and compile SystemTap scripts into portable kernel modules.
- Membership in the **stapusr** group allows users to run SystemTap instrumentation modules.
- Membership in the **stapdev** group allows users to compile SystemTap instrumentation modules.

CHAPTER 11

RED HAT ENTERPRISE LINUX DIAGNOSTICS AND TROUBLESHOOTING COMPREHENSIVE REVIEW

Overview	
Goal	Practice and demonstrate knowledge and skills learned in Red Hat Enterprise Linux Diagnostics and Troubleshooting.
Objectives	• Review and refresh knowledge and skills learned in Red Hat Enterprise Linux Diagnostics and Troubleshooting. • Practice and demonstrate knowledge and skills learned in Red Hat Enterprise Linux Diagnostics and Troubleshooting.
Sections	• Comprehensive Review of Red Hat Enterprise Linux Diagnostics and Troubleshooting • Comprehensive Review Labs
Lab	• Lab: Amsterdam — Can't Get Custom Application Working • Lab: Tokyo — Problems Logging In To The Console • Lab: Paris — Authentication Problems • Lab: London — A Web Server Issue • Lab: New York — Network Delays

Comprehensive Review of Red Hat Enterprise Linux Diagnostics and Troubleshooting

Objectives

After completing this section, students should be able to review and refresh knowledge and skills learned in Red Hat Enterprise Linux Diagnostics and Troubleshooting.

Reviewing Red Hat Enterprise Diagnostics and Troubleshooting

Before beginning the comprehensive review for this course, students should be comfortable with the topics covered in each chapter.

Students can refer to earlier sections in the textbook for extra study.

Chapter 1, What Is Troubleshooting

Describe a generalized strategy for troubleshooting.

- Identify a systematic approach to troubleshooting using the scientific method.
- Collect various pieces of information to aid in troubleshooting.
- Use Red Hat resources to aid in troubleshooting.

Chapter 2, Being Proactive

Prevent small issues from becoming large problems by employing proactive system administration techniques.

- Monitor systems for vital characteristics.
- Configure systems to send logging messages to a centralized host.
- Configure configuration management to aid in large-scale system administration.
- Implement change tracking to keep systems homogenous.

Chapter 3, Troubleshooting Boot Issues

Identify and resolve issues that can affect a system's ability to boot.

- Fix boot issues on traditional BIOS systems.
- Fix boot problems on UEFI systems.
- Identify and resolve service failures affecting boot.
- Regain root control of a system.

Chapter 4, Identifying Hardware Issues

Identify hardware problems that can affect a system's ability to operate normally.

- Identify hardware and hardware problems.
- Manage kernel modules and their parameters.
- Identify and resolve virtualization issues.

Chapter 5, Troubleshooting Storage Issues

Identify and fix issues related to storage.

- Describe the functions of the layers of the Linux storage stack
- Recover corrupted file systems.
- Recover from broken LVM configurations.
- Recover data from encrypted file systems.
- Identify and fix iSCSI issues.

Chapter 6, Troubleshooting RPM Issues

Identify and fix problems in, and using, the package management subsystem.

- Resolve package management dependency issues manually.
- Recover a corrupted Red Hat Package Management (RPM) database.
- Identify and restore changed files using the package management subsystem.
- Employ the Red Hat subscription management tools to receive updates.

Chapter 7, Troubleshooting Network Issues

Identify and resolve network connectivity issues.

- Verify network connectivity using standard tools.
- Identify and fix network connectivity issues.
- Inspect network traffic to aid troubleshooting.

Chapter 8, Troubleshooting Application Issues

Debug application issues.

- Identify library dependencies for third-party software.
- Identify if an application suffers from memory leaks.
- Debug an application using standard tools.

Chapter 9, Dealing with Security Issues

Identify and fix issues related to security subsystems.

- Identify and fix issues related to SELinux.
- Identify and fix issues in user authentication and authorization.
- Identify and fix issues related to LDAP and Kerberos identity management.

Chapter 10, Troubleshooting Kernel Issues

Identify kernel issues and assist Red Hat Support in resolving kernel issues.

- Create kernel crash dumps to assist Red Hat Support in resolving kernel issues.
- Compile and execute SystemTap modules to assist Red Hat Support in resolving kernel issues.

Comprehensive Review Labs

Objectives

After completing this section, students should be able to practice and demonstrate knowledge and skills learned in Red Hat Enterprise Linux Troubleshooting.

Comprehensive Reviews

In the following exercises, you are a senior systems administrator working for a large international company. During a recent tour of your main corporate offices, you encountered a number of issues that required your experience and skills to solve.

The following five exercises are some highlights of the problems you encountered. Some of these exercises might have more than one problem contained in them, as previous fix attempts can have left machines in a less-than-ideal state.

During these exercises, remember to use a structured approach, and to make copious notes on what you discover, and change.

Lab: Amsterdam — Can't Get Custom Application Working

In this lab, you will solve a problem on your **servera** system.

Resources	
Files	/usr/local/bin/pre-upower-version
Machines	servera

Outcome(s)

You should be able to successfully solve the issue(s) on your **servera** machine.

Before you begin

Prepare your **servera** machine for this exercise by running the **lab amsterdam setup** command on your **workstation** machine.

```
[student@workstation ~]$ lab amsterdam setup
```

Another junior system administrator installed a custom version of the **upower** application and has been unable to get it working. You have the task of getting it to work.

