





Join a community dedicated to learning open source

The Red Hat® Learning Community is a collaborative platform for users to accelerate open source skill adoption while working with Red Hat products and experts.



Network with tens of thousands of community members



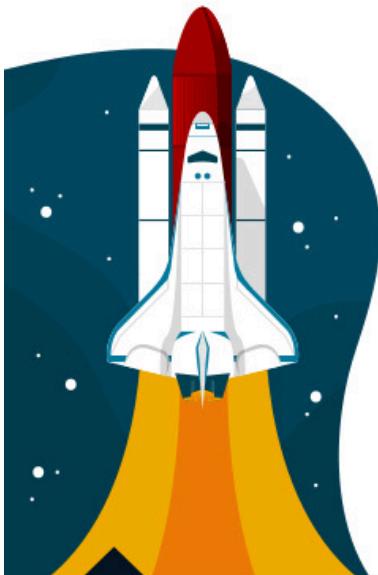
Engage in thousands of active conversations and posts



Join and interact with hundreds of certified training instructors



Unlock badges as you participate and accomplish new goals



This knowledge-sharing platform creates a space where learners can connect, ask questions, and collaborate with other open source practitioners.

Access free Red Hat training videos

Discover the latest Red Hat Training and Certification news

Connect with your instructor - and your classmates - before, after, and during your training course.

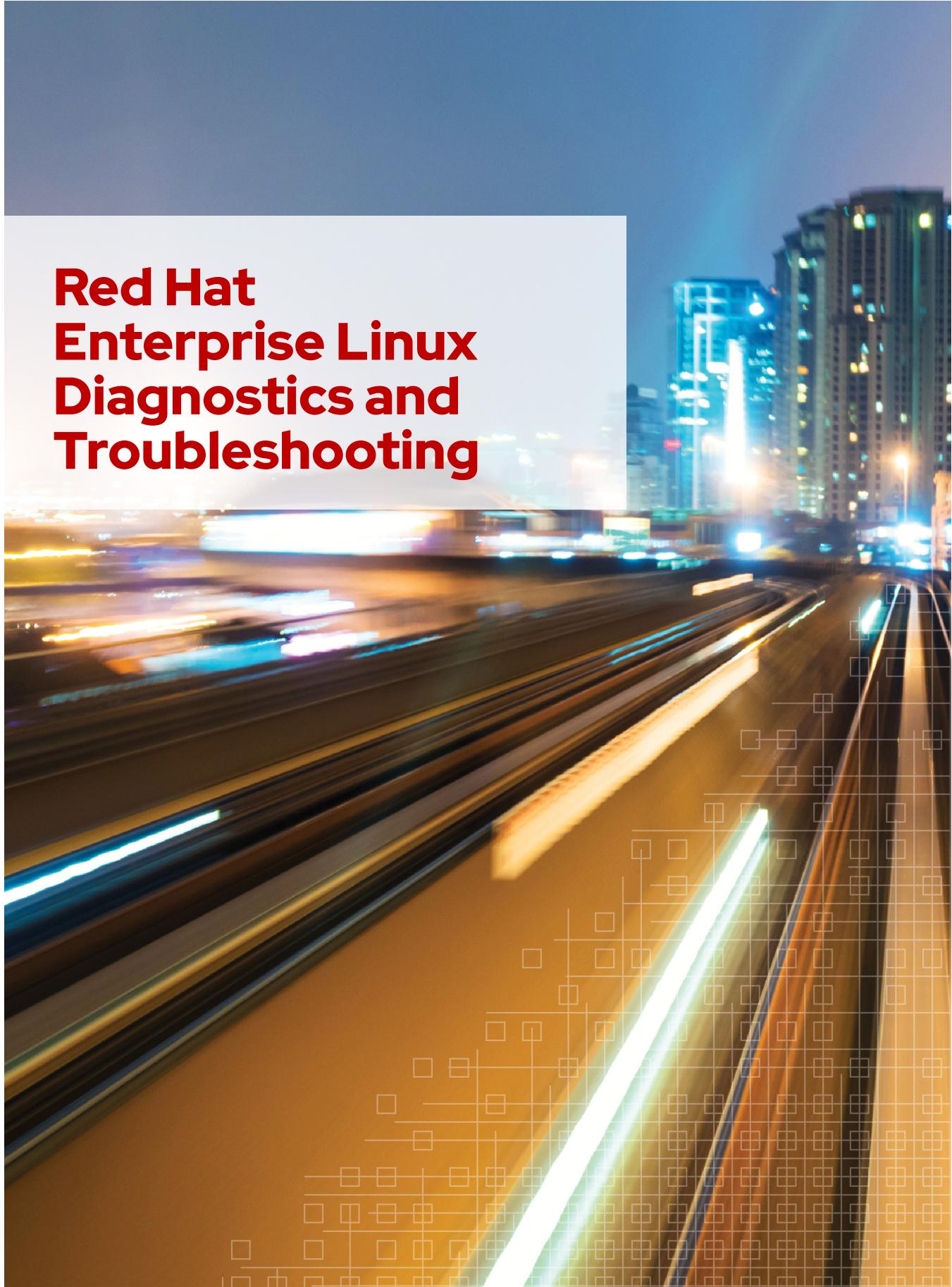
Join peers as you explore Red Hat products

Join the conversation learn.redhat.com



Copyright © 2020 Red Hat, Inc. Red Hat, Red Hat Enterprise Linux, the Red Hat logo, and Ansible are trademarks or registered trademarks of Red Hat, Inc. or its subsidiaries in the United States and other countries.

Red Hat Enterprise Linux Diagnostics and Troubleshooting



Red Hat Enterprise Linux 8.4 RH342
Red Hat Enterprise Linux Diagnostics and Troubleshooting
Edition 1 20211202
Publication date 20211202

Authors: Alex Callejas Garcia, Ashish Lingayat, Jacob Pelchat, Victor Costea
Course Architect: Philip Sweany
DevOps Engineer: Artur Glogowski
Editor: Julian Cable

Copyright © 2021 Red Hat, Inc.

The contents of this course and all its modules and related materials, including handouts to audience members, are
Copyright © 2021 Red Hat, Inc.

No part of this publication may be stored in a retrieval system, transmitted or reproduced in any way, including, but not limited to, photocopy, photograph, magnetic, electronic or other record, without the prior written permission of Red Hat, Inc.

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 send email to training@redhat.com or phone toll-free (USA) +1 (866) 626-2994 or +1 (919) 754-3700.

Red Hat, Red Hat Enterprise Linux, the Red Hat logo, JBoss, OpenShift, Fedora, Hibernate, Ansible, CloudForms, RHCA, RHCE, RHCSA, Ceph, and Gluster are trademarks or registered trademarks of Red Hat, Inc. or its subsidiaries 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.

MySQL® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js® is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

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: Adarsh Krishnan, David Sacco, Sajith Sugathan, Samik Sanyal

Document Conventions	xi
	xi
Introduction	xiii
Red Hat Enterprise Linux Diagnostics and Troubleshooting	xiii
Orientation to the Classroom Environment	xiv
Performing Lab Exercises	xviii
1. Introducing Troubleshooting Strategy	1
Using the Scientific Method	2
Guided Exercise: Using the Scientific Method to Solve a Login Issue	6
Collecting Information to Support Troubleshooting	10
Guided Exercise: Collecting Information to Support Troubleshooting	15
Troubleshooting with Red Hat Resources	19
Guided Exercise: Troubleshooting with Red Hat Resources	29
Lab: Introducing Troubleshooting Strategy	32
Summary	38
2. Configuring Baseline Data	39
Monitoring Systems	40
Guided Exercise: Monitoring Systems	46
Configuring Remote Logging	51
Guided Exercise: Configuring Remote Logging	56
Describing Configuration Management Automation	59
Guided Exercise: Implementing Configuration Changes with an Ansible Playbook	65
Configuring Change Tracking	70
Guided Exercise: Configuring Change Tracking	77
Lab: Configuring Baseline Data	84
Summary	97
3. Troubleshooting Boot Issues	99
Resolving Boot Loader Issues on BIOS Systems	100
Guided Exercise: Resolving Boot Loader Issues on BIOS Systems	106
Resolving Boot Loader Issues on UEFI Systems	109
Quiz: Resolving Boot Loader Issues on UEFI Systems	115
Identifying and Resolving Failing Services	117
Guided Exercise: Identifying and Resolving Failing Services	120
Resetting the root Password	124
Guided Exercise: Resetting the root Password	126
Lab: Troubleshooting Boot Issues	128
Summary	131
4. Identifying Hardware Issues	133
Identifying Hardware Issues	134
Guided Exercise: Identifying Hardware Issues	140
Managing Kernel Modules	144
Guided Exercise: Managing Kernel Modules	147
Resolving Virtualization Issues	150
Guided Exercise: Resolving Virtualization Issues	155
Lab: Identifying Hardware Issues	158
Summary	162
5. Troubleshooting Storage Issues	163
Describing the Linux Storage Stack	164
Guided Exercise: Configuring Storage with Stratis	171
Recovering from File System Corruption	175
Guided Exercise: Recovering from File System Corruption	181
Repairing LVM Issues	186

Guided Exercise: Repairing LVM Issues	189
Resolving Storage Device Encryption Issues	193
Guided Exercise: Resolving Storage Device Encryption Issues	197
Resolving iSCSI Issues	201
Guided Exercise: Resolving iSCSI Issues	207
Lab: Troubleshooting Storage Issues	211
Summary	221
6. Troubleshooting RPM Issues	223
Resolving Package Dependency Issues	224
Guided Exercise: Resolving Package Dependency Issues	228
Recovering a Corrupted RPM Database	233
Guided Exercise: Recovering a Corrupted RPM Database	236
Identifying and Recovering RPM Managed Files	240
Guided Exercise: Identifying and Recovering RPM Managed Files	243
Managing Red Hat Subscriptions	246
Quiz: Managing Red Hat Subscriptions	254
Lab: Troubleshooting RPM Issues	256
Summary	261
7. Troubleshooting Network Issues	263
Verifying Network Connectivity	264
Guided Exercise: Verifying Network Connectivity	273
Resolving Connectivity Issues	279
Guided Exercise: Resolving Connectivity Issues	287
Inspecting Network Traffic	293
Guided Exercise: Inspecting Network Traffic	297
Lab: Troubleshooting Network Issues	304
Summary	312
8. Troubleshooting Application Issues	313
Resolving Library Dependencies	314
Guided Exercise: Resolving Library Dependencies	318
Debugging Memory Leaks	322
Guided Exercise: Debugging Memory Leaks	326
Debugging Application Execution	330
Guided Exercise: Debugging Application Execution	335
Troubleshooting Containerized Applications	339
Guided Exercise: Troubleshooting Containerized Applications	347
Lab: Troubleshooting Application Issues	352
Summary	359
9. Troubleshooting Security Issues	361
Repairing SELinux Issues	362
Guided Exercise: Repairing SELinux Issues	368
Identifying Authentication Issues	376
Guided Exercise: Identifying Authentication Issues	379
Resolving Identity Management Issues	382
Guided Exercise: Resolving Identity Management Issues	389
Lab: Troubleshooting Security Issues	394
Summary	399
10. Troubleshooting Kernel Issues	401
Configuring Kernel Crash Dumps	402
Guided Exercise: Configuring Kernel Crash Dumps	411
Kernel Debugging with SystemTap	415
Guided Exercise: Kernel Debugging with SystemTap	422

Lab: Troubleshooting Kernel Issues	427
Summary	435
11. Comprehensive Review	437
Comprehensive Review	438
Lab: Repairing a Non-running Application	441
Lab: Resolving a Console Login Issue	446
Lab: Resolving Authentication Issues	451
Lab: Repairing a Web Server Issue	456
Lab: Resolving Network Delay Issues	462
Lab: Resolving Container Issues	467

Document Conventions

This section describes various conventions and practices used throughout all Red Hat Training courses.

Admonitions

Red Hat Training courses use the following admonitions:



References

These describe where to find external documentation relevant to a subject.



Note

These 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 something that makes your life easier.



Important

These provide details of information that is easily missed: configuration changes that only apply to the current session, or services that need restarting before an update will apply. Ignoring these admonitions will not cause data loss, but may cause irritation and frustration.



Warning

These should not be ignored. Ignoring these admonitions will most likely cause data loss.

Inclusive Language

Red Hat Training is currently reviewing its use of language in various areas to help remove any potentially offensive terms. This is an ongoing process and requires alignment with the products and services covered in Red Hat Training courses. Red Hat appreciates your patience during this process.

Introduction

Red Hat Enterprise Linux Diagnostics and Troubleshooting

Red Hat Enterprise Linux Diagnostics and Troubleshooting course (RH342) provides system administrators with the needed tools and techniques to diagnose successfully and fix various potential issues. Students work through hands-on problems in various subsystems to diagnose and fix common issues.

Course Objectives

- Diagnose problems in various subsystems on Red Hat Enterprise Linux 8 systems, with tools from the distribution.
- Gather information to assist Red Hat Support in diagnosing and fixing possible issues on a Red Hat Enterprise Linux system.
- Fix common issues on a Red Hat Enterprise Linux machine, with tools from the distribution.
- Prepare to attend a Red Hat Enterprise Linux Diagnostics and Troubleshooting Certification exam.

Audience

The Red Hat Enterprise Linux Diagnostics and Troubleshooting course is designed for senior system administrators who want to learn more about troubleshooting.

Prerequisites

- A Red Hat Certified System Administrator (RHCSA) certification, or equivalent knowledge, is required to succeed in this course.
- A Red Hat Certified Engineer (RHCE) certification, or equivalent knowledge, is recommended to increase topic retention in this course.

Orientation to the Classroom Environment

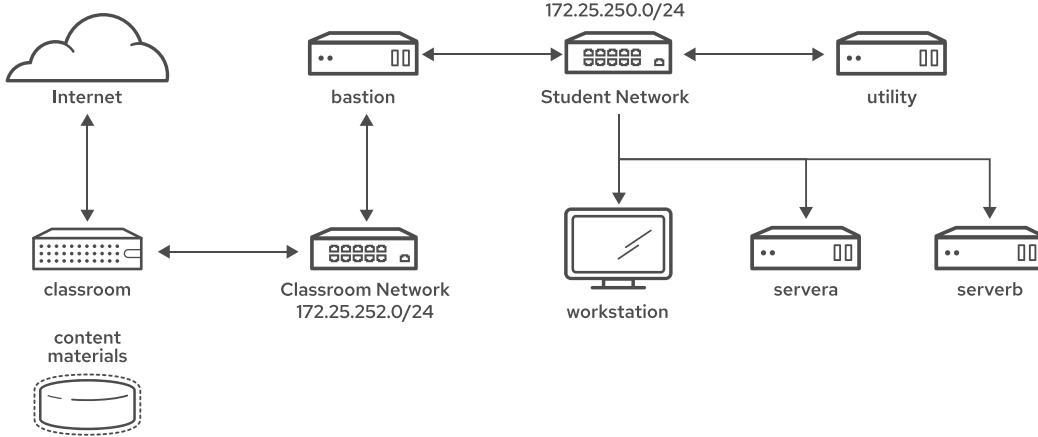


Figure 0.1: Classroom environment

In this classroom environment, your primary system for hands-on activities is **workstation**. The **workstation** virtual machine (VM) is the only system with a graphical desktop, which is required for using a browser to access web-based tools. Always log in directly to **workstation** first. From **workstation**, use **ssh** for command-line access to all other VMs.

Additional student VMs for hands-on exercises include **servera** and **serverb**. All of these systems are in the `lab.example.com` DNS domain.

The **utility** system hosts an IdM server with a .net realm.

All student computer systems have a standard **student** user account with a **student** password. The root password on all student systems is **redhat**.

As seen in *Figure 0.1*, all VMs share an external network, 172.25.250.0/24, with a gateway of 172.25.250.254 (bastion).

Classroom Machines

Machine name	IP addresses	Role
<code>bastion.lab.example.com</code>	172.25.250.254	Router to link VMs to central servers
<code>workstation.lab.example.com</code>	172.25.250.9	Graphical workstation for system administration
<code>servera.lab.example.com</code>	172.25.250.10	Managed server "A"
<code>serverb.lab.example.com</code>	172.25.250.11	Managed server "B"

Machine name	IP addresses	Role
classroom.lab.example.com	172.25.254.254	The classroom materials and content
utility.lab.example.com	172.25.250.17	Support services such as DNS and container registry

The primary function of **bastion** is to act as a router between the student systems network and the classroom network. If **bastion** is down, then student machines can access only systems on the individual student network.

Several systems in the classroom provide supporting services. Two server URLs, **content.example.com** and **materials.example.com**, provide software and lab materials for the hands-on activities, as described in the activity's instructions. These URLs are aliases of the **classroom.example.com** server. Both the **classroom** and **bastion** systems must remain running for the course environment to work properly.

Controlling Your Systems

You are assigned remote computers in a Red Hat Online Learning classroom. Self-paced courses are accessed through a web application that is hosted at **rol.redhat.com** [<http://rol.redhat.com>]. If your course is an instructor-led virtual training event, you will be provided with your course location URL. Log in to this site with your Red Hat Customer Portal user credentials.

Controlling the Virtual Machines

The virtual machines in your classroom environment are controlled through web page interface controls. The state of each classroom virtual machine is displayed on the **Lab Environment** tab.

Figure 0.2: An example course Lab Environment management page

Machine States

Virtual Machine State	Description
building	The virtual machine is being created.
active	The virtual machine is running and available. If just started, it might still be starting services.
stopped	The virtual machine is completely shut down. On starting, the virtual machine boots into the same state as it was before it was shut down. The disk state is preserved.

Classroom Actions

Button or Action	Description
CREATE	Create the ROLE classroom. Creates and starts all of the virtual machines that are needed for this classroom.

Button or Action	Description
CREATING	The ROLE classroom virtual machines are being created. Creates and starts all of the virtual machines that are needed for this classroom. Creation can take several minutes to complete.
DELETE	Delete the ROLE classroom. Destroys all virtual machines in the classroom. All saved work on that system's disks is lost.
START	Start all virtual machines in the classroom.
STARTING	All virtual machines in the classroom are starting.
STOP	Stop all virtual machines in the classroom.

Machine Actions

Button or Action	Description
OPEN CONSOLE	Connect to the system console of the virtual machine in a new browser tab. You can log in directly to the virtual machine and run commands, when required. Normally, log in to the workstation virtual machine only, and from there, use ssh to connect to the other virtual machines.
ACTION → Start	Start (power on) the virtual machine.
ACTION → Shutdown	Gracefully shut down the virtual machine, preserving disk contents.
ACTION → Power Off	Forcefully shut down the virtual machine, while still preserving disk contents. This is equivalent to removing the power from a physical machine.
ACTION → Reset	Forcefully shut down the virtual machine and reset the disk to its initial state. All saved work on that system's disks is lost.

At the start of an exercise, if instructed to reset a single virtual machine node, click ACTION → Reset for only the specific virtual machine.

At the start of an exercise, if instructed to reset all virtual machines, click ACTION → Reset on every virtual machine in the list.

If you want to return the classroom environment to its original state at the start of the course, you can click DELETE to remove the entire classroom environment. After the lab is deleted, you can click CREATE to provision a new set of classroom systems.



Warning

The DELETE operation cannot be undone. All completed work in the classroom environment is lost.

The Auto-stop and Auto-destroy Timers

The Red Hat Online Learning enrollment entitles you to a set allotment of computer time. To help to conserve your allotted time, the ROLE classroom uses timers, which shut down or delete the classroom when the appropriate timer expires.

To adjust the timers, locate the two + buttons at the bottom of the course management page. Click the auto-stop + button to add another hour to the auto-stop timer. Click the auto-destroy + button to add another day to the auto-destroy timer. Auto-stop has a maximum of 11 hours, and auto-destroy has a maximum of 14 days. Be careful to keep the timers set while you are working, so that your environment is not unexpectedly shut down. Be careful not to set the timers unnecessarily high, which could waste your subscription time allotment.

Performing Lab Exercises

Run the `lab` command from `workstation` to prepare your environment before each hands-on exercise, and again to clean up after an exercise. Each hands-on exercise has a unique name within a course.

There are two types of exercises. The first type, a *guided exercise*, is a practice exercise that follows a course narrative. If a narrative is followed by a quiz, it usually indicates that the topic did not have an achievable practice exercise. The second type, an *end-of-chapter lab*, is a gradable exercise to help to verify your learning. When a course includes a comprehensive review, the review exercises are structured as gradable labs.

The syntax for running an exercise script is as follows:

```
[student@workstation ~]$ lab action exercise
```

The `action` is a choice of `start`, `grade`, or `finish`. All exercises support `start` and `finish`. Only end-of-chapter labs and comprehensive review labs support `grade`.

start

A script's start logic verifies the required resources to begin an exercise. It might include configuring settings, creating resources, checking prerequisite services, and verifying necessary outcomes from previous exercises. With exercise start logic, you can perform any exercise at any time, even if you did not perform prerequisite exercises.

grade

End-of-chapter labs help to verify what you learned, after practicing with earlier guided exercises. The grade action directs the `lab` command to display a list of grading criteria, with a `PASS` or `FAIL` status for each. To achieve a `PASS` status for all criteria, fix the failures and rerun the grade action.

finish

A script's finish logic deletes exercise resources that are no longer necessary. With cleanup logic, you can repeatedly perform an exercise, and it helps course performance by ensuring that unneeded objects release their resources.

To list the available exercises, use tab completion in the `lab` command with an action:

```
[student@workstation ~]$ lab start Tab Tab
api-review           cluster-admin      comprehensive-review1
api-s3               cluster-maint     comprehensive-review2
api-swift            cluster-review    comprehensive-review3
block-devices        component-auth   comprehensive-review4
block-import          component-osd    comprehensive-review5
block-review          component-pool   configure-monitor
block-snapshot        component-review  configure-network
...output omitted...
```

Troubleshooting Lab Scripts

The `lab` command displays a list of action steps or grading criteria while it runs, with a PASS or FAIL status for each. If the status is FAIL, the script will display additional information captured from the step's command output. The output displayed, and how useful it might be for troubleshooting, will vary depending on what the step was doing.

```
[student@workstation ~]$ lab grade comprehensive-review

Grading lab.

· Checking lab systems ..... SUCCESS
· Verify that cluster is running on the hosts ..... SUCCESS
· Verify that cluster health is ok ..... SUCCESS
· Verify that number of OSDs is 11 ..... FAIL
  - '11 osds' not found in command output at 'clienta'
· Verify OSD public_network ..... FAIL
  - '172.25.250.0/24' not found in command output at 'clienta'
· Verify OSD cluster_network ..... SUCCESS

Overall lab grade: FAIL
```

All exercise scripts are stored on the `workstation` system in the folder `/home/student/.venv/labs/lib/python3.6/site-packages/SKU/`, where the SKU is the course code. When you run the `lab` command with a valid action and exercise, it creates an exercise log file in `/tmp/log/labs`, and captures command output and error messages into the file.

```
[student@workstation ~]$ lab start cluster-review
...output omitted...
[student@workstation ~]$ ls -l /tmp/log/labs/
-rw-rw-r-- 1 student student 7520 Nov 02 05:34 cluster-review
```

Although exercise scripts are always run from `workstation`, they perform tasks on other systems in the course environment. Many course environments, including OpenStack and OpenShift, use a command-line interface (CLI) that is invoked from `workstation` to communicate with server systems and components by using REST API calls. Because script actions typically distribute tasks to multiple systems, additional troubleshooting is necessary to determine where a failed task occurred. Log in to those other systems and use Linux diagnostic skills to read local system log files and determine the root cause of the lab script failure.

Updating the Lab Files

Performing maintenance on the `lab` command and script files should only be done on the advice of your instructor or Red Hat support personnel. Incorrect use of the `lab` command or versioning can make your course environment unable to run `lab` commands or for you to perform exercises. List all available `lab` actions by using the `--help` option.

```
[student@workstation ~]$ lab upgrade SKU
...output omitted...
[student@workstation ~]$ lab --help
...output omitted...
```


Chapter 1

Introducing Troubleshooting Strategy

Goal

Describe effective troubleshooting methods and data collection strategies.

Objectives

- Use a systematic approach to troubleshooting with the scientific method.
- Collect system information to support troubleshooting.
- Use Red Hat resources to support troubleshooting.

Sections

- Using the Scientific Method (and Guided Exercise)
- Collecting Information to Support Troubleshooting (and Guided Exercise)
- Troubleshooting with Red Hat Resources (and Guided Exercise)

Lab

Introducing Troubleshooting Strategy

Using the Scientific Method

Objectives

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

Defining Troubleshooting as a Scientific Method

Efficient and timely troubleshooting skills can be developed through practice of the widely recognized *scientific method*. The scientific method is an empirical process for using logic to hypothesize and test theories through observation, refined by experimentation and validation of deductions that are drawn from those hypotheses. With experience, scientific method users acquire knowledge and develop useful conclusions through refinement and elimination of tested hypotheses.

Many technical professionals solve problems by using past experience, having previously seen the same problem, or having been taught how to solve similar problems. Consequently, it is efficient to query colleagues and other knowledge sources about the scenario, after the problem is accurately defined. However, when a problem is difficult to define and is not recognized through experience, making unscientific guesses about the problem cause can waste significant time and effort.

The scientific method is a provable technique for resolving and fixing new and complex problems, but it might not be the quickest resolution for all scenarios. To solve technical problems, you must discover and gather information, discarding what does not fit observations, and produce logical conclusions to uncover root causes. The scientific method consists of these steps:

- Collect relevant information.
- Create an accurate problem statement.
- Formulate testable hypotheses.
- Test each hypothesis.
- Record and analyze the test results.
- Fix and verify the problem resolution.

Using the Scientific Method

In many scenarios, you might repeat the scientific method, either in whole or by iterating on certain steps, to discover and verify the root cause of the problem.

1. Collect relevant information.

The first step, before theorizing a problem statement, is to collect reliable and factual information. Common reasons for failed troubleshooting include incomplete information or misunderstood problem observations. Start by asking questions of the person who reported the problem and other relevant users or support personnel. Focus on, and **record in a readable form**, the verifiable facts that are related to the problem. Avoid opinions and judgments, but allow for suggestions from others who have proven to be successful with similar troubleshooting.

Useful information can also be found in screen outputs, system and application log files, error messages, and diagnostic tools. Diagnostic or error messages can be entered in Internet search engines to locate reports or resolutions for similar problems.

When applications or systems are known to have previously worked properly, then logic dictates that something must have changed. Use file, application, or system validation or comparison tools to locate changed files or to compare an errant file or system to a *known good* file or system that is expected to be configured the same.

When you have a clear perception of the problem, try to reproduce the error or failure. Use verbose logging or tool diagnostic modes to provide additional information about the errant process or observed behavior.

2. Create an accurate problem statement.

The process of creating a problem statement results in a specific definition of the problem in words, preferably written. Creating the statement as a grammatically correct and understood sentence contributes to accurately clarifying the problem. If you are unable to state the problem in a clear sentence that others agree defines that specific problem, then your problem statement needs work. An accurate problem statement explains the problem to be solved, such that a successful problem resolution is the inverse of that statement.

A problem statement includes answers to factual queries about the problem:

- What specific system, application, process, or function, is failing, degraded, or down?
- What actions or steps can reproduce the problem?
- When was the problem first noticed or reported?
- Where does the problem occur or where is the behavior observed?
- Who experiences the problem? Not who reported it, but what is the scope of its effect?

If the problem is reproducible, the problem statement should include the steps that cause the problem to occur. Here are examples of well-defined problem statements:

• **Problem 1**

Beginning last Friday, all marketing department users are reporting that they are unable to successfully launch or use the mail application, which displays the error message "Data store XYZ is not available." The problem can be consistently reproduced by selecting the mail icon from any marketing department user's menu.

• **Problem 2**

Today, userX reported that they are unable to print from application ABC to printer123, but can print to printer123 from any other application on the same system.

When the problem is resolved and fixed, the result is the inverse of the original problem statement. For example, the inverse of the previous problem statements would be:

• **Result for problem 1**

Currently, all marketing department users are able to successfully launch and use the mail application, without any displayed error messages."

• **Result for problem 2**

Currently, userX is able to print from application ABC to printer123, and can also print to printer123 from any other application on the same system.

3. Formulate testable hypotheses.

By using the problem statement that you created and the information that you collected and recorded, formulate one or more hypotheses as to the cause of the problem. This step is significantly more productive when performed in a brainstorming group that is comprised of individuals who are capable of effectively using this scientific method.

Do not rush this step. Although any single hypothesis might seem promising, it is more efficient to formulate, at the same time, all possible, practical hypotheses about the cause of the problem. No relevant, sincere suggestion should be dismissed without being tested.

When formulating and recording each hypothesis, also record a validation test method for each. Although it might appear faster to jump to performing each test as each hypothesis occurs, it is more productive to stay in brainstorming mode until all ideas are exhausted and each hypothesis and test method is recorded in an organized, readable form. Here are examples of hypotheses and test methods for the mail application problem:

- Data store XYZ is on a disk that failed. Test by locating data store XYZ and accessing other objects on that disk.
- Data store XYZ is on a network share that no longer exists or works properly. Test by locating the share and accessing the share directly by using a proper client.
- Data store XYZ is on a storage server that has stopped services or has frozen. Test by locating the correct server and accessing it with management tools.
- Data store XYZ is on a storage server that cannot be reached due to a network problem. Test by locating the correct network and accessing the interfaces on that network by using network tools.

4. Test each hypothesis.

Perform each of the tests that you recorded for your hypothesis. Prioritize the tests in the order that you or your group decide is the most likely to quickly find the problem's root cause. Performing each test should result in either discovering the problem's cause or in eliminating that hypothesis from your list.

If a test requires configuration changes or another form of system modification, follow this single, inviolate rule:

Only one change may be made during any single test run.

Never change more than one parameter at a time. If the test fails to verify the problem cause, reset that changed parameter to its original value, and then change only one new parameter before performing the next test run.

Record information about each test run, including the changed parameters, distinguishing test characteristics, and the observed result, in an organized, readable form. Failure to record each ongoing test result in an organized, readable form commonly creates an inability to distinguish or recall previous test results, which might cause you to need to repeat one or more tests from your hypotheses list.

5. Record and analyze the test results.

During testing, you record the results of each test run, including any observed behaviors and relevant test characteristics. You should also record any new information that you collect that appears to relate to the problem. This information is useful for creating system reliability methods for permanently mitigating this problem scenario.

If the problem cause is not discovered after performing all the hypothesis tests in your list, you could decide to repeat this scientific method, including any newly collected information in the brainstorming process.

You could also decide that you or your group has insufficient knowledge of the application or systems that you are troubleshooting. Accordingly, you could choose to obtain further training or perform additional research before continuing to troubleshoot this problem. If fixing this problem is time-critical, you might need to escalate the problem to an appropriate higher level of support.

6. Fix and verify the problem resolution.

If you discover the problem cause while performing a hypothesis test, you must still decide how to fix the problem. Some problems might require a temporary fix or workaround, with a permanent fix that is applied after preparation or during a maintenance window. Similar to the earlier steps, every change in your fix plan must include a validation test that conclusively proves that the change is working.

Fixes that you apply should follow the same inviolate rule used during hypothesis testing, that only one change can be made at a time and that the change must be validated before continuing with the next change. Red Hat recommends use of a change management system for applying and tracking changes, such as the Red Hat Ansible Automation Platform. Change management systems provide records that can verify earlier changes, and methods for accurately reverting changes or tested configurations.

After the temporary or permanent fix is applied, test the scenario again against the original problem statement. If the inverse of the problem statement is conclusively true, then you have successfully completed troubleshooting of your problem scenario.

► Guided Exercise

Using the Scientific Method to Solve a Login Issue

In this exercise, you solve a login issue by using the scientific method.

Outcomes

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

Before You Begin

As the student user on the workstation machine, use the lab command to prepare your system for this exercise.

```
[student@workstation ~]$ lab start strategy-scientificmethod
```

This command prepares workstation and servera for troubleshooting login issues.

Instructions

Over the weekend, colleagues ran user account maintenance. This morning, the student user reported that their student login no longer works on servera. The user states that their password is student, and that passwordless ssh normally works from the student account on workstation.

You verify that the user's expected settings are a home directory at /home/student, the UID and GID should both be 1000, and that this user's preferred shell is bash.

This exercise is intentionally simple, and is designed for you to practice including the scientific method in your troubleshooting work flow.

- 1. Begin by reproducing the problem, to collect relevant information. Record as much detail and information as possible about the issue.
 - 1.1. From a command prompt on workstation, attempt to use the ssh command to log in to the student account on servera.

What do you notice, and what can be hypothesized from this information?

```
[student@workstation ~]$ ssh student@servera
Activate the web console with: systemctl enable --now cockpit.socket
```

This system is not registered to Red Hat Insights. See <https://cloud.redhat.com/>
To register this system, run: insights-client --register

Connection to server closed.

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

"The messages indicate that authentication succeeded, but the connection was closed immediately after."

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

"The login proceeds without authentication error messages, but the session is closed immediately."

► 2. Gather more information about the target `servera` system.

- 2.1. Attempt to use the `ssh` command to log in to the `root` account on the `servera` system.

If required and unmodified, the password is `redhat`.

```
[student@workstation ~]$ ssh root@servera
...output omitted...
Last login: Wed Sep  1 17:06:02 2021 from 172.25.250.9
[root@servera ~]# whoami
root
```

"Logging in to the root account succeeds."

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

```
[root@servera ~]# su - student
Last login: Wed Sep  1 17:07:27 EDT 2021 from 172.25.250.9 on pts/0
[root@servera ~]# whoami
root
```

"Switching to the student account did not succeed."

- 2.3. Because the `root` account works, use it to gather information about the `student` account. Verify when `student` last successfully logged in.

```
[root@servera ~]# lastlog -u student
Username      Port      From          Latest
student       pts/0           Wed Sep  1 17:07:27 -0400 2021
```

"This output verifies that the student user successfully logged in during previous tests, suggesting that something is causing the session to end prematurely."

- 2.4. As `root` on `servera`, use the `getent` command to view the `student` user's settings.

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

"The student account is configured with the /bin/false shell, which restricts an account to non-interactive use only."

- 2.5. Create a problem statement. Although the solution appears obvious, creating a problem statement is a proven technique to verify that you can accurately define the problem. If you researched that other users do not have the same problem, and that the problem does not occur for `student` to other servers, then here is an example of an accurate problem statement:

"Since the previous weekend, the student user can access servera but is unable to create a working shell session. Because other users can start sessions on servera, and the student user can start sessions on other servers, the problem appears only with the student use of servera."

- ▶ **3.** With the information that you gathered, form one or more hypotheses of what might 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 shell is not a valid for interactive access, and is not the user's preferred shell."
 - 3.2. Formulate a test for your hypothesis.
"If the shell is the problem, then test by changing the user configuration to the user's preferred shell and attempting the login again."
- ▶ **4.** Reset the shell for **student** to **/bin/bash**, and then verify whether the problem is 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 use the **ssh** command from **workstation** to log in to the **student** account on **servera**.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

"Login to a working shell is successful."

- ▶ **5.** In this example, the only hypothesis proved to be the correct cause of the problem. In scenarios where there are more hypotheses, test each one and record the results.
"Test: Modified the user account to use /bin/bash in the servera local user configuration and logged in again. Result: success."
- ▶ **6.** After recording all test results, the final step is to choose and implement the best solution. During your testing, you verified on one test that the **student** user can access and create a shell on **servera**.
The problem is resolved when the inverse of the original problem statement is true. For this scenario, this is an example of the inverse problem statement as the solution.
"Currently, the student user can successfully access servera and create a working shell session."
- ▶ **7.** Return to **workstation** as the **student** user.

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

Finish

On the **workstation** machine, use the **lab** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish strategy-scientificmethod
```

This concludes the guided exercise.

Collecting Information to Support Troubleshooting

Objectives

After completing this section, you should be able to collect system information to support troubleshooting.

Locating Troubleshooting Information

Successfully solving troubleshooting problems requires not only evaluating current software and hardware behavior, but also discovering previous events that occurred, especially those that relate to the reported failure or issue. Current system behavior is evaluated with diagnostic commands, performance tools, and live monitoring, as discussed in later chapters. Previous events are discovered by viewing the system's journals and log files, as discussed in this section.

Journals and log files contain messages that are received from applications, services, and the kernel. Messages document events, errors, and configuration parameter changes as they occur. The following methods exist to store message information:

- `journald` is a logging service that was introduced with the `systemd` service architecture.
- `rsyslog` is a service that is designed to be a centralized message hub for other software to use.
- Some applications write their own private log files and do not use `journald` or `rsyslog`.

systemd-journald Journal Files

The `journald` service writes binary-structured, memory-based log files, referred to as *journals*, that are viewed with the `journalctl` command. The journals are indexed for faster performance, and the binary format is more secure than plain text log files. The `systemd-journald` service manages the journals, rotating them automatically to limit their storage space and to reduce the need for maintenance.

Messages that `journald` obtains can be forwarded to `rsyslog` by using a memory-based socket file in `/run/systemd/journal/`. Current `rsyslog` implementations use an integrated module to access messages that the `systemd` journal generates instead of by monitoring the socket file.

Viewing Journal Log Files

The `journalctl` command with no options shows all collected journal entries.

```
[user@host ~]$ journalctl
-- Logs begin at Tue 2021-09-14 22:51:26 EDT, end at Wed 2021-09-15 02:01:01 EDT.
--
Sep 14 22:51:26 localhost kernel: Linux version 4.18.0-305.el8.x86_64
(mockbuild@x86-vm-07.build.eng.bos.redhat.com) >
Sep 14 22:51:26 localhost kernel: Command line: BOOT_IMAGE=(hd0,gpt3)/boot/
vmlinuz-4.18.0-305.el8.x86_64 root=/dev/vd>
...output omitted...
```

View journal entries that relate to a specific device, executable, or special file.

```
[user@host ~]$ journalctl /dev/vda
-- Logs begin at Tue 2021-09-14 22:52:15 EDT, end at Wed 2021-09-15 02:11:18 EDT.
--
...output omitted...
Sep 14 22:52:17 localhost kernel: virtio_blk virtio2: [vda] 20971520 512-byte
logical blocks (10.7 GB/10.0 GiB)
```

Filter the journal logs for messages that relate to a specified `systemd` service or other type of `systemd` unit.

```
[user@host ~]$ journalctl -b _SYSTEMD_UNIT=httpd.service
-- Logs begin at Tue 2021-09-14 22:52:15 EDT, end at Wed 2021-09-15 02:14:43 EDT.
--
Sep 15 02:02:06 servera.lab.example.com httpd[5874]: Server configured, listening
on: port 80
```

View logs for a single `systemd` service instance by using that daemon's process ID (PID).

```
[user@host ~]$ journalctl -b _SYSTEMD_UNIT=httpd.service _PID=5874
-- Logs begin at Tue 2021-09-14 22:52:15 EDT, end at Wed 2021-09-15 02:15:28 EDT.
--
Sep 15 02:02:06 servera.lab.example.com httpd[5874]: Server configured, listening
on: port 80
```

Viewing Previous Boot Session Messages

Because the `systemd-journald` service is one of the first to start at boot, the journal logs contain the earliest events and messages that are generated during the boot process.

By default, `journalctl` displays messages since the most current boot only. To view messages from a session before the current boot, first locate the boot ID for the previous session.

The journal log files must be configured for persistent storage to be able to view previous boot session messages. Non-persistent journals are stored in the `/run` memory-mounted file system, and are lost when the system reboots.