Confirm the application is ready to be used by running the following commands:

```
[root@servera ~]# /usr/libexec/upowerd --help
[root@servera ~]# upower --version
```

The first command should produce a useful help message. The second command should display the current version of **upower**.

1. Investigate, and provide a workaround for, the issues that cause **/usr/libexec/upowerd --help** to fail. It is understood that a workaround will have to do until the upstream vendor has provided a more permanent fix.

2. Investigate, and fix, the system requirement needed to make **upower --version** work. Once the prerequisite condition is identified, create a shell script called **/usr/local/bin/pre-upower-version** that implements it.

3. Grade your work by running the **lab amsterdam grade** command on your **workstation** system.

   ```
   [student@workstation ~]$ lab amsterdam grade
   ```

4. Before you continue with the next exercise, either reset your **servera** virtual machine or run the following command on your **workstation** system.

   ```
   [student@workstation ~]$ lab amsterdam reset
   ```

Solution

In this lab, you will solve a problem on your **servera** system.

Resources	
Files	**/usr/local/bin/pre-upower-version**
Machines	**servera**

Outcome(s)

You should be able to successfully solve the issue(s) on your **servera** machine.

Before you begin

Prepare your **servera** machine for this exercise by running the **lab amsterdam setup** command on your **workstation** machine.

```
[student@workstation ~]$ lab amsterdam setup
```

Another junior system administrator installed a custom version of the **upower** application and has been unable to get it working. You have the task of getting it to work.

Confirm the application is ready to be used by running the following commands:

```
[root@servera ~]# /usr/libexec/upowerd --help
[root@servera ~]# upower --version
```

The first command should produce a useful help message. The second command should display the current version of **upower**.

1. Investigate, and provide a workaround for, the issues that cause **/usr/libexec/upowerd --help** to fail. It is understood that a workaround will have to do until the upstream vendor has provided a more permanent fix.

   1.1. First, run the command to see if any useful error messages display.

   ```
   [root@servera ~]# /usr/libexec/upowerd --help
   /usr/libexec/upowerd: error while loading shared libraries:
    libimobiledevice.so.4: cannot open shared object file: No such file
    or directory
   ```

   A shared library, **libimobiledevice.so.4**, cannot be resolved by the runtime linker.

   1.2. Use **ldd** to identify all of the shared libraries that cannot currently be resolved.

   ```
   [root@servera ~]# ldd /usr/libexec/upowerd | grep 'not found'
           libimobiledevice.so.4 => not found
           libplist.so.1 => not found
   ```

   1.3. Use Yum to identify any packages that might provide the required shared libraries, **libimobiledevice.so.4** and **libplist.so.1**.

   ```
   [root@servera ~]# yum whatprovides libimobiledevice.so.4 libplist.so.1
   Loaded plugins: langpacks, search-disabled-repos
   libimobiledevice-1.1.5-6.el7.i686 : Library for connecting to mobile devices
   ```

```
Repo        : rhel_dvd
Matched from:
Provides    : libimobiledevice.so.4
... Output omitted ...
libplist-1.10-4.el7.i686 : Library for manipulating Apple Binary and XML
                         : Property Lists
Repo        : rhel_dvd
Matched from:
Provides    : libplist.so.1
... Output omitted ...
```

1.4. Install the packages that will provide the required shared libraries.

```
[root@servera ~]# yum -y install libimobiledevice libplist
... Output omitted ...
```

1.5. Run the **/usr/libexec/upowerd --help** command again. Confirm that it works by producing a helpful usage message.

```
[root@servera ~]# /usr/libexec/upowerd --help
Usage:
  upowerd [OPTION...] upower daemon

Help Options:
  -h, --help           Show help options

Application Options:
  --timed-exit         Exit after a small delay
  --immediate-exit     Exit after the engine has loaded
  -v, --verbose        Show extra debugging information
```

2. Investigate, and fix, the system requirement needed to make **upower --version** work. Once the prerequisite condition is identified, create a shell script called **/usr/local/bin/pre-upower-version** that implements it.

2.1. Run the command to see if any useful error messages display.

```
[root@servera ~]# upower --version
Error: connect failed.: Connection refused
```

There is a runtime issue that needs further investigation. Some sort of connection is failing, and the error message is not that helpful.

2.2. Use **strace** to display the system calls that the program is making. Look for the system calls that occur just before the error message is printed and take note of their arguments.

```
[root@servera ~]# strace upower --version
execve("/usr/bin/upower", ["upower", "--version"], [/* 24 vars */]) = 0
brk(0)                                  = 0x7f03f8d89000
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
 0x7f03f7615000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
open("/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=29146, ...}) = 0
mmap(NULL, 29146, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f03f760d000
```

```
close(3)                                = 0
... Output omitted ...
socket(PF_INET, SOCK_STREAM, IPPROTO_IP) = 6
connect(6, {sa_family=AF_INET, sin_port=htons(8765),
 sin_addr=inet_addr("127.0.0.1")}, 16) = -1 ECONNREFUSED (Connection refused)
dup(2)                                  = 7
fcntl(7, F_GETFL)                       = 0x8402 (flags O_RDWR|O_APPEND|
O_LARGEFILE)
... Output omitted ...
write(7, "Error: connect failed.: Connecti"..., 43Error: connect failed.:
 Connection refused
) = 43
close(7)                                = 0
munmap(0x7f03f7614000, 4096)            = 0
exit_group(1)                           = ?
+++ exited with 1 +++
```

The system call that fails is the call to **connect()**. It is trying to connect to IPv4 address 127.0.0.1, TCP port 8765. This is what is causing **upower --version** to fail.

2.3. Use Yum to install the *nmap-ncat* RPM. It will be used to listen on port 8765 on **servera**.

```
[root@servera ~]# yum -y install nmap-ncat
... Output omitted ...
```

2.4. Open another window and log into **servera**. Launch **nc** and have it listen to port 8765 on the 127.0.0.1 address.

```
[student@servera ~]$ nc -l 127.0.0.1 8765
```

It will hang waiting for an incoming TCP connection.

2.5. In the first window, run the **upower --version** command again. Confirm that it works.

```
[root@servera ~]# upower --version
UPower client version 0.99.2
UPower daemon version 0.99.2
```

2.6. Create the **/usr/local/bin/pre-upower-version** program to listen to TCP port 8765 on 127.0.0.1. Use an editor to create the file so that it has the following contents, and make it executable.

```
[root@servera ~]# cat /usr/local/bin/pre-upower-version
#!/bin/bash
ncat -l 127.0.0.1 8765
[root@servera ~]# chmod 755 /usr/local/bin/pre-upower-version
```

3. Grade your work by running the **lab amsterdam grade** command on your **workstation** system.

```
[student@workstation ~]$ lab amsterdam grade
```

4. Before you continue with the next exercise, either reset your **servera** virtual machine or run the following command on your **workstation** system.

```
[student@workstation ~]$ lab amsterdam reset
```

Lab: Tokyo — Problems Logging In To The Console

In this lab, you will solve a problem on your **serverb** system.

Resources	
Machines	• **serverb**

Outcome(s)

You should be able to successfully solve the issue(s) on your **serverb** machine.

Before you begin

Prepare your **serverb** machine for this exercise by running the **lab tokyo setup** command on your **workstation** machine.

```
[student@workstation ~]$ lab tokyo setup
```

One of your users was reporting issues logging in directly to the console of your **serverb** machine as the user **student** with the password **student**.

The ticket for this issue was assigned to one of your junior admins, who has attempted to fix the issue. Unfortunately, during the "fixing" process, your **serverb** machine has become unreachable.

Shocked by this result, your junior admin cannot remember any of the steps that were taken up to this point.

Investigate this issue, and fix both the original problem and the one introduced by your junior admin.

1. Investigate, and fix, the case of the unreachable **serverb** machine.

2. Investigate, and fix, the original console login problem.

3. Grade your work by running the command **lab tokyo grade** on your **workstation** system.

   ```
   [student@workstation ~]$ lab tokyo grade
   ```

4. If you did *not* successfully solve the issue, but wish to continue with the next exercise, reset your **serverb** system.

Solution

In this lab, you will solve a problem on your **serverb** system.

Resources	
Machines	• **serverb**

Outcome(s)

You should be able to successfully solve the issue(s) on your **serverb** machine.

Before you begin

Prepare your **serverb** machine for this exercise by running the **lab tokyo setup** command on your **workstation** machine.

```
[student@workstation ~]$ lab tokyo setup
```

One of your users was reporting issues logging in directly to the console of your **serverb** machine as the user **student** with the password **student**.

The ticket for this issue was assigned to one of your junior admins, who has attempted to fix the issue. Unfortunately, during the "fixing" process, your **serverb** machine has become unreachable.

Shocked by this result, your junior admin cannot remember any of the steps that were taken up to this point.

Investigate this issue, and fix both the original problem and the one introduced by your junior admin.

1. Investigate, and fix, the case of the unreachable **serverb** machine.

   1.1. Verify that **serverb** is indeed unreachable.

   ```
   [student@workstation ~]$ ping serverb.lab.example.com
   ```

   1.2. Open a console to **serverb**.

   It appears that **serverb** is stuck on the **grub2** menu screen. Attempt to boot the default entry; if that does not work, attempt to boot a backup entry.

   1.3. What error did you receive when trying to boot those entries?

   ```
   error: can't find command `linux64'.
   error: you need to load the kernel first.
   ```

   Temporarily fix this issue: Press **e** to edit the default entry, change **linux64** to **linux16**, then press **Ctrl**+**x** to boot.

   1.4. Try to log in as **root** on the console, using the password **redhat**.

   It appears that the original login problem is still present. Attempt to log in as **root** over SSH.

```
[student@workstation ~]$ ssh root@serverb
```

Luckily, SSH still works. Permanently fix the boot issue by recreating the **grub2** configuration file.

```
[root@serverb ~]# grub2-mkconfig -o /boot/grub2/grub.cfg
```

2. Investigate, and fix, the original console login problem.

2.1. Attempt to log into the console on **serverb** as **student**, with the password **student**.

Luckily, SSH access seems to be unaffected.

2.2. Check **/var/log/secure** or **journalctl** for any relevant messages.

```
[root@serverb ~]# journalctl _COMM=login
...
login[1240]: FAILED LOGIN 1 FROM tty1 FOR student, Authentication failure
```

This message does not indicate an incorrect password, so something else must be wrong.