```
[user@host ~]$ journalctl --list-boots
0 00552ebbabad42c3a439a303138914ee Tue 2021-09-14 22:52:15 EDT-Wed 2021-09-15
02:19:28 EDT
0 30cd7eb9116d4078a1f11c0fbeac8082 Wed 2021-09-29 00:26:09 EDT-Wed 2021-09-29
00:22:47 EDT
```

View information about a specified `systemd` service by using a boot ID from a previous boot session.

```
[user@host ~]$ journalctl --boot 00552ebbabad42c3a439a303138914ee
_SYSTEMD_UNIT=httpd.service
-- Logs begin at Tue 2021-09-14 22:52:15 EDT, end at Wed 2021-09-15 02:20:28 EDT.
--
Sep 15 02:02:06 servera.lab.example.com httpd[5874]: Server configured, listening
on: port 80
```

Configuring `journald` for Persistent Storage

By default, Red Hat Enterprise Linux 8 stores the system journal logs in a ring-buffer in `/run/log/journal`.

Enable persistent journal storage by creating a disk directory and configuring the service to use it.

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

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

2. Edit the `/etc/systemd/journald.conf` file to enable the persistent storage policy.

```
[root@host ~]# sed -i  
's/#Storage=auto/Storage=persistent/' /etc/systemd/journald.conf
```

3. Restart the `systemd-journald` service to begin using the new persistent directory. If the configured directory does not exist or is not writable by `systemd-journald`, the service will continue to use the `/run/log/journal` location.

```
[root@host ~]# systemctl restart systemd-journald.service
```

`rsyslog` Log Files

The `rsyslog` service acts as a message processing hub. Applications can use system calls to send messages directly to `rsyslog`. The `rsyslog` service uses the `imjournal` input module to retrieve messages continuously from `journald`, and also forwards messages that it receives directly to `journald` by using the `omjournal` output module.

Messages that are sent to `rsyslog` include the `facility` and `level` fields, which are then used for sorting the messages into log files in the `/var/log/` directory. The facility value indicates where the message originated, and the level value indicates the severity of the message. These facilities, among some others, send messages to `rsyslog`:

- Kernel messages
- User-level messages
- Mail system
- System daemons
- Security and authorization messages
- Messages that are generated internally by `syslog`

Locations of the `syslog` Log Files

The `/etc/rsyslog.conf` file contains rules for sorting incoming messages by facility and level, and storing them into specific log files. The following subdirectories under the `/var/log` directory contain `rsyslog` messages:

- `/var/log/messages` stores all the `syslog` messages that are not explicitly configured for other `rsyslog` locations.
- `/var/log/secure` stores security and authentication-related messages and errors.
- `/var/log/maillog` stores mail server-related messages and errors.
- `/var/log/cron` stores log files that relate to periodically executed tasks.
- `/var/log/boot.log` stores log files that relate to system startup.

**Note**

You can access log files with the web console. Log in to the web console and then click Logs to view the default `rsyslog` log entries.

Private Log Files

Some applications maintain their own log files instead of using the `rsyslog` service. Those applications might need a custom message structure or might process their messages differently from `rsyslog` or `journald`. Typically, these log files are in subdirectories of `/var/log`. For example, the Apache Web Server on a RHEL server saves log messages to files under the `/var/log/httpd` directory. Samba is another service that uses private log files, with a default file location under the `/var/log/samba` directory.

Enabling Verbose Information

Many commands and services can increase the logging detail that they generate while running by including a log level option when they start. Examples of log level options include `-v`, `-vv`, `-vvv`, and `--debug`, which can be included in their startup configuration files. Refer to the documentation for individual services where you want to increase the verbosity or logging levels.

**Note**

Including a debug option for services that are configured under the `/etc/sysconfig/` directory might cause that service to be unable to disconnect from its controlling terminal. When such services are started by using `systemctl` and the service type is `forking`, the `systemctl` command does not return until the service is passed an interrupt signal such as `Ctrl-C`.

Alternatively, when troubleshooting, you can run the service manually from the command line with the debug option.

Troubleshooting the Audit Log for SELinux Events

The audit system security log file, at `/var/log/audit/audit.log`, contains audit event objects that relate to possible SELinux access denials. Since the file is securely stored in binary, inspect these audit records with a search utility. Use the `ausearch` tool to query the audit logs for SELinux events.

To resolve SELinux denials, first analyze the root cause as stored in the audit log. Use the `sealert` command from the `policycoreutils-python-utils` and `setroubleshoot-server` packages to help in resolving the problem. SELinux denials might be caused by an incorrect SELinux label, context, Boolean, or port number.

SELinux context issues also occur when a service uses non-standard directories. For example, use of a web server with the non-standard `/home/user/myweb` directory requires that directory to be set with the correct SELinux label.

The `semanage fcontext` command stores a specified SELinux context with the directory location in the SELinux database.

```
[root@host ~]# semanage fcontext -a -t httpd_sys_content_t "/home/user/myweb"
```

The `restorecon` command looks up a specified directory in the database, retrieves the configured context for that directory, and applies the context on the directory.

Use the following command to apply the context change:

```
[root@host ~]# restorecon -R -v /home/user/myweb
```

SELinux Booleans allow exceptions to runtime policy that can be enabled or disabled without restarting a service. In this example, enabling this Boolean enables access to the home directories through a local web server, which is normally a policy behavior that is allowed to web servers.

```
[root@host ~]# setsebool httpd_enable_homedirs on
```

Many services are allowed by policy to run only on specific port numbers. The `semanage port` command allows the service to operate through a non-standard port. In this example, the default port of a web service is changed to use port 9876.

```
[root@host ~]# semanage port -a -t http_port_t -p tcp 9876
```



References

`journalctl(1)`, `systemd.journal-fields(7)`, and `systemd-journald.service(8)`, `auditd(8)`, `ausearch(8)`, `sealert(8)` man pages.

For further information, refer to *Chapter 10. Troubleshooting Problems Using Log Files* at

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html/configuring_basic_system_settings/assembly_troubleshooting-problems-using-log-files_configuring-basic-system-settings

For further information, refer to *Chapter 5. Troubleshooting Problems Related to SELinux* at

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html/using_selinux/troubleshooting-problems-related-to-selinux_using-selinux

► Guided Exercise

Collecting Information to Support Troubleshooting

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

Outcomes

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

Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

```
[student@workstation ~]$ lab start strategy-collectinginfo
```

This command prepares `workstation` and `servera` for troubleshooting file access issues.

Instructions

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

Investigate this issue by using the log files on `servera`, and then fix the issue. For testing from the command line on `workstation`, as an alternative to opening a graphical browser, you can use the command `curl 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 with a Firefox browser, or by executing the following command:

```
[student@workstation ~]$ curl http://servera.lab.example.com/test.html
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>403 Forbidden</title>
</head><body>
<h1>Forbidden</h1>
<p>You don't have permission to access this resource.</p>
</body></html>
```

- 1.2. Consider the possible causes for the HTTP 403 Forbidden error that you encountered. This error can have various reasons: file permissions, SELinux types, internal `httpd` configurations, and others.

Chapter 1 | Introducing Troubleshooting Strategy

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

- 2. Log in to `servera` and switch to the root user.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

- 3. 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.

- 3.1. Check `/var/log/httpd/access_log` for any message about this failure.

```
[root@servera ~]# grep test.html /var/log/httpd/access_log
...output omitted...
172.25.250.9 - - [02/Sep/2021:22:39:46 -0400] "GET /test.html HTTP/1.1" 403 199
 "-" "curl/7.61.1"
...output omitted...
```

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

- 3.2. Check `/var/log/httpd/error_log` for any message about this failure.

```
[root@servera ~]# tail /var/log/httpd/error_log
...output omitted...
[Thu Sep 02 22:39:46.533187 2021] [core:error] [pid 6082:tid 140537780700928]
(13)Permission denied: [client 172.25.250.9:51002] AH00035: access to /test.html
denied (filesystem path '/var/www/html/test.html') because search permissions are
missing on a component of the path
...output omitted...
```

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

- 4. Inspect the file permissions on `/var/www/html/test.html`, and fix if necessary.

- 4.1. Inspect the file permissions on `/var/www/html/test.html`.

```
[root@servera ~]# ls -l /var/www/html/test.html
-rw-----. 1 root root 8 Sep 2 22:39 /var/www/html/test.html
```

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

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

- 4.3. Confirm that the permissions are now correct.

```
[root@servera ~]# ls -l /var/www/html/test.html  
-rw-r--r-- 1 root root 8 Sep 2 23:49 /var/www/html/test.html
```

- 4.4. Return to workstation as the student user.

```
[root@servera ~]# exit  
[student@servera ~]$ exit  
[student@workstation ~]$
```

- 5. Test access to the file again, with either Firefox or curl.

```
[student@workstation ~]$ curl http://servera.lab.example.com/test.html  
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">  
<html><head>  
<title>403 Forbidden</title>  
</head><body>  
<h1>Forbidden</h1>  
<p>You don't have permission to access this resource.</p>  
</body></html>
```

File permissions were an issue, but the problem is still not solved. Therefore one likely culprit remains: SELinux.

- 6. Log in to servera and switch to the root user.

```
[student@workstation ~]$ ssh student@servera  
...output omitted...  
[student@servera ~]$ sudo -i  
[sudo] password for student: student  
[root@servera ~]#
```

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

- 7.1. Check the SELinux log for any denials today.

```
[root@servera ~]# ausearch -i -m avc -ts today  
...output omitted...  
type=AVC msg=audit(09/02/2021 22:39:46.532:4380) : avc: denied  
{ getattr } for pid=6082 comm=httpd path=/var/www/html/test.html  
dev="vda3" ino=1067530 scontext=system_u:system_r:httpd_t:s0  
tcontext=unconfined_u:object_r:samba_share_t:s0 tclass=file permissive=0  
...output omitted...
```

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

- 7.2. Fix this issue by running a recursive `restorecon` on `/var/www`.

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

7.3. Return to workstation as the student user.

```
[root@servera ~]# exit  
[student@servera ~]$ exit  
[student@workstation ~]$
```

- 8. Test whether you can now access <http://servera.lab.example.com/test.html> from workstation.

```
[student@workstation ~]$ curl http://servera.lab.example.com/test.html  
ServerA
```

Finish

On the workstation machine, use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish strategy-collectinginfo
```

This concludes the guided exercise.

Troubleshooting with Red Hat Resources

Objectives

After completing this section, you should be able to use Red Hat resources to support troubleshooting.

Red Hat Customer Portal

The Red Hat Customer Portal <https://access.redhat.com> provides customers with access to their subscription benefits in one place. Customers can search for solutions, FAQs, Knowledgebase articles, and official product documentation. Many features are accessible to everyone, while some are exclusive to customers with active subscriptions.

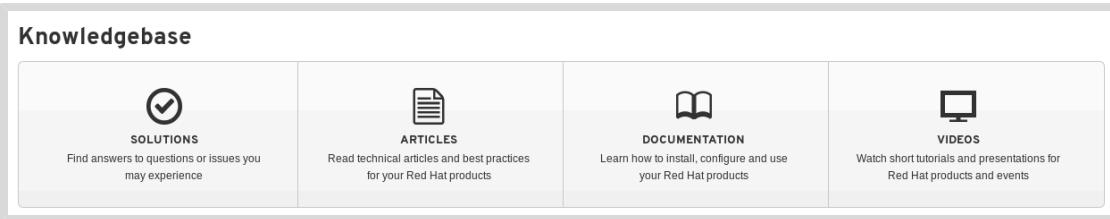


Figure 1.1: Knowledgebase at the Red Hat Customer Portal

At the Customer Portal, customers can manage Red Hat product subscriptions on registered systems, download software, upgrades, and evaluations, and submit and manage support cases for those systems. The Red Hat Customer Portal supports using the command-line `redhat-support-tool` to access the same services. Help for obtaining access is available at <https://access.redhat.com/help/>.

The Red Hat Knowledgebase is at <https://access.redhat.com/knowledgebase/>.

Collecting Information for Red Hat Support

Red Hat Enterprise Linux includes the `sos report` tool in the `sos` package. This tool collects configuration details, log files, system information, diagnostic information, then combines the results in a tarball to attach to an open Red Hat Support case. The command can collect information from multiple systems. The tool requires `root` privileges.

The `sos clean` subcommand obfuscates potentially sensitive system information that is not removed by standard `sos report` postprocessing. This data includes IP addresses, networks, MAC addresses, and more.

When run without options, the `sos report` command prompts the user for a case number, and then collects a default set of files and settings, and creates the tarball. Options can enable or disable plug-ins, configure plug-in options, and make the process run without interaction.

Run `sos report -l` to view all plug-ins and those which are currently enabled, and configurable plug-in options. To enable additional plug-ins, use the `-e ENABLE_PLUGINS` option. To configure plug-in options, use the `-k PLUGOPTS` option. The `-n SKIP_PLUGINS` option disables unwanted plug-ins in the report execution.

Use the `--encrypt-key` and `--encrypt-pass` options with `sos report` command to generate encrypted reports.

**Note**

If the `sos report` command fails, you can collect the data manually. Follow the steps in Knowledgebase solution 68996.

Sosreport fails. What data should I provide in its place? at <https://access.redhat.com/solutions/68996>

Searching the Knowledgebase

The Red Hat Support Tool utility `redhat-support-tool` provides a text-console interface to the Red Hat Access subscription services. Internet access is required. The `redhat-support-tool` is a text-based tool, which can be run using SSH or from any terminal.

The `redhat-support-tool` can use an interactive shell or be invoked as individual commands with options and arguments. The syntax is identical for both methods. By default, the program launches in shell mode. Use the `help` subcommand to view available subcommands. Shell mode supports tab completion and calling other programs from the parent shell.

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

When first invoked, `redhat-support-tool` prompts for the 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 many users share a Red Hat Access account, 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.

Subscribers can use the `redhat-support-tool` to search and display the same Knowledgebase content as 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 output is an initial configuration and basic search demonstration:

```
[user@host ~]$ 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/user/.redhat-support-tool/redhat-support-tool.conf (y/
n): y
Please enter the password for subscriber: password
Save the password for subscriber in /home/user/.redhat-support-tool/redhat-
support-tool.conf (y/n): y
```

After prompting for the user configuration, the tool continues with the 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 the Red Hat Customer  
Portal using Red Hat Subscription-Manager  
2 [3121571:VER] How to register and subscribe a system offline to the Red Hat  
Customer Portal?  
2 of 50 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: 1
```

You can select specific sections of solution documents for viewing.

```
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 the Red Hat Customer Portal using Red  
Hat Subscription-Manager  
URL: https://access.redhat.com/solutions/253273  
Created On: 2012-10-30T04:24:25-04:00  
Modified On: 2017-11-29T10:33:51-05:00  
(END) q
```

Directly Access Knowledgebase Articles by Document ID

Locate online articles using the tool's kb command with the Knowledgebase document ID. Documents scroll without pagination, allowing a user to redirect the output to other commands.

```
[user@host ~]$ redhat-support-tool kb 253273 | less  
  
Title  
=====  
How to register and subscribe a system to the Red Hat Customer Portal using Red  
Hat Subscription-Manager  
URL: https://access.redhat.com/solutions/253273  
  
Issue  
=====  
* How to register a new `Red Hat Enterprise Linux` system to the Customer Portal  
using `Red Hat Subscription-Manager`  
* How to un-register a system using `Red Hat Subscription-Manager`  
  
: q
```

Documents that are retrieved in unpaginated format are easy to send to a printer, or convert to PDF or other document format. Documents can also be redirected to a data entry program for

an incident tracking or change management system, by using other utilities that are installed and available in Red Hat Enterprise Linux.

Managing Support Cases

One subscription benefit 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 information about the support process.

Preparing a Bug Report

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

- *Define the problem.* Clearly state the problem and its symptoms. Be as specific as possible. Detail the steps to reproduce the problem.
- *Gather background information.* Which products and versions are affected? Be ready to provide relevant diagnostic information. The files can include the output of `sos report`, as discussed earlier in this section. For kernel problems, the files could include the system's `kdump` crash dump or a digital photo of the kernel backtrace that is 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.

Severity	Description
<i>Urgent</i> (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.
<i>High</i> (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.
<i>Medium</i> (Severity 3)	A problem that involves partial, non-critical loss of use of the software in a production environment or development environment. For production environments, the situation has a medium-to-low impact on the business, but the business continues to function, including by using a procedural workaround. For development environments, the situation is causing the project to no longer continue or migrate into production.
<i>Low</i> (Severity 4)	A general usage question, reporting of a documentation error, or recommendation for a future product enhancement or modification. For production environments, the situation has a low-to-no impact on the business or on 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 Bug Reports

Subscribers can create, view, modify, and close Red Hat Support cases with `redhat-support-tool`. When support cases are opened or maintained, users can include files or documentation, such as diagnostic reports (`sos report`). The tool uploads and attaches files to online cases.

Chapter1 | Introducing Troubleshooting Strategy

Case details including product, version, summary, description, severity, and case group may be assigned with command options or by allowing the tool to prompt for required information. In this example, the `--product` and `--version` options are specified, but `redhat-support-tool` could provide choices if the `opencase` command had not specified them.

```
[student@demo ~]$ redhat-support-tool
Welcome to the Red Hat Support Tool.
Command (? for help): opencase --product="Red Hat Enterprise Linux"
--version="8.4"
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 use the default (Ungrouped Case) Case Group (y/N)? : y
Would see if there is a solution to this problem before opening a support case?
(y/N) N
-----
Support case 03022708 has successfully been opened.
```

Including Diagnostic Information to Support Cases

Including diagnostic information when you create a support case contributes to quicker problem resolution. The `sos report` command generates a compressed tarball archive of diagnostic information that is gathered from the running system. The `redhat-support-tool` prompts to include another tarball if an archive was created previously:

```
Please attach a SoS report to support case 03022708. 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 03022708 <path to sosreport>

Would you like to attach a file to 03022708 at this time? (y/N) N
Command (? for help):
```

If an `sos report` archive is not already prepared, then an administrator can generate and attach one later, with the tool's `addattachment` command. Subscribers can view, modify, and close support cases.

```
Command (? for help): listcases

Type the number of the case to view or 'e' to return to the previous menu.
1 03022708 [Waiting on Red Hat ] [sev4] 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 Get Attachment
5 Add Attachment
6 Add Comment
End of options.
Option: q

Select a Case: q

Command (? for help): q

[user@host ~]$ redhat-support-tool modifycase --status=Closed 03022708
[user@host ~]$
```

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

The tool analyzes log files. With the tool's `analyze` command, you can parse log files of many applications to recognize problem symptoms. You can use these log files to inspect and diagnose unexpected individual behaviors.

Red Hat Customer Portal Tools

The Red Hat Customer Portal home page provides a **Tools** menu to troubleshoot Red Hat products, making it easy to search for common issues and possible solutions.

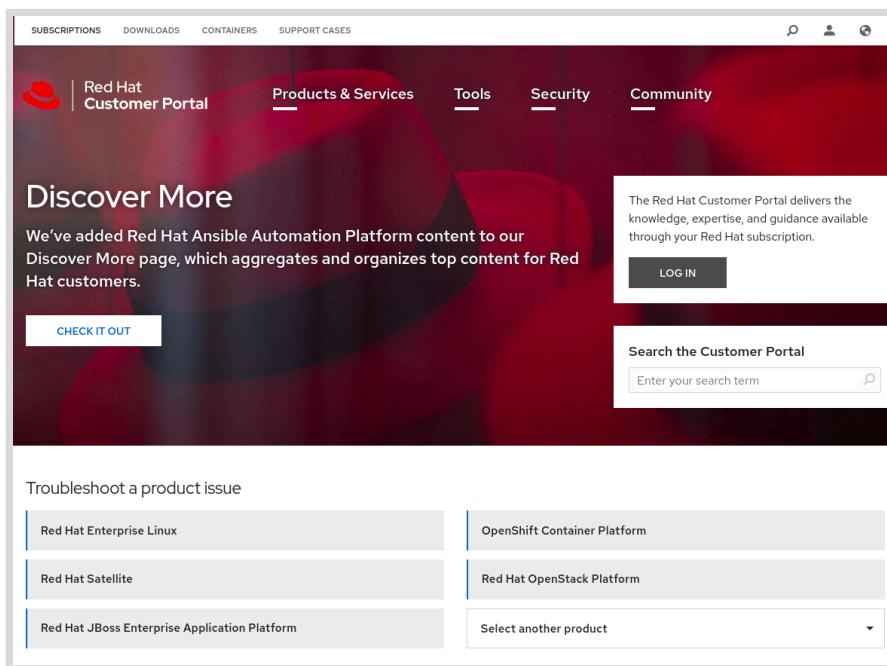


Figure 1.2: Troubleshoot a product issue

The Red Hat Customer Support Troubleshoot tool, at <https://access.redhat.com/support/cases/#/troubleshoot/describe-issue>, helps with searching for common solutions in the Knowledgebase.

The screenshot shows the 'Customer support' interface with the 'Troubleshoot' tab selected. On the left, there are two numbered steps: '1 Select a product' and '2 Describe your issue'. Step 1 has a dropdown for 'Product *' and another for 'Version *'. Below these are dropdowns for 'Select a top product' with options for Red Hat Enterprise Linux, OpenShift Container Platform, Red Hat Satellite, Red Hat OpenStack Platform, and Red Hat JBoss Enterprise Application Platform. A note at the bottom says 'Have an account, billing, or subscription issue? Contact customer service for help.' A 'Continue' button is located at the bottom right of the form area.

Figure 1.3: Troubleshoot main page

Red Hat Customer Portal Labs

Red Hat Customer Portal Labs, at <https://access.redhat.com/labs>, provide web-based applications to aid in configuration, deployment, security, and troubleshooting.

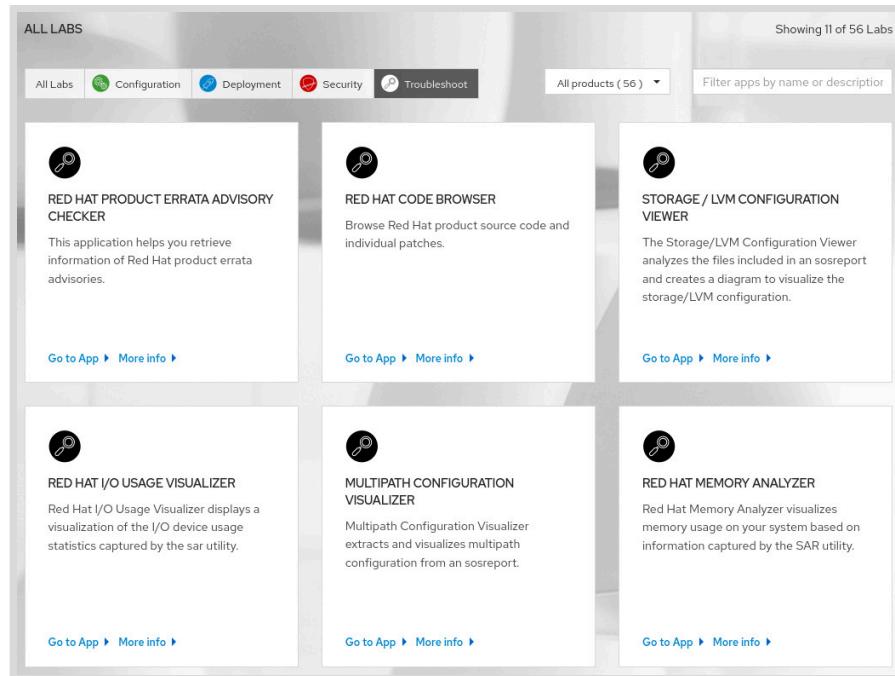


Figure 1.4: Troubleshooting Labs at the Red Hat Customer Portal

The Customer Portal Troubleshooting Labs include various analysis tools:

Red Hat I/O Usage Visualizer

Provides a visualize I/O usage report.

Red Hat Memory Analyzer

Provides a visualize memory usage report.

JVM Peg

Help to analyze JVM threads that are overworking a CPU above a specified threshold.

Log Reaper

The app provides log analysis that emphasizes identification of errors in log files, presented in a view tailored for each log type, with automatic solution recommendations, and targeted analysis.

Kernel Oops Analyzer

Help to diagnose a kernel fault by using the input of an error message or file that contains one or more kernel oops messages.

Red Hat Insights

Red Hat Insights is a hosted service for system administrators and managers to 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 are made. These recommendations can help to keep systems stable and performing, by spotting any potential issues and giving remediation advice before they can become larger problems that cause disruption.

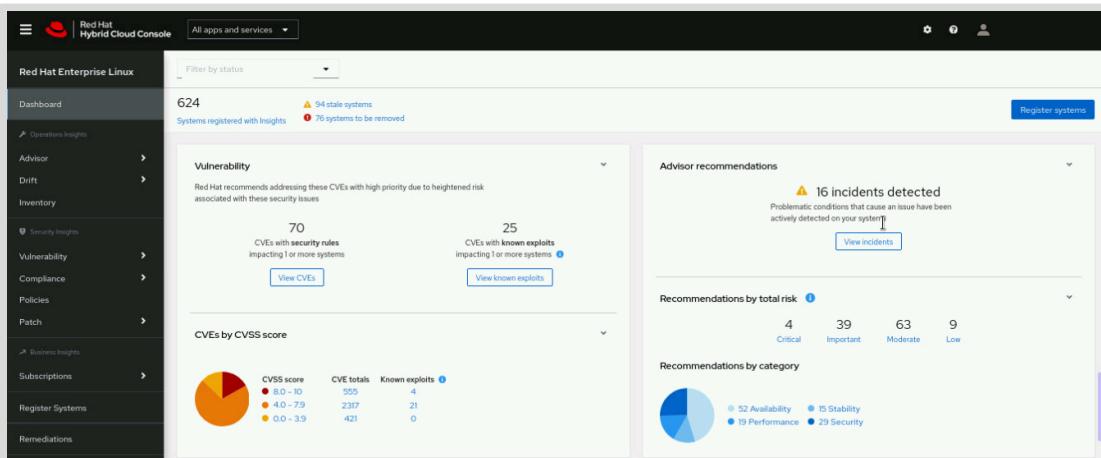


Figure 1.5: Red Hat Insights

The web interface at <https://console.redhat.com/insights> outlines issues that can affect registered systems, ordered by severity and type. Administrators can drill down into issues for tailored recommendations. Administrators can also choose to permanently ignore certain rules.

Red Hat Insights Services

Red Hat Insights provides services to administer and monitor registered systems to troubleshoot and remediate identified issues.

Advisor

Advisor identifies known configuration risks in the operating system, underlying infrastructure, or workloads that impact performance, stability, availability, or security best practices.

Vulnerability

Vulnerability assesses, remediates, and reports on *Common Vulnerabilities and Exposures* (CVEs) that impact Red Hat Enterprise Linux environments in the cloud or on premises.

Compliance

Compliance analyzes the compliance level of a Red Hat Enterprise Linux environment to an OpenSCAP policy, based on the version of SCAP Security Guide (SSG), supported by Red Hat Enterprise Linux.

Patch

Patch determines which Red Hat product advisories apply to an organization's specific Red Hat Enterprise Linux instances. The patch provides guidance for remediation either manually or via Ansible Playbooks for patching.

Drift

Drift compares systems to baselines, system histories, and to each other to troubleshoot or identify differences.

Policies

Organizations can use the Policies service to define and monitor policies that are important internally, with alerts for environments that are not aligned to a policy.

Inventory

The Inventory service lists all of the hosts that are registered to Insights.

Remediations

The Remediation service lists all remediation plans, concentrating them in one place for easy access and analysis. You can download the remediation plans from this service. Alternatively, if Smart Management with Cloud Connector is also configured, then you can execute the playbook directly from the Remediations service.

Subscription Watch

Subscription Watch provides unified reporting of Red Hat subscription usage for easier and more efficient management of subscriptions to Red Hat Enterprise Linux and Red Hat OpenShift Platform.

Registering Systems Using Red Hat Insights

To register systems with Red Hat Insights, follow these steps:

1. Ensure that the `insights-client` package is installed:

```
[root@host ~]# yum install insights-client
```

2. Register the system with Red Hat Insights.

```
[root@host ~]# insights-client --register
You successfully registered 4578cb7f-47a9-4203-bef4-42651a257984 to account
5662036.
Successfully registered host host.lab.example.com
Automatic scheduling for Insights has been enabled.
Starting to collect Insights data for host.lab.example.com
Uploading Insights data.
Successfully uploaded report from host.lab.example.com to account 5662036.
View the Red Hat Insights console at https://cloud.redhat.com/insights/
```

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



References

For more information, refer to the **Generating sos Reports for Technical Support Guide** at

https://access.redhat.com/documentation/en/red_hat_enterprise_linux/8/html-single/generating_sos_reports_for_technical_support/index

Red Hat Access: Red Hat Support Tool

<https://access.redhat.com/site/articles/445443>

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 Customer Portal Labs

<https://access.redhat.com/labs>

Red Hat Insights

https://access.redhat.com/documentation/en/red_hat_insights/2021

sos(1) man page

► Guided Exercise

Troubleshooting with Red Hat Resources

In this exercise, you generate and inspect an `sos` report on your `servera` system.

Outcomes

You should be able to generate an `sos` report to aid Red Hat Support in resolving an issue.

Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

```
[student@workstation ~]$ lab start strategy-redhatresources
```

This command ensures that your systems are reachable from the `workstation` machine.

Instructions

While troubleshooting an issue on `servera`, you contacted Red Hat Support for assistance. Support requested that you generate an `sos` report using the `xfs` plugin.

For the purpose of this exercise, your case number is 123456.

- 1. Log in to `servera` and switch to the `root` user.

```
[student@workstation ~]$ ssh student@servera  
...output omitted...  
[student@servera ~]$ sudo -i  
[sudo] password for student: student  
[root@servera ~]#
```

- 2. Verify that the `sos` package is installed on your `servera` system, and install it if necessary.

- 2.1. Verify that the `sos` package is installed.

```
[root@servera ~]# rpm -q sos  
sos-4.0-11.el8.noarch
```

- 2.2. If the previous command returned package `sos` is not installed, then install the `sos` package.

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

- 3. View the available options, plug-ins, and plug-in options for `sos report`.

- 3.1. View the available options for the `sos report` command.

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

3.2. View the available plug-ins and plug-in options for the `sos report` command.

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

► 4. Generate an `sos report` on `servera`. Enable only the `xfs` plug-in.

4.1. Run the `sos report` tool with the `-o xfs` option. Use the interactive prompts to enter all required information.

```
[root@servera ~]# sos report -o xfs
```

`sosreport` (version 4.0)

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/sos.og2eg7gq` 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 the case id that you are generating this report for []: **123456**

Setting up archive ...

Setting up plugins ...

Running plugins. Please wait ...

Starting 1/1 xfs [Running: xfs]

Finished running plugins

Creating compressed archive...

Your `sosreport` has been generated and saved in:

`/var/tmp/sosreport-servera-123456-2021-08-30-pgstdzb.tar.xz`

Size 3.76KiB

Owner root

```
md5 8d66fd763d397ce8de263bfdd042b7aa
```

Please send this file to your support representative.

4.2. Return to workstation as the student user.

```
[root@servera ~]# exit  
[student@servera ~]$ exit  
[student@workstation ~]$
```

► 5. Copy the generated file to the student user's home directory on workstation, and extract it. The file name used in this exercise is an example, and yours will be different.

5.1. Copy the generated file to the student user's home directory on workstation.

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

5.2. Extract the generated file.

```
[student@workstation ~]$ tar xvf sosreport-servera-123456-2021-09-02-  
ytgpkbx.tar.xz
```

► 6. Inspect the extracted sos report and validate that the xfs plug-in was used.

6.1. Inspect the extracted sos report by listing its contents.

```
[student@workstation ~]$ cd sosreport-servera-123456-2021-09-02-ytgpkbx  
[student@workstation ~]$ ls -l
```

6.2. Confirm the use of xfs plug-in inspecting the xfs configuration of servera.

```
[student@workstation ~]$ cat sos_commands/xfs/xfs_info
```

Finish

On the workstation machine, use the lab command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish strategy-redhatresources
```

This concludes the guided exercise.

► Lab

Introducing Troubleshooting Strategy

In this lab, you solve a Secure Copy Protocol (SCP) file transfer issue.

Outcomes

You should be able to solve an SCP file transfer issue by using the scientific method.

Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your systems for this lab.

```
[student@workstation ~]$ lab start strategy-review
```

This command creates the necessary files to perform this lab.

Instructions

The student user on workstation reported that they are unable to copy a file from serverb by using SCP.

The user provided the following information:

- The file that they are trying to copy is `/home/student/document.txt`.
- They are trying to copy the file to the student user's home directory on workstation.
- They are connecting to serverb as the student user.



Note

The purpose of this lab is to implement the scientific method to troubleshoot an issue, regardless of how simple the issue is. Force yourself to exaggerate the process to understand the importance of each step.

1. Collect relevant information.
2. Create a problem statement.
3. Formulate testable hypotheses.
4. Test each hypothesis.
5. Record and analyze the test results.
6. Fix and verify the problem resolution.

Evaluation

On the workstation machine, use the `lab` command to grade your work. Correct any reported failures and rerun the script until you receive a passing grade.

```
[student@workstation ~]$ lab grade strategy-review
```

Finish

On the **workstation** machine, use the **lab** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish strategy-review
```

This concludes the lab.

► Solution

Introducing Troubleshooting Strategy

In this lab, you solve a Secure Copy Protocol (SCP) file transfer issue.

Outcomes

You should be able to solve an SCP file transfer issue by using the scientific method.

Before You Begin

As the **student** user on the **workstation** machine, use the **lab** command to prepare your systems for this lab.

```
[student@workstation ~]$ lab start strategy-review
```

This command creates the necessary files to perform this lab.

Instructions

The **student** user on **workstation** reported that they are unable to copy a file from **serverb** by using SCP.

The user provided the following information:

- The file that they are trying to copy is **/home/student/document.txt**.
- They are trying to copy the file to the **student** user's home directory on **workstation**.
- They are connecting to **serverb** as the **student** user.



Note

The purpose of this lab is to implement the scientific method to troubleshoot an issue, regardless of how simple the issue is. Force yourself to exaggerate the process to understand the importance of each step.

1. Collect relevant information.

1.1. Login to **serverb** and switch to the **root** user.

```
[student@workstation ~]$ ssh student@serverb
...output omitted...
[student@serverb ~]$ sudo -i
[sudo] password for student: student
[root@serverb ~]$
```

1.2. Gather information about the **sshd** service. SCP relies on **sshd** to function and the status would report any recent errors.

```
[root@serverb ~]$ systemctl status sshd.service
● sshd.service - OpenSSH server daemon
  Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; vendor preset: enabled)
  Active: active (running) since Wed 2021-09-15 09:07:20 EDT; 7h ago
    ...output omitted...
```

- 1.3. Gather information about the /home/student/document.txt file.

```
[root@serverb ~]$ ls -l /home/student
total 0
-rw-r----- 1 consultant consultant 0 Sep 20 09:41 document.txt
```

- 1.4. Gather information about potential SELinux alerts. If the output is blank, then no alerts were found.

```
[root@serverb ~]$ cat /var/log/messages | grep sealert
```

- 1.5. Return to workstation as the student user.

```
[root@serverb ~]# exit
[student@serverb ~]$ exit
[student@workstation ~]$
```

- 1.6. From workstation, attempt to re-create the issue.

```
[student@workstation ~]$ scp student@serverb:~/document.txt .
scp: /home/student/document.txt: Permission denied
```

Observe the error message and consider the information that you gathered from the previous steps.

2. Create a problem statement.

- 2.1. The following would be an appropriate problem statement. "Today, the student user reported that they are unable to copy the /home/student/document.txt file from serverb to workstation with the scp command. A permission denied error message can be consistently reproduced."

3. Formulate testable hypotheses.

- 3.1. The following hypotheses are exaggerated to demonstrate other plausible real-world scenarios. Some of these hypotheses ignore information from the information gathering step for the sake of testing multiple hypotheses.
 - serverb is not reachable over the network. Test by accessing network interfaces with network tools.
 - The sshd process on serverb is not running or is misconfigured. Test by using SSH to log in to the system.
 - SELinux on serverb is preventing the file transfer. Test by temporarily disabling SELinux.

Chapter 1 | Introducing Troubleshooting Strategy

- /home/student/document.txt has incorrect file permissions. Test by adding read permissions.

4. Test each hypothesis.

- 4.1. Test the hypothesis that serverb cannot be reached over the network with network tools.

```
[student@workstation ~]$ ping 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=0.861 ms
...output omitted...
```

The hypothesis that serverb is not reachable can be eliminated because the ping command succeeds.

- 4.2. Test the hypothesis that the sshd process on serverb is not running properly by attempting to run a remote SSH command on the system.

```
[student@workstation ~]$ ssh student@serverb echo test message on serverb
test message on serverb
```

The hypothesis that the sshd process on serverb is not functioning can be eliminated because the ssh command succeeds.

- 4.3. Test the hypothesis that SELinux on serverb is preventing the file transfer by disabling SELinux.

```
[student@workstation ~]$ ssh root@serverb setenforce 0
[student@workstation ~]$ scp student@serverb:~/document.txt .
scp: /home/student/document.txt: Permission denied
[student@workstation ~]$ ssh root@serverb setenforce 1
```

The hypothesis that SELinux is preventing the file transfer can be eliminated because disabling SELinux did not resolve the issue. Notice that the change was immediately reversed after it was tested and eliminated as a possible issue.

- 4.4. Test the hypothesis that a file permission issue is preventing the file transfer.

```
[student@workstation ~]$ ssh root@serverb chown
student:student /home/student/document.txt
[student@workstation ~]$ scp student@serverb:~/document.txt .
document.txt          100%    0     0.0KB/s   00:00
```

It appears that the issue was likely caused by incorrect file permissions.

5. Record and analyze the test results.

- 5.1. Based on the preceding tests, it can be determined that the issue was caused by incorrect file permissions. If none of the tests had succeeded, then the scientific method would have to be repeated with new hypotheses, or the issue would need to be escalated.

6. Fix and verify the problem resolution.

- 6.1. In this lab's case, no further fixes are required to resolve the issue permanently. The issue can be marked as resolved, given that the current state is the inverse of the problem statement: "Now, the student user reports that they are able to use the scp command to transfer the /home/student/document.txt file from serverb to workstation, without any error messages."

Evaluation

On the **workstation** machine, use the **lab** command to grade your work. Correct any reported failures and rerun the script until you receive a passing grade.

```
[student@workstation ~]$ lab grade strategy-review
```

Finish

On the **workstation** machine, use the **lab** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish strategy-review
```

This concludes the lab.

Summary

In this chapter, you learned:

- The scientific method is an efficient form of troubleshooting.
- The `journalctl` command outputs system logs generated by systemd services and other units.
- The `sos report` command gathers detailed system information into a tarball.
- The `redhat-support-tool` command interfaces with Red Hat support services.
- The Red Hat Customer Portal provides tools to diagnose and troubleshoot issues.
- The Red Hat Insights service provides system analytic and remediation strategies.

Chapter 2

Configuring Baseline Data

Goal

Configure baseline data collection with monitoring, logging, and change tracking.

Objectives

- Monitor systems to gather information.
- Configure systems for remote logging to a central log host.
- Describe configuration management with Red Hat Satellite and Red Hat Ansible Automation Platform.
- Implement change tracking to monitor system modifications.