2.3. What file(s) control(s) authentication for the **login** process? Check them for any weird settings.

/etc/pam.d/login and **/etc/pam.d/system-auth**. On a rough inspection, they seem to be fine, but **/etc/pam.d/system-auth-ac** does seem to have had recent changes.

2.4. Use **authconfig** to recreate all configuration files.

```
[root@serverb ~]# authconfig --updateall
```

2.5. Verify that the previous step solved the issue by logging into the console of **serverb** as **student**, with the password **student**.

3. Grade your work by running the command **lab tokyo grade** on your **workstation** system.

```
[student@workstation ~]$ lab tokyo grade
```

4. If you did *not* successfully solve the issue, but wish to continue with the next exercise, reset your **serverb** system.

Lab: Paris — Authentication Problems

In this lab, you will solve an authentication-related issue.

Resources	
Machines	• **workstation** • **servera**

Outcome(s)
You should be able to diagnose and fix an issue related to authentication.

Before you begin
Prepare your systems for this exercise by running the command **lab paris setup** on your **workstation** system.

```
[student@workstation ~]$ lab paris setup
```

Recently a new server, **servera**, has been joined to your authentication domain; unfortunately, this process was not completed successfully. Reports have been coming in from your users that they cannot log into this system. You have been given a test account **ldapuser** with a password of **kerberos**.

The following table lists all settings as they were given to the person joining **servera** to your authentication domain.

Settings	
LDAP server	**ldap://workstation.lab.example.com**
LDAP search base	**dc=lab,dc=example,dc=com**
LDAP CA certificate	**ftp://workstation.lab.example.com/pub/example-ca.crt**
Kerberos realm	**LAB.EXAMPLE.COM**
Kerberos KDC	**workstation.lab.example.com**
NFSv4 home directories	**workstation.lab.example.com:/home/guests/***

As with all other systems in your organization, the use of **sssd.service** is required.

1. Reproduce the problem, taking note of any error messages.

2. Collect more information, then form a hypothesis.

3. Test your hypothesis, then, if correct, fix the issue.

4. Verify your work by running the command **lab paris grade** from your **workstation** machine.

```
[student@workstation ~]$ lab paris grade
```

5. *Important*: Clean up your work by resetting both your **`workstation`** and your **`servera`** machine.

Solution

In this lab, you will solve an authentication-related issue.

Resources	
Machines	• **workstation** • **servera**

Outcome(s)

You should be able to diagnose and fix an issue related to authentication.

Before you begin

Prepare your systems for this exercise by running the command **lab paris setup** on your **workstation** system.

```
[student@workstation ~]$ lab paris setup
```

Recently a new server, **servera**, has been joined to your authentication domain; unfortunately, this process was not completed successfully. Reports have been coming in from your users that they cannot log into this system. You have been given a test account **ldapuser** with a password of **kerberos**.

The following table lists all settings as they were given to the person joining **servera** to your authentication domain.

Settings	
LDAP server	**ldap://workstation.lab.example.com**
LDAP search base	**dc=lab,dc=example,dc=com**
LDAP CA certificate	**ftp://workstation.lab.example.com/pub/example-ca.crt**
Kerberos realm	**LAB.EXAMPLE.COM**
Kerberos KDC	**workstation.lab.example.com**
NFSv4 home directories	**workstation.lab.example.com:/home/guests/***

As with all other systems in your organization, the use of **sssd.service** is required.

1. Reproduce the problem, taking note of any error messages.

    1.1. Attempt to log in as **ldapuser** on **servera**, using the password **kerberos**, both using **ssh** and on the console.

    1.2. Using **ssh ldapuser@servera** asks for a password, but fails when the password is entered.

    1.3. Directly logging into the console gives a **Login incorrect** message directly after entering the username.

2. Collect more information, then form a hypothesis.

    2.1. Read the logs related to **sshd** and **login**. What is said about the user **ldapuser**?

```
[root@servera ~]# journalctl _COMM=sshd + _COMM=login
```

Both the entries for **sshd** and **login** list **ldapuser** as an invalid user.

2.2. Verify that there is indeed no user named **ldapuser** on **servera**.

```
[root@servera ~]# getent passwd ldapuser
```

2.3. Use the **ldapsearch** command to query the configured LDAP server.

```
[root@servera ~]# ldapsearch -x
ldap_bind: Confidentiality required (13)
        additional info: TLS confidentiality required
```

This server requires the use of TLS; try again, enforcing TLS usage.

```
[root@servera ~]# ldapsearch -x -ZZ
ldap_start_tls: Connect error (-11)
        additional info: TLS error -8179:Peer's Certificate issuer is not
 recognized.
```

2.4. Hypothesis: The CA certificate for the LDAP server is not installed properly on **servera**.

3. Test your hypothesis, then, if correct, fix the issue.

3.1. Inspect the location for storing OpenLDAP CA certificates.

```
[root@servera ~]# ls -l /etc/openldap/cacerts
```

That directory seems to be empty. That would explain the failure to verify the TLS certificate for our LDAP server.

3.2. Download the correct certificate to that location, then add the hashes need by the OpenLDAP client tools.

```
[root@servera ~]# wget -O /etc/openldap/cacerts/example-ca.crt ftp://
workstation/pub/example-ca.crt
[root@servera ~]# cacertdir_rehash /etc/openldap/cacerts
```

3.3. Verify that **ldapsearch** now works as expected.

```
[root@servera ~]# ldapsearch -x -ZZ
....
dn: uid=ldapuser,ou=People,dc=lab,dc=example,dc=com
....
```

3.4. Restart the **sssd.service**.

```
[root@servera ~]# systemctl restart sssd.service
```

3.5. Verify that the issue is now fixed with **getent**.

```
[root@servera ~]# getent passwd ldapuser
ldapuser:*:1701:1701:LDAP Test User:/home/guests/ldapuser:/bin/bash
```

3.6. Restart the **autofs.service** service.

```
[root@servera ~]# systemctl restart autofs.service
```

3.7. Verify that you can now log in as **ldapuser** with the password **kerberos**. When Using **ssh** from **workstation** SSH key-based authentication will be used, and thus the home directory for **ldapuser** will be inaccessible. Either test with key-based authentication disables, or try on the console of **servera**.

4. Verify your work by running the command **lab paris grade** from your **workstation** machine.

```
[student@workstation ~]$ lab paris grade
```

5. *Important*: Clean up your work by resetting both your **workstation** and your **servera** machine.

Lab: London — A Web Server Issue

In this lab, you will troubleshoot and resolve a service failure and a corrupted file system.

Resources	
Application	`http://servera.lab.example.com`
Machines	• `servera`

Outcome(s)

You should be able to resolve the failure of a service to start and recover from file system inconsistency.

Before you begin

Prepare your systems for this exercise by running the command **`lab london setup`** on your **`workstation`** machine.

```
[student@workstation ~]$ lab london setup
```

After a fellow administrator decided to harden the **`httpd`** service on **`servera`**, users complain that the website running on the server is no longer available. Determine and resolve the failure of the service. Once the source of the problem is determined, be sure to validate that the same issue is not also impacting other contents of the *httpd* package. If other contents in the package are also affected, fix them accordingly.

Once the **`httpd`** service starts successfully, verify that the website content can be retrieved using **`http://servera.lab.example.com`**. If the website content cannot be retrieved, troubleshoot and resolve the issue.

1. Verify the user complaints by attempting to connect to the website from **`serverb`**.

2. On **`servera`**, troubleshoot the cause of the failed connection from **`serverb`** to the web server.

3. On **`servera`**, determine if the issue with the **`/usr/sbin/httpd`** file is also impacting other files from the same package.

4. On **`servera`**, fix the issue with the **`/usr/sbin/httpd`** file, as well as other affected files from the same package.

5. On **`servera`**, determine if the **`httpd`** service can now start successfully.

6. On **`serverb`**, determine if the **`index.html`** file can now be successfully retrieved from **`servera`**.

7. On **`servera`**, troubleshoot why the **`index.html`** is not retrievable by **`serverb`**.

8. On **`servera`**, fix the inability to access the **`index.html`** file.

9. On **`servera`**, restore access to the **`index.html`** file.

10. Verify the user complaints are now resolved by attempting to connect to the website from **serverb**.

11. Grade your work, then clean up your systems for the next exercise.

    11.1. Grade your work.

    ```
    [student@workstation ~]$ lab london grade
    ```

    11.2. Clean up your systems for the next exercise.

    ```
    [student@workstation ~]$ lab london reset
    ```

Solution

In this lab, you will troubleshoot and resolve a service failure and a corrupted file system.

Resources	
Application	`http://servera.lab.example.com`
Machines	• `servera`

Outcome(s)

You should be able to resolve the failure of a service to start and recover from file system inconsistency.

Before you begin

Prepare your systems for this exercise by running the command **`lab london setup`** on your **`workstation`** machine.

```
[student@workstation ~]$ lab london setup
```

After a fellow administrator decided to harden the **`httpd`** service on **`servera`**, users complain that the website running on the server is no longer available. Determine and resolve the failure of the service. Once the source of the problem is determined, be sure to validate that the same issue is not also impacting other contents of the *httpd* package. If other contents in the package are also affected, fix them accordingly.

Once the **`httpd`** service starts successfully, verify that the website content can be retrieved using **`http://servera.lab.example.com`**. If the website content cannot be retrieved, troubleshoot and resolve the issue.

1. Verify the user complaints by attempting to connect to the website from **`serverb`**.

   1.1. Log into **`serverb`** as the **`student`** user.

   1.2. Attempt to retrieve the **`index.html`** file from the web server running on **`servera`**.

```
[student@serverb ~]$ curl http://servera.lab.example.com
curl: (7) Failed to connect to fd37:5265:6448:6174::a: Network is unreachable
```