Sections

- Monitoring Systems (and Guided Exercise)
- Configuring Remote Logging (and Guided Exercise)
- Describing Configuration Management Automation (and Guided Exercise)
- Configuring Change Tracking (and Guided Exercise)

Lab

Configuring Baseline Data

Monitoring Systems

Objectives

After completing this section, you should be able to monitor systems to gather information.

System Monitoring with the web console

The web console is a Red Hat Enterprise Linux 8 web-based interface. It is built with Cockpit technology, and is designed for managing and monitoring a Linux host. Users administer their system with the graphical interface, without requiring command-line tool knowledge. To perform privileged tasks in the web console, users must be configured for privileges with RHEL sudo configuration.

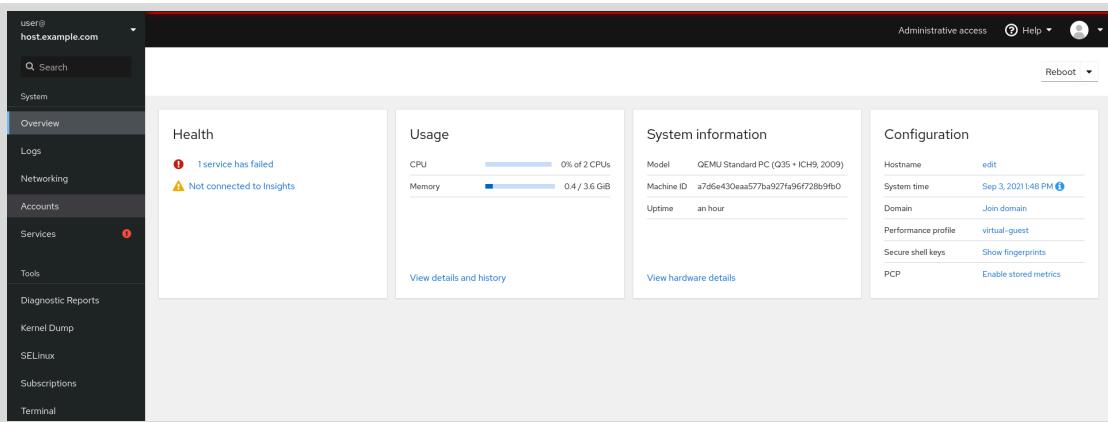


Figure 2.1: web console

Most Red Hat Enterprise Linux 8 deployments install the web console by default. If necessary, install the `cockpit` package to obtain the web console.

```
[root@host ~]# yum install cockpit
```

Use the `systemctl` command to enable and start the `cockpit.socket` service.

```
[root@host ~]# systemctl enable --now cockpit.socket
```

If you manually installed the web console, then also add the `cockpit` service to `firewalld` to open port 9090 tcp.

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

The web console uses existing system user names and passwords. If you are logging in with a user account with `sudo` privileges, you can perform privileged tasks in the web console, such as installing software or configuring SELinux.

Running privileged tasks requires entering your user password each time, as you do when running `sudo` commands on a command line. Alternatively, to avoid re-entering your password for each privileged task, select `Reuse my password for privileged tasks` on the login screen.

After successful authentication, the web console displays the **Overview** page.

web console Add-ons

Extend web console capabilities by installing available add-on packages with `yum`.

```
[root@host ~]# yum install add-on
```

This table lists available add-on applications for the web console.

Feature	Package	Usage
Composer	<code>cockpit-composer</code>	Building custom OS images
PackageKit	<code>cockpit-packagekit</code>	Managing packages, and installing updates and applications
Performance Metrics	<code>cockpit-pcp</code>	Collecting performance metrics
Podman containers	<code>cockpit-podman</code>	Managing podman containers
Session Recording	<code>cockpit-session-recording</code>	Recording and managing user sessions
Storage	<code>cockpit-storaged</code>	Managing system storage
Virtual Machines	<code>cockpit-machines</code>	Managing <code>libvirt</code> virtual machines

Monitoring Performance in the web console

The web console provides observable data for troubleshooting with the Utilization Saturation and Errors (USE) Method. The USE method is a checklist for solving performance issues quickly, by observing utilization, saturation, and errors for each expected resource.

- **Utilization:** the average time that the resource was busy performing work
- **Saturation:** the measurement of how much extra work is queued and not being serviced by the resource
- **Errors:** the number of recorded error events

The **Overview → Performance Metrics** page displays a chronological view of historical data, and lists events, errors, and graphical representations for resource utilization and saturation.

To access the data, navigate to the **Overview** page, and click the **View details and history** link.

Chapter 2 | Configuring Baseline Data

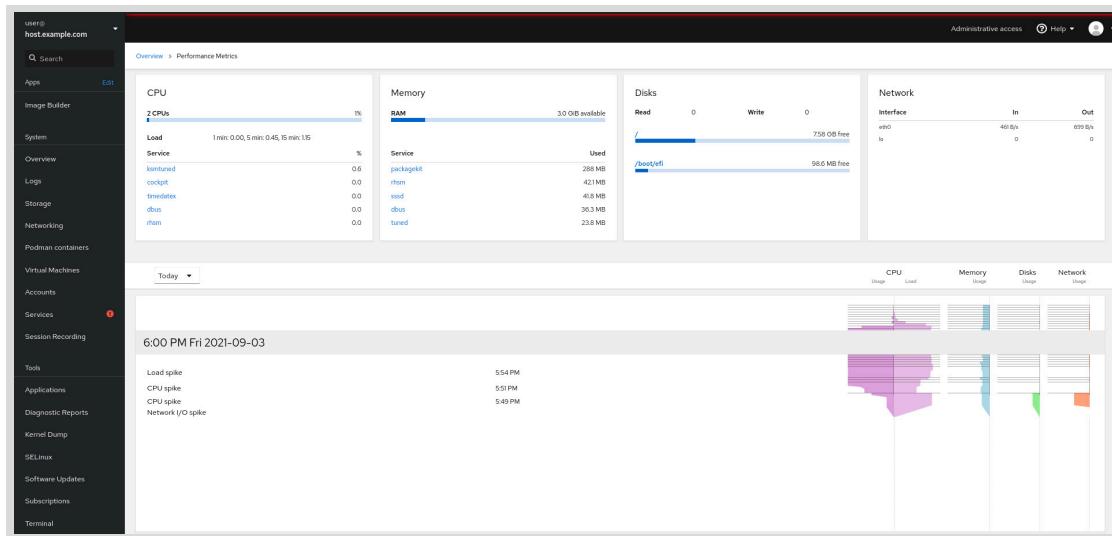


Figure 2.2: Performance metrics in the web console

Reviewing Logs in the web console

The web console Logs page provides an interface for displaying `journalctl` log journals. Log entries can be filtered by *Time*, *Priority*, *Identifier*, and *Text* patterns.

Time

Filter for a time range.

Priority

Select a minimum priority level, similar to `journalctl --priority`. The default is **Error and above**.

Identifier

Select a `systemd` unit or service name identifier, similar to `journalctl --identifier`.

Text

Enter a text pattern to match to filter messages.

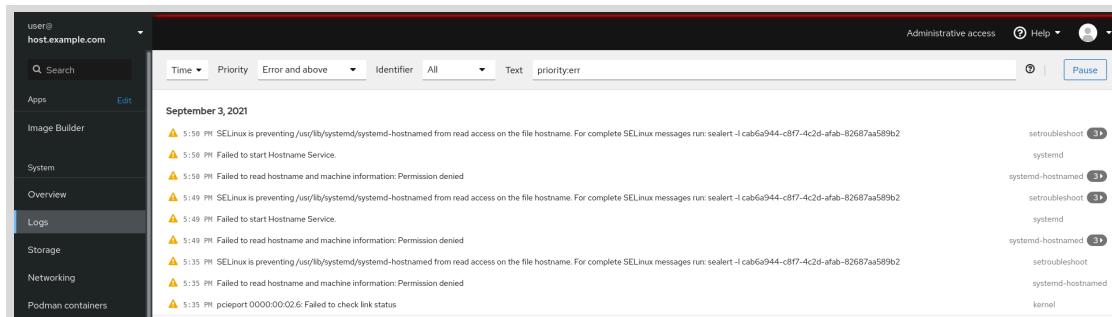


Figure 2.3: Logs in the web console

Session Recording in the web console

The Session Recording capability is new with the web console, and is based on the `tlog` package. The web console session player can record and play back user terminal sessions. The session player captures terminal input and output and stores the session in a text format in the system journal.

**Important**

Session recording is disabled by default. When enabled, passwords and sensitive information are stored in plain text.

To enable session recording, install the `cockpit-session-recording` and `tlog` packages, and ensure that the `sssd` service is enabled and started.

```
[root@host ~]# yum install cockpit-session-recording tlog
[root@host ~]# systemctl status sssd
```

Configure the `sssd-session-recording.conf` file to specify the user or user groups to record, and the session scope:

- `none` to record no sessions.
- `some` to record only specified sessions.
- `all` to record all sessions.

```
[root@host ~]# cat /etc/sssd/conf.d/sssd-session-recording.conf
[session_recording]
scope=none
users=
groups=
```

You can configure session recording for users and scope in the web console, on the **Session Recording** page. Click the gear icon to open the **General Config** form and select the wanted options.

Figure 2.4: Session recording in the web console

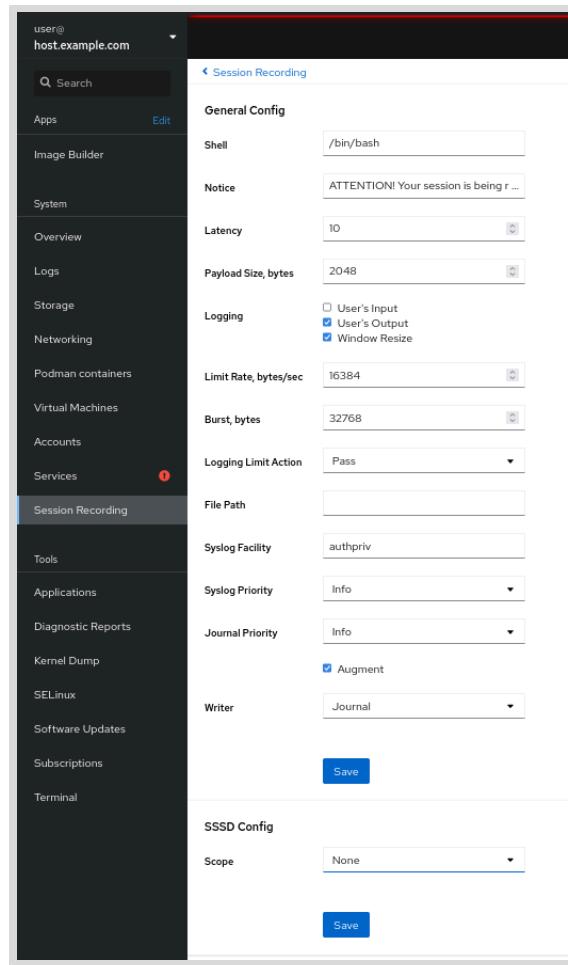


Figure 2.5: Session Recording General Config

Replay Recorded Sessions

Play back recorded sessions for analysis with web console or with the `tlog-play` tool that is included in the `tlog` package.

Play back with the web console

The Session Recording page lists previously recorded sessions for playback and analysis. To play a session, select and click a session entry in the list. The web console opens a page with playback and analysis controls for the session.

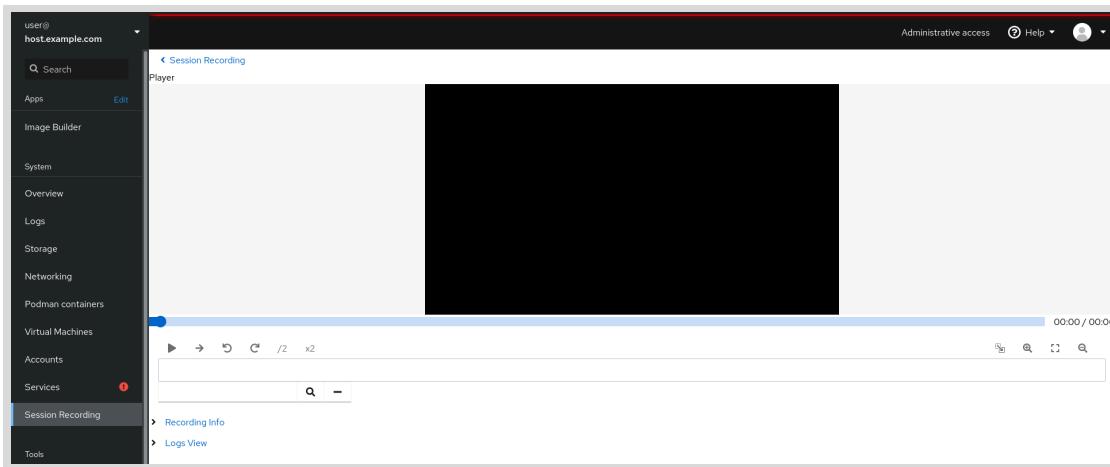


Figure 2.6: Session playback controls

Play back with tlog-play

Working from the command line, the `tlog-play` command is the playback program for terminal input and output that are recorded with the `tlog-rec` command.

When recording a terminal session, specify a file name for capturing the activity.

```
[user@host ~]$ tlog-rec --file-path=tlog.log
```

All terminal typing is stored in the log file, until the user types `exit` to end the recording.

To play back the session, specify the stored log file.

```
[user@host ~]$ tlog-play --file-path=tlog.log
```



References

For further information, refer to the *Managing Systems Using the RHEL 8 Web Console Guide* at
https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html/managing_systems_using_the_rhel_8_web_console/index

For further information, refer to the *Recording Sessions Guide* at
https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/recording_sessions/index

For more information about Cockpit, visit **Cockpit Project**
<http://www.cockpit-project.org>

► Guided Exercise

Monitoring Systems

In this lab, you monitor system performance and record the terminal session of a consultant by using the web console.

Outcomes

You should be able to display current and historical performance indicators on a system and play back a terminal session with the web console.

Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

```
[student@workstation ~]$ lab start baseline-monitoring
```

This command confirms that the required hosts for this exercise are accessible and creates the consultant user.

Instructions

In this exercise, you install and configure the web console software on `servera`. You use web console features to gather current performance statistics of the system. You record the terminal session of a consultant.

- 1. Log in to `servera` and switch to the root user.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

- 2. Install the web console software packages on `servera`.

- `cockpit-composer`
- `cockpit-machines`
- `cockpit-pcp`
- `cockpit-podman`
- `cockpit-session-recording`
- `tlog`

```
[root@servera ~]# yum install cockpit-composer cockpit-machines cockpit-pcp
cockpit-podman cockpit -session-recording tlog
```

- 3. Restart the pmlogger service to configure persistent logging of performance metrics.

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

- 4. Start and enable the web console.

```
[root@servera ~]# systemctl enable --now cockpit.socket
```

- 5. Log out from the root user and from servera.

```
[root@servera ~]# exit  
[student@servera ~]$ logout  
Connection to servera closed.  
[student@workstation ~]$
```

- 6. Inspect the system performance status of servera on the web console.

- 6.1. On workstation, open a web browser and navigate to <https://servera:9090>. Accept the certificate warning. Log in with user name student and password student.

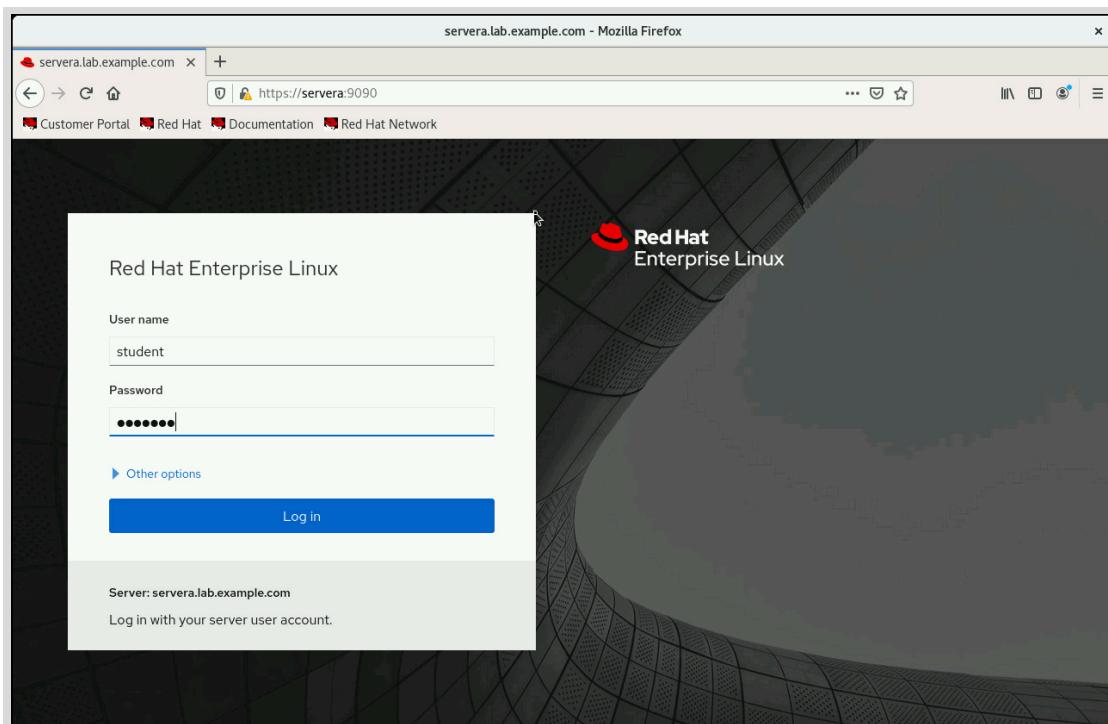


Figure 2.7: web console login page

- 6.2. Review the alerts that are reported in the overview.

Chapter 2 | Configuring Baseline Data

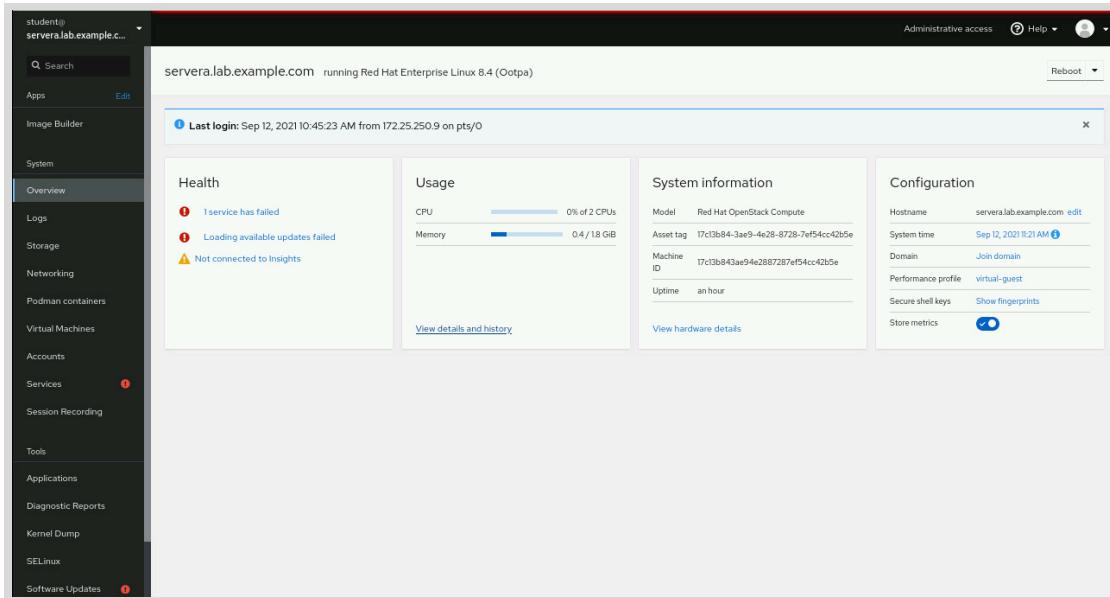


Figure 2.8: web console overview

- 6.3. Inspect the historical performance of the system. Click the **View details and history** link in the **Usage** box.

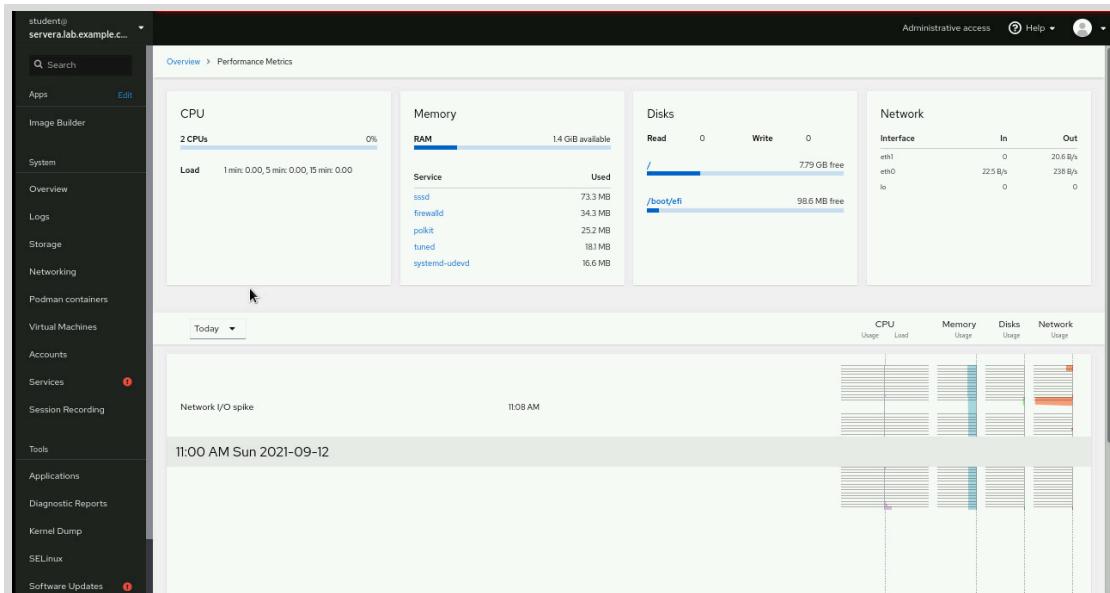


Figure 2.9: web console performance metrics

- 7. Configure the terminal session recording of the consultant user.

- 7.1. Click **Session Recording** from the main menu, and then click the gear icon. Navigate to **SSSD Config** and select **Scope → Some**, and then define **consultant** as **Users and Groups**.

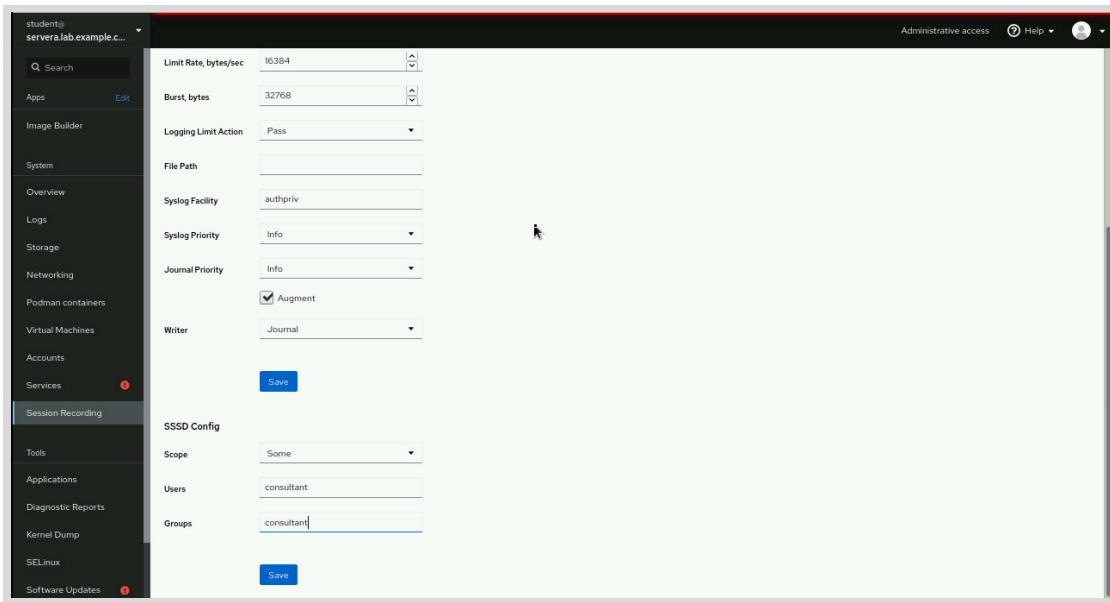


Figure 2.10: web console session recording

- 7.2. Click **Save** and click **Session Recording** at the top of the page.

- 8. Log in as the **consultant** user and perform some tasks on the **servera** host to be monitored.

- 8.1. Open a terminal and log in to **servera** as the **consultant** user.

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

- 8.2. Run various commands as the **consultant** user.

```
[consultant@servera ~]$ ps auxf
...output omitted...
[consultant@servera ~]$ top
...output omitted...
[consultant@servera ~]$ df -h
...output omitted...
```

- 8.3. Return to **workstation** as **student**.

```
[consultant@servera ~]$ exit
[student@workstation ~]$
```

- 9. Review the tasks that the **consultant** user performed in the web console.

- 9.1. In the web console of **servera**, click **Session Recording**.

- 9.2. Click the recorded terminal session of the **consultant** user.

The screenshot shows the Red Hat web console interface. On the left, a sidebar menu lists various system components: Apps, Image Builder, System, Overview, Logs, Storage, Networking, Podman containers, Virtual Machines, Accounts, Services, Session Recording (which is selected), Tools, Applications, Diagnostic Reports, Kernel Dump, SELinux, and Software Updates. The main content area displays session recording details for a user named 'consultant'. The session started at 2021-09-12 11:52:10 and ended at 2021-09-12 11:52:26, with a duration of 00:16. There are filter options for 'Since' (Filter since), 'Until' (Filter until), 'Search' (Filter by content), 'Username' (Filter by username), and a duration filter.

Figure 2.11: web console session recording

9.3. Play back the recorded session. Click Play.

Finish

On the **workstation** machine, use the **lab** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish baseline-monitoring
```

This concludes the guided exercise.

Configuring Remote Logging

Objectives

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

Remote logging

Standard configuration of system log management 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 for tasks that are executed just once a month, such as month-end financial closing. 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 useful for monitoring the state of systems and for identifying problems. Remote storage provides a backup location for log messages in case a system suffers a catastrophic failure that might cause the local logs to be unavailable. In these situations, the copy of the log messages on the central log host can be used to diagnose the issue that caused the problem.

Standardized system logging is implemented in Red Hat Enterprise Linux 8 by the `rsyslog` service. System programs can send syslog messages to the local `rsyslogd` service, which then redirects those messages to files in `/var/log`, remote log servers, or other databases based on the settings in its configuration file, `/etc/rsyslog.conf`.

Two characteristics of log messages categorize them: facility and severity. The facility indicates the type of log message. The severity indicates the importance of the event that is logged in the message.

Syslog Priority Levels

Severity	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

Implementing a central log host requires configuring the `rsyslog` service on the systems where the log messages originate and on the central log host that receives the messages. On the central log host, the `rsyslog` service is configured to accept log messages from the originating hosts.

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:

```
module(load="imudp")
input(type="imudp" port="514")
```

or for TCP reception:

```
module(load="imtcp")
input(type="imtcp" port="514")
```

In the example syntax, the `module` object loads the input modules for TCP and UDP, and the `input` object configures them. Although TCP provides more reliable delivery of remote log messages, UDP is supported by more operating systems and networking devices.

The default rules in `/etc/rsyslog.conf` accommodate message logging on a single host, sorting and bundling messages by facility. For example, mail messages are stored in `/var/log/maillog` while messages generated by the `cron` daemon are stored in `/var/log/cron` to easily locate each type of message.

Although sorting of messages by facility is ideal on a single host, it produces an undesirable result on a central log host because it causes messages from different remote hosts to be mixed together. On a central log host, it might be optimal for log messages from remote systems to remain separated. This separation is achieved by using the template function of `rsyslog`.

Templates are defined in `/etc/rsyslog.conf` by using the `template()` statement and are used to generate dynamic log files. A `template()` statement must contain a name and a `type` parameter. Different template types use different syntax, but still have the same functionality. Template parameters depend on the template type. Templates can dynamically construct log file names and paths based on the properties of a log message.

For example, to route `cron` syslog messages from different systems to different files on a central log host, use a template to generate dynamic log directories based on the `hostname` property of each log message, as in this example:

```
template (name="HostTemplate" type="string" string="/var/log/loghost/%HOSTNAME%/
cron.log")
```

Alternatively, use the `list` type parameter to specify a consistent and preferred syntax:

```
template(name="HostTemplate" type="list") {
    constant(value="/var/log/loghost/")
    property(name="hostname")
    constant(value="/cron.log")
}
```

**Note**

A list of template types is available in the references. The `list` type is unique because its configuration is defined with curly braces.

A template can be referenced by name in a rule:

```
cron.* action(type="omfile" DynaFile="HostTemplate")
```

This example rule indicates that log messages of the `cron` facility type, of any severity level, should be handled by the `action` object. The `action` object uses the File Output Module, `omfile`, to write to the template that the `DynaFile` parameter indicates.

The following template example generates dynamic log file names. Remote log messages are sorted by originating host name and facility values by referencing the `hostname` and `syslogfacility-text` properties. Log messages are written to the dynamically generated log file names.

```
template (name="HostTemplate" type="string" string="/var/log/loghost/%HOSTNAME%/
%syslogfacility-text%.log")
*.* action(type="omfile" DynaFile="HostTemplate")
```

**Note**

A list of the syslog message properties for `rsyslog` are found in the Available Properties section of the `rsyslog.conf(5)` man page.

After activating syslog message reception and creating the rules for log separation by host, restart the `rsyslog` service to implement the changes. In addition, configure appropriate UDP or TCP firewall rules to permit incoming syslog traffic, and reload `firewalld` to implement those rules.

```
[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
```

Newly created log files might not be included in the log host's current log rotation schedule, which can cause those new logs to grow too large. Include the new log files in the log rotation by adding a new entry to the list of log files in the `/etc/logrotate.d/rsyslog` configuration file.

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

Redirecting Logging to Central Log Host

After configuring the central log host to accept incoming remote messages, configure the `rsyslog` service on originating systems to send logs remotely to the central log host. Add entries to the rules section in the `/etc/rsyslog.conf` file to indicate the facility, severity, and an `action` object to specify how to communicate with the central log host.

For example, to send messages with `info` or higher severity that are sent to `loghost.example.com` via UDP, use this entry:

```
*.info action(type="omfwd" target="loghost.example.com" port="514" protocol="udp")
```

The `action` object uses the Forwarding Output Module, `omfwd`, to send syslog messages to the central log host.

To send *all* messages to `loghost.example.com` via TCP, use the following entry:

```
*.* action(type="omfwd" target="loghost.example.com" port="514" protocol="tcp")
```

After adding rules, restart the `rsyslog` service. Use the `logger` command to send a test message, which can optionally specify the facility and severity to test.

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

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

Export Session Recordings to Central Log Host

Sessions that are recorded with the web console tool can be exported to a central log host for centralization. Use the `systemd export` option to forward *encrypted* journal entries over the network with HTTP/HTTPS and to write journal files from the serialized journal content.

This option is integrated into `systemd`, by installing the `systemd-journal-remote` package.

```
[root@host ~]# yum install systemd-journal-remote
```

Create a temporary directory where the journal output with all its entities are exported into a file.

```
[root@host ~]# mkdir /tmp/dir  
[root@host ~]# journalctl -o export | /usr/lib/systemd/systemd-journal-remote  
-o /tmp/dir/example.journal -
```

In the central log host, create a directory to store the files of the nodes with terminal session recordings and copy or rsync them securely to the directory.

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

To playback the recorded sessions, use the tool included in the `tlog` package.

```
[root@host ~]# tlog-play -r journal -M TLOG_REC=<your-unique-host-id>
```

Automate the installation of the terminal session recording tool with the RHEL `tlog` system role. Sessions are sent to the central log host for further analysis.



References

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

For further information, refer to the *Recording Sessions Guide* at
https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/recording_sessions/index

► Guided Exercise

Configuring Remote Logging

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

Outcomes

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

Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your systems for this exercise.

```
[student@workstation ~]$ lab start baseline-remotelogging
```

This command confirms that your systems are reachable from the `workstation` machine.

Instructions

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

Create a rule to organize syslog messages into subdirectories under `/var/log/loghost`.

Create subdirectories by using the originating host name of each syslog message. Within each subdirectory, maintain a separate log file for each syslog facility. Configure log rotation for the new log files.

Configure `serverb` for remote logging to the central log host. Test the configuration by generating syslog messages from `serverb`. Verify that the generated messages are routed to the appropriate log file on `servera`.

- 1. Log in to `servera` and switch to the root user.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

- 2. Verify that the `rsyslog` service is running and enabled to start at boot.

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

- 3. Configure the `rsyslog` service on `servera` to accept remote syslog messages with TCP, and that the messages are written to separate files that are named for the originating host and message facility.

- 3.1. Enable TCP syslog reception in `/etc/rsyslog.conf` by uncommenting these lines in `/etc/rsyslog.conf`.

```
module(load="imtcp")
input(type="imtcp" port="514")
```

- 3.2. In the "##### RULES #####" section, add a `template()` statement to organize log messages by host name and syslog facility. Below the `template()` statement, create a rule to apply the template to all syslog messages.

```
template(name="ExerciseTemplate" type="string"
        string="/var/log/loghost/%HOSTNAME%/%syslogfacility-text%.log")
*.* action(type="omfile" DynaFile="ExerciseTemplate")
```



Note

A `template()` statement can span multiple lines for readability, if the statement's parameters are contained within the `()` parentheses.

- 3.3. To apply the configuration changes, restart the `rsyslog` service.

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

- 4. Add this entry to the `/etc/logrotate.d/syslog` file to add the new log files to the log rotation schedule.

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

- 5. Modify the `servera` firewall to allow receiving incoming syslog messages from remote hosts on TCP port 514.

- 5.1. Allow incoming packets on TCP port 514.

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

- 5.2. To apply the change, reload `firewalld`.

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

- 6. From `workstation`, open a new terminal. Login to `serverb` and switch to the `root` user.

```
[student@workstation ~]$ ssh student@serverb  
...output omitted...  
[student@serverb ~]$ sudo -i  
[sudo] password for student: student  
[root@serverb ~]#
```

- 7. Configure serverb to send syslog messages remotely to servera with the TCP protocol.

- 7.1. Create this rule in /etc/rsyslog.conf to send syslog messages to servera with TCP port 514.

```
*.* action(type="omfwd" target="servera" port="514" protocol="tcp")
```

- 7.2. To apply the change, restart the rsyslog service on serverb.

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

- 8. Verify that remote logging from serverb to the central log host on servera is working.

- 8.1. On serverb, generate some syslog messages that use different facilities.

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

- 8.2. On servera, verify that the syslog messages routed to the correct files in 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  
[root@servera ~]# grep 'cron\..crit' /var/log/loghost/serverb/cron.log  
Dec 11 00:44:40 serverb root: Test cron.crit message
```

- 9. Close the additional terminal session to serverb and return to workstation as the student user.

```
[root@servera ~]# exit  
[student@servera ~]$ exit  
[student@workstation ~]$
```

Finish

On the workstation machine, use the lab command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish baseline-remotelogging
```

This concludes the guided exercise.

Describing Configuration Management Automation

Objectives

After completing this section, you should be able to describe configuration management with Red Hat Satellite and Red Hat Ansible Automation Platform.

Configuration Management Solutions

As an organization's IT infrastructure grows, system configuration becomes unmanageable when still performed manually. Manual system administration fails to scale, introduces increased human errors, and might negatively impact system stability and function.

A recommended alternative is to use a configuration management solution. Configuration management tools can greatly enhance operational stability and efficiency, and expedite the deployment of configuration changes to many systems. Because changes are automated and programmatic, the accuracy of those changes is ensured.

Red Hat Satellite Server

Red Hat Satellite is a system management solution that deploys, configures, and maintains systems across physical, virtual, and cloud environments. Red Hat Satellite provides provisioning, remote management, and monitoring of multiple Red Hat Enterprise Linux deployments with a centralized tool. A Satellite Server organizes lifecycle management by using organizations as principal division units. Organizations isolate content for groups of hosts with specific requirements and administration tasks.

Red Hat Ansible Automation Platform

Organizations with broad configuration management requirements might also find Red Hat Ansible Automation Platform suitable. Ansible Automation Platform is a simple automation language that can describe a general IT infrastructure by using playbook tasks. Ansible Automation Platform is based on Python, is agentless, and relies on SSH connections to push configurations to designated remote systems.

Ansible Automation Platform includes an automation controller (formerly known as Red Hat Ansible Tower), which adds a dashboard and graphical tools to the solution. The automation controller can integrate server inventory from Red Hat Satellite, and can host and provide configuration roles and playbooks for Red Hat Satellite to launch.

Using Ansible Automation Platform for Configuration Management

System administrators can use Ansible Automation Platform to implement *Infrastructure as Code* by using a descriptive language to configure systems, instead of using individual, customized tasks. Ansible Automation Platform uses YAML syntax as its playbook language, which makes playbooks simple to read and write.

Playbooks can manage remote system configuration and deployment. Playbooks can also sequence multi-tier rollouts that involve rolling updates; can delegate actions to other hosts; and can interact with multiple host types while running tasks. Playbooks not only manage

configurations, but can also orchestrate any ordered process, even when process steps execute and communicate on and between multiple hosts.

Because Ansible Automation Platform is agentless, the only requirement for communication between hosts is to connect via SSH with user accounts with sufficient permissions for the requested tasks.

Ansible Automation Platform Architecture

The following nodes are defined for configuration management processes:

Control node

A host where Ansible Automation Platform is installed, from which playbooks are launched to execute tasks.

Managed node

Hosts that are reachable by the control node, where the launched tasks are running.

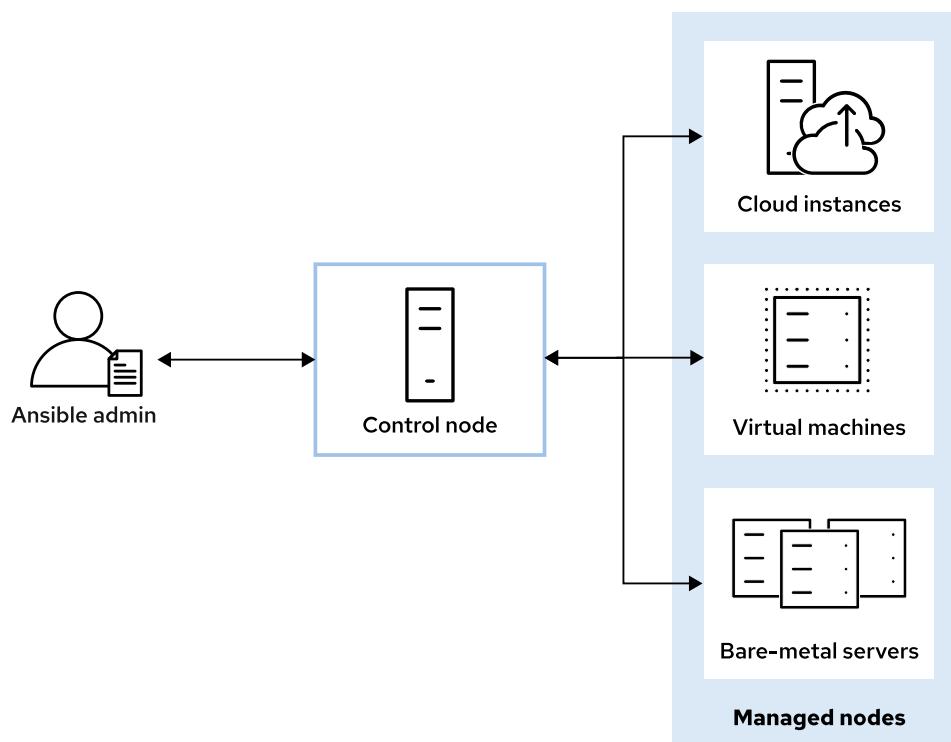


Figure 2.12: Ansible Automation Platform nodes

The control node stores the configuration files:

- `ansible.cfg`. Ansible Automation Platform configuration file.
- *Inventory*. A list of managed nodes.
- *Playbooks, modules, or roles*. Units or sets of defined configuration tasks.

Customized Configuration

Ansible Playbooks retrieve or discover variables, called **facts**, which contain information from the managed system, and that influence some behavior or state change on the system. For example,

the discovered IP address from one system can be used as a service configuration value in another system.

Run this example ad hoc command to verify the available variables on `server.example.com`:

```
[user@host ~]$ ansible server -m ansible.builtin.setup
```

**Note**

To reach managed nodes, the control node must know the hostname, short name, or IP address of each node. In the control node inventory, aliases can be also used for identification and for grouping managed hosts.

Facts are variable data that are stored in memory by default. Ansible Automation Platform uses a memory cache plug-in to maintain facts in memory for the duration of the current playbook run.

The *Jinja2* templating system uses the facts as substitution variables in configuration files. *Jinja2* templating enables dynamic expressions and access to variables, as in this example:

```
- name: Template example configuration file
  template:
    source: file.cfg.j2
    dest: {{ remote_install_path }}/file.cfg
```

The `remote_install_path` variable defines a file location, which might vary from one system to another.

A playbook can disable fact gathering, with this syntax:

```
- hosts: servers
  gather_facts: no
```

RHEL System Roles

RHEL System Roles is a collection of Ansible Roles and modules to manage system configurations across multiple versions of RHEL. Because roles are consistent, they are useful for troubleshooting by creating a known-good configuration for comparison. Roles can be used to test problem cause hypotheses, or to eliminate common configuration problems that cause other service failures. For example, improper time synchronization is a common root cause for network application failure in secure environments.

Red Hat Enterprise Linux 8 includes these roles by default:

`certificate`

Manages TLS/SSL certificate issuance and renewal.

`crypto_policies`

Manages system-wide crypto security policies.

`ha_cluster`

Manages High Availability (Corosync) Clustering.

`kdump`

Replaces the kdump configuration of the managed host.

kernel_settings

Modifies kernel settings, with `tuned`.

logging

Provisions and configures the logging system, with `rsyslog`.

metrics

Configures performance analysis services for the managed host.

nbde_client, nbde_server

Configures Network-Bound Disk Encryption clients with `clevis`, and Network-Bound Disk Encryption servers with `tang`.

network

Configures the network on target machines. Non-privileged users can use the role to configure different types of interfaces, such as:

- Ethernet
- Bridge
- Bond
- VLAN
- MacVLAN
- InfiniBand
- Wireless (WiFi)

Non-privileged users can also use the role for other network configurations, such as:

- IP address
- 802.1x authentication

postfix

Provides `postfix.conf` Postfix configuration dictionary with key/value parameters.

selinux

Manages local customization:

- Set enforcing modes
- Manage and restore file contexts
- Manage Booleans, logins, and ports

ssh, sshd

Manages SSH client configuration and configures the OpenSSH server daemon.

storage

Configures local storage with minimal input. Non-privileged users can use this role to manage file systems and mount entries from different types of devices, such as:

- Unpartitioned disks
- Lvm (unpartitioned whole-disk physical volumes only)

timesync

Manages an NTP or PTP implementation to operate as an NTP or PTP client for system clock synchronization.

tlog

Configures a system for Terminal session recording, by configuring `tlog` to log recording data to the `systemd` journal.

To use RHEL System Roles, verify that the `rhel-system-roles` package, which is available from the AppStream repository, is installed on the controller node:

```
[root@host ~]# yum install rhel-system-roles
```

To implement an Ansible Role, include a `roles` task in an Ansible Playbook:

```
- hosts: servers
  roles:
    - rhel-system-roles.network
    - rhel-system-roles.timesync
```

On the control node, Ansible Roles are hosted in the `/usr/share/ansible/roles` directory, with a directory for each role with all the information and requirements to use the role. Every role includes Markdown and HTML documentation files. Some roles include use case templates and examples, such as for the `certificate` role:

```
[user@host ~]# tree /usr/share/ansible/roles/linux-system-roles.certificate
/usr/share/ansible/roles/linux-system-roles.certificate
├── ansible_pytest_extra_requirements.txt
├── custom_requirements.txt
├── defaults
│   └── main.yml
└── examples
    ├── basic_self_signed.yml
    ├── dns_ip_and_email.yml
    ├── key_usage_and_extended_key_usage.yml
    ├── many_self_signed.yml
    ├── no_auto_renew.yml
    ├── not_wait_for_certificate.yml
    ├── owner_and_group.yml
    ├── principal.yml
    ├── subject.yml
    └── wrong_provider.yml
└── library
    └── certificate_request.py
└── LICENSE
└── meta
    └── main.yml
└── module_utils
    └── certificate_lsr
        ├── init.py
        └── providers
            ├── base.py
            ├── certmonger.py
            └── init.py
└── README.html
└── README.md
...output omitted...
```

The `README` file describes the role's use cases, parameters, templates, example syntax, and structure. To view the HTML version, use your browser to open the file from the command line:

```
[user@host ~]# cd /usr/share/ansible/roles
[user@host roles]# firefox linux-system-roles.certificate/README.html
```

Table of Contents

Certificate System Role

- Variables
- CAs and Providers
- Examples
- Compatibility
- Dependencies
- License
- Author Information

Certificate System Role

Role for managing TLS/SSL certificate issuance and renewal

Linux system role to issue and renew SSL certificates.

Basic usage:

```

---
- hosts: webserver

  vars:
    certificate_requests:
      - name: mycert
        dns: www.example.com
        ca: self-sign

  roles:
    - rhel-system-roles.certificate
  
```

On a RPM-based system this will place the certificate in `/etc/pki/tls/certs/mycert.crt` and the key in `/etc/pki/tls/private/mycert.key`.

Figure 2.13: RHEL System Role certificates README page



References

For further information, refer to the *Red Hat Satellite* product page at
<https://access.redhat.com/products/red-hat-satellite/#knowledge>

For further information, refer to *Configuring Satellite to Use Ansible* at
https://access.redhat.com/documentation/en-us/red_hat_satellite/6.9/html-single/configuring_satellite_to_use_ansible/index

For further information, refer to the *Working With Playbooks* section in *Ansible Documentation* at
https://docs.ansible.com/ansible/2.9/user_guide/playbooks.html

For further information, refer to *Administration and Configuration Tasks Using System Roles in RHEL* at
https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/administration_and_configuration_tasks_using_system_roles_in_rhel/index

► Guided Exercise

Implementing Configuration Changes with an Ansible Playbook

In this lab, you configure a system with Ansible as a configuration manager.

Outcomes

You should be able to install and configure the web service on the system by using Ansible.

Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

```
[student@workstation ~]$ lab start baseline-configmanagement
```

This command confirms that the required hosts for this exercise are accessible and gets the templates for the web service configuration.

Instructions

In this exercise, you configure `workstation` as an Ansible control node, and then install and configure the web service on the `servera` managed node.

- 1. On `workstation` as student user, create the `workdir` directory. Change directory to it.

```
[student@workstation ~]$ mkdir workdir  
[student@workstation ~]$ cd workdir
```

- 2. Create an `ansible.cfg` configuration file.

```
[defaults]  
inventory = inventory  
remote_user = root  
host_key_checking = False  
deprecation_warnings = False
```

- 3. Create the `inventory` file and set `servera` as the `web_prod` production web server.

```
[webservers]  
web_prod ansible_host=servera
```

- 4. Verify connectivity with the `web_prod` managed node.

```
[student@workstation workdir]$ ansible web_prod -m ping
web_prod | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false,
    "ping": "pong"
}
```

- 5. Move the downloaded templates to the `workdir` directory and view the included substitution variables.

```
[student@workstation workdir]$ mv ~/apache_* .
[student@workstation workdir]$ grep "{{" apache_httpdconf.j2
# {{ ansible_managed }}
[student@workstation workdir]$ grep "{{" apache_indexhtml.j2
<!-- {{ ansible_managed }} -->
Hello from {{ inventory_hostname }}
```

- 6. Create the `mywebserver.yaml` Ansible Playbook for the web service configuration, to include these tasks:

- Install `httpd` package if not installed.
- Use the templates as the web service configuration files.
- Ensure that the firewall service is enabled for the web service.

- 6.1. Set the managed hosts to configure. It is not necessary to gather facts for these tasks.

```
- name: Install and configure a customized web server
hosts: webservers
gather_facts: False
```

- 6.2. Configure tasks to verify that the `httpd` service is the latest version, and verify the presence of the `firewalld` service.

```
- name: Install httpd package
ansible.builtin.yum:
  name: httpd
  state: latest

- name: Validate firewall
ansible.builtin.yum:
  name: firewalld
  state: present
```

- 6.3. Configure tasks to inject templates as the `httpd` configuration files.

```

- name: Template out httpd configuration file
  template:
    src: apache_httpdconf.j2
    dest: /etc/httpd/conf/httpd.conf
    owner: root
    group: root
    mode: '0444'

- name: Template out httpd index file
  template:
    src: apache_indexhtml.j2
    dest: /var/www/html/index.html
    owner: root
    group: root
    mode: '0444'

```

- 6.4. Configure tasks to ensure that the `httpd` and `firewalld` services are active and enabled at system startup.

```

- name: Start and enable httpd daemon
  ansible.builtin.service:
    name: httpd
    state: started
    enabled: true

- name: Start and enable firewalld daemon
  ansible.builtin.service:
    name: firewalld
    state: started
    enabled: true

```

- 6.5. Configure a task to ensure that the `http` service is allowed in the firewall.

```

- name: Open http firewalld port
  firewalld:
    service: http
    immediate: yes
    permanent: yes
    state: enabled

```

- 6.6. Verify that the final Ansible Playbook `mywebserver.yaml` contains this content:

```

- name: Install and configure a customized web server
  hosts: webservers
  gather_facts: False

  tasks:
    - name: Install httpd package
      ansible.builtin.yum:
        name: httpd
        state: latest

```

```

- name: Validate firewall
ansible.builtin.yum:
  name: firewalld
  state: present

- name: Template out httpd configuration file
template:
  src: apache_httpdconf.j2
  dest: /etc/httpd/conf/httpd.conf
  owner: root
  group: root
  mode: '0444'

- name: Template out httpd index file
template:
  src: apache_indexhtml.j2
  dest: /var/www/html/index.html
  owner: root
  group: root
  mode: '0444'

- name: Start and enable httpd daemon
ansible.builtin.service:
  name: httpd
  state: started
  enabled: true

- name: Start and enable firewalld daemon
ansible.builtin.service:
  name: firewalld
  state: started
  enabled: true

- name: Open http firewalld port
firewalld:
  service: http
  immediate: yes
  permanent: yes
  state: enabled

```

**Note**

You can verify the syntax of an Ansible Playbook without implementing it.

```
[student@workstation workdir]$ ansible-playbook --syntax-check
mywebserver.yaml

playbook: mywebserver.yaml
```

- 7. Run the Ansible Playbook and verify its successful execution.

```
[student@workstation workdir]$ ansible-playbook mywebserver.yaml
PLAY [Install and configure a customized web server] ****
TASK [Install httpd package] ****
changed: [web_prod]

TASK [Validate firewall] ****
ok: [web_prod]

TASK [Template out httpd configuration file] ****
changed: [web_prod]

TASK [Template out httpd index file] ****
ok: [web_prod]

TASK [Start and enable httpd daemon] ****
changed: [web_prod]

TASK [Start and enable firewalld daemon] ****
ok: [web_prod]

TASK [Open http firewalld port] ****
changed: [web_prod]

PLAY RECAP ****
web_prod : ok=7      changed=5      unreachable=0      failed=0      skipped=0
             rescued=0     ignored=0
```

- 8. Verify that the web server can be accessed.

```
[student@workstation workdir]$ curl servera
<!-- ansible managed -->
<html>
<head><title>Apache is running!</title></head>
<body>
<h1>
Hello from web_prod
</h1>
</body>
</html>
[student@workstation workdir]$ cd ~
```

Finish

On the workstation machine, use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish baseline-configmanagement
```

This concludes the guided exercise.

Configuring Change Tracking

Objectives

After completing this section, you should be able to implement change tracking to monitor system modifications.

Using Intrusion Detection Software for Change Monitoring

Advanced Intrusion Detection Environment (AIDE) is a tool that compares and reports file and directory modifications. Configure AIDE to monitor files for changes including to permissions or ownership, timestamps (modification or access), or file contents. When changes are detected, a log entry is created in the `/var/log/aide/aide.log` file.

Installing AIDE

Install AIDE from the `aide` RPM package. The `aide` package provides the `aide` utility, configuration files, and documentation on tuning to monitor specific changes of interest.

```
[root@host ~]# yum install aide
```

Configuring AIDE

The `/etc/aide.conf` file is the primary configuration file for AIDE. It has three types of configuration directive: configuration, selection, and macro.

Configuration

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. By default, AIDE provides the following group definitions in the `/etc/aide.conf` file:

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

This 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

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 its subdirectories. The second line performs the group of checks on `/dir2`. The equal sign indicates to check the directory only and not to recurse below it. The third line excludes from checks `/dir3` and everything below it.

Macro

Macro lines define variables, with this syntax:

```
@@define VAR value
```

`@@{VAR}` refers to the defined macro.

The `aide.conf` configuration file can include comments, which start with the `#` character.

Manually Deploying AIDE

After configuring the `aide.conf` file, initialize the AIDE database, and initially scan the file system.

Initializing the AIDE Database

To initialize the AIDE database, use the `aide --init` command.

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

Initializing AIDE creates the database in a `/var/lib/aide/aide.db.new.gz` file. Rename the file to `/var/lib/aide/aide.db.gz`, by removing the `.new` string, to create the expected database name for the AIDE service.

```
[root@host ~]# mv /var/lib/aide/aide.db.new.gz /var/lib/aide/aide.db.gz
```



Warning

Because users with local system access might manipulate the database, recommended security practice is to move or copy the database to a secured location between validation runs. To perform an AIDE check, the database must be restored or compared to the secure archived version to detect any tampering.

Checking for File System Changes

Use the `aide --check` command to check a file system. After the database is created, you can check a file system by running the `aide --check` command. This command scans the file system and compares the current state of files with the information in the AIDE database. Any differences are displayed to the terminal, and show both the original file state and the new condition.

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



Note

Several applications or services can create, touch, or modify many files. For example, the `prelink` cron job creates dynamic library links to launch executable file launching faster. AIDE captures these changes in an AIDE database, with the impression that the binaries changed. Either disable the `prelink` cron job, if appropriate, or learn to recognize non-intrusive file changes.

Automating File Integrity Alerts

Create a shell script that checks the `/var/log/aide/aide.log` file for alerts and then emails selected users.

```
if ! grep "Looks okay" /var/log/aide/aide.log &>/dev/null
then
    cat /var/log/aide/aide.log | /usr/bin/mail -s "AIDE Alert" user@host
fi
```

To automate the file system check, create the `/etc/cron.d/aide` cron job with the following content:

```
# AIDE Checks and alert
*/2 * * * * root /sbin/aide --check & /root/aide_mon.sh
```

Manually Updating the AIDE Database

Use the `aide --update` command to update the AIDE database with wanted changes to files.

```
[root@host ~]# aide --update
```

For each database update, you must rename the new `/var/lib/aide/aide.db.new.gz` file to `/var/lib/aide/aide.db.gz` for the next run of an AIDE check. After completing the update, run the `aide --check` command to verify that the database and the file system match.

Update the database with wanted changes:

```
[root@host ~]# aide --update
```



Note

It is recommended to back up the previous database file before renaming the new database file.

Create a backup copy of the current database:

```
[root@host ~]# mv /var/lib/aide/aide.db.gz /var/lib/aide/aide.db.old.gz
```

Rename the new database by removing the `.new` substring:

```
[root@host ~]# mv /var/lib/aide/aide.db.new.gz /var/lib/aide/aide.db.gz
```

Verify that no differences exist between the current database and the file system:

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

Using auditd for System Auditing

The `auditd` daemon is the user-space component of the Linux auditing subsystem. When `auditd` is running, kernel-generated audit messages are collected in the `auditd` file, by default `/var/log/audit/audit.log`. If `auditd` is not running for any reason, kernel audit messages are sent to `rsyslog`.

By default, limited messages are passed through the audit system, such as authentication and authorization messages when users log in or use `sudo`, and SELinux messages. Security administrators can use `auditctl` to add auditing rules for any system call, including file operations. By default, auditing rules are added to the bottom of the current list. Rules can be inserted to manage rule order.



Important

Because audit rules work on a first-match-wins basis, ordering is important for rules to function as intended. For example, if one rule logs access to all files in the `/etc` directory and a later rule logs access to all files in the `/etc/sysconfig` directory, then the later rule would never be reached because the earlier rule was already triggered for any access at or below the `/etc` directory. By switching the order of these two rules, both rules would work as intended.

A watch rule is one of the most basic rules. Watch rules are set on files and directories, and can be configured for specific types of access, such as read, write, attribute change, and execute. Watches are triggered on the initial open system call, which requests the access type, and not on the subsequent, repetitive `read` or `write` calls to the same open file.

Some examples of watches:

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

This line adds a watch rule for `/etc/passwd` write and attribute change access, and adds a custom `user-edit` key to all log messages.

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

This line adds a recursive watch to the `/etc/sysconfig` directory and to all files and directories beneath it. Logs read, write, and attribute change access, and add a custom `sysconfig-access` key to all log messages.

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

This line audits the execution of binaries under `/bin`, and does not add a key phrase to the log messages.

Removing Rules

Use the uppercase `-W` option to remove a file system watch rule. Do not confuse the uppercase `W` option with the lowercase `-w` option, which is used to *write* the original rule. Use the lowercase `-d` option to remove rules that were added with lowercase `-a` or uppercase `-A`.

Use the `auditctl -D` command to remove all rules.

Persistent Rules

To make auditing rules persistent, add them to the `/etc/audit/rules.d/audit.rules` file. This file contains `auditctl` command syntax as entered on the command line, without the `auditctl` command itself. Empty lines are permitted, and comment lines start with the hash character (#).

When the `auditd` service is started, it activates all rules from `/etc/audit/rules.d/audit.rules`. The `auditctl` command can load rules from a file with the `-R` option.



Important

By design, `auditd` cannot be reloaded with `systemctl restart auditd`, because this action would log the credentials of the kernel (PID 1) as the process that sends the kill signal. Instead, use `service auditd restart` to reload changes to persistent rules. The service command intentionally logs the UID of the user who initiated the action.

Reading Audit Messages

Audit messages that are logged to `/var/log/audit/audit.log` contain significant information. The fields, which are each labeled, differ by message types. The following example is a typical audit record from `/var/log/audit/audit.log`.

```
type=SYSCALL ①
msg=audit(1371716130.596:28708) ②
: arch=c000003e syscall=2 success=yes exit=4 a0=261b130 a1=90800 a2=e a3=19
items=1 ppid=2548 pid=26131 auid=500 ③
uid=0 gid=0 euid=0 ④
suid=0 fsuid=0 egid=0 sgid=0 fsgid=0 tty=pts0 ses=1 comm="aureport" ⑤
exe="/sbin/aureport" ⑥
subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 key="audit-access" ⑦
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
```

- ① The type of audit message. In this example, a system call was audited.
- ② The timestamp and ID of the audit message. The number before the colon (in this case, 1371716130.596) is a timestamp in seconds since the epoch. The number after the colon (28708) is the audit event number. To convert a timestamp from epoch time to local time zone, use the `date --date=@<epoch-time>` command. A single event might trigger multiple lines of logging, with different types.
- ③ The original, or Audit, UID of the user who triggered this audit message. The audit system tracks the original UID even when a user elevates their privileges through `su` or `sudo`.
- ④ The Effective UID of the user who triggered this audit message. The Effective UID might be different from the Audit UID, and even from the regular UID; for example, when running a binary with the SUID or SGID bit set.
- ⑤ The executable name that triggered this audit message, as reported by `ps` or `top`.
- ⑥ The name of the stored binary file for the process that triggered this audit message.

- 7 An identifier to search for events. Custom auditing rules typically set keys to help to filter for event types.

Searching for Events

The auditing system includes the ausearch audit log searching tool. Not only does ausearch easily search for and filter various event types, it can also interpret events by translating numeric values into readable values such as user names or system call names. Common ausearch options are listed here, and more options are available to search for events based on user, terminal, or virtual machine.

Option	Description
-i	Interpret log line, and translate numeric values into names.
--raw	Print raw log entries, without record separators between entries.
-a <EVENT-ID>	Show all lines for the event with this event ID.
--file <FILENAME>	Include all events for a specific file name.
-k <KEY>	Search for all events that are labeled with a key.
--start [start-date] [start-time]	Include only events after the date and time. If no start time is included, then assume <code>midnight</code> . Omitting a starting date assumes <code>today</code> .

The appropriate time format depends on your system's current locale. Other acceptable time values include `recent` (past 10 minutes), `this-week`, `this-month`, and `this-year`. The `--end` time option specifies all remaining audit entries.

In this example, ausearch displays an interpreted version of the previous example, filtered by the `audit-access` key value.

```
[root@host ]# 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 uid=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 about `aide`, see the `aide(1)` and `aide.conf(5)` man pages.

For more information about `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 configure and perform system auditing with the Linux audit system. In this exercise, you configure and perform system auditing with the Linux audit system.

Outcomes

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

Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

```
[student@workstation ~]$ lab start baseline-changetracking
```

Instructions

- 1. Log in to servera and switch to the root user.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

- 2. Configure AIDE to monitor file integrity.

- 2.1. Initialize the AIDE database.

```
[root@servera ~]# aide -i
Start timestamp: 2021-09-14 22:00:07 -0400 (AIDE 0.16)
AIDE initialized database at /var/lib/aide/aide.db.new.gz

Number of entries: 5

-----
The attributes of the (uncompressed) database(s):
-----

/var/lib/aide/aide.db.new.gz
MD5      : LLSaHUKYsWzCDA94nbLurQ==
SHA1     : mv+HidAoRw+kp1NC2vLgapVKdJg=
RMD160   : CzX5WCHz8L5UXb4Amzx5CJvtzsw=
TIGER    : 1T8DFsev/GgTMj/gY8MExRvLJvGoWXT
SHA256   : reYx+x0uZyataCNPvgksaLfu0GA1rL0G
dKEpxfQybL0=
```

```
SHA512 : xlxRXa1reUuGVhP1QxVUoVvtBULc2skU  
tpNI5imXj/XeIF8LwJbAop8CF/JTi7SE  
mAnwzY2919NeaialwaOUTg==
```

```
End timestamp: 2021-09-14 22:00:07 -0400 (run time: 0m 0s)
```

- 2.2. Remove the new substring from the initial database name to enable it for the aide command to use.

```
[root@servera ~]# mv /var/lib/aide/aide.db.new.gz /var/lib/aide/aide.db.gz
```

- 3. View the /etc/aide.conf file to learn the default monitored files and rules.

```
[root@servera ~]# cat /etc/aide.conf  
...output omitted...  
# Custom Rules  
# Extended content + file type + access.  
CONTENT_EX = sha512+ftype+p+u+g+n+acl+selinux+xattrs  
  
# trusted databases  
/etc/hosts$ CONTENT_EX  
/etc/issue$ CONTENT_EX  
/etc/passwd$ CONTENT_EX  
/etc/group$ CONTENT_EX  
/etc/shadow$ CONTENT_EX
```

- 4. Generate a file integrity report for the default monitored files and verify that the report is present in the logs.

- 4.1. Generate the file integrity report.

```
[root@servera ~]# aide --check  
Start timestamp: 2021-09-14 22:10:19 -0400 (AIDE 0.16)  
AIDE found NO differences between database and filesystem. Looks okay!!
```

```
Number of entries: 5
```

```
-----  
The attributes of the (uncompressed) database(s):  
-----
```

```
/var/lib/aide/aide.db.gz  
MD5 : LLSaHUKYsWzCDA94nbLurQ==  
SHA1 : mv+HidAoRw+kp1NC2vLgapVKdJg=  
RMD160 : CzX5WCHz8L5UXb4Amzx5CJvtzsw=  
TIGER : 1T8DFsev/GgTMj/gY8MExRvLJVvGoWXT  
SHA256 : reYx+xOuZyataCNPvgksaLfu0GA1rLOG  
          dKEpxfQybL0=  
SHA512 : xlxRXa1reUuGVhP1QxVUoVvtBULc2skU  
          tpNI5imXj/XeIF8LwJbAop8CF/JTi7SE  
          mAnwzY2919NeaialwaOUTg==
```

```
End timestamp: 2021-09-14 22:10:19 -0400 (run time: 0m 0s)
```

- 4.2. Verify that the file integrity report is recorded in the log file.

```
[root@servera ~]# cat /var/log/aide/aide.log
Start timestamp: 2021-09-14 22:10:19 -0400 (AIDE 0.16)
AIDE found NO differences between database and filesystem. Looks okay!!

Number of entries: 5

-----
The attributes of the (uncompressed) database(s):
-----

/var/lib/aide/aide.db.gz
MD5      : LLSaHUKYsWzCDA94nbLurQ==
SHA1     : mv+HidAoRw+Kp1NC2vLgapVKdJg=
RMD160   : CzX5WCHz8L5UXb4Amzx5CJvtzsw=
TIGER    : 1T8DFsev/GgTMj/gY8MEXrVLJVvGoWXT
SHA256   : reYx+x0uZyataCNPvg skaLf u0GA1rLOG
           dKEpxfQybL0=
SHA512   : xlxRXa1reUuGVhP1QxVUoVvtBULc2skU
           tpNI5imXj/XeIF8LwJbAop8CF/JTi7SE
           mAnwzY2919NeaialwaoUTg==
```

```
End timestamp: 2021-09-14 22:10:19 -0400 (run time: 0m 0s)
```

- 5. Create a script to send an email alert whenever a change in the monitored files is found.

```
[root@servera ~]# vim /root/aide_mon.sh
if ! grep "Looks okay" /var/log/aide/aide.log &>/dev/null
then
  cat /var/log/aide/aide.log | /usr/bin/mail -s "AIDE Alert" \
  student@servera.lab.example.com
fi
```

Make the script executable.

```
[root@servera ~]# chmod +x /root/aide_mon.sh
```

- 6. Schedule the AIDE execution for every two minutes. Create the /etc/cron.d/aide file and add the following lines:

```
[root@servera ~]# vim /etc/cron.d/aide
# AIDE Checks and alert
*/2 * * * * root /sbin/aide --check & /root/aide_mon.sh
```

- 7. Verify that the integrity report is recorded in the log file. Check that no differences exist between the database and the file system.

```
[root@servera ~]# cat /var/log/aide/aide.log
Start timestamp: 2021-09-14 22:18:01 -0400 (AIDE 0.16)
AIDE found NO differences between database and filesystem. Looks okay!!

Number of entries: 5

-----
The attributes of the (uncompressed) database(s):
-----

/var/lib/aide/aide.db.gz
MD5      : LLSaHUKYsWzCDA94nbLurQ==
SHA1     : mv+HidAoRw+kp1NC2vLgapVKdJg=
RMD160   : CzX5WCHz8L5UXb4Amzx5CJvtzsw=
TIGER    : 1T8DFsev/GgTMj/gY8MExRvLJVvGoWXT
SHA256   : reYx+xOuZyataCNPvgksaLfuOGA1rLOG
          dKEpxfQybL0=
SHA512   : xlxRXa1reUuGVhP1QxVUoVvtBULc2skU
          tpNI5imXj/XeIF8LwJbAop8CF/JT17SE
          mAnwzY2919NeaialwaOUTg==

End timestamp: 2021-09-14 22:18:01 -0400 (run time: 0m 0s)
```

- 8. Add a user, which causes a change in the systems account database, which is a monitored file. This action generates an inconsistency in file integrity.

```
[root@servera ~]# useradd user01
```

- 9. As the **student** user on the **servera** machine, verify that an AIDE alert email is received. The first email might take up to five minutes to arrive. The complete log report might not finish processing until the second email is received. Optionally, you could monitor the email activity that is sent to the **student** user.

**Note**

Log in to the **servera** machine in a separate terminal. Use the **journalctl** command to monitor the email that is sent to the **student** user.

```
[root@servera ~]# journalctl -f -u postfix --since today
```

- 9.1. Switch to the **student** user on the **servera** machine to check for AIDE alert emails.

```
[root@servera ~]# exit
[student@servera ~]$
```

- 9.2. Check the email.

```
[student@servera ~]$ mail
Heirloom Mail version 12.5 7/5/10. Type ? for help.
"/var/spool/mail/student": 1 message 1 new
>N 1 root           Tue Sep 14 22:54 62/2250 "AIDE Alert"
```

Read the most recent email.

```
& 1
Message 1:
From root@servera.lab.example.com Tue Sep 14 22:54:22 2021
Return-Path: <root@servera.lab.example.com>
From: root <root@servera.lab.example.com>
Date: Tue, 14 Sep 2021 22:54:01 -0400
To: student@servera.lab.example.com
Subject: AIDE Alert
User-Agent: Heirloom mailx 12.5 7/5/10
Content-Type: text/plain; charset=us-ascii
Status: R
```

```
Start timestamp: 2021-09-14 22:52:01 -0400 (AIDE 0.16)
AIDE found differences between database and filesystem!!
```

Summary:

```
Total number of entries: 5
Added entries: 0
Removed entries: 0
Changed entries: 3
```

Changed entries:

```
f ... .C... : /etc/group
f ... .C... : /etc/passwd
f ... .C... : /etc/shadow
```

...output omitted...

Quit the mail interactive interface.

```
& q
...output omitted...
[student@servera ~]$
```

- ▶ 10. Switch to the `root` user on the `servera` machine. Update the AIDE database to add the recent file changes. Back up the AIDE database. Remove the new substring from the updated database name to enable it for the `aide` command to use.

- 10.1. Switch to the `root` user.

```
[student@servera ~]$ sudo -i  
[sudo] password for student: student  
[root@servera ~]#
```

10.2. Update the database.

```
[root@servera ~]# aide --update
```

10.3. Back up the current database.

```
[root@servera ~]# mv /var/lib/aide/aide.db.gz /var/lib/aide/aide.db.old.gz
```

10.4. Enable the new database for the aide command to use.

```
[root@servera ~]# mv /var/lib/aide/aide.db.new.gz /var/lib/aide/aide.db.gz
```

► 11. Generate an integrity report of the monitored files.

```
[root@servera ~]# aide --check  
Start timestamp: 2021-09-14 23:23:06 -0400 (AIDE 0.16)  
AIDE found NO differences between database and filesystem. Looks okay!!  
  
Number of entries: 5  
  
-----  
The attributes of the (uncompressed) database(s):  
-----  
  
/var/lib/aide/aide.db.gz  
MD5      : /tJ6rCYNS+gHPpAqP+cdFQ==  
SHA1     : v0sEePADH6VbXAFPuGrrVG+tQNM=  
RMD160   : UDvr4Gvti4Layc8imAlHhAgZIWo=  
TIGER    : VCSYuYBw98r81861ip0iVVE37mdp03g1  
SHA256   : 9wyBh7ir0dLWeacsPHKRHYPLYKbt0zND  
          42xuIn4/I3w=  
SHA512   : nw8mnWzPdlodiPVHtgcPOH1jcrDuVnXg  
          wt9zFVBrAr8nFURYEnm12QFt7snGn03n  
          mx7M8Dj5RBydOMetEyoSaA==  
  
End timestamp: 2021-09-14 23:23:06 -0400 (run time: 0m 0s)
```

► 12. Switch to the student user on the servera machine. Verify that an alert email is no longer sent to the student user. Verify that the integrity report is recorded in the log file.

12.1. Return to the student user.

```
[root@servera ~]# exit  
[student@servera ~]$
```

- 12.2. Check email to confirm that no new alerts are present.

The total number of messages varies. If the database is correctly updated, then no new messages are present.

```
[student@servera ~]$ mail  
>N 12 root Tue Sep 14 23:22 21/970 "AIDE Alert"
```

- 12.3. Use the `sudo cat` command to view the log file to verify that no differences are found between the database and file system.

```
[student@servera ~]$ sudo cat /var/log/aide/aide.log  
Start timestamp: 2021-09-14 23:26:01 -0400 (AIDE 0.16)  
AIDE found NO differences between database and filesystem. Looks okay!!  
  
Number of entries: 5  
  
-----  
The attributes of the (uncompressed) database(s):  
-----  
  
/var/lib/aide/aide.db.gz  
MD5      : /tJ6rCYNS+gHPpAqP+cdFQ==  
SHA1     : v0sEePADH6VbXAFPuGrrVG+tQNM=  
RMD160   : UDvr4Gvti4Layc8imAlHhAgZIWo=  
TIGER    : VCSYuYBw98r81861ip0iVVE37mdp03g1  
SHA256   : 9wyBh7ir0dLWeacsPHKRHYPLYKbt0zND  
          42xuIn4/I3w=  
SHA512   : nw8mnWzPdlodiPVHtgcPOH1jcrDuVnXg  
          wt9zFVBrAr8nFURYEnm12QFt7snGn03n  
          mx7M8Dj5RByd0MetEyoSaA==  
  
End timestamp: 2021-09-14 23:26:01 -0400 (run time: 0m 0s)
```

- 13. Return to workstation as student.

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

Finish

On the `workstation` machine, use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish baseline-changetracking
```

This concludes the guided exercise.

► Lab

Configuring Baseline Data

In this lab, you configure systems for monitoring, remote logging, and auditing with Ansible Automation Platform as the configuration manager.

Outcomes

You should be able to configure servers as an Ansible Automation Platform control node and managed nodes. You should also be able to configure the servers as a central and remote log server with a file access audit system.

Before You Begin

As the student user on the **workstation** machine, use the **lab** command to prepare your system for this exercise.

```
[student@workstation ~]$ lab start baseline-review
```

This command confirms that the required hosts for this exercise are accessible and creates the **workdir** directory to contain the basic minimum configuration files for the activities to perform.

Instructions

Configure **workstation** as a control node, and verify communication with **servera** and **serverb** as managed nodes. Use Ansible Playbooks to configure the managed nodes:

servera

- Configure the **rsyslog** service to serve as a central log host.
- Create a rule that writes the syslog messages that each host generates to separate files under the **/var/log/loghost** directory.
- Name each log file for the originating host of the messages.
- Add the new log files to the log rotation schedule for size management.
- Configure terminal session recording for the **consultant** user.

serverb

- Install AIDE to report file and directory changes.
 - Configure **serverb** to send log messages to the central log host.
1. Configure **workstation** as a control node and **servera** and **serverb** as managed nodes. Verify that the control node can reach both.
 2. Configure terminal session recording for the **consultant** user.
 3. Use an Ansible Playbook to configure **servera** as a central log host. Use the provided configuration template.

4. Use an Ansible Playbook to configure `serverb` as a remote log server, and configure AIDE to monitor file integrity. Use the provided configuration templates for each service.
5. Run the playbook, and then view remote logging and file integrity on `serverb`.

Evaluation

On the `workstation` machine, use the `lab` command to grade your work. Correct any reported failures and rerun the script until you receive a passing grade.

```
[student@workstation ~]$ lab grade baseline-review
```

Finish

On the `workstation` machine, use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish baseline-review
```

This concludes the lab.

► Solution

Configuring Baseline Data

In this lab, you configure systems for monitoring, remote logging, and auditing with Ansible Automation Platform as the configuration manager.

Outcomes

You should be able to configure servers as an Ansible Automation Platform control node and managed nodes. You should also be able to configure the servers as a central and remote log server with a file access audit system.

Before You Begin

As the student user on the **workstation** machine, use the **lab** command to prepare your system for this exercise.

```
[student@workstation ~]$ lab start baseline-review
```

This command confirms that the required hosts for this exercise are accessible and creates the **workdir** directory to contain the basic minimum configuration files for the activities to perform.

Instructions

Configure **workstation** as a control node, and verify communication with **servera** and **serverb** as managed nodes. Use Ansible Playbooks to configure the managed nodes:

servera

- Configure the **rsyslog** service to serve as a central log host.
- Create a rule that writes the syslog messages that each host generates to separate files under the **/var/log/loghost** directory.
- Name each log file for the originating host of the messages.
- Add the new log files to the log rotation schedule for size management.
- Configure terminal session recording for the **consultant** user.

serverb

- Install AIDE to report file and directory changes.
 - Configure **serverb** to send log messages to the central log host.
- Configure **workstation** as a control node and **servera** and **serverb** as managed nodes. Verify that the control node can reach both.
 - On **workstation**, change to the **workdir** directory and review the configuration files.

```
[student@workstation ~]$ cd workdir
[student@workstation workdir]$ cat ansible.cfg
[defaults]
inventory = inventory
remote_user = root
host_key_checking = False
deprecation_warnings = False
[student@workstation workdir]$ cat inventory
[servers]
central_localhost ansible_host=servera
```

- 1.2. Add serverb as remote_localhost in the inventory file.

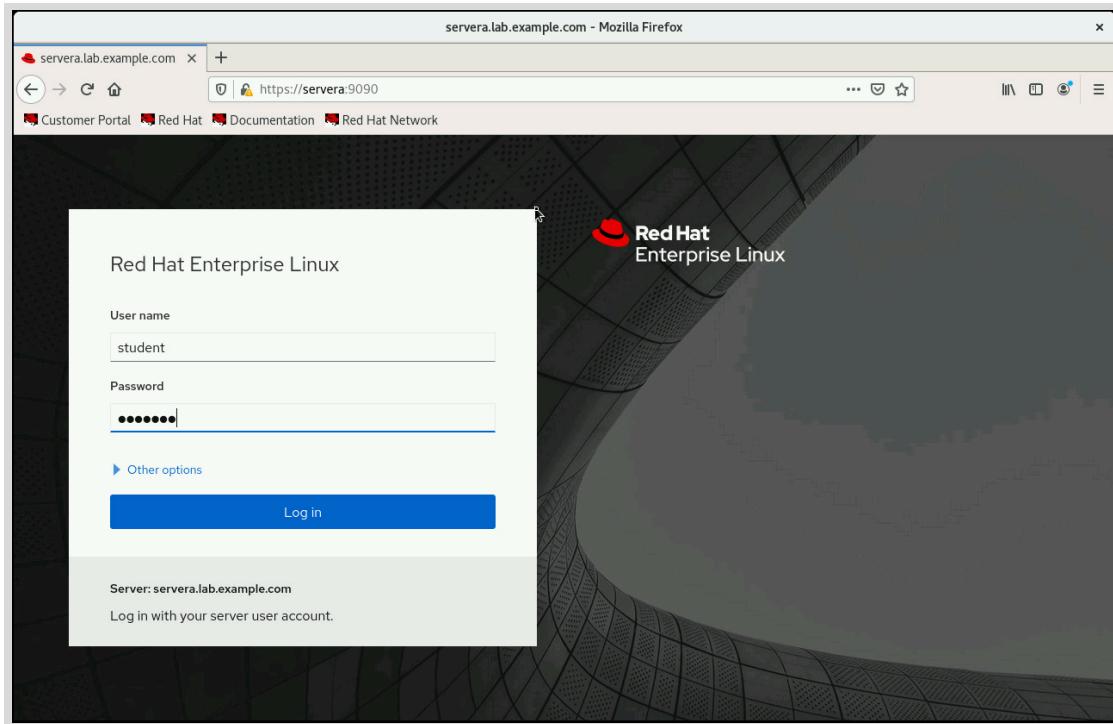
```
[servers]
central_localhost ansible_host=servera
remote_localhost ansible_host=serverb
```

- 1.3. Confirm that the managed nodes are reached.