2. On **`servera`**, troubleshoot the cause of the failed connection from **`serverb`** to the web server.

   2.1. Verify the status of the **`httpd`** service.

```
[root@servera ~]# systemctl status httpd
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; vendor
 preset: disabled)
   Active: failed (Result: exit-code) since Fri 2016-02-26 00:29:56 EST; 9min
 ago
     Docs: man:httpd(8)
           man:apachectl(8)
  Process: 1633 ExecStop=/bin/kill -WINCH ${MAINPID} (code=exited, status=1/
FAILURE)
  Process: 1631 ExecStart=/usr/sbin/httpd $OPTIONS -DFOREGROUND (code=exited,
 status=203/EXEC)
 Main PID: 1631 (code=exited, status=203/EXEC)

Feb 26 00:29:56 servera.lab.example.com systemd[1]: Starting The Apache HTTP
 Server...
Feb 26 00:29:56 servera.lab.example.com systemd[1631]: Failed at step EXEC
 spawning /usr/sbin/httpd: Permission denied
Feb 26 00:29:56 servera.lab.example.com systemd[1]: httpd.service: main process
 exited, code=exited, status=203/EXEC
Feb 26 00:29:56 servera.lab.example.com kill[1633]: kill: cannot find process ""
Feb 26 00:29:56 servera.lab.example.com systemd[1]: httpd.service: control
 process exited, code=exited status=1
Feb 26 00:29:56 servera.lab.example.com systemd[1]: Failed to start The Apache
 HTTP Server.
Feb 26 00:29:56 servera.lab.example.com systemd[1]: Unit httpd.service entered
 failed state.
Feb 26 00:29:56 servera.lab.example.com systemd[1]: httpd.service failed.
```

2.2. Determine the cause of the failure to spawn **`/usr/sbin/httpd`**.

```
[root@servera ~]# ls -la /usr/sbin/httpd
-rw-r--r--. 1 root root 507000 Sep 17 09:07 /usr/sbin/httpd
```

The **`/usr/sbin/httpd`** file lacks execute permission.

3. On **`servera`**, determine if the issue with the **`/usr/sbin/httpd`** file is also impacting other files from the same package.

3.1. Determine the package that provided the **`/usr/sbin/httpd`** file.

```
[root@servera ~]# rpm -qf /usr/sbin/httpd
httpd-2.4.6-40.el7.x86_64
```

3.2. Determine the files from the *httpd* package that have had the permissions modified.

```
[root@servera ~]# rpm -qV httpd
.M.......    /usr/sbin/apachectl
.M.......    /usr/sbin/fcgistarter
.M.......    /usr/sbin/htcacheclean
.M.......    /usr/sbin/httpd
.M.......    /usr/sbin/rotatelogs
.M.......    /usr/sbin/suexec
```

4. On **`servera`**, fix the issue with the **`/usr/sbin/httpd`** file, as well as other affected files from the same package.

4.1. Use the **`rpm`** command to set the permissions of the files belonging to the *httpd*package back to their original settings.

```
[root@servera ~]# rpm --setperms httpd
```

4.2. Verify that the permission issues have been fixed.

```
[root@servera ~]# rpm -qV httpd
```

5. On **servera**, determine if the **httpd** service can now start successfully.

   5.1. Restart the **httpd** service.

```
[root@servera ~]# systemctl restart httpd
```

   5.2. Verify the status of the **httpd** service.

```
[root@servera ~]# systemctl status httpd
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; vendor
 preset: disabled)
   Active: active (running) since Fri 2016-02-26 00:56:44 EST; 1s ago
     Docs: man:httpd(8)
           man:apachectl(8)
  Process: 2558 ExecStop=/bin/kill -WINCH ${MAINPID} (code=exited, status=0/
SUCCESS)
 Main PID: 2563 (httpd)
   Status: "Processing requests..."
   CGroup: /system.slice/httpd.service
           ├─2563 /usr/sbin/httpd -DFOREGROUND
           ├─2564 /usr/sbin/httpd -DFOREGROUND
           ├─2565 /usr/sbin/httpd -DFOREGROUND
           ├─2566 /usr/sbin/httpd -DFOREGROUND
           ├─2567 /usr/sbin/httpd -DFOREGROUND
           └─2568 /usr/sbin/httpd -DFOREGROUND

Feb 26 00:56:44 servera.lab.example.com systemd[1]: Starting The Apache HTTP
 Server...
Feb 26 00:56:44 servera.lab.example.com systemd[1]: Started The Apache HTTP
 Server.
```

6. On **serverb**, determine if the **index.html** file can now be successfully retrieved from **servera**.

   6.1. Log into **serverb** as the **student** user.

   6.2. Attempt to retrieve the **index.html** file from the web server running on **servera**.

```
[student@serverb ~]$ curl http://servera.lab.example.com
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/
DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
        <head>
                <title>Test Page for the Apache HTTP Server on Red Hat
 Enterprise Linux</title>
... Output omitted ...
```

   Since the contents of the Apache test page are received rather than the contents of the **index.html** page, something is still not right with the web service on **servera**.