```
[student@workstation workdir]$ ansible all -m ping
central_localhost | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false,
    "ping": "pong"
}
remote_localhost | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false,
    "ping": "pong"
}
```

2. Configure terminal session recording for the consultant user.

- 2.1. On workstation, open a web browser and navigate to <https://servera:9090>. Log in with user name student and password student.



- 2.2. From the left menu, click **Session Recording**, and then click the gear icon.

Navigate to **SSSD Config** and select **Scope → Some**. Define **consultant** as **Users** and **Groups**.

The screenshot shows the 'Session Recording' configuration screen. The 'SSSD Config' section is active, displaying the following settings:

- Scope: Some
- Users: consultant
- Groups: consultant

At the bottom of the screen, there are two 'Save' buttons.

- 2.3. Click **Save**, and then at the top of the page click **Session Recording**.

- 2.4. On workstation, open another terminal and log in to **servera** as the **consultant** user.

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

- 2.5. Run some commands as the **consultant** user, such as `ps auxf`, `top`, and `df -h`.

```
[consultant@servera ~]$ ps auxf
...output omitted...
[consultant@servera ~]$ top
...output omitted...
[consultant@servera ~]$ df -h
...output omitted...
```

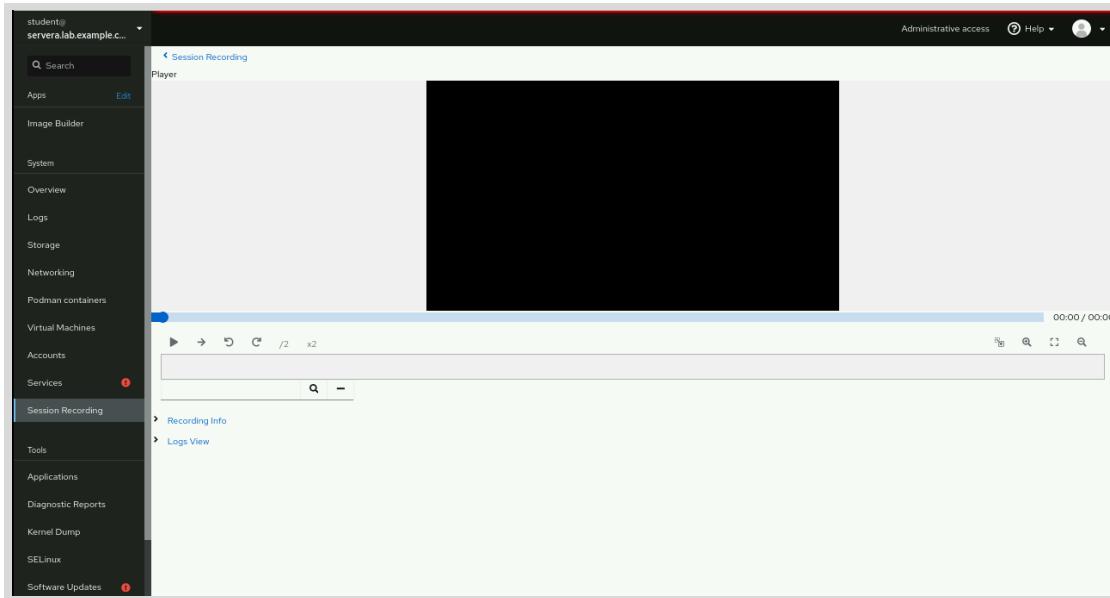
- 2.6. Return to **workstation** as **student**.

```
[consultant@servera ~]$ exit
[student@workstation workdir]$
```

- 2.7. Confirm the terminal session recording. In the web console of **servera**, click **Session Recording**. Refresh the page if necessary.
- 2.8. Click the **consultant** user's recorded terminal session.

User	Start	End	Duration
consultant	2021-09-12 11:52:10	2021-09-12 11:52:26	00:16

- 2.9. Play back the recorded session by clicking **Play**.



3. Use an Ansible Playbook to configure `servera` as a central log host. Use the provided configuration template.

- 3.1. Inside the `workdir` directory, create the `mybaseline.yaml` Ansible Playbook for the `rsyslog` service configuration on `servera`.

```
- name: Configure central loghost
  hosts: central_loghost
  gather_facts: False
```

- 3.2. Confirm that the `rsyslog` service is active and enabled at system startup.

```
- name: Ensure that rsyslog is active and enabled
  ansible.builtin.service:
    name: rsyslog
    state: started
    enabled: True
```

- 3.3. Use the `rsyslog.conf.j2` template file as the `rsyslog` service configuration file to allow remote logging.

```
- name: Template out rsyslog configuration file
  template:
    src: rsyslog.conf.j2
    dest: /etc/rsyslog.conf
    owner: root
    group: root
    mode: '0444'

- name: Restart rsyslog service
  ansible.builtin.service:
    name: rsyslog
    state: restarted
```

3.4. Ensure that the `firewalld` service is enabled for the `rsyslog` service.

```
- name: Open rsyslog firewalld port
  firewalld:
    port: 514/tcp
    immediate: yes
    permanent: yes
    state: enabled
```

3.5. Add the remote logs to the configured log rotation.

```
- name: Add entry for that the new logs have rotation
  lineinfile:
    path: /etc/logrotate.d/syslog
    line: /var/log/loghost/*/*.log
```

4. Use an Ansible Playbook to configure `serverb` as a remote log server, and configure AIDE to monitor file integrity. Use the provided configuration templates for each service.

4.1. Add a task to the `mybaseline.yaml` playbook to configure `serverb` as the remote log server.

```
- name: Configure remote logging
  hosts: remote_loghost
  gather_facts: False
```

4.2. Verify that the `rsyslog` service is active and enabled at system startup.

```
- name: Ensure that rsyslog is active and enabled
  ansible.builtin.service:
    name: rsyslog
    state: started
    enabled: True
```

4.3. Add an entry in the `/etc/rsyslog.conf` file to redirect logs to the central log host.

```
- name: Add entry for redirecting logs
  lineinfile:
    path: /etc/rsyslog.conf
    line: '*.* action(type="omfwd" target="servera" port="514" protocol="tcp")'

- name: Restart rsyslog service
  ansible.builtin.service:
    name: rsyslog
    state: restarted
```

4.4. Add a task to the `mybaseline.yaml` playbook to configure AIDE on `serverb`.

```
- name: Install AIDE
  hosts: remote_loghost
  gather_facts: False
```

- 4.5. Ensure that the `aide` package is installed. Use the `aide.conf.j2` template file as the `aide` service configuration file.

```
- name: Ensure that AIDE package is installed
ansible.builtin.yum:
  name: aide
  state: present

- name: Template out AIDE configuration file
  template:
    src: aide.conf.j2
    dest: /etc/aide.conf
    owner: root
    group: root
    mode: '0444'
```

- 4.6. Initialize the `aide` database to start monitoring configured files.

```
- name: Init AIDE database
ansible.builtin.command: aide --init

- name: Enable AIDE database
  ansible.builtin.command: mv /var/lib/aide/aide.db.new.gz /var/lib/aide/
aide.db.gz
```

5. Run the playbook, and then view remote logging and file integrity on `serverB`.

- 5.1. Verify that the final Ansible Playbook `mybaseline.yaml` contains this content:

```
- name: Configure central loghost
hosts: central_loghost
gather_facts: False

tasks:
  - name: Ensure that rsyslog is active and enabled
    ansible.builtin.service:
      name: rsyslog
      state: started
      enabled: True

  - name: Template out rsyslog configuration file
    template:
      src: rsyslog.conf.j2
      dest: /etc/rsyslog.conf
      owner: root
      group: root
      mode: '0444'

  - name: Restart rsyslog service
    ansible.builtin.service:
      name: rsyslog
      state: restarted

  - name: Open rsyslog firewalld port
```

```
firewalld:
  port: 514/tcp
  immediate: yes
  permanent: yes
  state: enabled

  - name: Add entry for that the new logs have rotation
    lineinfile:
      path: /etc/logrotate.d/syslog
      line: /var/log/loghost/*/*.log

  - name: Configure remote logging
    hosts: remote_loghost
    gather_facts: False

  tasks:
    - name: Ensure that rsyslog is active and enabled
      ansible.builtin.service:
        name: rsyslog
        state: started
        enabled: True

    - name: Add entry for redirecting logs
      lineinfile:
        path: /etc/rsyslog.conf
        line: '.* action(type="omfwd" target="servera" port="514"
protocol="tcp")'

    - name: Restart rsyslog service
      ansible.builtin.service:
        name: rsyslog
        state: restarted

  - name: Install AIDE
    hosts: remote_loghost
    gather_facts: False

  tasks:
    - name: Ensure that AIDE package is installed
      ansible.builtin.yum:
        name: aide
        state: present

    - name: Template out AIDE configuration file
      template:
        src: aide.conf.j2
        dest: /etc/aide.conf
        owner: root
        group: root
        mode: '0444'

    - name: Init AIDE database
      ansible.builtin.command: aide --init
```

```
- name: Enable AIDE database
  ansible.builtin.command: mv /var/lib/aide/aide.db.new.gz /var/lib/aide/
  aide.db.gz
  tags: enable_aidedb
```

5.2. Run the playbook and verify that it finishes successfully.

```
[student@workstation workdir]$ ansible-playbook mybaseline.yaml
PLAY [Configure central loghost] ****
TASK [Ensure that rsyslog is active and enabled] ****
ok: [central_loghost]

TASK [Template out rsyslog configuration file] ****
changed: [central_loghost]

TASK [Restart rsyslog service] ****
changed: [central_loghost]

TASK [Open rsyslog firewalld port] ****
ok: [central_loghost]

TASK [Add entry for that the new logs have rotation] ****
ok: [central_loghost]

PLAY [Configure remote logging] ****
TASK [Ensure that rsyslog is active and enabled] ****
ok: [remote_loghost]

TASK [Add entry for redirecting logs] ****
changed: [remote_loghost]

TASK [Restart rsyslog service] ****
changed: [remote_loghost]

PLAY [Install AIDE] ****
TASK [Ensure that AIDE package is installed] ****
changed: [remote_loghost]

TASK [Template out AIDE configuration file] ****
changed: [remote_loghost]

TASK [Init AIDE database] ****
changed: [remote_loghost]

TASK [Enable AIDE database] ****
changed: [remote_loghost]

PLAY RECAP ****
central_loghost      : ok=5    changed=2    unreachable=0    failed=0
remote_loghost       : ok=7    changed=6    unreachable=0    failed=0
```

**Note**

Output may vary depending on changes or re-run of the Ansible Playbook.

- 5.3. On **workstation**, open another terminal and log in to **serverb** as the consultant user.

```
[student@workstation workdir]$ ssh consultant@serverb
[consultant@serverb ~]$
```

- 5.4. Generate some syslog messages with different log facilities and priorities. Then, return to **workstation** as student.

```
[consultant@serverb ~]$ logger -p user.info "Test user.info message from serverb"
[consultant@serverb ~]$ logger -p cron.crit "Test cron.crit message from serverb"
[consultant@serverb ~]$ exit
[student@workstation workdir]$
```

- 5.5. Log in to **servera** and use **sudo** to verify the remote logging. Use **student** as password when prompted.

```
[student@workstation workdir]$ ssh student@servera
[student@servera ~]$ sudo grep consultant /var/log/localhost/serverb/user.log
[sudo] password for student: student
Sep 22 00:09:37 serverb consultant[39113]: Test user.info message from serverb
[student@servera ~]$ sudo grep consultant /var/log/localhost/serverb/cron.log
Sep 22 00:09:56 serverb consultant[39202]: Test cron.crit message from serverb
```

- 5.6. Return to **workstation** as student.

```
[student@servera ~]$ exit
[student@workstation workdir]$
```

- 5.7. ssh to **serverb**. Use **sudo** to verify the integrity of the monitored files. Enter **student** as password.

```
[student@workstation workdir]$ ssh student@serverb
[student@serverb ~]$ sudo aide --check
[sudo] password for student: student
Start timestamp: 2021-09-22 00:15:27 -0400 (AIDE 0.16)
AIDE found NO differences between database and filesystem. Looks okay!!

Number of entries: 5

-----
The attributes of the (uncompressed) database(s):
-----

/var/lib/aide/aide.db.gz
MD5      : Q6Ba4uEAwnVokSR5IoGcGA==
```

Chapter 2 | Configuring Baseline Data

```
SHA1      : gtNsinq9mM5ljKC0tsjdyJRVIobw=
RMD160    : DhFPdM+jfYw7uHqnNRmwbEwQbxY=
TIGER     : IvUi93X18F+0/nkkHBhgrpc0/EobIvVc
SHA256    : 0a6zINluSdksWHL4jFLkqT7XI6waQgrS
           xt+TmWcl61w=
SHA512    : 65bpzDil3IYXMOjzb4zepETkfW9r906k
           hmiBKGXoE3aPjS2F8e0c/smbUiU9sc9/
           NFyNOKhXZtNC65WN6fDxdQ==
```

```
End timestamp: 2021-09-22 00:15:27 -0400 (run time: 0m 0s)
```

- 5.8. Return to **workstation** as the **student** user and change to its home directory.

```
[student@serverb ~]$ exit
[student@workstation workdir]$ cd ~
[student@workstation ~]$
```

Evaluation

On the **workstation** machine, use the **lab** command to grade your work. Correct any reported failures and rerun the script until you receive a passing grade.

```
[student@workstation ~]$ lab grade baseline-review
```

Finish

On the **workstation** machine, use the **lab** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish baseline-review
```

This concludes the lab.

Summary

In this chapter, you learned:

- The web console provides system monitoring, administration, and session recording through a GUI.
- The `rsyslog` service enables centralized local and remote logging.
- The Red Hat Ansible Automation Platform improves the efficiency and stability of system deployments and configuration changes.
- The `aide` utility tracks file and directory attributes to facilitate detecting unauthorized changes.
- The `auditctl` command manages rules to select which security events are collected in the audit log.
- The `ausearch` command queries the binary audit log and displays messages in readable form.

Chapter 3

Troubleshooting Boot Issues

Goal

Identify and resolve issues that can affect a system's ability to boot.

Objectives

- Repair boot issues on BIOS systems.
- Repair boot issues on UEFI systems.
- Identify and resolve service failures that affect the boot process.
- Reset the root password.

Sections

- Resolving Boot Loader Issues on BIOS Systems (and Guided Exercise)
- Resolving Boot Loader Issues on UEFI Systems (and Quiz)
- Identifying and Resolving Failing Services (and Guided Exercise)
- Resetting the root Password (and Guided Exercise)

Lab

Troubleshooting Boot Issues

Resolving Boot Loader Issues on BIOS Systems

Objectives

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

Describing the BIOS Firmware Interface

The *Basic Input/Output System (BIOS)* is a firmware interface that controls the first step of the boot process and provides the lowest-level interface to external devices.

The BIOS checks the system and locates a bootable device with a Master Boot Record (MBR) partition to start the boot process. In Red Hat Enterprise Linux 8, the MBR boot loader use the *GRand Unified Boot loader version 2 (GRUB2)*. The BIOS loads the boot loader that is stored in the MBR into memory. The boot loader then loads and passes control to the GRUB2 boot menu.

Storing the Master Boot Record (MBR)

GRUB2 provides two options to boot from a disk on a BIOS system: 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.

Limitation of Using MBR

Using the MBR has a size limitation. A typical MBR has only 512 bytes and holds the disk's partition table, leaving only 446 bytes for the boot loader. To work around this limitation, GRUB2 can use the MBR gap, also known as the boot track or embedding area, where extra boot modules and files are stored. The MBR gap is the space between the MBR block and the first disk partition.

To work reliably for modern boot loaders, the MBR gap must be at least 31 KiB (63 sectors on a 512-byte sector disk). Disks prepared using anaconda, the RHEL installation utility, typically start the first partition at sector 2048, creating a 1 MiB MBR gap for storing GRUB2 elements.

Booting RHEL 8 BIOS Systems Using GRUB2

The boot loader is the first program executed when the system starts. It finds, loads, and transfers control to an operating system. GRUB2 can boot any compatible operating system or transfer control to other boot loaders for unsupported operating systems.

For systems with an existing operating system, the anaconda installer attempts to automatically detect and configure the boot loader to start another operating system. If the boot loader is not detected, then you can manually configure additional operating systems after finishing the installation. Partition requirements vary, depending on whether a system is BIOS or UEFI. Modern systems may also require GPT partitioning, which handles larger disks and offers more flexible configuration than MBR partitioning while still supporting MBR boot loaders. If a user selects automatic partitioning in anaconda, then the installer creates the required partition type with sufficient MBR gap sizing.

GRUB 2 reads its configuration from the `/boot/grub2/grub.cfg` file on traditional BIOS-based machines and from the `/boot/efi/EFI/redhat/grub.cfg` file on UEFI-based machines.

The GRUB2 utilities stores files on a BIOS system in various locations:

- The `/boot/` directory stores the kernel and initial ramdisk.
- The `/boot/grub2/` directory contains configuration files and extension modules.
- The `/boot/grub2/grub.cfg` file is the configuration file, with a symbolic link to `/etc/grub2.cfg`.
- The `/etc/grub.d/` directory contains a helper script to generate the `/boot/grub2/grub.cfg` file.
- The `/etc/default/grub` file contains variables used by the configuration helper script.
- The `/boot/grub2/grubenv` file is exactly 1 KiB, and stores variables such as the default or saved boot entry.

Describing the GRUB2 Configuration File

Typically, a system administrator does not manually configure the GRUB2 boot loader. During installation, a new kernel is added to the GRUB2 configuration file automatically. Similarly, the corresponding boot loader entry is removed when a kernel is removed. To temporarily change the GRUB2 menu at boot time, pass parameters to the kernel during startup using the GRUB2 menu. Some parameter options are available in `/etc/default/grub` file.

GRUB_TIMEOUT

The number of seconds that the GRUB2 menu is displayed before the default entry gets selected automatically.

GRUB_DEFAULT

The default entry that starts whenever a user does not select another entry. GRUB2 entries are counted starting with 0.

GRUB_CMDLINE_LINUX

This variable contains a list of extra kernel command-line options to add to every Linux kernel.

After updating the `/etc/default/grub` file, run the `grub2-mkconfig -o /boot/grub2/grub.cfg` command to apply the changes. This command uses the configuration helper scripts and settings to generate a new configuration file. The `-o` option is used to redirect the command's standard output to the named file.

The Red Hat Enterprise Linux 8 Boot Sequence

Booting Red Hat Enterprise Linux 8 on a physical x86_64 system is a sequential set of tasks. For x86_64 virtual machines, the hypervisor handles some of the hardware-specific steps.

1. When the machine powers on, the BIOS or UEFI system firmware runs a *Power On Self Test (POST)* and initializes the critical hardware.
2. The firmware searches for a bootable device, which is configured in the firmware, by searching for a *Master Boot Record (MBR)* on each disk.
3. The firmware searches for a boot loader on the bootable device and passes control to the boot loader. In a RHEL 8 system, the boot loader is the *GRand Unified Boot loader version 2 (GRUB2)*, installed with the `grub2-install` command.
4. GRUB2 loads its configuration from the `/boot/grub2/grub.cfg` file and displays a menu with options to select a kernel to boot.
5. The boot loader loads the kernel and the `initramfs` file system from the device into memory after selecting a kernel or when the timeout expires. The `initramfs` file system

- contains a minimal but complete set of critical drivers and kernel modules necessary to bootstrap the system.
6. The boot loader passes control to the kernel, with specified kernel command line options and the memory location of the `initramfs` file system.
 7. The kernel initializes critical hardware using drivers in the `initramfs` file system and then executes `/sbin/init` from the `initramfs` file system as PID 1. In RHEL 8, `/sbin/init` is a link to `systemd`.
 8. The `systemd` daemon executes all units for the `initrd.target` and mounts the configured root file system on the `/sysroot` directory.
 9. The kernel switches (pivots) the root file system from the `initramfs` file system to the root file system in `/sysroot`. The `systemd` daemon re-executes itself with the `systemd` copy from the root file system.
 10. The `systemd` daemon activates the default target, which is either passed from the kernel command line or from the target unit linked to `/etc/systemd/system/default.target`. Systemd starts or stops units to comply with the target's configuration, and solves dependencies between units automatically.
 11. Systemd targets that behave as run levels (with an `AllowIsolate=true` parameter) will activate its units and display a text-based login prompt or a graphical login screen as the result of the intended system state. The system is now booted and ready for user login.

Updating Boot Loader Configuration Files

The Boot Loader Specification (BLS) manages boot loader configuration without manipulating boot loader configuration files, by defining a schema and file format.

With BLS, each boot entry references a single configuration file in the drop-in directory. This drop-in architecture extends the configuration without needing to edit or regenerate the main boot loader configuration files.

BLS also uses this concept to configure boot menu entries. Manage the boot loader menu options by adding, removing, or editing individual boot entry files in the drop-in directory. BLS significantly simplifies the kernel installation process and is consistent across multiple architectures.

Generate BLS Configuration Files

1. Add the following entry in the `/etc/default/grub` file:

```
GRUB_ENABLE_BLSCFG=true
```

2. For a legacy BIOS system, generate the `/boot/grub2/grub.cfg` file.

```
[root@host ~]# grub2-mkconfig -o /boot/grub2/grub.cfg
Generating grub configuration file ...
done
```

3. Generate the `/boot/loader/entries`.

```
[root@host ~]# kernel-install add $(uname -r) /lib/modules/$(uname -r)/vmlinuz
```

Making Persistent Changes to the GRUB2 Boot Menu

The `grubby` utility manipulates boot loader-specific configuration files. `grubby` works as a wrapper for the BLS operations. Use `grubby` for changing the default boot entry, and for adding or removing arguments in a GRUB2 menu entry.

If `grubby` is invoked manually without specifying a GRUB 2 configuration file, it defaults to locating `grub2.cfg`.

Obtain the file name of the default kernel with `grubby`:

```
[root@host ~]# grubby --default-kernel
/boot/vmlinuz-4.18.0-305.el8.x86_64
```

Find the index number of the default kernel:

```
[root@host ~]# grubby --default-index
0
```

Use the `grubby` command to persistently change the default kernel:

```
[root@host ~]# grubby --set-default /boot/vmlinuz-4.18.0-305.el8.x86_64
The default is /boot/loader/entries/
ffffffffffffffffff-4.18.0-305.el8.x86_64.conf with index 0 and
kernel /boot/vmlinuz-4.18.0-305.el8.x86_64
```

List all the kernel menu entries:

```
[root@host ~]# grubby --info=ALL
index=0
kernel="/boot/vmlinuz-4.18.0-305.el8.x86_64"
args="ro no_timer_check net.ifnames=0 crashkernel=auto $tuned_params"
root="/dev/vda3"
initrd="/boot/initramfs-4.18.0-305.el8.x86_64.img $tuned_initrd"
title="Red Hat Enterprise Linux (4.18.0-305.el8.x86_64) 8.4 (Ootpa)"
id="ffffffffffffffffff-4.18.0-305.el8.x86_64"
```

View the GRUB 2 menu entry for a specific kernel:

```
[root@host ~]# grubby --info /boot/vmlinuz-4.18.0-305.el8.x86_64
index=0
kernel="/boot/vmlinuz-4.18.0-305.el8.x86_64"
args="ro no_timer_check net.ifnames=0 crashkernel=auto $tuned_params"
root="/dev/vda3"
initrd="/boot/initramfs-4.18.0-305.el8.x86_64.img $tuned_initrd"
title="Red Hat Enterprise Linux (4.18.0-305.el8.x86_64) 8.4 (Ootpa)"
id="ffffffffffffffffff-4.18.0-305.el8.x86_64"
```

Reinstalling the GRUB2 Boot Loader on the MBR

If the MBR or the MBR gap is damaged, then reinstall GRUB2 into the MBR to overwrite it. If a system is not running when this issue occurs, the repair must be performed from within a

Chapter 3 | Troubleshooting Boot Issues

Live CD or the *rescue* environment available from the *anaconda* installer. If a user with root permissions has access to a running system with this issue, the solution is to use the `grub2-install` command.

To handle any scenario, practice booting into a rescue environment to reinstall GRUB2 into the MBR:

1. Boot from an installation source; such as a RHEL DVD image, netboot CD, or a Preboot eXecution Environment (PXE) provided by a RHEL installation tree.
2. Append `inst.rescue` to the boot command line. Alternatively, select **Troubleshooting** and then the **Rescue a Red Hat Enterprise Linux system** option from the installation media menu, and press **Enter**.
3. From the menu, select option 1 to mount the file system under the `/mnt/sysroot` directory. To open a shell and obtain a prompt, press **Enter**.
4. Use `chroot` to switch the rescue mode environment to use the mounted root partition (`/mnt/sysroot`) of the system being repaired.

```
sh-4.4# chroot /mnt/sysroot
```

5. Reinstall the GRUB2 boot loader on the previously-used block device.

```
bash-4.4# /sbin/grub2-install /dev/vda
```

6. Reboot the system to verify that the GRUB2 boot loader is now operational.

Describing Early Kdump Support in RHEL 8

Older RHEL versions were unable to capture kernel crashes at early stages in the boot process, because the kdump service's dependencies required it to start later in the process. Without a kernel dump, it is difficult to troubleshoot early boot process problems.

RHEL 8 introduces early kdump support to address this issue. Early kdump support can log kernel crashes during kernel booting, by storing a *crash kernel* `vmlinuz` and `initramfs` inside the `initramfs` of the normal, booting kernel. Early kdump loads directly into reserved memory (`crashkernel`) during the early boot stage. Two `dracut` modules are provided in the `kexec-tools` package to load the crash kernel and `initramfs` as early as possible during the boot sequence to be available to capture a crash dump of the normal, booting kernel.



References

`systemd(1)`, `grub2-install(1)`, `grub2-mkconfig(1)`, `grubby(8)`, `kernel-install(8)`, and `grub2-editenv(1)` man and info pages

For further information, refer to *What is early kdump support and how do I configure it?* at

<https://access.redhat.com/solutions/3700611>

For further information, refer to *26.4. Making Persistent Changes to a GRUB 2 Menu Using the grubby Tool* at

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/system_administrators_guide/ch-working_with_the_grub_2_boot_loader#sec-Making_Persistent_Changes_to_a_GRUB_2_Menu_Using_the_grubby_Tool

For further information, refer to *G.10.3. Reinstalling the GRUB2 boot loader* at

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/performing_a_standard_rhel_installation#reinstalling-the-grub2-boot-loader_using-rescue-mode

For further information, refer to *How to generate BLS configuration files under /boot/loaders/entries in Red Hat Enterprise Linux 8?* at

<https://access.redhat.com/solutions/5847011>

► Guided Exercise

Resolving Boot Loader Issues on BIOS Systems

In this lab, you fix a GRUB2 boot issue on a BIOS-based machine.

Outcomes

You should be able to restore the GRUB2 boot menu on a BIOS-based system with the `grubby` command.

Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

```
[student@workstation ~]$ lab start boot-biosgrub
```

Instructions

The `servera` machine was updated to boot with an incorrect kernel. The incorrect boot entry has a label of **server**.

Change the default GRUB2 boot menu entry to use the kernel entry that does not have the **server** label.

- ▶ 1. Open the `servera` machine console. View the error message, which shows that the kernel is absent.

```
error: ../../grub-core/loader/i386/pc/linux.c:170:invalid magic number.  
error: ../../grub-core/loader/i386/pc/linux.c:421:you need to load the kernel  
first.
```

Press any key to continue...

- ▶ 2. Boot the `servera` machine with the kernel entry without the **server** label. You should see the login screen of the `servera` machine.

```
...output omitted...  
servera login:
```

- ▶ 3. Log in to `servera` as the student user. Switch to root and run the `grubby` command to query the boot entries.

```
servera login: student  
Password: student  
[student@servera ~]$ sudo -i
```

```
[sudo] password for student: student
[root@servera ~]# grubby --info=ALL
index=0
kernel="/boot/vmlinuz-server"
args="ro no_timer_check net.ifnames=0 crashkernel=auto"
root="/dev/vda3"
initrd="/boot/initramfs-server.img"
title="Red Hat Enterprise Linux (server) 8.4 (Ootpa)"
id="81a31fe4555b4603a2e19b6ec313f8b9-server.0-custom"
index=1
kernel="/boot/vmlinuz-server"
args="ro no_timer_check net.ifnames=0 crashkernel=auto"
root="/dev/vda3"
initrd="/boot/initramfs-server.img"
title="Red Hat Enterprise Linux (server) 8.4 (Ootpa)"
id="81a31fe4555b4603a2e19b6ec313f8b9-server"
index=2
kernel="/boot/vmlinuz-4.18.0-305.el8.x86_64"
args="ro no_timer_check net.ifnames=0 crashkernel=auto $tuned_params"
root="/dev/vda3"
initrd="/boot/initramfs-4.18.0-305.el8.x86_64.img $tuned_initrd"
title="Red Hat Enterprise Linux (4.18.0-305.el8.x86_64) 8.4 (Ootpa)"
id="ffffffffffffffffffff-4.18.0-305.el8.x86_64"
```

- 4. Query the default GRUB2 boot entry with the **grubby** command.

```
[root@servera ~]# grubby --default-kernel
/boot/vmlinuz-server
```

- 5. Change the default GRUB2 menu entry to use the kernel that does not have the **server** label.

```
[root@servera ~]# grubby --set-default=/boot/vmlinuz-4.18.0-305.el8.x86_64
The default is /boot/loader/entries/
ffffffffffffffffffff-4.18.0-305.el8.x86_64.conf with index 2 and
kernel /boot/vmlinuz-4.18.0-305.el8.x86_64
```

- 6. Verify that the default kernel is set correctly.

```
[root@servera ~]# grubby --default-kernel
/boot/vmlinuz-4.18.0-305.el8.x86_64
```

- 7. Reboot the system and verify that the default GRUB2 menu entry boots correctly and displays the login prompt.

```
[root@servera ~]# reboot
...output omitted...
servera login:
```

Finish

On the `workstation` machine, use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish boot-biosgrub
```

This concludes the guided exercise.

Resolving Boot Loader Issues on UEFI Systems

Objectives

After completing this section, you should be able to repair boot issues on UEFI systems.

Describing the UEFI Firmware Interface

Introduced in 2005, the *Unified Extensible Firmware Interface* (UEFI) replaces the original *Extensible Firmware Interface* (EFI) and the legacy BIOS. UEFI overcomes the limits of BIOS, such as 16-bit processor mode and 1 MiB of addressable space. UEFI supports booting from larger disks than 2 TiB, if the disk and its partitions use a *GUID Partition Table* (GPT) format.

UEFI systems and BIOS systems boot differently. A BIOS system must search for bootable operating systems on devices, while UEFI systems register bootable operating systems in the UEFI firmware. Users of UEFI systems can more easily select an operating system in a multiboot environment.

Using GRUB2 to Boot UEFI Systems

To boot a UEFI system with GRUB2, an *EFI System Partition* (ESP) must be present on the installed disk. It is recommended to use GPT partitioning for UEFI systems, although you can still use MBR partitioning. The ESP partition must be formatted with a FAT file system and be large enough to store the boot loader and all of the bootable kernels for operating systems that are installed on this system. The recommended size is 512 MiB, but a smaller size might be sufficient. The ESP partition is mounted at `/boot/efi` and is typically created by the anaconda installer.

The UEFI Boot Chain

A Secure Boot-capable system uses signatures on boot loaders, kernels, and other boot objects. The system firmware verifies the signatures as authentication that only unmodified code is being loaded, with a component that is known as the shim UEFI boot loader.

The `shim.efi` application is signed with a key that is trusted by common UEFI firmware. When executed, the `shim.efi` application attempts to load `grubx64.efi` with standard UEFI firmware calls. If the UEFI firmware cannot load the code because the signature does not authenticate, then the shim attempts to verify the application with other keys.

These alternative keys are either compiled into the shim, or read from a user-generated key that is stored in the UEFI NVRAM as the *Machine Owner Key* (MOK). When attempting to start a signed application for which no keys are registered, the `MokManager.efi` application starts automatically so that an end-user can register a personal key.

The `shim.efi` application registers an additional UEFI system call that uses GRUB2 to verify kernel signatures. If `shim.efi` is not registered with the UEFI firmware, then booting from the ESP partition executes the `/boot/efi/EFI/BOOT/BOOTX64.efi` application. This application loads 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`.

Main GRUB2 Difference Between BIOS and UEFI

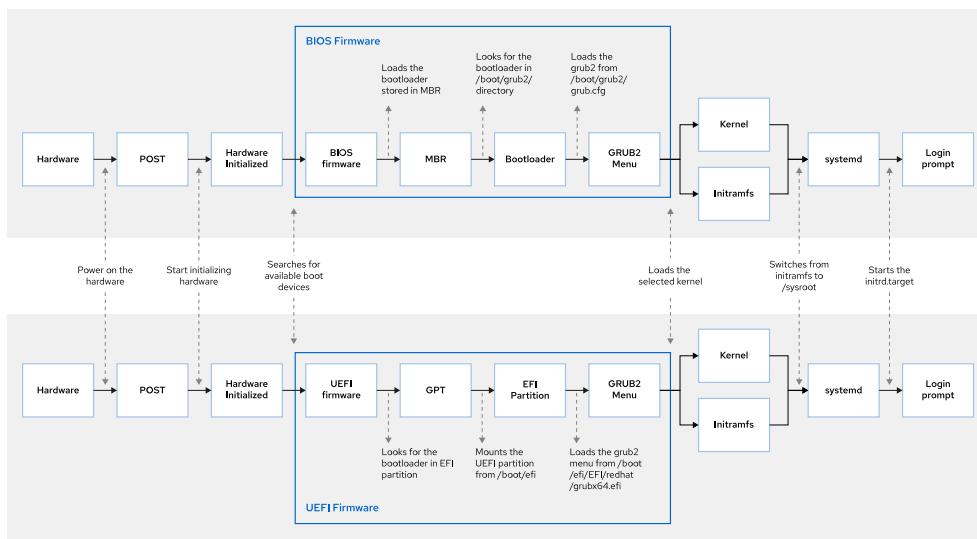


Figure 3.1: Boot process for BIOS-based and UEFI-based systems

While most configuration syntax and tools remain the same for both BIOS and UEFI boot processes, some differences exist.

linux16 and /initrd16 changed to linuxefi and initrdefi

The configuration commands to load a kernel and initial ramdisk switch from `linux16` and `initrd16` for BIOS to `linuxefi` and `initrdefi` for UEFI.

This change is necessary because kernels must load differently on a UEFI system from on a BIOS system. The `grub2-mkconfig` command automatically recognizes a UEFI system and uses the correct commands.

/boot/grub2 changed to /boot/efi/EFI/redhat

The directory for holding the UEFI GRUB2 configuration files and objects is `/boot/efi/EFI/redhat`. This directory is on the ESP partition for UEFI firmware access.

grub2-install

Do not use the `grub2-install` command directly to install the UEFI boot loader. RHEL 8 provides a prebuilt `/boot/efi/EFI/redhat/grubx64.efi` file, which contains the required authentication signatures for a Secure Boot system. Executing `grub2-install` directly on a UEFI system generates a new `grubx64.efi` file without those required signatures. The correct `grubx64.efi` file can be restored from the `grub2-efi` package.

Use of `grub2-install` directly registers a bootable target in the UEFI firmware with this new `grubx64.efi` application, instead of the required `shim.efi` application.



Warning

Use of the `grub2-install` command generates and configures a generic `grubx64.efi` file. If the system is configured for Secure Boot, then the system fails to boot.

/boot/grub2/grub.cfg **changed to** /boot/efi/EFI/redhat/grub.cfg

The GRUB2 configuration file is moved from the /boot/grub2 directory for BIOS to the /boot/efi/EFI/redhat/ directory on the ESP partition for UEFI. The symbolic link /etc/grub.cfg is moved to /etc/grub2-efi.cfg.

Managing UEFI Boot Targets

RHEL 8 provides the efibootmgr utility to manage and edit the available boot targets and logical boot order. The tool is in the efibootmgr package, which is installed by default on UEFI systems.

Viewing Current Targets

View the current targets with the efibootmgr command.

```
[root@host ~]# efibootmgr
BootCurrent: 0003
Timeout: 0 seconds
BootOrder: 0003,0002,0000①
Boot0000* UiApp
Boot0002* UEFI Misc Device
Boot0003* Red Hat Enterprise Linux②
```

- ① The list defines the order of the boot entries to try, from left to right. The numbers refer to the listed Boot0000 entries.
- ② This number is an individual boot entry; in this case, hexadecimal number 0x0003. An asterisk (*) on an entry marks it as active and available to select and boot.

View more information about boot entries by using the -v option.

```
[root@host ~]# efibootmgr -v
BootCurrent: 0003
Timeout: 0 seconds
BootOrder: 0003,0002,0000
Boot0000* UiApp FvVol(7cb8bdc9-f8eb-4f34-aaea-3ee4af6516a1)/
FvFile(462caa21-7614-4503-836e-8ab6f4662331)
Boot0002* UEFI Misc Device PciRoot(0x0)/Pci(0x2,0x3)/Pci(0x0,0x0)N.....YM....R,Y.
Boot0003* Red Hat Enterprise Linux HD(1,GPT,e7e64bf7-6799-4791-
bdc0-28147761e11d,0x800,0x12c000)/File(\EFI\redhat\shimx64.efi)①
```

- ① The Boot0003 entry now displays the hard disk, partition, and file for booting.

Removing an Entry

To remove a boot entry from the list, use the efibootmgr command with the -B option. In this example, the 0x0004 entry is deleted.

```
[root@host ~]# efibootmgr -b 4 -B
```

Selecting a Temporary Boot Target

Use the -n option to override the regular boot order for the current boot. For example, the following command forces the system to boot with entry 0x0002.

```
[root@host ~]# efibootmgr -n 2
```

Adding an Entry

Use the `-c` option for creating new entries and boot numbers. Unless overridden on the command line, the option defaults to an ESP location on `/dev/sda1`, an intended label of `Linux`, and the boot loader application is `elilo.efi`.

The following example adds an `Example` entry, with `/dev/sda2` as the ESP location, and with the `/EFI/example.efd` application on the ESP. In the application file name, you must replace forward slashes (/) with backslashes (\).

```
[root@host ~]# efibootmgr -c -d /dev/sda -p 2 -L "Example" -l "\EFI\example.efd"
```

Generating BLS Configuration Files in UEFI

1. To generate BLS configuration files, enable the feature in the `/etc/default/grub` file.

```
GRUB_ENABLE_BLSCFG=true
```

2. For a UEFI system, regenerate the `/boot/efi/EFI/redhat/grub.cfg` file.

```
[root@host ~]# grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg
Generating grub configuration file ...
done
```

3. Generate a kernel boot configuration in the `/boot/loader/entries` directory.

```
[root@host ~]# kernel-install add $(uname -r) /lib/modules/$(uname -r)/vmlinuz
```

Using the `grubby` Tool for UEFI Firmware

On UEFI systems, using `grubby` requires `root` privileges.

Use `grubby` to obtain the file name of the default kernel.

```
[root@host ~]# grubby --default-kernel
/boot/vmlinuz-4.18.0-305.el8.x86_64
```

Find the index number of the default kernel.

```
[root@host ~]# grubby --default-index
0
```

Use the `grubby` command to persistently change the default kernel.

```
[root@host ~]# grubby --set-default /boot/vmlinuz-4.18.0-305.el8.x86_64
The default is /boot/loader/entries/
e1a2445950a34804857a9407715ed7ab-4.18.0-305.el8.x86_64.conf with index 0 and
kernel /boot/vmlinuz-4.18.0-305.el8.x86_64
```

List the kernel menu entries with details.

```
[root@host ~]# grubby --info=ALL
index=0
kernel="/boot/vmlinuz-4.18.0-305.el8.x86_64"
args="ro crashkernel=auto resume=/dev/mapper/rhel-swap rd.lvm.lv=rhel/root
rd.lvm.lv=rhel/swap rhgb quiet $tuned_params"
root="/dev/mapper/rhel-root"
initrd="/boot/initramfs-4.18.0-305.el8.x86_64.img $tuned_initrd"
title="Red Hat Enterprise Linux (4.18.0-305.el8.x86_64) 8.4 (Ootpa)"
id="e1a2445950a34804857a9407715ed7ab-4.18.0-305.el8.x86_64"
index=1
kernel="/boot/vmlinuz-0-rescue-e1a2445950a34804857a9407715ed7ab"
args="ro crashkernel=auto resume=/dev/mapper/rhel-swap rd.lvm.lv=rhel/root
rd.lvm.lv=rhel/swap rhgb quiet"
root="/dev/mapper/rhel-root"
initrd="/boot/initramfs-0-rescue-e1a2445950a34804857a9407715ed7ab.img"
title="Red Hat Enterprise Linux (0-rescue-e1a2445950a34804857a9407715ed7ab) 8.4
(Ootpa)"
id="e1a2445950a34804857a9407715ed7ab-0-rescue"
```

View the GRUB 2 menu entry for a specific kernel.

```
[root@host ~]# grubby --info /boot/vmlinuz-4.18.0-305.el8.x86_64
index=0
kernel="/boot/vmlinuz-4.18.0-305.el8.x86_64"
args="ro crashkernel=auto resume=/dev/mapper/rhel-swap rd.lvm.lv=rhel/root
rd.lvm.lv=rhel/swap rhgb quiet $tuned_params"
root="/dev/mapper/rhel-root"
initrd="/boot/initramfs-4.18.0-305.el8.x86_64.img $tuned_initrd"
title="Red Hat Enterprise Linux (4.18.0-305.el8.x86_64) 8.4 (Ootpa)"
id="e1a2445950a34804857a9407715ed7ab-4.18.0-305.el8.x86_64"
```

Reinstalling GRUB2 on UEFI-based Machines

To recover damaged or missing /boot/efi files, reinstall the `grub2-efi` and `shim` packages.

```
[root@host ~]# yum reinstall grub2-efi shim
--output omitted--
Complete!
```

If the `/boot/efi/EFI/redhat/grub.cfg` file was removed, then you can regenerate it with `grub2-mkconfig`.

```
[root@host ~]# grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg
Generating grub configuration file ...
done
```

If the `Red Hat Enterprise Linux 8` entry is missing from the UEFI boot menu, it is automatically restored when you next boot from the UEFI-configured system disk.



References

`efibootmgr(1)`, `grub2-mkconfig(1)`, `kernel-install(8)` and `grubby(8)` man page

For further information, refer to *Chapter 5. Configuring Kernel Command-Line Parameters* at

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html/managing_monitoring_and_updating_the_kernel/configuring-kernel-command-line-parameters_managing-monitoring-and-updating-the-kernel

For further information, refer to *How to reinstall GRUB and GRUB2 on UEFI-based machines?* at

<https://access.redhat.com/solutions/3486741>

► Quiz

Resolving Boot Loader Issues on UEFI Systems

Choose the correct answers to the following questions:

- ▶ 1. **On which directory does the Anaconda installer mount the UEFI partition?**
 - a. /boot/grub2
 - b. /boot/efi
 - c. /etc/default
 - d. /etc/grub.d/

- ▶ 2. **Which directory stores the GRUB2 configuration files and the UEFI firmware applications?**
 - a. /boot/grub2
 - b. /etc/default
 - c. /boot/efi/EFI/redhat
 - d. /boot/loader

- ▶ 3. **Which packages are needed to reinstall deleted files from /boot/efi on UEFI-based systems?**
 - a. grub2-efi and shim
 - b. grub2-mkconfig and grub2-efi
 - c. shim and grub2-install
 - d. grub2-efi and grub2-install

- ▶ 4. **Which command sets the default kernel?**
 - a. grub2-mkconfig --default
 - b. kernel-install add
 - c. grub2-install
 - d. grubby --set-default

► Solution

Resolving Boot Loader Issues on UEFI Systems

Choose the correct answers to the following questions:

- ▶ 1. **On which directory does the Anaconda installer mount the UEFI partition?**
 - a. /boot/grub2
 - b. /boot/efi
 - c. /etc/default
 - d. /etc/grub.d/

- ▶ 2. **Which directory stores the GRUB2 configuration files and the UEFI firmware applications?**
 - a. /boot/grub2
 - b. /etc/default
 - c. /boot/efi/EFI/redhat
 - d. /boot/loader

- ▶ 3. **Which packages are needed to reinstall deleted files from /boot/efi on UEFI-based systems?**
 - a. grub2-efi and shim
 - b. grub2-mkconfig and grub2-efi
 - c. shim and grub2-install
 - d. grub2-efi and grub2-install

- ▶ 4. **Which command sets the default kernel?**
 - a. grub2-mkconfig --default
 - b. kernel-install add
 - c. grub2-install
 - d. grubby --set-default

Identifying and Resolving Failing Services

Objectives

After completing this section, you should be able to identify and resolve service failures that affect the boot process.

Service Dependencies

When a system starts up, many `systemd` services automatically start based on the selected boot configuration. Each of these services might depend on other services, units, or targets.

A `systemd` unit's configuration lists its dependencies, typically a drop-in file under `/etc/systemd/system/<UNITNAME>.d/`, or by using a `requires` or `wants` subdirectory, such as `/etc/systemd/system/<UNITNAME>.requires` or `/etc/systemd/system/<UNITNAME>.wants`, for drop-in file configuration.

Dependency Directives

The following list describes the common types of unit dependency directives. For more detailed information, consult the `systemd.unit(5)` man page.

Requires=

Starting this unit automatically pulls the listed required units into the transaction. Stopping a required unit causes this unit to stop. Unless `After=` or `Before=` relationships are used, units are all started at the same time. You can configure this dependency type by creating symbolic links to the required units in `/etc/systemd/system/<UNITNAME>.requires/`.

Requisite=

Similar to `Requires` but stronger. While a `Requires` starts a required unit if it is not already running, a `Requisite` fails if the required unit is not running.

Wants=

Similar to `Requires` but weaker. When a wanted unit fails to start, this unit still starts without it. You can configure this dependency type by creating symbolic links to the wanted units in `/etc/systemd/system/<UNITNAME>.wants/`.

Conflicts=

A negative dependency. Starting this unit stops the conflicting units, if possible.

Before=, After=

These dependency types control ordering. Without these dependencies, required units are started at the same time. If the `a.service` unit has the `Before=b.service` dependency, then `b.service` startup is delayed until starting of the `a.service` is completed. `After=` indicates that starting of the listed units must complete before this unit can be started.

Viewing Dependencies

You can view a unit's dependencies manually, automatically, or graphically. To manually view a unit's dependencies, use the `systemctl show UNIT` command. An administrator can view the unit's configuration, and repeat this process for each listed dependency.

The automated method uses the `systemctl list-dependencies UNIT` command, which displays a recursive dependency tree. On a color-capable terminal, each line displays a green or red dot, to indicate whether that unit is currently active.

```
[root@demo ~]# systemctl list-dependencies nfs-server.service
nfs-server.service
● └─auth-rpcgss-module.service
● └─nfs-idmapd.service
● └─nfs-mountd.service
● └─nfsdcl.d.service
● └─proc-fs-nfsd.mount
● └─rpc-statd-notify.service
● └─rpc-statd.service
● └─rpcbind.socket
● └─system.slice
● └─network-online.target
●   └─NetworkManager-wait-online.service
● └─network.target
```



Note

Many systemd units depend on starting after reaching a `target`, such as `basic.target`, which is a configuration of a group of units that are used as a synchronization point in the startup process. Running `systemctl list-dependencies` on a unit with a target dependency also lists all dependencies for the included targets.

When viewing units with large dependency chains, a graphical dependency representation might be easier to analyze. This method uses the `systemd-analyze dot` command with the `dot` rendering command from the `graphviz` package. The following example installs the tools, and then creates a graphical representation of the `sshd.service` unit's dependencies:

```
[root@demo ~]# yum 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
```

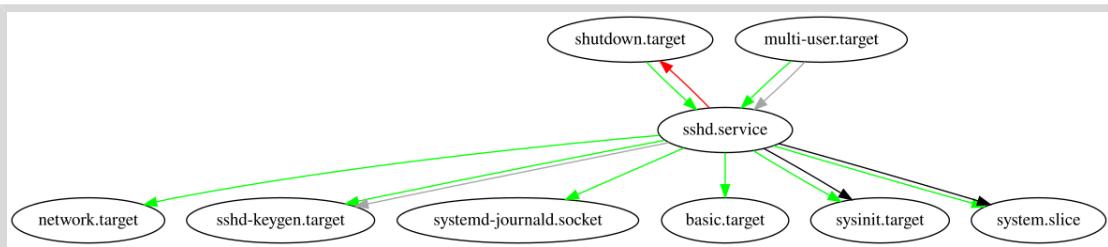


Figure 3.2: SSHD dependencies

Failing Dependency Scenarios

You can configure dependencies so that some services never start while adhering to those dependencies. For example, if the `Requires` and `Before` directives of two units are each set for the other unit, then startup would fail because neither unit would be able to satisfy both configurations. To maintain a system's ability to boot, `systemd` can detect these issues and attempt to resolve them. The method to resolve these issues is to remove one or more offending units from the transaction set until a non-conflicting configuration is reached. This can result in some expected services not starting. Failing services send output to the console, but reading the system logs is required to identify and resolve missing services in this scenario.

After identifying and verifying such issues, modify the affected unit dependencies to resolve the situation. In many cases, changing `Requires` to `Wants`, or editing a circular use of `Before` or `After`, can resolve the situation.



Important

You must inform `systemd` of your static configuration file edits by running `systemctl daemon-reload` to implement your changes on the active system.

Using Early root Shell to Debug Access

Sometimes, startup tasks take much longer than expected, or fail to complete after minutes or hours. The `systemctl list-jobs` command displays all jobs that `systemd` is currently executing. An administrator can stop or kill selected jobs, or resolve and repair the root cause for these job delays.

For privileged access to the system before startup completes, you can enable the `debug-shell` service, which provides a root shell *during* startup. This service starts a virtual console with a logged-in root shell that is attached to it on `/dev/tty9`, during the early `sysinit.target` startup. An administrator can use this early root shell to analyze, debug, and possibly resolve a failing service or unit.



Warning

Do NOT leave the `debug-shell` service enabled and running after your authorized troubleshooting is completed. Any user with console access, whether physical or via a management card or KVM switch, can use the service to obtain unrestricted root access to the system.



References

`systemd-analyze(1)` and `systemd.unit(5)` man pages

Working with `systemd` unit files

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/configuring_basic_system_settings/index#working-with-systemd-unit-files_configuring-basic-system-settings

► Guided Exercise

Identifying and Resolving Failing Services

In this lab, you resolve an issue where two services fail to start at boot.

Outcomes

You should be able to resolve an issue where two services fail to start at boot.

Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

```
[student@workstation ~]$ lab start boot-failingservices
```

Instructions

Your servera machine is set up to serve both web and FTP content with the `httpd` and `vsftpd` services. The specification states that both daemons operate simultaneously on the same system.

Your colleague attempted to satisfy this requirement but now both services fail to start at boot time. You are asked to resolve this issue. You are only required to ensure that both services start. You do not need to verify or configure that both services always operate simultaneously.

- 1. On the servera system, check the status of the `httpd` and `vsftpd` services to troubleshoot the issue.

- 1.1. Log in to servera and switch to the `root` user.

```
[student@workstation ~]$ ssh student@servera  
...output omitted...  
[student@servera ~]$ sudo -i  
[sudo] password for student: student  
[root@servera ~]#
```

- 1.2. Check the status of the `httpd` service. Because status lines sometimes contain useful information past the 80-character limit, use the `-l` option to display 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.service(8)
```

Chapter 3 | Troubleshooting Boot Issues

```
Sep 24 21:50:42 servera.lab.example.com systemd[1]: httpd.service: Found ordering
cycle on vsftpd.service/start
Sep 24 21:50:42 servera.lab.example.com systemd[1]: httpd.service: Found
dependency on httpd.service/start
Sep 24 21:50:42 servera.lab.example.com systemd[1]: httpd.service: Unable to break
cycle starting with httpd.service/start
Sep 24 21:50:42 servera.lab.example.com systemd[1]: vsftpd.service: Found ordering
cycle on httpd.service/start
Sep 24 21:50:42 servera.lab.example.com systemd[1]: vsftpd.service: Found
dependency on vsftpd.service/start
Sep 24 21:50:42 servera.lab.example.com systemd[1]: vsftpd.service: Unable to
break cycle starting with vsftpd.service/start
...output omitted...
```

1.3. Check the status of the vsftpd service.

```
[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)

...output omitted...
Sep 24 21:50:42 servera.lab.example.com systemd[1]: httpd.service: Unable to break
cycle starting with httpd.service/start
...output omitted...
```

- 2. Both services have start dependencies on each other. The `systemctl status` command output includes the files that are responsible for the dependencies. View the contents of those drop-in configuration files.

- 2.1. From the output of the `systemctl status` commands, observe that two nonstandard drop-in files were added to the services:

- `/etc/systemd/system/httpd.service.d/10-dependencies.conf`
- `/etc/systemd/system/vsftpd.service.d/10-dependencies.conf`

- 2.2. View 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 of why these two services fail to start.

- 3.1. Both services have a `Requires` directive for the other service, but this directive ensures only that if one of them is started, then the other is started as well.
 - 3.2. Both services have an `After` requirement on the other service. This requirement causes a cyclic dependency, which `systemd` resolves by not starting these services.
- ▶ 4. Describe two possible solutions.
- 4.1. Removing both drop-in directories and files entirely, and then reloading `systemd`, and finally starting the services. This approach removes all dependencies.
 - 4.2. Changing one of the two `After` lines to `Before` ensures that 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 to `Before`. Reload `systemd` and start both services to verify.
- 5.1. Change the `After` line for the `httpd` service to `Before`. Edit `/etc/systemd/system/httpd.service.d/10-dependencies.conf` to match this text:

```
[Unit]
Before=vsftpd.service
Requires=vsftpd.service
```

- 5.2. Reload the active `systemd` configuration.

```
[root@servera ~]# systemctl daemon-reload
```

- 5.3. Start the `httpd.service` and `vsftpd.service` services.

```
[root@servera ~]# systemctl start httpd.service vsftpd.service
[root@servera ~]#
```

- ▶ 6. Reboot the `servera` machine to verify that both services successfully start during boot.

- 6.1. Reboot `servera`.

```
[root@servera ~]# reboot
Connection to servera closed by remote host.
Connection to servera closed.
[student@workstation ~]$
```

- 6.2. Log in to `servera` and switch to the `root` user.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

- 6.3. Check that the `httpd` and `vsftpd` services are active.

```
[root@servera ~]# systemctl status httpd.service vsftpd.service
● 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: active (running) since Fri 2021-09-24 23:22:35 EDT; 2min 23s ago
              ...
              ...output omitted...
● 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: active (running) since Fri 2021-09-24 23:22:35 EDT; 2min 23s ago
              ...
              ...output omitted...
```

► 7. Return to workstation as the student user.

```
[root@servera ~]# exit
[student@servera ~]$ exit
[student@workstation ~]$
```

Finish

On the **workstation** machine, use the **lab** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish boot-failingservices
```

This concludes the guided exercise.

Resetting the root Password

Objectives

After completing this section, you should be able to reset the root password.

Resetting the root Password

When the `root` password for a system is lost or forgotten, an authorized administrator can reset it. Some methods can work remotely, such as through an SSH connection, while others require physical console access.

If any user is still logged in as the `root` account on an unlocked terminal, then use that active session to change the password. Similarly, use any accessible account that has sufficient sudo shell or `passwd` command access to reset the `root` password.

A more complex method is to manually edit the `/etc/shadow` file by copying in a known password hash from any account that has sudo text editor access, or by editing the virtual machine's disk image with the `guestfish` command.

Rescue Mode

If the previous methods are not available or unsuccessful, then rescue mode is an alternative. An administrator with physical system access can use the anaconda installation program's rescue mode to boot from installation media or an external media device, and then use that access to change the `root` password. This procedure requires that either the system's disks are not encrypted or that the encryption password is known. If the firmware password is configured and known, then you can boot from alternative devices on both BIOS and UEFI firmware. The anaconda rescue mode is accessed by booting from a Red Hat Enterprise Linux 8 boot media device, such as a USB installation device, and selecting the `Troubleshooting` option in the boot menu.

Resetting the root Password without External Media

When the external boot media method is not available or unsuccessful, an administrator can use the `systemd` startup sequence to halt the initial ramdisk (`initramfs`) startup sequence. This method requires physical console access, or access through a remote management card or KVM switch, and knowledge of passwords for disk encryption and the boot loader, if configured.

This method for resetting a `root` password consists of these steps:

1. Reboot the system and interrupt the boot loader timer by pressing any key except Enter.
2. Find the entry that is normally booted, and change it so that it halts execution during the initial ramdisk startup sequence.
 - a. Use the cursor keys to highlight the entry that would normally be booted, and press `e`.
 - b. Use the cursor keys to move to the line that has the kernel and the kernel arguments. This line normally starts with `Linux`.
 - c. Move the cursor to the end of the line by pressing `Ctrl+e`, and add `rd.break`.

**Note**

Classroom virtual machine images have a `console=kernel` setting for a serial console. If you are not using a serial console, then remove this setting to force the initial ramdisk to use the virtual console. If you keep this setting, then control passes to a serial console that you cannot see, and a black screen appears.

- d. Press `Ctrl+x` to boot with the modified parameters.
3. The system now boots, but exits the process during initial ramdisk execution. If a prompt does not appear shortly, press `Enter` to see whether the prompt is obscured by kernel output.
4. Remount the root file system with read and write capabilities. The file system is currently mounted on the `/sysroot` directory mount point.

```
switch_root:/# mount -o remount,rw /sysroot
```

5. Change the working root directory to `/sysroot`.

```
switch_root:/# chroot /sysroot
```

6. Reset the `root` password to a known value.

```
sh-4.2# echo "root:newpassword" | chpasswd
```

7. Force SELinux to relabel during the next boot.

```
sh-4.2# touch /.autorelabel
```

**Important**

The SELinux relabel in this method is required. SELinux detects whether an alternative access sequence occurred because the SELinux contexts are no longer present on the modified files. To trust the system again, SELinux will not boot until all files are properly relabeled.

8. Reboot the system by exiting from the `chroot` environment and from the `switch_root` prompt by typing `exit` twice.
9. Verify that the `root` password access is reset by either logging in as `root` or by logging in as a non-privileged user and switching to `root` with any method that requires entering the `root` password.

**References****Resetting the root Password of RHEL8**

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html/configuring_basic_system_settings/changing-and-resetting-the-root-password-from-the-command-line_configuring-basic-system-settings

► Guided Exercise

Resetting the root Password

In this exercise, you reset a root password.

Outcomes

You should be able to reset a root password without having another form of root access to a machine.

Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

```
[student@workstation ~]$ lab start boot-rootpw
```

This command changes your system's root password and increases the grub2 boot menu timeout value.

Instructions

A colleague installed a new `servera` system, but does not remember the `root` password. The server is otherwise configured properly, but the documentation does not include the password.

You are asked to reset the `root` password to `redhat`.



Important

For this exercise, assume that you do not have ssh access to `servera`, and that the student account on `servera` does not have full sudo access. Use a direct virtual console connection to `servera`. Obtain access to the `servera` console with an appropriate method for your classroom environment.

- 1. Open a console to your `servera` system, log in as the `student` user, and reboot the system. Press any key except `Enter` to pause the GRUB menu timer when it appears, and force the system to pause the boot sequence during execution of the initial ramdisk.
 - 1.1. From the console, log in to your `servera` system as the `student` user and run the `reboot` command.

```
servera login: student
Password: student
[student@servera ~]$ reboot
```

- 1.2. Press any key except `Enter` to pause the GRUB menu 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 `linux`, press `Ctrl+e` to jump to the end of the line, remove all `console=` settings (if any), and append `rd.break`.
 - 1.5. Press `Ctrl+x` to boot with these modified settings.
- ▶ 2. Change the root password to `redhat`, force SELinux to relabel all files during the next boot, and reboot the system.
- 2.1. Remount the file system on `/sysroot` with read-write capabilities.

```
switch_root:/# mount -o remount,rw /sysroot
```

- 2.2. Change the root directory to `/sysroot`.

```
switch_root:/# chroot /sysroot
```

- 2.3. Set the root password to `redhat`.

```
sh-4.2# echo "root:redhat" | chpasswd
```

- 2.4. Force SELinux to relabel all files during the next boot.

```
sh-4.2# touch /.autorelabel
```

- 2.5. Reboot your system by typing `exit` twice.

```
sh-4.2# exit  
exit  
switch_root:/# exit  
exit
```

- ▶ 3. Log in to the console as the `root` user with the password `redhat`.

```
servera login: root  
Password: redhat  
[root@servera ~]#
```

Finish

On the `workstation` machine, use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish boot-rootpw
```

This concludes the guided exercise.

► Lab

Troubleshooting Boot Issues

In this lab, you fix an issue where a system fails to boot past the boot loader screen.

Outcomes

You should be able to repair issues where a system fails to boot past the boot loader.

Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

```
[student@workstation ~]$ lab start boot-review
```

This command confirms that the required hosts for this exercise are accessible and provides a working kernel.

Instructions

Someone decided to shorten the boot process of the `servera` system. Doing this process, they broke the ability of `servera` to boot to its default target.

You are asked to make `servera` boot properly again.

1. Open a console to your `servera` machine and troubleshoot the problem.
2. Temporarily boot your `servera` system to a working configuration.
3. Permanently fix the issue.

Evaluation

On the `workstation` machine, use the `lab` command to grade your work. Correct any reported failures and rerun the script until you receive a passing grade.

```
[student@workstation ~]$ lab grade boot-review
```

Finish

On the `workstation` machine, use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish boot-review
```

This concludes the lab.

► Solution

Troubleshooting Boot Issues

In this lab, you fix an issue where a system fails to boot past the boot loader screen.

Outcomes

You should be able to repair issues where a system fails to boot past the boot loader.

Before You Begin

As the student user on the **workstation** machine, use the **lab** command to prepare your system for this exercise.

```
[student@workstation ~]$ lab start boot-review
```

This command confirms that the required hosts for this exercise are accessible and provides a working kernel.

Instructions

Someone decided to shorten the boot process of the **servera** system. Doing this process, they broke the ability of **servera** to boot to its default target.

You are asked to make **servera** boot properly again.

1. Open a console to your **servera** machine and troubleshoot the problem.
 - 1.1. What is displayed on the **servera** console? If a GRUB menu is displayed, attempt to boot the default (select) entry.

```
error: ../../grub-core/loader/i386/pc/linux.c:170:invalid magic number.
error: ../../grub-core/loader/i386/pc/linux.c:421:you need to load the kernel
      first.
```

Press any key to continue...

- 1.2. What can this message mean?

This error message typically indicates that no kernel is defined in the menu entry, or that the ramdisk is being loaded before the kernel.
2. Temporarily boot your **servera** system to a working configuration.
 - 2.1. What are the fix options? Verify whether a first entry boots, 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 another working entry.
3. Permanently fix the issue.
 - 3.1. Log in to the **servera** system and switch to the **root** user.

```
[student@workstation ~]$ ssh student@servera  
...output omitted...  
[student@servera ~]$ sudo -i  
[sudo] password for student: student  
[root@servera ~]#
```

- 3.2. Use the list of installed kernels to re-create a working /boot/grub2/grub.cfg file.

```
[root@servera ~]# grub2-mkconfig -o /boot/grub2/grub.cfg
```

- 3.3. Use grubby to review the newly added entry and set it as the default boot entry.

```
[root@servera ~]# grubby --info=ALL  
[root@servera ~]# grubby --set-default=/boot/vmlinuz-4.18.0-305.el8.x86_64
```

- 3.4. Verify the solution by rebooting the servera system.

```
[root@servera ~]# reboot
```

Evaluation

On the **workstation** machine, use the **lab** command to grade your work. Correct any reported failures and rerun the script until you receive a passing grade.

```
[student@workstation ~]$ lab grade boot-review
```

Finish

On the **workstation** machine, use the **lab** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish boot-review
```

This concludes the lab.

Summary

In this chapter, you learned:

- The RHEL 8 boot process flow and the purpose of the GRUB2 bootloader.
- The `grubby` command manipulates bootloader configuration files.
- The `efibootmgr` command manages the boot order and boot targets on UEFI systems.
- The `systemctl list-dependencies` and `systemd-analyze` commands show a `systemd` unit's dependencies.
- The `debug-shell` service grants early root access to a system as it boots.
- The `root` user's password is reset by adding `rd.break` to the kernel options at boot.

Chapter 4

Identifying Hardware Issues

Goal

Identify hardware issues that can affect a system's ability to operate normally.

Objectives

- Identify system hardware devices and hardware issues.
- Manage kernel modules and parameters.
- Identify and resolve virtualization issues.

Sections

- Identifying Hardware Issues (and Guided Exercise)
- Managing Kernel Modules (and Guided Exercise)
- Resolving Virtualization Issues (and Guided Exercise)

Lab

Identifying Hardware Issues

Identifying Hardware Issues

Objectives

After completing this section, you should be able to identify system hardware devices and hardware issues.

Reviewing Kernel Messages

A simple and direct source of hardware information is the running Linux kernel, which functions as the mediator of all hardware access. The kernel exposes information structures through the `/proc` and `/sys` file systems, as well as by sending kernel messages and events.

Kernel messages are written to a preallocated ring buffer that is known as the `dmesg` buffer. A ring buffer is a sequential memory structure where data overflow starts again at the top of the buffer. Over time, recent messages fill the buffer and overwrite original messages, but the buffer never grows in size.

Examine the `dmesg` buffer for the following reasons:

- Reviewing hardware that was detected at boot time.
- Observing driver messages that are sent as hardware is detected, attached, or detached.
- Observing warning or error messages as events occur.

To display ring buffer messages, use the `dmesg` or the `journalctl -k` command. The `journald` service can be configured for persistent storage, which is preferred over using the `/var/log/dmesg` file pointer to the ring buffer. The `systemd-journal` service captures the kernel ring buffer. The `rsyslog` service retrieves information from the kernel ringer buffer with the `imjournal` plug-in.

Exploring Kernel Messages

The first two lines of the `dmesg` command contain the kernel version and kernel parameters that are used on the last system boot, which is useful for troubleshooting boot issues.

```
[root@host ~]# dmesg | head -n2
[    0.000000] Linux version 4.18.0-305.el8.x86_64 (mockbuild@x86-
vm-07.build.eng.bos.redhat.com) (gcc version 8.4.1 20200928 (Red Hat 8.4.1-1)
(GCC)) #1 SMP Thu Apr 29 08:54:30 EDT 2021
[    0.000000] Command line: BOOT_IMAGE=(hd0,gpt3)/boot/
vmlinuz-4.18.0-305.el8.x86_64 root=/dev/vda3 ro no_timer_check net.ifnames=0
crashkernel=auto
```

Because of the quantity and complexity of `dmesg` log entries, view the log with filters to focus on relevant information. This example displays the memory that was made available during booting.

```
[root@host ~]# dmesg | grep "Memory"
[    0.000000] Memory: 261668K/2096600K available (12292K kernel code, 2100K
rwdata, 3816K rodata, 2348K init, 3320K bss, 271600K reserved, 0K cma-reserved)
[    0.108065] x86/mm: Memory block size: 128MB
```

You can filter messages by their syslog facility and severity by using the **-f** and **-l** options.

```
[root@host ~]# dmesg -f kern -l warn
[    0.025358] acpi PNP0A03:00: fail to add MMCONFIG information, can't access
extended PCI configuration space under this bridge.
[    6.095391] printk: systemd: 16 output lines suppressed due to ratelimiting
```

Make message time stamps easier to read by using the **-T** option.

```
[root@host ~]# dmesg -f kern -l warn -T
[Wed Sep 22 07:17:39 2021] acpi PNP0A03:00: fail to add MMCONFIG information,
can't access extended PCI configuration space under this bridge.
[Wed Sep 22 07:17:45 2021] printk: systemd: 16 output lines suppressed due to
ratelimiting
```

Identifying CPU Capabilities

Modern systems typically have multiple CPUs, each with multiple cores per socket, and possibly multiple hyper-threads per core, each with different levels of local and shared caches. The **lscpu** command provides a quick summary of the local configuration. The following example shows **lscpu** output, and an explanation of the pertinent information.

```
[root@host ~]# lscpu
Architecture:          x86_64      ①
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                4           ②
On-line CPU(s) list:   0-3
Thread(s) per core:    1           ③
Core(s) per socket:    4           ④
Socket(s):             1           ⑤
NUMA node(s):          1           ⑥
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 23
Model name:            Intel(R) Core(TM)2 Quad CPU     Q9550 @ 2.83GHz
Stepping:               10
CPU MHz:               2833.000
BogoMIPS:              5665.57
Virtualization:        VT-x      ⑦
L1d cache:             32K       ⑧
L1i cache:             32K
L2 cache:              6144K
NUMA node0 CPU(s):    0-3       ⑨
Flags:                 fpu vme vmx ... ⑩
```

- ① The architecture of the CPU.
- ② The number of logical cores that are available to the kernel for task scheduling.
- ③ The number of hyper-threads per core.
- ④ The number of cores per socket.

- ➅ The number of physical sockets.
- ➆ The number of distinct memory buses on NUMA systems.
- ➇ The hardware-assisted virtualization support for the CPU.
- ➈ The sizes of the relative caches.
- ➉ The mapping of the logical CPUs to the NUMA address buses.
- ➊ The list of extended technologies that the CPU supports.

**Note**

A CPU might display support for a specific feature flag, but that flag does not guarantee that the feature is available or in use. For example, the Intel vmx flag indicates a processor that is capable of supporting hardware virtualization, but the feature might be disabled in the firmware.

Identifying Memory

Use the `dmidecode -t memory` command to retrieve information about physical memory banks, including the type, speed, and location.

```
[root@host ~]# dmidecode -t memory
# dmidecode 2.12
...output omitted...
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
...output omitted...
```

Identifying Storage Devices

To identify physical storage devices, use the `lsscsi` command, to list the physical SCSI driver compatible devices that are attached to the system, including SSD, USB, SATA, and SAS disks.

```
[root@host ~]# 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 GUBON     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]
```

The `hdparm` command can provide detailed information for individual storage devices.

```
[root@host ~]# 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
...output omitted...
```

Identifying Peripherals on the USB and PCI Buses

The `lspci` and `lsusb` commands query system buses to discover connected, active peripherals. The `lspci` command displays hardware devices that are connected to the system PCI bus. The hardware might be integrated on the motherboard, or be physically connected through a PCI host adapter. To display further device details, use the increasing verbose options (`-v`, `-vv`, `-vvv`).

```
[root@host ~]# 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)
...output omitted...
```

The `lsusb` command displays hardware devices that are connected to any system USB bus. The hardware might be integrated on the motherboard, or be physically connected through a USB port. This command also uses verbose options to show more device details.

```
[root@host ~]# 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
...output omitted...
```

Hardware Error Reporting

Red Hat Enterprise Linux 8 provides the `rasdaemon` daemon to report hardware errors. The `rasdaemon` service processes *reliability*, *availability*, and *serviceability* (RAS) error events that are generated by kernel tracing. These trace events are logged in `/sys/kernel/debug/tracing` and are reported by `rsyslog` and `journald`.

To use `rasdaemon`, install the package, and then start and enable the service.

```
[root@host ~]# yum install rasdaemon
...output omitted...
[root@host ~]# systemctl enable --now rasdaemon
Created symlink /etc/systemd/system/multi-user.target.wants/rasdaemon.service → /
usr/lib/systemd/system/rasdaemon.service.
```

The `ras-mc-ctl` command from the `rasdaemon` package works with *error detection and correction* (EDAC) drivers. The `--help` option displays the command options.

```
[root@host ~]# ras-mc-ctl --help
Usage: ras-mc-ctl [OPTIONS...]
--quiet           Quiet operation.
--mainboard       Print mainboard vendor and model for this hardware.
--status          Print status of EDAC drivers.
--print-labels    Print Motherboard DIMM labels to stdout.
--guess-labels   Print DMI labels, when bank locator is available.
--register-labels Load Motherboard DIMM labels into EDAC driver.
--delay=N         Delay N seconds before writing DIMM labels.
--labeldb=DB      Load label database from file DB.
--layout          Display the memory layout.
--summary         Presents a summary of the logged errors.
--errors          Shows the errors stored at the error database.
--error-count    Shows the corrected and uncorrected error counts using sysfs.
--help            This help message.
```

This example summarizes memory controller events:

```
[root@host ~]# ras-mc-ctl --summary
Memory controller events summary:
  Corrected on DIMM Label(s): 'CPU_SrcID#0_Ha#0_Chan#0_DIMM#0'
  location: 0:0:0:-1 errors: 1

  No PCIe AER errors.

  No Extlog errors.

  MCE records summary:
    1 MEMORY CONTROLLER RD_CHANNEL0_ERR Transaction: Memory read errors
    2 No Errors
```

This example lists errors that the memory controller reports:

```
[root@host ~]# ras-mc-ctl --errors
Memory controller events:
1 3172-02-17 00:47:01 -0500 1 Corrected error(s): memory read error at
CPU_SrcID#0_Ha#0_Chan#0_DIMM#0 location: 0:0:0:-1, addr 65928, grain 7, syndrome
0 area:DRAM err_code:0001:0090 socket:0 ha:0 channel_mask:1 rank:0
...output omitted...
MCE events:
1 3171-11-09 06:20:21 -0500 error: MEMORY CONTROLLER RD_CHANNEL0_ERR
Transaction: Memory read error, mcg mcgstatus=0, mci Corrected_error,
n_errors=1, mcgcaps=0x01000c16, status=0x8c00004000010090, addr=0x1018893000,
misc=0x15020a086, walltime=0x57e96780, cpuid=0x00050663, bank=0x00000007
2 3205-06-22 00:13:41 -0400 error: No Error, mcg mcgstatus=0, mci Corrected_error
Error_enabled, mcgcaps=0x01000c16, status=0x9400000000000000, addr=0x0000abcd,
walltime=0x57e967ea, cpuid=0x00050663, bank=0x00000001
3 3205-06-22 00:13:41 -0400 error: No Error, mcg mcgstatus=0, mci Corrected_error
Error_enabled, mcgcaps=0x01000c16, status=0x9400000000000000, addr=0x00001234,
walltime=0x57e967ea, cpu=0x00000001, cpuid=0x00050663, apicid=0x00000002,
bank=0x00000002
```

Memory Testing

When physical memory is suspected of causing errors or being damaged, you can run a thorough, repetitive memory test with the `memtest86+` package. Because live memory testing on an active system is not recommended or might cause instability, `memtest86+` installs a separate boot entry that loads a `memtest86+` bootable program instead of a regular Linux kernel. The MemTest86 open-source project now supports BIOS, UEFI, and ARM systems, and can use a USB device to load the bootable memory test application.

For BIOS-based RHEL systems, enable the boot entry by using standard boot loader utilities:

1. Install the `memtest86+` package to add the `memtest86+` application to the `/boot` directory.

```
[root@host ~]# yum install memtest86+
```

2. Run the `memtest-setup` command to add a new template to the `/etc/grub.d/` directory.

```
[root@host ~]# memtest-setup
```

3. Update the `grub2` boot loader configuration.

```
[root@host ~]# grub2-mkconfig -o /boot/grub2/grub.cfg
```



References

`dmesg(1)`, `lscpu(1)`, `dmidecode(8)`, `lspci(8)`, `lsusb(8)`, `rasdaemon(8)`, and `ras-mc-ctl(8)` man pages

► Guided Exercise

Identifying Hardware Issues

In this exercise, you identify system hardware and view hardware error reports.

Outcomes

You should be able to identify your system's hardware and view hardware errors with rasdaemon.

Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

```
[student@workstation ~]# lab start hardware-identify
```

This command verifies that your systems are reachable and generates the necessary files for this exercise.

Instructions

You collect hardware information from servera, and list hardware errors that rasdaemon reports. Use the `/root/systeminfo.txt` file on servera to record servera hardware information.

- 1. Login to servera and switch to the root user.

```
[student@workstation ~]$ ssh student@servera  
...output omitted...  
[student@servera ~]$ sudo -i  
[sudo] password for student: student  
[root@servera ~]#
```

- 2. Record servera hardware information.