7. On **servera**, troubleshoot why the **index.html** is not retrievable by **serverb**.

   7.1. Consult the **/var/log/httpd/error_log** log file for error messages.

```
[root@servera ~]$ tail /var/log/httpd/error_log
```

```
[Fri Feb 26 00:53:52.185570 2016] [mpm_prefork:notice] [pid 2545] AH00163:
 Apache/2.4.6 (Red Hat Enterprise Linux) configured -- re
suming normal operations
[Fri Feb 26 00:53:52.185583 2016] [core:notice] [pid 2545] AH00094: Command
 line: '/usr/sbin/httpd -D FOREGROUND'
[Fri Feb 26 00:56:43.135815 2016] [mpm_prefork:notice] [pid 2545] AH00170:
 caught SIGWINCH, shutting down gracefully
[Fri Feb 26 00:56:44.172637 2016] [core:notice] [pid 2563] SELinux policy
 enabled; httpd running as context system_u:system_r:httpd
_t:s0
[Fri Feb 26 00:56:44.173107 2016] [suexec:notice] [pid 2563] AH01232: suEXEC
 mechanism enabled (wrapper: /usr/sbin/suexec)
[Fri Feb 26 00:56:44.178433 2016] [auth_digest:notice] [pid 2563] AH01757:
 generating secret for digest authentication ...
[Fri Feb 26 00:56:44.178855 2016] [lbmethod_heartbeat:notice] [pid 2563]
 AH02282: No slotmem from mod_heartmonitor
[Fri Feb 26 00:56:44.180023 2016] [mpm_prefork:notice] [pid 2563] AH00163:
 Apache/2.4.6 (Red Hat Enterprise Linux) configured -- re
suming normal operations
[Fri Feb 26 00:56:44.180032 2016] [core:notice] [pid 2563] AH00094: Command
 line: '/usr/sbin/httpd -D FOREGROUND'
[Fri Feb 26 00:57:36.538719 2016] [core:error] [pid 2564] (22)Invalid argument:
 [client 172.25.250.11:33294] AH00036: access to /in
dex.html failed (filesystem path '/var/www/html/index.html')
```

It appears there was an issue accessing the **/var/www/html/index.html** file.

7.2. Attempt to list the contents of the **/var/www/html** directory.

```
[root@servera ~]$ ls -la /var/www/html
ls: cannot access /var/www/html/index.html: Invalid argument
total 12
drwxr-xr-x. 3 root root 96 Feb 24 23:55 .
drwxr-xr-x. 4 root root 31 Feb 26 00:29 ..
-rw-r--r--. 1 root root 15 Feb 24 23:52 about.html
??????????? ? ?    ?     ?            ? index.html
drwxr-xr-x. 2 root root  6 Feb 24 23:55 lost+found
-rw-r--r--. 1 root root 18 Feb 24 23:53 products.html
-rw-r--r--. 1 root root 17 Feb 24 23:52 support.html
```

8. On **servera**, fix the inability to access the **index.html** file.

8.1. Determine which file system is used for **/var/www/html**.

```
[root@servera ~]$ df -T /var/www/html
Filesystem     Type 1K-blocks  Used Available Use% Mounted on
/dev/vdb1      XFS      98988  5296     93692   6% /var/www/html
```

8.2. Umount the XFS file system mounted at **/var/www/html**.

```
[root@servera ~]$ umount /var/www/html
```

8.3. Run a file system check on the file system residing on the **/dev/vdb1** partition.

```
[root@servera ~]$ xfs_repair -n /dev/vdb1
Phase 1 - find and verify superblock...
Phase 2 - using internal log
```

```
        - scan filesystem freespace and inode maps...
... Output omitted ...
entry "index.html" in shortform directory 128 references invalid inode 123456789
would have junked entry "index.html" in directory inode 128
... Output omitted ...
        - moving disconnected inodes to lost+found ...
disconnected inode 137, would move to lost+found
Phase 7 - verify link counts...
No modify flag set, skipping filesystem flush and exiting.
```

8.4. Repair the XFS file system residing on the **/dev/vdb1** partition.

```
[root@servera ~]$ xfs_repair /dev/vdb1
Phase 1 - find and verify superblock...
Phase 2 - using internal log
        - zero log...
... Output omitted ...
entry "index.html" in shortform directory 128 references invalid inode 123456789
junking entry "index.html" in directory inode 128
... Output omitted ...
disconnected inode 137, moving to lost+found
Phase 7 - verify and correct link counts...
done
```

9. On **servera**, restore access to the **index.html** file.

9.1. Mount the XFS file system contained in the **/dev/vdb1** partition at **/var/www/html**.

```
[root@servera ~]$ mount /dev/vdb1 /var/www/html
```

9.2. Attempt to list the contents of the **/var/www/html** directory.

```
[root@servera ~]$ ls -la /var/www/html
total 12
drwxr-xr-x. 3 root root 79 Feb 24 23:55 .
drwxr-xr-x. 4 root root 31 Feb 26 00:29 ..
-rw-r--r--. 1 root root 15 Feb 24 23:52 about.html
drwxr-xr-x. 2 root root 16 Feb 24 23:55 lost+found
-rw-r--r--. 1 root root 18 Feb 24 23:53 products.html
-rw-r--r--. 1 root root 17 Feb 24 23:52 support.html
```

9.3. Look for the missing **/var/www/html/index.html** file.

```
[root@servera ~]$ ls -la /var/www/html/lost+found
total 4
drwxr-xr-x. 2 root root 16 Feb 24 23:55 .
drwxr-xr-x. 3 root root 79 Feb 24 23:55 ..
-rw-r--r--. 1 root root 14 Feb 24 23:55 137
```

9.4. Restore the orphaned file **index.html** to the **/var/www/html** location.

```
[root@servera ~]$ mv /var/www/html/lost+found/137 /var/www/html/index.html
```

10. Verify the user complaints are now resolved by attempting to connect to the website from **serverb**.

10.1. Log into **serverb** as the **student** user.

10.2. Attempt to retrieve the **index.html** file from the web server running on **servera**.

```
[student@serverb ~]$ curl http://servera.lab.example.com
<h1>Home</h1>
```

11. Grade your work, then clean up your systems for the next exercise.

11.1. Grade your work.

```
[student@workstation ~]$ lab london grade
```

11.2. Clean up your systems for the next exercise.

```
[student@workstation ~]$ lab london reset
```

Lab: New York — Network Delays

In this lab, you will solve a problem involving weird network pauses.