- 2.1. View which boot image was last used.

```
[root@servera ~]# dmesg | grep BOOT_IMAGE  
[    0.000000] Command line: BOOT_IMAGE=(hd0,gpt3)/boot/  
vmlinuz-4.18.0-305.el8.x86_64 root=/dev/vda3 ro no_timer_check net.ifnames=0  
crashkernel=auto  
[    0.000000] Kernel command line: BOOT_IMAGE=(hd0,gpt3)/boot/  
vmlinuz-4.18.0-305.el8.x86_64 root=/dev/vda3 ro no_timer_check net.ifnames=0  
crashkernel=auto
```

- 2.2. Record the boot image in the `/root/systeminfo.txt` file.

```
# Boot Image
(hd0,gpt3)/boot/vmlinuz-4.18.0-305.el8.x86_64
```

2.3. View the CPU model name.

```
[root@servera ~]# lscpu
Architecture:           x86_64
CPU op-mode(s):         32-bit, 64-bit
Byte Order:             Little Endian
CPU(s):                 2
On-line CPU(s) list:   0,1
Thread(s) per core:    1
Core(s) per socket:    1
Socket(s):              2
NUMA node(s):           1
Vendor ID:              GenuineIntel
BIOS Vendor ID:        Red Hat
CPU family:             6
Model:                  85
Model name:             Intel(R) Xeon(R) Gold 6248 CPU @ 2.50GHz
BIOS Model name:       RHEL 7.6.0 PC (i440FX + PIIX, 1996)
Stepping:                7
CPU MHz:                2494.048
...output omitted...
```

2.4. Record the CPU model name in the /root/systeminfo.txt file.

```
# CPU Model
Intel(R) Xeon(R) Gold 6248 CPU @ 2.50GHz
```

2.5. View the size and form factor of the memory devices.

```
[root@servera ~]# dmidecode -t memory
# dmidecode 3.2
Getting SMBIOS data from sysfs.
SMBIOS 2.8 present.
...output omitted...
Handle 0x1100, DMI type 17, 40 bytes
Memory Device
  Array Handle: 0x1000
  Error Information Handle: Not Provided
  Total Width: Unknown
  Data Width: Unknown
  Size: 2 GB
  Form Factor: DIMM
  Set: None
  Locator: DIMM 0
...output omitted...
```

2.6. Record the size and form factor of the memory devices in the /root/systeminfo.txt file.

```
# Memory  
Size: 2 GB  
Form Factor: DIMM
```

2.7. View the devices that are connected via any USB bus.

```
[root@servera ~]# lsusb  
Bus 001 Device 002: ID 0627:0001 Adomax Technology Co., Ltd  
Bus 001 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
```

2.8. Record the USB devices in the /root/systeminfo.txt file.

```
# USB Devices  
Bus 001 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub  
Bus 001 Device 002: ID 0627:0001 Adomax Technology Co., Ltd
```

2.9. View the devices that are connected via the PCI bus.

```
[root@servera ~]# lspci  
00:00.0 Host bridge: Intel Corporation 440FX - 82441FX PMC [Natoma] (rev 02)  
00:01.0 ISA bridge: Intel Corporation 82371SB PIIX3 ISA [Natoma/Triton II]  
00:01.1 IDE interface: Intel Corporation 82371SB PIIX3 IDE [Natoma/Triton II]  
00:01.2 USB controller: Intel Corporation 82371SB PIIX3 USB [Natoma/Triton II]  
00:01.3 Bridge: Intel Corporation 82371AB/EB/MB PIIIX4 ACPI (rev 03)  
00:02.0 VGA compatible controller: Cirrus Logic GD 5446  
00:03.0 Ethernet controller: Red Hat, Inc. Virtio network device  
00:04.0 Ethernet controller: Red Hat, Inc. Virtio network device  
00:05.0 SCSI storage controller: Red Hat, Inc. Virtio block device  
00:06.0 SCSI storage controller: Red Hat, Inc. Virtio block device  
00:07.0 Unclassified device [00ff]: Red Hat, Inc. Virtio memory balloon
```

2.10. Record the PCI devices in the /root/systeminfo.txt file.

```
# PCI Devices  
00:00.0 Host bridge: Intel Corporation 440FX - 82441FX PMC [Natoma] (rev 02)  
00:01.0 ISA bridge: Intel Corporation 82371SB PIIX3 ISA [Natoma/Triton II]  
00:01.1 IDE interface: Intel Corporation 82371SB PIIX3 IDE [Natoma/Triton II]  
00:01.2 USB controller: Intel Corporation 82371SB PIIX3 USB [Natoma/Triton II]  
00:01.3 Bridge: Intel Corporation 82371AB/EB/MB PIIIX4 ACPI (rev 03)  
00:02.0 VGA compatible controller: Cirrus Logic GD 5446  
00:03.0 Ethernet controller: Red Hat, Inc. Virtio network device  
00:04.0 Ethernet controller: Red Hat, Inc. Virtio network device  
00:05.0 SCSI storage controller: Red Hat, Inc. Virtio block device  
00:06.0 SCSI storage controller: Red Hat, Inc. Virtio block device  
00:07.0 Unclassified device [00ff]: Red Hat, Inc. Virtio memory balloon
```

- 3. Install `rasdaemon`, start and enable the service, and add a summary of all hardware errors to the /root/systeminfo.txt file.

3.1. Install the `rasdaemon` package.

```
[root@servera ~]# yum install rasdaemon  
...output omitted...
```

3.2. Start and enable the `rasdaemon` service.

```
[root@servera ~]# systemctl enable --now rasdaemon  
Created symlink /etc/systemd/system/multi-user.target.wants/rasdaemon.service → /  
usr/lib/systemd/system/rasdaemon.service.
```

3.3. View a summary of all hardware errors.

```
[root@servera ~]# ras-mc-ctl --errors  
No Memory errors.  
  
No PCIe AER errors.  
  
No Extlog errors.  
  
No MCE errors.
```

3.4. Record the error summary in the `/root/systeminfo.txt` file.

```
# Hardware Errors  
No Memory errors.  
  
No PCIe AER errors.  
  
No Extlog errors.  
  
No MCE errors.
```

► 4. Return to workstation as the student user.

```
[root@servera ~]# exit  
[student@servera ~]$ exit  
[student@workstation ~]$
```

Finish

On the `workstation` machine, use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish hardware-identify
```

This concludes the guided exercise.

Managing Kernel Modules

Objectives

After completing this section, you should be able to manage kernel modules and parameters.

Managing Kernel Modules

The running Linux kernel is constructed from feature and driver modules that can be loaded and unloaded on demand. Use of kernel modules keeps the base kernel memory footprint small and ensures that code is loaded into memory only when required.

Most device drivers are built as kernel modules, as are file system drivers, encryption algorithms, and many other kernel features. Modules are found in the `/lib/modules/<KERNEL_VERSION>/kernel/<SUBSYSTEM>` directory as compressed `<KERNEL_MODULE_NAME>.ko.xz` object files.

Kernel modules have regular names and aliases. Aliases are used to provide shorter name alternatives, or for standardizing names for management routines to recognize. For example, USB devices create aliases in the form `usb:v0123d0001` to display the vendor ID 0123 and device ID 0001 in the device name. Similar letter patterns are standardized for all kernel modules.

Viewing Loaded Modules

The `lsmod` command lists the currently loaded modules. The output includes the loaded module name, its size in bytes, the number of other modules that currently use this module, and an optional list of those other modules.

```
[user@host ~]$ lsmod
Module           Size  Used by
nft_fib_inet     16384  1
nft_fib_ipv4     16384  1 nft_fib_inet
nft_fib_ipv6     16384  1 nft_fib_inet
nft_fib          16384  3 nft_fib_ipv6,nft_fib_ipv4,nft_fib_inet
nft_reject       16384  1 nft_reject_inet
nft_ct           20480  10
nf_tables_set    49152  13
nft_chain_nat   16384  12
nf_nat           45056  1 nft_chain_nat
nfit              65536  0
libnvdimm        192512  1 nfit
kvm_intel        315392  0
kvm               847872  1 kvm_intel
...output omitted...
```

Loading and Unloading Modules

Developers use the `insmod` and `rmmmod` commands to load and unload modules. However, recommended practice for administrators is to use the `modprobe` command instead, which is more capable and handles module dependencies. The `modprobe` command handles options or aliases that are defined in configuration files in `/etc/modprobe.d/`. To unload a module, use

`modprobe -r`. A module in use cannot be unloaded until components that use that module are exited or detached.

With the `-v` option, `modprobe` can display the `insmod` and `rmmmod` commands in use.

```
[root@host ~]$ modprobe mce-inject -v
insmod /lib/modules/4.18.0-305.el8.x86_64/kernel/arch/x86/kernel/cpu/mce/mce-
inject.ko.xz
[root@host ~]$ modprobe -r mce-inject -v
rmmod mce_inject
```



Note

The `modprobe` exit codes indicate whether the end result was achieved, rather than whether the action was performed. For example, when loading an already loaded module, `modprobe` returns true (exit code 0), even though no load action occurred.

Viewing Module Parameters

Many kernel modules support optional parameters to alter their behavior. To view a module's parameters, use the `modinfo -p <MODULE>` command. Alternatively, most modules set their parameters within the `/sys/module/<MODULE_NAME>/parameters` directory, with each parameter as a separate file. If a parameter is modifiable at runtime, the corresponding file is root writable. If a parameter is configurable only at module load time, the corresponding file is read-only.

To view the `kvm` module's parameters, use `modinfo -p`. The output includes parameter names, descriptions, and value types, such as integer, Boolean, or string.

```
[root@host ~]# modinfo -p kvm
tdp_mmu: (bool)
nx_huge_pages: (bool)
nx_huge_pages_recovery_ratio: (uint)
flush_on_reuse: (bool)
ignore_msrs: (bool)
report_ignored_msrs: (bool)
min_timer_period_us: (uint)
kvmclock_periodic_sync: (bool)
tsc_tolerance_ppm: (uint)
lapic_timer_advance_ns: (int)
vector_hashing: (bool)
enable_vmware_backdoor: (bool)
force_emulation_prefix: (bool)
pi_inject_timer: (bint)
halt_poll_ns: (uint)
halt_poll_ns_grow: (uint)
halt_poll_ns_grow_start: (uint)
halt_poll_ns_shrink: (uint)
```

You can view the same `kvm` module parameters with the `/sys/module/kvm/parameters` directory.

```
[root@host ~]# ls -l /sys/module/kvm/parameters/
total 0
-r--r--r--. 1 root root 4096 Sep 24 12:23 enable_vmware_backdoor
-rw-r--r--. 1 root root 4096 Sep 24 12:23 flush_on_reuse
-r--r--r--. 1 root root 4096 Sep 24 12:23 force_emulation_prefix
-rw-r--r--. 1 root root 4096 Sep 24 12:23 halt_poll_ns
-rw-r--r--. 1 root root 4096 Sep 24 12:23 halt_poll_ns_grow
-rw-r--r--. 1 root root 4096 Sep 24 12:23 halt_poll_ns_grow_start
-rw-r--r--. 1 root root 4096 Sep 24 12:23 halt_poll_ns_shrink
-rw-r--r--. 1 root root 4096 Sep 24 12:23 ignore_msrs
-r--r--r--. 1 root root 4096 Sep 24 12:23 kvmclock_periodic_sync
-rw-r--r--. 1 root root 4096 Sep 24 12:23 lapic_timer_advance_ns
-rw-r--r--. 1 root root 4096 Sep 24 12:23 min_timer_period_us
-rw-r--r--. 1 root root 4096 Sep 24 12:23 mmu_audit
-rw-r--r--. 1 root root 4096 Sep 24 12:23 nx_huge_pages
-rw-r--r--. 1 root root 4096 Sep 24 12:23 nx_huge_pages_recovery_ratio
-rw-r--r--. 1 root root 4096 Sep 24 12:23 pi_inject_timer
-rw-r--r--. 1 root root 4096 Sep 24 12:23 report_ignored_msrs
-rw-r--r--. 1 root root 4096 Sep 24 12:23 tdp_mmu
-rw-r--r--. 1 root root 4096 Sep 24 12:23 tsc_tolerance_ppm
-r--r--r--. 1 root root 4096 Sep 24 12:23 vector_hashing
```

Detailed parameter descriptions and values for many kernel modules are found in files in the *kernel-doc* package, which is not installed by default.

Managing Module Parameters

To pass a parameter to a module, add the `parameter=value` option to the `modprobe` command when loading the module. This example sets the `buffer_kbs` parameter to 64 for the `st` module:

```
[root@host ~]# 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 set this parameter each time that the module is loaded, use a configuration file in the `/etc/modprobe.d/` directory. Module configuration files require that file names end in `.conf`. Files are parsed in alphanumeric order. Add parameters with an `options` line.

```
options st buffer_kbs=64 max_sg_segs=512
```

Group options into files that are named for their module, such as `/etc/modprobe.d/st.conf`, or into files that relate to a function, such as `/etc/modprobe.d/throughput.conf`.



References

Managing kernel modules

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/managing_monitoring_and_updating_the_kernel/index#managing-kernel-modules_managing-monitoring-and-updating-the-kernel

`lsmod(8)` and `modprobe(8)` man pages

► Guided Exercise

Managing Kernel Modules

In this exercise, you configure parameters for a kernel module.

Outcomes

You should be able to configure module options for loadable kernel modules.

Before You Begin

As the student user on the workstation machine, use the `lab start hardware-kernelmodules` command to prepare your system for this exercise.

```
[student@workstation ~]# lab start hardware-kernelmodules
```

This command confirms that your systems are reachable.

Instructions

The `servera` system has an attached SAS RAID array that uses the `megaraid_sas.ko` kernel module. Unfortunately, performance is unpredictable, and sometimes error messages about MSI-X interrupts appear in the logs.

After opening a support case, you are asked to disable MSI-X interrupt handling in the driver to test whether this configuration fixes the issue.

- 1. Log in to `servera` and switch to the `root` user.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

- 2. View the kernel module options for the `megaraid_sas.ko` kernel module. Locate the option for MSI-X interrupt handling, load the module, and verify that this option is applied. Unload the module before continuing.

- 2.1. View the options for the `megaraid_sas.ko` kernel module:

```
[root@servera ~]# modinfo -p megaraid_sas
lb_pending_cmds:Change raid-1 load balancing outstanding threshold. Valid Values
are 1-128. Default: 4 (int)
max_sectors:Maximum number of sectors per IO command (int)
msix_disable:Disable MSI-X interrupt handling. Default: 0 (int)
msix_vectors:MSI-X max vector count. Default: Set by FW (int)
allow_vf_ioctl:Allow ioctl in SR-IOV VF mode. Default: 0 (int)
...output omitted...
```

2.2. Locate the MSI-X interrupt handling option.

```
[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)
```

The `msix_disable` option is the relevant choice.

2.3. Load the `megaraid_sas.ko` kernel module manually.

```
[root@servera ~]# modprobe -v megaraid_sas
insmod /lib/modules/4.18.0-305.el8.x86_64/kernel/drivers/scsi/megaraid/
megaraid_sas.ko.xz
```

2.4. Verify that the `msix_disable` default (value 0) is the current value.

```
[root@servera ~]# cat /sys/module/megaraid_sas/parameters/msix_disable
0
```

2.5. Unload the `megaraid_sas.ko` module.

```
[root@servera ~]# modprobe -rv megaraid_sas
rmmod megaraid_sas
```

- 3. Configure the `megaraid_sas.ko` module to always set `msix_disable=1`. This option is applied whether the module is loaded manually or automatically. For easy copying of this configuration to other systems, use a separate configuration file. Verify that the new configuration is applied correctly.

3.1. Create a file called `/etc/modprobe.d/megaraid.conf` with the following entry:

```
options megaraid_sas msix_disable=1
```

3.2. Load the `megaraid_sas.ko` kernel module.

```
[root@servera ~]# modprobe -v megaraid_sas
insmod /lib/modules/4.18.0-305.el8.x86_64/kernel/drivers/scsi/megaraid/
megaraid_sas.ko.xz
```

3.3. Verify that your new setting is applied correctly.

```
[root@servera ~]# cat /sys/module/megaraid_sas/parameters/msix_disable
1
```

- 4. Return to workstation as the student user.

```
[root@servera ~]# exit
[student@servera ~]$ exit
[student@workstation ~]$
```

Finish

On the **workstation** machine, use the **lab** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish hardware-kernelmodules
```

This concludes the guided exercise.

Resolving Virtualization Issues

Objectives

After completing this section, you should be able to identify and resolve virtualization issues.

Virtualization Issues

Kernel-based Virtual Machine (KVM) is a virtualization technology that is available across all Red Hat products. KVM provides the virtualization kernel drivers that turn a Linux host into a hypervisor that can run multiple, isolated virtual machines (VMs). User-space components of virtualization include the QEMU emulator that simulates a virtualized hardware platform and manages resources between the host and VMs. The `libvirt` collection of tools provides an interface for management and communication to help to interact with QEMU for configuring and running VMs.

Issues can arise when running multiple virtual machines with KVM and `libvirt`. Some issues relate to hardware or firmware, while others are configuration issues. This section identifies some issues, their symptoms, and solutions.

Hardware Virtualization Support

KVM requires hardware virtualization support in both the CPU and the host system firmware. To identify support for KVM, search the `flags` section of the `/proc/cpuinfo` file. For Intel machines, the `vmx` flag is present. For AMD-based machines, the `svm` flag is present.

View the CPU flags for each processor on the host machine:

```
[root@host ~]# egrep 'processor|vmx|svm' /proc/cpuinfo
processor : 0
flags  : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat
        pse36 clflush mmx fxsr sse sse2 ss syscall nx pdpe1gb rdtscp lm constant_tsc
        arch_perfmon rep_good nopl xtopology cpuid tsc_known_freq pni pclmulqdq vmx ssse3
        fma cx16 pcid sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx
        f16c rdrand
...output omitted...
processor : 1
flags  : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat
        pse36 clflush mmx fxsr sse sse2 ss syscall nx pdpe1gb rdtscp lm constant_tsc
        arch_perfmon rep_good nopl xtopology cpuid tsc_known_freq pni pclmulqdq vmx ssse3
        fma cx16 pcid sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx
        f16c rdrand
...output omitted...
```

If the virtualization flag is available, then you can test support for KVM by loading the `kvm-intel` or `kvm-amd` kernel module manually. If the module loads without errors, or is already loaded, then hardware virtualization support should be available. If the module load fails with an error message, then either the CPU does not support hardware virtualization or the feature is disabled in the host system firmware:

```
[root@host ~]# modprobe -v kvm-intel
modprobe: ERROR: could not insert 'kvm_intel': Operation not supported
```

Use the `virsh capabilities` command to check for hardware virtualization support. The command lists all supported virtual machine types on the host system.

```
[root@host ~]# virsh capabilities
<capabilities>

<host>
...output omitted...
<cpu>
  <arch>x86_64</arch>
  <model>Cascadelake-Server-noTSX</model>
  <vendor>Intel</vendor>
  <microcode version='83898374' />
  <topology sockets='2' dies='1' cores='1' threads='1' />
  <feature name='ss' />
  <feature name='vmx' /> ①
...output omitted...
</host>
...output omitted...
<guest>
  <os_type>hvm</os_type>
  <arch name='x86_64'> ②
    <wordsize>64</wordsize>
    <emulator>/usr/libexec/qemu-kvm</emulator>
...output omitted...
    <machine maxCpus='384'>pc-q35-rhel8.2.0</machine>
    <machine canonical='pc-q35-rhel8.2.0' maxCpus='384'>q35</machine>
...output omitted...
    <domain type='qemu' /> ③
    <domain type='kvm' /> ④
  </arch>
<features>
  <acpi default='on' toggle='yes' />
  <apic default='on' toggle='no' />
  <cpuselection />
  <deviceboot />
  <disksnapshot default='on' toggle='no' />
</features>
</guest>

</capabilities>
```

- ① The `vmx` feature indicates that support for `kvm-intel` is available.
- ② The architecture name indicates the type of hardware that is virtualized; in this case, a 64-bit `x86` machine. In some cases, the output contains multiple architecture names; for example, both 32-bit and 64-bit VMs can be emulated.
- ③ This domain type indicates that this type of machine can be *emulated* with a `qemu` software CPU. Use of emulation can be measurably slower than full hardware virtualization.

- ④ This domain type indicates support for hardware virtualization by using KVM. If the value is missing, then this type of machine cannot be virtualized in hardware and requires emulation.

**Important**

If hardware virtualization is not available, and cannot be turned on in the host system firmware, then virtualized machines run on an emulated processor.

Using Overcommitted Resources

Administrators can use overcommit features in `Libvirt` to assign more virtual resources to VMs than are physically available on the host system.

When overcommit limitations are respected, overcommitting resources is efficient and should not negatively affect system performance.

- VM CPU usage is rarely 100%, and typically less than 50% in many use cases. Assigning more virtual CPUs than physical cores is a recommended practice for increasing CPU utilization, within sensible 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 again with Copy-on-Write (COW) when one VM writes to that page.
- Sparse disk images can have a larger total size than the available space on the underlying storage, provided that the size in use cannot exceed the available physical space on the host storage.

These overcommit methods should not impact the overall performance of the running VMs, unless a resource becomes stressed. For example, if many virtual machines used 100% of their virtual CPU allocation at the same time, the VMs would compete for physical CPU time and experience queue delays due to CPU saturation.

When resources become scarce, overall performance and latency can significantly decrease. The host must balance the available resources between the VMs, while also using those resources for the balancing.

Viewing Resource Use

KVM virtual machines are implemented as regular processes on the host. To view resource usage, use normal performance metrics tools such as `top`. 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>`.

Other monitoring tools, such as `collectd` and `web console`, have plug-ins to monitor virtual machine resource usage.

Troubleshooting Scarce Resources

Several approaches are used to resolve scarce resources:

- Scale up by adding more resources, with more memory, CPU, disk space, or other resources.
- Add more hypervisor hosts to scale out the virtual machines.

- Limit the resource usage of specific virtual machines, either by reducing their allocation or by imposing limits by using cgroups.
- Disable or power off unnecessary or noncritical applications or virtual machines until resource usage returns to a manageable state.

Libvirt XML Configuration

The `libvirt` service stores virtual machine definitions and related configuration as XML files. You can validate these XML files with the Relax NG schemas in the `/usr/share/libvirt/schemas/` directory.

If files were updated with libvirt tools such as `virsh` and `virt-manager` only, and not by direct file editing, then the files should successfully validate against the provided schema files. Manually changing files under the `/etc/libvirt/` directory is not recommended, as this approach can introduce configuration issues.

Validate the syntax of an XML file, and then validate the file against the libvirt schema:

- To ensure that the file is valid XML, use the `xmllint` tool.

```
[root@host ~]# xmllint --noout FILENAME
```

It is recommended to solve errors in the `xmllint` reported order, because earlier failures commonly cause later related failures that are not the real problem source.

- After the file validates as well-structured XML, use the `virt-xml-validate` tool to validate conformity with a schema. The tool parses the XML file to identify its type of configuration, and then validates it against the correct schema.

```
[root@host ~]# virt-xml-validate FILENAME
```

Libvirt Networking

Libvirt uses software bridges to provide virtual networks to virtual machines. These bridges act as virtual switches, connecting all virtual network interfaces that are assigned to them.

Networks are defined in files in the `/etc/libvirt/qemu/networks/` directory, and can be configured to autostart by adding a symbolic link in the `/etc/libvirt/qemu/networks/autostart/` directory. Use tools such as `virsh` or `virt-manager` to edit these files and to avoid common mistakes during manual editing.

Apart from virtual networks created by the `libvirt` command, administrators can use `NetworkManager` to configure regular bridges with one or more physical network interfaces assigned to them.

Virtual Networking Issues

If networking on virtual machines is not working as expected, various issues might be the cause:

A virtual machine is unreachable from the outside

If a connection from outside the hypervisor machine to a virtual machine cannot be established, the following issues might be the cause:

- The virtual network is operating in NAT mode.
- A firewall on the hypervisor, or on the VM, is blocking connections.

- The client machine is missing a defined route to reach the VM.

The outside world is unreachable from the virtual machine

If the virtual machine itself cannot reach the outside world, the following issues might be the cause:

- The virtual network is operating in isolated mode.
- A firewall rule on the hypervisor might be blocking outgoing connections.

Connection issues in both directions

To allow network traffic to virtual networks of the type NAT and routed, the `libvirt` command creates firewall rules with `iptables`. If an administrator clears all firewall rules, or adds blocking rules at the top of chains, an interruption of regular network traffic to and from virtual machines might occur. These issues might be solved by restarting the `libvirtd` service, or by restarting the individual `libvirt` networks that encountered the error.



References

For more information, refer to the *Configuring Virtual Machine Network Connections* chapter in the *Red Hat Enterprise Linux 8 Virtualization Guide* at

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/configuring_and_managing_virtualization/index#configuring-virtual-machine-network-connections_configuring-and-managing-virtualization

For further information, refer to the *Viewing Information About Virtual Machines* chapter in the *Red Hat Enterprise Linux 8 Virtualization Guide* at

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/configuring_and_managing_virtualization/index#viewing-information-about-virtual-machines_configuring-and-managing-virtualization

► Guided Exercise

Resolving Virtualization Issues

In this exercise, you fix an invalid domain XML specification for a virtual machine.

Outcomes

You should be able to fix a malformed libvirt domain XML file.

Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

```
[student@workstation ~]$ lab start hardware-virtualization
```

Instructions

One of your colleagues is unable to import a virtual machine on the `servera` system. This virtual machine is defined in the `/root/vm-guest1.xml` file. Importing it with `virsh define /root/vm-guest1.xml` returns errors about a malformed XML file.

You are asked to investigate and fix this issue.



Important

For the purpose of this exercise, the virtual machine needs only to be defined, not started. The disk images that are referenced in the virtual machine definition do not exist, and do not need to be created.

- 1. Begin by re-creating the issue. Note any errors, and try to interpret them.

- 1.1. Log in to `servera` and switch to the `root` user.

```
[student@workstation ~]$ ssh student@servera  
...output omitted...  
[student@servera ~]$ sudo -i  
[sudo] password for student: student  
[root@servera ~]#
```

- 1.2. Define the virtual machine by using the `vm-guest1.xml` file.

```
[root@servera ~]# virsh define vm-guest1.xml  
error: Failed to define machine from vm-guest1.xml  
error: (domain_definition):20: Opening and ending tag mismatch: acpi line 17 and  
features  
  </features>  
-----^
```

13. 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 the `vm-guest1.xml` file is malformed.

► 2. Locate and correct the XML errors in the `vm-guest1.xml` file.

- 2.1. Use the `xmllint` command to locate errors in the `vm-guest1.xml` file. Remember that the first reported error can have a cascading effect; therefore fix errors one at a time, starting with the first error that the `xmllint` command reports.

```
[root@servera ~]# xmllint vm-guest1.xml
vm-guest1.xml:20: parser error : Opening and ending tag mismatch: acpi line 17 and
features
</features>
^
...output omitted...
```

It appears that an unclosed `<acpi>` tag exists on or around line 20.

- 2.2. Edit the file to close the unclosed tag on line 17 in the `vm-guest1.xml` file:

```
<acpi/>
```

- 2.3. Repeat the previous steps to locate and correct errors in the `vm-guest1.xml` file.

```
[root@servera ~]# xmllint vm-guest1.xml
vm-guest1.xml:132: parser error : Unescaped '<' not allowed in attributes values
<address type='pci' domain='0x0000' bus='0x05' slot='0x00' function='0x0'/
^
...output omitted...
```

Errors might appear before where the `xmllint` command identifies an issue. In this output, it appears that there is a syntax issue near line 132.

- 2.4. Use your chosen editor to locate the error near line 132 in the `vm-guest1.xml` file:

```
<memballoon model='virtio'>
```

- 2.5. Run the `xmllint` command again to verify that no more XML issues exist.

```
[root@servera ~]# xmllint vm-guest1.xml
<?xml version="1.0"?>
<domain type="kvm">
  <name>vm-guest1</name>
  <uuid>9b94dde1-d145-45ee-8c65-0c133c3d5b01</uuid>
  <metadata>
    <libosinfo:libosinfo xmlns:libosinfo="http://libosinfo.org/xmlns/libvirt/
domain/1.0">
      <libosinfo:os id="http://redhat.com/rhel/8.4"/>
    </libosinfo:libosinfo>
  </metadata>
  <memory unit="KiB">1048576</memory>
```

```
<currentMemory unit="KiB">1048576</currentMemory>
<vcpu placement="static">2</vcpu>
<os>
  <type arch="x86_64" machine="pc-q35-5.2">hvm</type>
  <boot dev="hd"/>
</os>
<features>
  <acpi/>
  <apic/>
  <vmpart state="off"/>
</features>
...output omitted...
<memballoon model="virtio">
  <address type="pci" domain="0x0000" bus="0x05" slot="0x00" function="0x0"/>
</memballoon>
<rng model="virtio">
  <backend model="random">/dev/urandom</backend>
  <address type="pci" domain="0x0000" bus="0x06" slot="0x00" function="0x0"/>
</rng>
</devices>
</domain>
```

- 3. Verify that the `vm-guest1.xml` file now adheres to the `libvirt` XML schema.

```
[root@servera ~]# virt-xml-validate vm-guest1.xml
vm-guest1.xml validates
```

- 4. Define a virtual machine by using the `vm-guest1.xml` file.

```
[root@servera ~]# virsh define vm-guest1.xml
Domain vm-guest1 defined from vm-guest1.xml
```

- 5. Return to workstation as the student user.

```
[root@servera ~]# exit
[student@servera ~]$ exit
[student@workstation ~]$
```

Finish

On the `workstation` machine, use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish hardware-virtualization
```

This concludes the guided exercise.

▶ Lab

Identifying Hardware Issues

In this lab, you configure options for the `usb-storage` kernel module.

Outcomes

You should be able to configure a module option for the `usb-storage` driver.

Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

```
[student@workstation ~]$ lab start hardware-review
```

This command confirms that the required host for this exercise is accessible and sets up the needed modules for the requested activities.

Instructions

Your `servera` system has issues with a removable storage device that uses the `usb-storage` driver. After testing and research, you determined that your device incorrectly reports 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 is in the `kernel-parameters.txt` file from the `kernel-doc` package.

You are asked to configure the `servera` system so that the `usb-storage` module ignores the write-protect mode from that device.

1. Install the `kernel-doc` package, and search for options that relate to `usb-storage` in the `kernel-parameters.txt` file.
2. Create a permanent configuration to force-enable the `NO_WP_DETECT` flag for the USB device with vendor ID `0513` and product ID `0132`.

Evaluation

On the `workstation` machine, use the `lab` command to grade your work. Correct any reported failures and rerun the script until you receive a passing grade.

```
[student@workstation ~]$ lab grade hardware-review
```

Finish

On the `workstation` machine, use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish hardware-review
```

This concludes the lab.

► Solution

Identifying Hardware Issues

In this lab, you configure options for the `usb-storage` kernel module.

Outcomes

You should be able to configure a module option for the `usb-storage` driver.

Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

```
[student@workstation ~]$ lab start hardware-review
```

This command confirms that the required host for this exercise is accessible and sets up the needed modules for the requested activities.

Instructions

Your `servera` system has issues with a removable storage device that uses the `usb-storage` driver. After testing and research, you determined that your device incorrectly reports 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 is in the `kernel-parameters.txt` file from the `kernel-doc` package.

You are asked to configure the `servera` system so that the `usb-storage` module ignores the write-protect mode from that device.

1. Install the `kernel-doc` package, and search for options that relate to `usb-storage` in the `kernel-parameters.txt` file.

- 1.1. Log in to `servera` and switch to the `root` user.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

- 1.2. Install the `kernel-doc` package.

```
[root@servera ~]# yum install kernel-doc
```

- 1.3. Open the `/usr/share/doc/kernel-doc*/Documentation/admin-guide/kernel-parameters.txt` file and search for `usb-storage`. Identify which parameter might be useful, and which flag to use.
The `quirks` parameter seems useful, with the `w (NO_WP_DETECT)` flag added.
2. Create a permanent configuration to force-enable the `NO_WP_DETECT` flag for the USB device with vendor ID `0513` and product ID `0132`.
 - 2.1. Create a file called `/etc/modprobe.d/usb-storage.conf` with the following content:

```
options usb-storage quirks=0513:0132:w
```

- 2.2. Return to `workstation` as the `student` user.

```
[root@servera ~]# exit  
[student@servera ~]$ exit  
[student@workstation ~]$
```

Evaluation

On the `workstation` machine, use the `lab` command to grade your work. Correct any reported failures and rerun the script until you receive a passing grade.

```
[student@workstation ~]$ lab grade hardware-review
```

Finish

On the `workstation` machine, use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish hardware-review
```

This concludes the lab.

Summary

In this chapter, you learned:

- The `lscpu`, `lspci`, and `dmidecode` commands are useful for identifying system hardware components.
- The `rasdaemon` tool reports hardware errors generated by kernel tracing.
- The `memtest86+` package checks system memory thoroughly and can be run repetitively.
- The `modprobe` command loads and unloads kernel modules.
- The `modprobe` and `virsh capabilities` commands identify hardware virtualization support.
- The `xmllint` and `virt-xml-validate` commands validate `libvirt` configuration files.

Chapter 5

Troubleshooting Storage Issues

Goal

Identify and resolve issues related to storage.

Objectives

- Describe the Linux storage layers and their function, and identify tools to examine activity at different layers.
- Detect and recover from file system corruption.
- Revert an LVM storage misconfiguration.
- Recover data from an encrypted device.
- Identify and repair iSCSI issues.

Sections

- Describing the Linux Storage Stack (and Guided Exercise)
- Recovering from File System Corruption (and Guided Exercise)
- Repairing LVM Issues (and Guided Exercise)
- Resolving Storage Device Encryption Issues (and Guided Exercise)
- Resolving iSCSI Issues (and Guided Exercise)

Lab

Troubleshooting Storage Issues

Describing the Linux Storage Stack

Objectives

After completing this section, you should be able to describe the Linux storage layers and their function, and identify tools to examine activity at different layers.

The Linux Storage Stack

Troubleshooting storage issues in Red Hat Enterprise Linux starts with understanding how I/O is passed from applications through the Linux storage stack to storage devices.

Applications request system calls to read and write from storage. The kernel processes these system calls through the storage stack of software and hardware layers, to move application data to storage devices. Application programmers can use the storage stack to access data as standardized data structures without the need to interact directly with specific file system and storage device implementations. The storage stack also provides features that improve the performance and reliability of I/O operations.

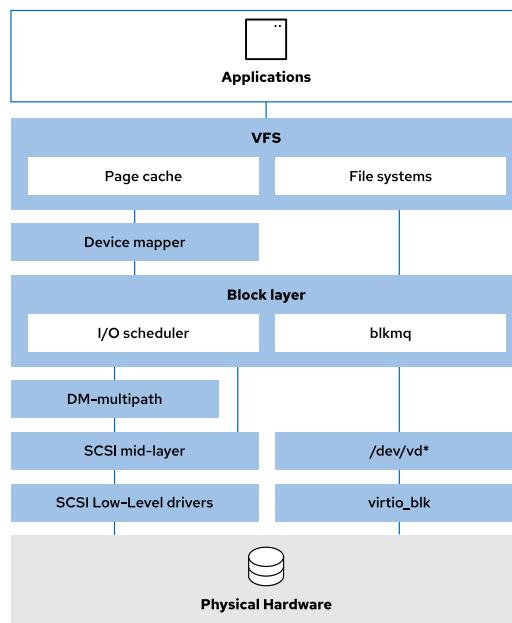


Figure 5.1: The Linux storage stack

The Virtual File System (VFS)

The *Virtual File System* (VFS) provides support for standard POSIX system calls to read and write files. VFS implements a *common file model* so that applications use the same system calls, such as `creat()`, `open()`, `read()`, `write()`, and `mmap()`, to access files without knowing the underlying file system's implementation details. File system implementations such as XFS, ext4, FAT32, and others plug in to VFS as modules and so VFS provides a common abstraction for the data on those file systems.

Linux applications expect that directories are accessed like any other file, but some file systems store directory data differently. VFS ensures that applications access directories as files while using the file system's driver to correctly implement directories in that file system.

VFS maintains caches to improve storage I/O performance, including an *inode cache*, *dentry cache*, *buffer cache*, and a *page cache*. The page cache is dynamically allocated by using free memory, and caches disk blocks during file system reads and writes. The buffer cache is unified with the page cache, except for memory structures that are not backed by files, such as metadata or raw block I/O. The inode cache and directory cache ease access to inodes (file metadata) and directory entries. The `/proc/slabinfo` file records the memory usage of both caches.

Applications that implement their own caching, such as databases, use direct I/O to bypass the page cache. In these applications, the only process to access files or volumes has the `O_DIRECT` flag. Multiple files that attempt to use the same direct I/O files or that conflict with the page cache for that file can cause corruption.

Viewing VFS Memory Use

VFS issues usually involve memory shortages that constrain cache use or `sysctl` tuning mistakes. Use the `free` command for a simple cache overview, or view `/proc/meminfo` for details.

```
[root@host ~]# cat /proc/meminfo
MemTotal:      1860624 kB
MemFree:       1480952 kB
MemAvailable:  1515816 kB
Buffers:        2240 kB
Cached:        164108 kB
```

Cache is dynamically allocated from unused free memory as needed to support I/O. Cache pages are released for applications to use when the system is under memory pressure. The page, inode, and dentry caches can be cleared, but short-term performance drops until the caches are repopulated with frequently accessed data.

```
[root@host ~]# echo 3 > /proc/sys/vm/drop_caches
```

Many kernel `sysctl` tunables in `/proc/sys/vm` affect disk cache performance and memory management. These topics are discussed in detail in the **Red Hat Performance Tuning: Linux in Physical, Virtual, and Cloud (RH442)** training course.

File Systems

File systems provide the logical structures for organizing and naming metadata and data in storage, and how they are secured against compromise and corruption. The default file systems in RHEL, XFS and ext4, share many basic POSIX features to integrate with VFS and the common file model. File systems might be backed by block storage (XFS, ext4, GFS2, FAT32), network storage (NFS, SMB), or might be memory-based *pseudo-filesystems* (procfs, sysfs) or special purpose file systems (tmpfs).

Device Mapper

The device mapper creates mapping tables of blocks from one device layer to blocks in another logical device. Device layers build complex storage structures with LVM volumes, LUKS disk encryption, RAID, and other compatible layers. Device mapper use is optional; you can directly format physical block devices with a file system without using it.

Chapter 5 | Troubleshooting Storage Issues

In this example, an LVM logical volume named /dev/mapper/myvg1-my lv1 is built from two physical volumes, /dev/vdb1 and /dev/vdb2. When initially created with LVM utilities, the device mapper mapped the /dev/vdb1 and /dev/vdb2 physical block device partitions to the /dev/dm-0 higher logical device. Use the dmsetup command to view device mappings:

```
[root@host ~]# dmsetup ls
myvg1-my lv1      (252:0)
[root@servera ~]# ls -l /dev/mapper/myvg1-my lv1
lrwxrwxrwx. 1 root root 7 Sep 30 18:30 /dev/mapper/myvg1-my lv1 -> ../dm-0
[root@host ~]# dmsetup table /dev/mapper/myvg1-my lv1
0 1015808 linear 253:17 2048
1015808 1015808 linear 253:18 2048
```

The /dev/mapper/myvg1-my lv1 volume has 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-my lv1. 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-my lv1, starting at block 1015808. The major:minor numbers 253:17 and 253:18 correspond to /dev/vdb1 and /dev/vdb2:

```
[root@host ~]# ls -l /dev/vdb*
brw-rw----. 1 root disk 253, 16 Sep 30 18:28 /dev/vdb
brw-rw----. 1 root disk 253, 17 Sep 30 18:30 /dev/vdb1
brw-rw----. 1 root disk 253, 18 Sep 30 18:30 /dev/vdb2
```

The resulting /dev/dm-0 logical volume can be represented visually:

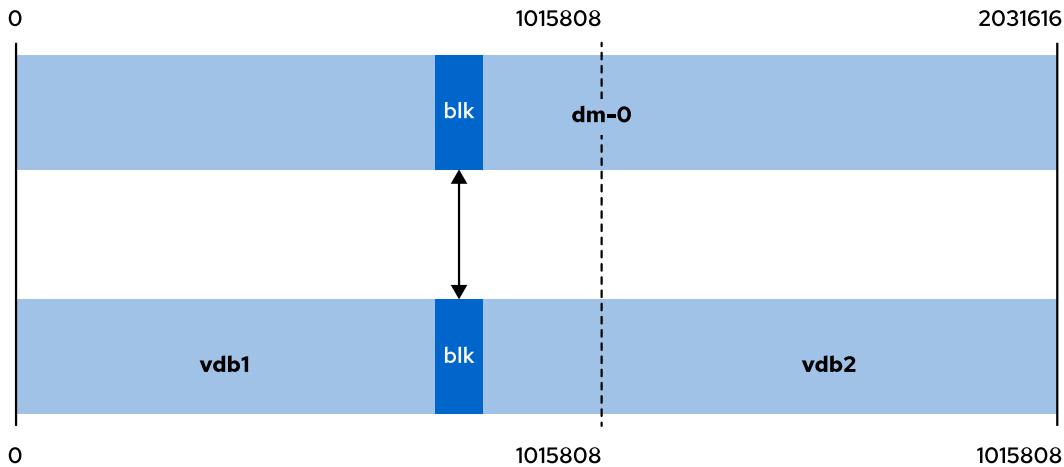


Figure 5.2: Device mapping between a logical volume and its two virtual storage devices

The logical device that the device mapper created in this example (/dev/dm-0) therefore has 2031616 sectors (1015808 sectors from /dev/vdb1 and 1015808 sectors from /dev/vdb2).