Resources	
Machines	• **workstation** • **servera** • **serverb**

Outcome(s)
You should be able to resolve a network issues involving seemingly random delays.

Before you begin
Prepare your systems for this exercise by running the command **lab newyork setup** on your **workstation** machine.

```
[student@workstation ~]$ lab newyork setup
```

Users are reporting random pauses when connecting to network services on your **serverb** system. Until recently, this system was under the control of another admin, who has since left the company. This former admin was known for distrusting any new technology, and, contrary to company policy, never kept any notes or work logs.

Both **httpd** and **sshd** services are available for testing. A similar machine, **servera**, is unaffected by this issue, but that server is not running **httpd** services.

Investigate, and fix, this issue.

1. Recreate the issue.
2. Gather information.
3. Form a hypothesis.
4. Test your hypothesis.
5. Implement a fix.
6. Grade your work by running the command **lab newyork grade** from your **workstation** machine.

   ```
   [student@workstation ~]$ lab newyork grade
   ```

7. Clean up your machines by running the command **lab newyork reset** from your **workstation** machine. If that does not work, reset your machines.

   ```
   [student@workstation ~]$ lab newyork reset
   ```

Solution

In this lab, you will solve a problem involving weird network pauses.

Resources	
Machines	• **workstation** • **servera** • **serverb**

Outcome(s)
You should be able to resolve a network issues involving seemingly random delays.

Before you begin
Prepare your systems for this exercise by running the command **lab newyork setup** on your **workstation** machine.

```
[student@workstation ~]$ lab newyork setup
```

Users are reporting random pauses when connecting to network services on your **serverb** system. Until recently, this system was under the control of another admin, who has since left the company. This former admin was known for distrusting any new technology, and, contrary to company policy, never kept any notes or work logs.

Both **httpd** and **sshd** services are available for testing. A similar machine, **servera**, is unaffected by this issue, but that server is not running **httpd** services.

Investigate, and fix, this issue.

1. Recreate the issue.

   1.1. Use **ssh** to connect to **serverb**. What do you notice?

   ```
   [student@workstation ~]$ ssh serverb
   ```

   There is a delay of a couple of seconds before the connections goes through.

   1.2. Time this delay.

   ```
   [student@workstation ~]$ time ssh serverb echo

   real    0m3.190s
   user    0m0.015s
   sys     0m0.004s
   ```

   The delay seems to be about three seconds.

   1.3. Time loading a web page from **serverb** using **curl**.

   ```
   [student@workstation ~]$ time curl http://serverb.lab.example.com
   ```

   The delay is again around three seconds.

2. Gather information.

    2.1. Run the **curl** command under **strace** to see if it blocks somewhere.

```
[student@workstation ~]$ strace curl http://serverb.lab.example.com
```

    The delays seem to come from some **poll** system calls, on a socket created with the following system call:

```
connect(3, {sa_family=AF_INET6, sin6_port=htons(80), inet_pton(AF_INET6,
 "fd37:5265:6448:6174::b", &sin6_addr), sin6_flowinfo=0, sin6_scope_id=0}, 28) =
 -1 EINPROGRESS (Operation now in progress)
```

    2.2. Take another look at the **strace** output. Is the socket ever opened successfully?

    No, the **connect** fails, and a regular IPv4 socket is opened successfully afterwards.

3. Form a hypothesis.

    3.1. If **serverb** did not have a working IPv6 address, or a working DNS entry for that address, tools like **ssh** and **curl** would never try to connect there.

    3.2. Logically, it follows that although IPv6 is configured correctly for **serverb**, something is blocking access to it.

4. Test your hypothesis.

    4.1. Test by forcing both **ssh** and **curl** to connect using IPv4 only. Do you still see the delay?

```
[student@workstation ~]$ time ssh -4 serverb
```

```
[student@workstation ~]$ time curl -4 http://serverb.lab.example.com
```

    The delay is now completely gone.

5. Implement a fix.

    5.1. Check for any weird firewall rules on **serverb**.

```
[root@serverb ~]# firewall-cmd --list-all
public (default, active)
  interfaces: eth0 eth1
....
  rich rules:
        rule family="ipv6" source address="0::0/0" reject
```

    5.2. Remove the offending rule both from the in-memory firewall configuration and from the permanent configuration.

```
[root@serverb ~]# firewall-cmd --remove-rich-rule='rule family="ipv6" source
 address="0::0/0" reject'
```

```
[root@serverb ~]# firewall-cmd --permanent --remove-rich-rule='rule
 family="ipv6" source address="0::0/0" reject'
```

6. Grade your work by running the command **lab newyork grade** from your **workstation** machine.

   ```
   [student@workstation ~]$ lab newyork grade
   ```

7. Clean up your machines by running the command **lab newyork reset** from your **workstation** machine. If that does not work, reset your machines.

   ```
   [student@workstation ~]$ lab newyork reset
   ```