Disk Schedulers

Disk schedulers are responsible for ordering the I/O requests that are submitted to a storage device. In Red Hat Enterprise Linux 8, block devices support only *multi-queue scheduling*. This

enables block layer performance to scale with fast solid-state drives (SSDs) and multi-core systems.

Traditional, single-queue schedulers, which were available in RHEL 7 and earlier versions, are removed. These multi-queue schedulers are available in RHEL 8:

```
[root@host ~]# dmesg | grep -i 'io scheduler'
[    1.136832] io scheduler mq-deadline registered
[    1.137038] io scheduler kyber registered
[    1.137273] io scheduler bfq registered
```

none

Implements a first-in first-out (FIFO) scheduling algorithm, merging requests at the generic block layer through a simple last-hit cache.

mq-deadline

Sorts queued I/O requests into a read or write batch and schedules them for execution in increasing logical block addressing (LBA) order. After the scheduler processes a batch, it checks how long write operations have been starved of processor time and schedules the next read or write batch appropriately. This scheduler is especially suitable where read operations are more frequent than write operations.

kyber

Tunes itself to achieve a latency goal by calculating the latencies of every I/O request that is submitted to the block I/O layer.

bfq

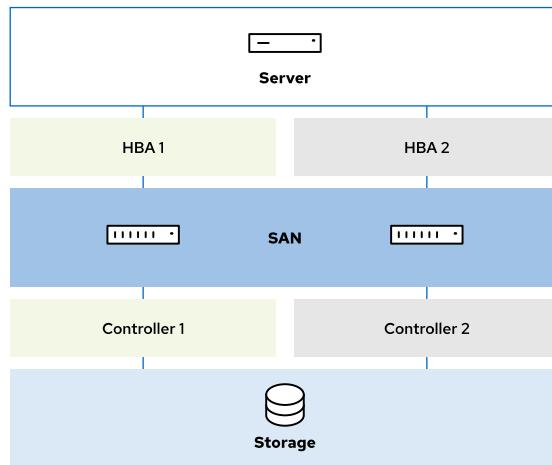
Ensures that a single application never uses all the bandwidth. Focuses on delivering the lowest latency rather than achieving the maximum throughput.

To determine which disk scheduler is currently active on a given block device, view the `/sys/block/device/queue/scheduler` file:

```
[root@host ~]# cat /sys/block/sda/queue/scheduler
[mq-deadline] kyber bfq none
```

Device Mapper Multipath

Device mapper multipath (DM Multipath) configures multiple I/O paths between servers and storage arrays so that they appear as a single device. These paths can be physical connections with separate cables, switches, and controllers. Multipathing aggregates the I/O, creating a new device that consists of the redundant, aggregated paths.

**Figure 5.3: Device mapper multipath**

The RHEL server has two host bus adapters (HBAs) that are attached to a SAN, typically Fibre Channel connections. The SAN connects to separate storage controllers. This configuration supports four possible paths to the back-end storage, improving performance and fault tolerance.

In RHEL, the device mapper generates separate block devices to represent different paths to the same storage device. The `multipathd` daemon and the `multipath` command manage these devices. These paths are accessed through a higher-level multipath block device, which sends the request to a particular path by selecting the block device for that path. Multipath devices can be named for the World Wide ID (WWID) of the storage device, a more convenient name **mpath** plus a sequence or path number, or custom names.

The SCSI Mid-layer

The SCSI mid-layer is a bridge between the SCSI targets that present storage devices and the host bus adapters or hardware interface card drivers that communicate with the storage device. The block device drivers are the SCSI disk (`sd`) and the SCSI CDROM (`sr`) driver. The SCSI mid-layer also provides one SCSI driver for character-based tape devices (`st`) and one for generic SCSI devices such as scanners (`sg`).

All devices that can use or emulate the SCSI protocol can use the SCSI mid-layer. This mid-layer includes seemingly unrelated devices such as SATA devices, USB storage, and virtual machine disks. These devices are presented as SCSI devices and use appropriate device names, such as `/dev/sda`. Some storage devices bypass this layer. For example, the `/dev/vd*` devices are paravirtualized devices that use the `virtio_blk` driver, which does not emulate the SCSI protocol.

Low-level Drivers

Low-level drivers communicate with physical system hardware. Examples include SCSI drivers for Qlogic (`qla2xxx`), Adaptec (`aacraid`), or Emulex (`lpfc`) devices, or local SATA or USB (`libata` and `ahci`) devices. An iSCSI device that uses TCP/IP transport would use a SCSI driver (such as `iscsi_tcp`) before passing its traffic to the network stack. Some paravirtualized devices are presented as SCSI (`virtio_scsi` and `vmw_pvscsi`) disk devices. Paravirtualized drivers such as `virtio_blk` interact directly with the scheduler at the block layer.

The low-level driver receives I/O from the scheduler at the block layer and dispatches it to the storage hardware. The controller takes incoming I/O requests and forwards them to the underlying hardware controller. The driver does not queue the I/O, but tracks the active I/O requests.

Stratis Storage Management

Stratis is a solution that eases local storage management with a focus on simplicity. The service manages pools of physical storage devices, which are created from one or more local disks or partitions. Volumes are created from the pools, with many useful features, such as file system snapshots, thin provisioning, and data tiering. Stratis pools have been tested on these block device types:

- LUKS
- LVM logical volumes
- MD RAID
- DM Multipath
- iSCSI
- HDDs and SSDs
- NVMe devices

Installing and Creating Stratis Pools

Install Stratis packages and enable the service:

```
[root@host ~]# yum install stratisd stratis-cli  
[root@host ~]# systemctl enable --now stratisd
```

Prepare devices for use in pools by erasing any existing file system, partition table, or RAID signatures on each block device:

```
[root@host ~]# wipefs --all block-device
```

Create the Stratis pool by selecting the block devices to use. You can also create Stratis pools in encrypted form, with the kernel key ring as the primary encryption mechanism.

```
[root@host ~]# stratis pool create my-pool block-device
```

You can attach further block devices to a pool to increase the storage capacity for file systems.

Stratis File System

Stratis file systems are thinly provisioned without a fixed total size. If the size of the data approaches the virtual size of the file system, then Stratis grows the thin volume and its XFS file system automatically. Create a Stratis file system on a pool, by using the `stratis fs create` command:

```
[root@host ~]# stratis fs create my-pool my-fs
```

To persistently mount a Stratis file system, add an entry in the `/etc/fstab` file, to use the Stratis file system UUID. To retrieve the Stratis file system UUID, use `lsblk`.

```
[root@servera ~]# lsblk --output=UUID /dev/stratis/my-pool/my-fs
UUID
b65883bf-cd83-420d-bb78-433b6545c053
```



References

A detailed storage stack diagram is available under the CC-BY-SA license at
Linux Storage Stack Diagram

https://www.thomas-krenn.com/en/wiki/Linux_Storage_Stack_Diagram

For further information, refer to the *Managing Storage Devices Guide* at
https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html/managing_storage_devices/index

For further information, refer to the **Managing layered local storage with Stratis** section in *System Design Guide* at
https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/system_design_guide/index

free(1), dmsetup(8), multipath(8), blktrace(8), blkparse(1), btt(1), and stratis(1) man pages

► Guided Exercise

Configuring Storage with Stratis

In this exercise, you set up and manage complex storage configurations integrated by the Stratis high-level system.

Outcomes

You should be able to install and configure file systems with the Stratis service.

Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

This command confirms that the required hosts for this exercise are accessible and have the required devices for the file system configuration.

```
[student@workstation ~]$ lab start storage-overview
```

Instructions

- 1. Log in to `servera` and switch to the `root` user.

```
[student@workstation ~]$ ssh student@servera  
... output omitted...  
[student@servera ~]$ sudo -i  
[sudo] password for student: student  
[root@servera ~]#
```

- 2. Install and enable the Stratis service.

- 2.1. Install the `stratisd` and `stratis-cli` packages.

```
[root@servera ~]# yum install stratisd stratis-cli
```

- 2.2. Ensure that the `stratisd` service is enabled.

```
[root@servera ~]# systemctl enable --now stratisd
```

- 3. Create a Stratis pool named `my-pool` that consists of the `/dev/vdb` and `/dev/vdc` devices.

- 3.1. Wipe devices from any file system, partition table, or RAID signatures.

```
[root@servera ~]# wipefs --all /dev/vdb /dev/vdc
```

- 3.2. Create a Stratis pool.

```
[root@servera ~]# stratis pool create my-pool /dev/vdb /dev/vdc
```

3.3. Verify the Stratis pool.

```
[root@servera ~]# stratis blockdev
Pool Name   Device Node   Physical Size   Tier
my-pool     /dev/vdb       1 GiB        Data
my-pool     /dev/vdc       1 GiB        Data
[root@servera ~]# stratis pool list
Name          Total Physical  Properties
my-pool      2 GiB / 41.63 MiB / 1.96 GiB    ~Ca,~Cr
```

► 4. Create the Stratis file system.

4.1. Create the file system in the pool.

```
[root@servera ~]# stratis fs create my-pool my-fs
```

4.2. Confirm the Stratis file system creation.

```
[root@servera ~]# stratis fs list my-pool
Pool Name  Name  Used   Created           Device           UUID
my-pool    my-fs  546 MiB Oct 04 2021 23:47 /dev/stratis/my-pool/my-fs b6588...
```

4.3. Create a /test directory and mount the Stratis file system.

```
[root@servera ~]# mkdir /test
[root@servera ~]# mount /dev/stratis/my-pool/my-fs /test
```

4.4. Verify the Stratis file system.

```
[root@servera ~]# df -h /test
Filesystem           Size  Used Avail Use% Mounted on
/dev/mapper/stratis-1-8c1...-thin-fs-b6...  1.0T  7.2G 1017G  1% /test
```

► 5. Explore the features of Stratis file systems.

5.1. Inspect the file system format type.

```
[root@servera ~]# blkid /dev/stratis/my-pool/my-fs
/dev/stratis/my-pool/my-fs: UUID="b658..." BLOCK_SIZE="512" TYPE="xfs"
```

5.2. Create a snapshot of the file system and verify it.

```
[root@servera ~]# stratis fs snapshot my-pool my-fs my-fs-snapshot
[root@servera ~]# stratis fs
Pool Name Name           Used     Created          Device
        UUID
my-pool   my-fs            546 MiB Oct 04 2021 23:47 /dev/stratis/my-pool/my-fs
          b6588...
my-pool   my-fs-snapshot  546 MiB Oct 05 2021 00:33 /dev/stratis/my-pool/my-fs-
snapshot  1cf51...
```

5.3. Create a test file and use a snapshot as a file system backup.

```
[root@servera ~]# echo "Hello, I am a test file" > /test/mytestfile
[root@servera ~]# cat /test/mytestfile
Hello, I am a test file
```

Back up the current file system state and verify it.

```
[root@servera ~]# stratis filesystem snapshot my-pool my-fs my-fs-backup
[root@servera ~]# stratis fs
my-pool my-fs            546 MiB Oct 04 2021 23:47 /dev/stratis/my-pool/my-fs
          b6588...
my-pool my-fs-snapshot  546 MiB Oct 05 2021 00:33 /dev/stratis/my-pool/my-fs-
snapshot  1cf51...
my-pool my-fs-backup    546 MiB Oct 05 2021 00:39 /dev/stratis/my-pool/my-fs-backup
          d25b5...
```

Create a /bkp directory and mount the backup file system on it.

```
[root@servera ~]# mkdir /bkp
[root@servera ~]# mount /dev/stratis/my-pool/my-fs-backup /bkp
[root@servera ~]# df -h /bkp
Filesystem                      Size  Used Avail Use% Mounted on
/dev/mapper/stratis-1-8cc...-thin-fs-d2...  1.0T  7.2G 1017G  1% /bkp
[root@servera ~]# cat /bkp/mytestfile
Hello, I am a test file
```

5.4. Use the file system backup to restore the original file system.

Unmount and destroy the file system.

```
[root@servera ~]# umount /dev/stratis/my-pool/my-fs
[root@servera ~]# stratis filesystem destroy my-pool my-fs
[root@servera ~]# stratis fs
Pool Name Name           Used     Created          Device
        UUID
my-pool   my-fs-snapshot  546 MiB Oct 05 2021 00:33 /dev/stratis/my-pool/my-fs-
snapshot  1cf51...
my-pool   my-fs-backup    546 MiB Oct 05 2021 00:39 /dev/stratis/my-pool/my-fs-
backup    d25b5...
```

Use the original file system name to back up the my-fs-backup snapshot and mount it on the /test directory. Confirm the restore.

```
[root@servera ~]# stratis filesystem snapshot my-pool my-fs-backup my-fs
[root@servera ~]# stratis fs
my-pool my-fs-snapshot 546 MiB Oct 05 2021 00:33 /dev/stratis/my-pool/my-fs-
snapshot 1cf51...
my-pool my-fs-backup 546 MiB Oct 05 2021 00:39 /dev/stratis/my-pool/my-fs-backup
d25b5...
my-pool my-fs 546 MiB Oct 05 2021 00:53 /dev/stratis/my-pool/my-fs
ca12d...
[root@servera ~]# mount /dev/stratis/my-pool/my-fs /test
[root@servera ~]# cat /test/mytestfile
Hello, I am a test file
```

► **6.** Return to workstation as the student user.

```
[root@servera ~]# exit
[student@servera ~]$ exit
[student@workstation ~]$
```

Finish

On the **workstation** machine, use the **lab** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish storage-overview
```

This concludes the guided exercise.

Recovering from File System Corruption

Objectives

After completing this section, you should be able to detect and recover from file system corruption.

File System Choices

Red Hat Enterprise Linux provides several file systems to fit various workloads and use cases. In RHEL 8, the default is XFS, and ext4 remains popular.



Note

Although the ext4 driver can read and write to ext2 and ext3 file systems, Red Hat recommends to use only ext4 for support and stability.

XFS and ext4 are journaling file systems. Uncommitted file activity is written to a journal first. If the system loses power or cannot finish transactions, then the journal can recover unsaved information and avoid file system metadata corruption. Although journaling improves fault tolerance, it does not mitigate all file system corruption. File system corruption can still occur due to hardware failure, human error, or software bugs.

Identifying File System Corruption

When file system corruption occurs or is suspected, use file system checking tools to check the file system for consistency. When the data corruption is identified, use file system repair tools to restore file system consistency. Use recommended practices when recovering from file system corruption:

- If a hardware failure caused the data corruption, then resolve the hardware issue before checking or repairing a file system. For example, if a storage disk is failing, then move the file system to a functional disk before performing file system maintenance. Moving a corrupted file system requires copying the data partition block by block with utilities such as dd.
- Always check a file system before repairing it. This check serves as a dry run for the file system repair, and can report discovered file system issues and suggested corrective actions.
- A file system must be unmounted before file system administration. To successfully restore file system consistency, file system repair tools must have exclusive access to the partition table. Unmounting a file system ensures that non-storage operations, other than the file system repair, can occur. Running a file system repair on a mounted file system often leads to further data corruption.

Checking ext4 File Systems

After a power loss or system crash, RHEL 8 systems automatically invoke the e2fsck utility to recover the journal of ext4 file systems. On replaying the uncommitted changes in the journal, e2fsck marks whether the file systems are consistent. If inconsistencies are discovered, e2fsck fully checks the file system and requests user intervention if it cannot safely resolve an issue. Both the e2fsck and the dumpe2fs utilities are part of the e2fsprogs package.

Chapter 5 | Troubleshooting Storage Issues

Administrators can manually execute e2fsck to check a file system. Unmount the file system, and then run e2fsck with the -n option and the device name for the file system. The -n option places the file system in read-only mode, and answers no to all questions during the file system check.

```
[root@host ~]# umount /dev/vdb1
[root@host ~]# e2fsck -n /dev/vdb1
```

Checking a file system requires a usable superblock. If the default superblock location is corrupted, then locate a backup copy of the superblock to use for the file system check.

```
[root@host ~]# 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 when created. To determine the location of backup superblocks, use the dumpe2fs utility.

```
[root@host ~]# 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
```

After locating the backup superblocks, use the -b option and select one to use as an alternative superblock during the file system check.

```
[root@host ~]# 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@host ~]# echo $?
0
```

Determine the file system check result with the e2fsck command's exit status. The reported exit status is the sum of the triggered exit codes.

Exit code	Description
0	No errors.
1	File system errors corrected.
4	File system errors uncorrected.
8	Operational error.
16	Usage error.
32	Cancelled by user request.
128	Shared library error.

Checking XFS File Systems

Unlike ext4 file systems, RHEL 8 systems do not automatically initiate checks and repairs on XFS file systems. Check XFS file systems manually with the `xfs_repair` utility, which the `xfsporgs` package provides. The `xfs_repair` command is invoked with the `-n` option to check a file system. With the `-n` option, `xfs_repair` scans the file system and only reports potential repairs without taking corrective action. This example checks the XFS file system on `/dev/vdb1`:

```
[root@host ~]# xfs_repair -n /dev/vdb1
```

The `xfs_repair` command does not execute on an XFS file system that does not have a clean journal log. Journal logs can be corrupted by unclean system shutdowns or system crashes. Mount and unmount an XFS file system to clean its journal logs.

Like ext4 file systems, an XFS file system check can fail to execute due to a corrupted primary superblock. However, unlike e2fsck, with `xfs_repair` you do not need to locate and specify an alternative superblock. The `xfs_repair` command automatically scans the XFS file system, locates a secondary superblock, and uses it to recover the primary superblock.

```
[root@host ~]# 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@host ~]# echo $?
0
```

An XFS file system check returns an exit code of 1 if file system corruption was detected, and an exit code of 0 if the file system is clean.

Repairing File Systems

After a file system check, identify the file system errors and the corrective actions to take. Repair the file system with the correct utility for your file system.

Repairing ext4 File Systems

The ext4 file systems are repaired with the same e2fsck command that checks a file system. Without the -n option, e2fsck applies all safe corrective actions. For operations that cannot be done safely, e2fsck prompts the user for whether to take the action.

The file system must be unmounted during file system repair to ensure file system consistency and to prevent further corruption. Depending on the severity of the file system corruption, you might choose to execute e2fsck with additional options.

Option	Description
-b <i>location</i>	Use an alternative superblock at the specified location.
-p	Automatically repair the file system. Prompt user only for problems that cannot be safely fixed.
-v	Verbose mode.
-y	Run in noninteractive mode and answer yes to all questions. This option cannot be used with the -p or -n options.

This example repairs an ext4 file system with the -y and -b options to execute a noninteractive file system check with an alternative superblock.

```
[root@host ~]# 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@host ~]# echo $?
1
```

Determine the result of the file system repair by the e2fsck exit code.

Repairing XFS File Systems

XFS file systems are repaired with the same `xfs_repair` command to check a file system. Without the `-n` option, `xfs_repair` takes all safe corrective actions. Unmount the file system before repairing it to ensure file system consistency and to prevent further corruption.

An `xfs_repair` can be executed only on an XFS file system with a clean journal log. If mounting and unmounting an XFS file system does not result in a clean journal, then the journal might be corrupted. Because a clean log is required, this scenario might require adding the `-L` option to the `xfs_repair` command, which clears the journal log. When the journal is unrecoverable, this step is necessary to proceed, but this option discards all journal metadata, which might result in further issues with recently written data.

Unlike `e2fsck`, `xfs_repair` is not an interactive utility. When initiated, the XFS file system repair will perform all operations automatically without any user input. An `xfs_repair` always returns an exit code of `0` on completion of the file system repair.

```
[root@host ~]# 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 "/"
    subscription-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@host ~]# echo $?
0
```

During an XFS file system repair, you might discover files and directories in use with allocated inodes but 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 at the root of that file system. If files are missing after a file system repair, then review whether they were relocated to the `lost+found` directory.

**Note**

RHEL 8 does not provide any automated ways to recover files in the `lost+found` directory. To recover files from `lost+found`, use the `mv` command to move files manually to their correct directory.

File System Backup and Recovery

File system checking and repair cannot guarantee data recovery, and do not replace a tested backup and recovery strategy. If severely damaged inodes or directories are encountered during file system repair, then the tools might permanently discard the inodes that cannot be fixed. Discarded data can be recovered only from recent backups of the file system's 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 exercise, you verify a file system and repair file system corruption.

Outcomes

You should be able to check and repair an XFS file system.

Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

```
[student@workstation ~]$ lab start storage-filesystem
```

This command creates the necessary file systems and files for this exercise.

Instructions

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 and then mount the file system at `/mnt/etc_restore`. If the file system repair relocates orphaned files to the file system's `lost+found` directory, then use the backup file, `/root/etc.tgz`, to determine the original file location and name. Restore the orphaned file to its proper location.

- 1. Log in to `servera` and switch to the `root` user.

```
[student@workstation ~]$ ssh student@servera  
...output omitted...  
[student@servera ~]$ sudo -i  
[sudo] password for student: student  
[root@servera ~]#
```

- 2. Check the XFS file system on the `/dev/vdb1` partition.

- 2.1. Unmount the `/dev/vdb1` file system, because a file system check cannot run on a mounted file system.

```
[root@servera ~]# umount /mnt/etc_restore
```

- 2.2. Use the `xfs_repair` command with the `-n` option to check the file system with no corrective actions taken.

```
[root@servera ~]# xfs_repair -n /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 (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 0x564c8914a8e8, xfs_dir3_block 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 ...
Invalid inode number 0x499602d2
xfs_dir_ino_validate: XFS_ERROR_REPORT
Metadata corruption detected at 0x564c8914a8e8, xfs_dir3_block block 0xa7e0/0x1000
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 reported errors with information that might be helpful for file system recovery.

► 3. 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

```

Chapter 5 | Troubleshooting Storage Issues

```
Metadata corruption detected at 0x56258b0938e8, xfs_dir3_block 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 inodes...
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
```

- ▶ 4. Mount the repaired file system at /mnt/etc_restore to analyze the file system repair results.

```
[root@servera ~]# mount /dev/vdb1 /mnt/etc_restore
```

- ▶ 5. Determine whether the file system's lost+found directory has orphaned files.

```
[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
```

- ▶ 6. Determine the correct name and location of the orphaned file in lost+found with the output that was generated during the file system repair.

- 6.1. Based on the generated output, 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...
```

Chapter 5 | Troubleshooting Storage Issues

```
entry at block 0 offset 456 in directory inode 144973 has illegal name "/"
ubscription-manager": - process newly discovered i
nodes...
... Output omitted ...
```

- 6.2. Based on the generated output, 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
```

- 7. Use the file system's content backup at `/root/etc.tgz` to verify the orphaned file's name and location.

- 7.1. Extract the backup file to view its contents for comparison.

```
[root@servera ~]# tar -C /tmp -xzf /root/etc.tgz
```

- 7.2. Verify the contents of the orphaned file against the backup copy of the identified file.

```
[root@servera ~]# cd /tmp/etc/security/console.apps
[root@servera console.apps]# diff -s subscription-manager /mnt/etc_restore/lost
+found/145282
Files subscription-manager and /mnt/etc_restore/lost+found/145282 are identical
```

- 8. When verified, move the orphaned file to its proper location.

```
[root@servera console.apps]# mv /mnt/etc_restore/lost+found/145282
subscription-manager
mv: overwrite 'subscription-manager'? y
```

- 9. Return to workstation as the student user.

```
[root@servera ~]# exit
[student@servera ~]$ exit
[student@workstation ~]$
```

Finish

On the **workstation** machine, use the **lab** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish storage-filesystem
```

This concludes the guided exercise.

Repairing LVM Issues

Objectives

After completing this section, you should be able to revert an LVM storage misconfiguration.

Logical Volume Management

Logical volume management (LVM) provides administrators with a comprehensive storage virtualization framework. One or more data storage devices (physical volumes) are aggregated into a storage pool (volume group), from which volumes are carved out to act as block devices (logical volumes). LVM supports data striping or mirroring, or both, between physical volumes.

Similar to `dm-multipath`, logical volume management uses the kernel Device Mapper subsystem to create the LVM devices. Logical volumes are created in the `/dev/` directory as `dm-*` device nodes, but with symlinks in both the `/dev/mapper/` and `/dev/<vgname>/` directories with persistent names.

Configuration Files

LVM behavior is configured in the `/etc/lvm/lvm.conf` file. Settings in this file control locking behavior, device scanning, names of multipathed physical volumes, and other LVM attributes. The following table lists some common options:

<code>/etc/lvm/lvm.conf</code>	Options
<code>dir</code>	Which directory to scan for physical volume device nodes.
<code>obtain_device_list_from</code>	If udev should be used to obtain a list of eligible block devices for scanning.
<code>preferred_names</code>	A list of regular expressions to show which device names should have preference when displaying devices. If multiple device names (nodes) are found for the same PV, then the LVM tools give preference to those names that appear earlier in this list.
<code>filter</code>	A list of regular expressions prefixed with <code>a</code> for Add or <code>r</code> for Remove. This list determines which device nodes are scanned for the presence of a PV signature. The default is <code>["a/.*/"]</code> , which adds every device. You can use this option to remove a device that should never be scanned.
<code>backup</code>	This setting determines whether to store a text-based backup of volume group metadata after every change. You can use this backup to restore a volume group if the on-disk metadata gets corrupted.
<code>backup_dir</code>	This setting specifies where to store the backup of volume group metadata.

/etc/lvm/lvm.conf	Options
archive	This setting determines whether to archive old volume group configurations or layouts to use later to revert changes.
archive_dir	This setting specifies where to store archives of old configurations.
retain_min	This setting specifies the minimum number of archives to store.
retain_days	This setting specifies the minimum number of days to keep archive files.

Reverting LVM Changes

During routine storage management, administrators might make a change to a volume group that they immediately regret, for example, shrinking a logical volume before shrinking the file system on that logical volume. In this situation, the administrator can try to extend the logical volume to the previous size; however, with possible corruption of the file system on the resized logical volume, the same blocks on the disk are not guaranteed to be available for use.

The LVM archive feature can be configured to store copies of volume group metadata. If the `archive` option is set to `1` in the `/etc/lvm/lvm.conf` file, then the LVM tools create an archived copy of the current volume group metadata *before* making any changes on disk. In a default configuration, these archives are in the `/etc/lvm/archive/` directory. For a list of all archived metadata for a volume group, examine the files in the `/etc/lvm/archive/` directory.

Alternatively, you can display the same information with the `vgcfgrestore -l <vgname>` command. In this example, the **File:** line provides the name and location of the metadata file for the described action. Carefully interpret the **Description:** lines to determine the order in which the archive is created.

```
[root@host ~]# vgcfgrestore -l vg_example
File: /etc/lvm/archive/vg_example_00000-943759797.vg ①
VG name: vg_example
Description: Created before executing 'vgcreate vg_example /dev/sda5'
Backup Time: Mon Oct 4 07:01:47 2021

File: /etc/lvm/archive/vg_example_00001-1528176681.vg ②
VG name: vg_example
Description: Created before executing 'lvcreate -L 1G -n lv_example vg_example'
Backup Time: Mon Oct 4 07:02:00 2021

File: /etc/lvm/archive/vg_example_00002-1484695080.vg ③
VG name: vg_example
Description: Created before executing 'lvresize -L -256M /dev/vg_example/
lv_example'
Backup Time: Mon Oct 4 07:02:34 2021

File: /etc/lvm/backup/vg_example ④
```

```
VG name:      vg_example
Description:  Created after executing 'lvresize -L -256M /dev/vg_example/
lv_example'
Backup Time:  Mon Oct 4 07:02:34 2021
```

The archive and backup files contain metadata for the following actions:

- ➊ The `vg_example` volume group is created.
- ➋ An `lv_example` logical volume is created in the volume group.
- ➌ An archive of the current volume group is created *before* reducing the logical volume by 256M.
- ➍ A backup of the current volume group is created *after* reducing the logical volume by 256M.

To undo an LVM change:

1. Unmount any file systems that were created on the logical volume to ensure that the logical volume is not in current use.
2. To restore the volume group, use the `vgcfgrestore` command with the `-f <archive_file>` option.

The following example uses the metadata file that was created *before* reducing the logical volume size.

```
[root@host ~]# vgcfgrestore -f /etc/lvm/archive/vg_example_00002-1484695080.vg
vg_example
Restored volume group vg_example
```

In some scenarios, it might be necessary to deactivate and then reactivate the logical volume to ensure that all changes are also updated in memory.

```
[root@host ~]# lvchange -an /dev/vg_example/lv_example
[root@host ~]# lvchange -ay /dev/vg_example/lv_example
```



References

`lvm.conf(5)`, `vgbackup(8)`, and `vgcfgrestore(8)` manual pages

For more information, refer to the *Red Hat Enterprise Linux 8 Configuring and Managing Logical Volumes Guide* at

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/configuring_and_managing_logical_volumes/index

► Guided Exercise

Repairing LVM Issues

In this lab, you troubleshoot and resolve an LVM issue.

Outcomes

You should be able to identify and revert the cause of an LVM failure.

Before You Begin

As the student user on the workstation machine, use the `lab start storage-lvm` command to prepare your system for this exercise.

```
[student@workstation ~]$ lab start storage-lvm
```

Instructions

A request was submitted to allocate an additional 20 MiB of storage to the `/mnt/lvm` directory on `servera`. After the request was completed, a user reported that this directory is missing content. The user suggests that critical files were deleted during the requested allocation. The problem was escalated and you are assigned to identify the root cause of the problem. When root cause analysis is complete, restore the system to proper working order.

- 1. Log in to `servera` and switch to the `root` user.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

- 2. Attempt to list the content of the `/mnt/lvm` directory to confirm the issue that the user reported.

```
[root@servera ~]# ll /mnt/lvm
total 0
```

The directory is empty, which means that the content is missing.

- 3. Check the contents of the `/etc/fstab` file to determine whether the directory contents mount on a separate file system.

```
[root@servera ~]# grep /mnt/lvm /etc/fstab
/dev/vg00/lvol0      /mnt/lvm      xfs      defaults      0 0
```

- 4. Verify that the file system on the logical volume is mounted at `/mnt/lvm`.

```
[root@servera ~]# mount | grep /mnt/lvm
[root@servera ~]#
```

- ▶ 5. Because the file system is not mounted, attempt to mount it manually.

```
[root@servera ~]# mount /mnt/lvm
mount: /mnt/lvm: can't read superblock on /dev/mapper/vg00-lvol0.
```

- ▶ 6. Consult the list of metadata backup and archive files for the volume group where the /dev/vg00/lvol0 logical volume resides to determine any changes to the logical volume configuration.

```
[root@servera ~]# vgcfgrestore -l vg00
...output omitted...

File: /etc/lvm/archive/vg00_00002-1457658928.vg
VG name: vg00
Description: Created before executing 'lvresize -L20M -f /dev/vg00/lvol0'
Backup Time: Mon Oct 4 23:55:42 2021

File: /etc/lvm/backup/vg00
VG name: vg00
Description: Created after executing 'lvresize -L20M -f /dev/vg00/lvol0'
Backup Time: Mon Oct 4 23:55:42 2021

...output omitted...
```

- ▶ 7. Analyze the descriptions of each archive log and identify the archive that existed before the space allocation on the logical volume. Determine whether the recorded command in the archive log caused the issue that the user observed.

```
...output omitted...

File: /etc/lvm/archive/vg00_00002-1457658928.vg
VG name: vg00
Description: Created before executing 'lvresize -L20M -f /dev/vg00/lvol0'
Backup Time: Mon Oct 4 23:55:42 2021

...output omitted...
```

Rather than allocating another 20 MiB to the logical volume with the **-L+20M** option, the administrator erroneously reduced the logical volume to 20 MiB.

- 8. 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_00002-1457658928.vg vg00
Volume group vg00 has active volume: lvol0.
WARNING: Found 1 active volume(s) in volume group "vg00".
Restoring VG with active LVs, may cause mismatch with its metadata.
Do you really want to proceed with restore of volume group "vg00", while 1
volume(s) are active? [y/n]: y
Restored volume group vg00
```

- 9. Deactivate and reactivate the /dev/vg00/lvol0 logical volume to ensure that everything is updated.

```
[root@servera ~]# lvchange -an /dev/vg00/lvol0
[root@servera ~]# lvchange -ay /dev/vg00/lvol0
```

- 10. Mount the file system to determine whether the issue is 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
```

- 11. Correctly fulfill the user's request by allocating another 20 MiB to the /dev/vg00/lvol0 file system.

11.1. Resize the logical volume:

```
[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.
```

11.2. Increase the file system on the logical volume:

```
[root@servera ~]# xfs_growfs /dev/vg00/lvol0
meta-data=/dev/mapper/vg00-lvol0 isize=512    agcount=2, agsize=5120 blks
          =                      sectsz=512   attr=2, projid32bit=1
          =                      crc=1      finobt=1, sparse=1, rmapbt=0
          =                      reflink=1
data     =                      bsize=4096   blocks=10240, imaxpct=25
          =                      sunit=0     swidth=0 blks
naming   =version 2           bsize=4096   ascii-ci=0, ftype=1
log      =internal log        bsize=4096   blocks=1368, 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
```

- 12. Return to workstation as the student user.

```
[root@servera ~]# exit  
[student@servera ~]$ exit  
[student@workstation ~]$
```

Finish

On the **workstation** machine, use the **lab** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish storage-lvm
```

This concludes the guided exercise.

Resolving Storage Device Encryption Issues

Objectives

After completing this section, you should be able to recover data from an encrypted device.

Using LUKS to Encrypt Devices

Linux Unified Key Setup (LUKS) is a specification for block device encryption. In Red Hat Enterprise Linux 8, LUKS2 is the default LUKS encryption format. A LUKS-encrypted block device must be decrypted with a passphrase or key before the contained file systems can be mounted.



Note

The legacy LUKS1 format remains fully supported in RHEL 8 for compatibility with earlier RHEL releases.

Add entries to the `/etc/crypttab` and `/etc/fstab` files to persistently mount an encrypted device. The `/etc/crypttab` file must reference the device and, if used, its key file.

- The system consults the `/etc/crypttab` file during booting to determine which devices to decrypt. The file lists one device per line, with space-separated fields.

```
name  /dev/<device>  /path/to/keyfile
```

- `name` is the name that the device mapper creates for the decrypted device.
- `/dev/<device>` is the underlying encrypted device.
- `/path/to/keyfile` is the key file for decrypting the device. If the last field is blank or set to none, then the user is prompted for the decryption password during boot.
- The system decrypts the device, by using the passphrase from the user or the key file, and names the device `/dev/mapper/name`, with the specified name in `/etc/crypttab`.
- The system mounts the file system in the decrypted mapped device, by using the relevant entry in `/etc/fstab`. The entry can reference either the device mapper name or the device's UUID.

```
# Using device mapper name  
/dev/mapper/name  /secret  ext4  defaults  1 2  
  
# Using device UUID  
"2460ab6e-e869-4011-acae-31b2e8c05a3b"  /another-secret  ext4  defaults  1 2
```

Troubleshooting LUKS Devices

Manually entering commands and passwords can decrypt LUKS-encrypted devices and mount their filesystems. When the automated persistent mounting fails, verify the file configurations and attempt to access and mount the device from the command line. Failures are caused by either an incorrect file configuration or an incorrect password or key.

Incorrect /etc/crypttab Configuration

An incorrect /etc/crypttab entry causes decryption issues. Entries in this file state how encrypted devices are decrypted and how they are named after decryption.

The fields in an /etc/crypttab entry are the device mapper name, the encrypted device, and the path to the key file (or blank or "none" to use a passphrase). If decryption fails, then verify the second and third fields. An incorrect encrypted device path or an incorrect key file causes a decryption failure.

If the decryption is successful, but mounting the device fails, then verify the first field. A mismatch between the specified name in /etc/crypttab and the device mapper name in /etc/fstab causes the device's file system to fail to mount. The `dmsetup ls --crypt` command lists the encrypted mapped devices in use.

```
[root@host ~]# dmsetup ls --target crypt
example-device (252, 0)
```

Decryption Failure

Devices might fail decryption due to problems with a key or passphrase.

- The LUKS passphrase is forgotten or changed to an unknown password.

The LUKS2 format offers 32 key slots for encrypted devices. If other keys or passphrases exist, then they can reset the forgotten or unknown password. To display associated keys, use the `cryptsetup luksDump` command.

```
[root@host ~]# cryptsetup luksDump /dev/vdb
LUKS header information
Version:          2
Epoch:            3
Metadata area:   16384 [bytes]
Keyslots area:   16744448 [bytes]
UUID:             1bb6e35d-c1dd-4afa-a028-ee4fe8f4c5ef
Label:            (no label)
Subsystem:        (no subsystem)
Flags:            (no flags)

Data segments:
  0: crypt
    offset: 16777216 [bytes]
    length: (whole device)
    cipher: aes-xts-plain64
    sector: 512 [bytes]

Keyslots:
  0: luks2
    Key:      512 bits
    Priority: normal
    Cipher:   aes-xts-plain64
    Cipher key: 512 bits
    PBKDF:    argon2i
    Time cost: 4
    Memory:   930312
```

```
Threads: 2
Salt: aa 78 ac dc 79 b7 3a 1b 77 dc 7d 7a 92 59 53 d7
       1b 0d 13 94 3d ea 30 48 f8 97 db 8e fd d4 50 4a
AF stripes: 4000
AF hash: sha256
Area offset:32768 [bytes]
Area length:258048 [bytes]
Digest ID: 0
...output omitted...
```

If a backup of the LUKS header exists, then restoring it can revert to a previously working passphrase.

- The LUKS header is corrupted. LUKS stores a metadata header and key slots at the beginning of each encrypted device. A corrupted LUKS header can render the encrypted data inaccessible. If a known-good backup of the LUKS header exists, then restore the header from the backup.
- The needed hash functions and ciphers to execute the decryption are not in the kernel. View the contents of `/proc/crypto` for a list of installed cryptographic ciphers that the kernel supports.

Incorrect /etc/fstab Configuration

The decrypted mapped device is configured for mounting in `/etc/fstab`. For an incorrectly specified device name, the mount fails. A common mistake is to configure the encrypted device name, instead of the decrypted name, for mounting.

Restoring LUKS Headers

For commonly encountered LUKS issues, LUKS header backups can mean the difference between a simple fix and permanently unrecoverable data. You should routinely back up LUKS-encrypted device headers and practice the procedure to restore headers.

LUKS Header Backup

Backup LUKS headers using the `cryptsetup` tool with the `luksHeaderBackup` subcommand. Specify the backup file location with the `--header-backup-file` option.

```
[root@host ~]# cryptsetup luksHeaderBackup /path/to/encrypted/device
--header-backup-file /path/to/backup/file
```

Make LUKS header backups before every administrative task on LUKS-encrypted devices.

Testing and Recovering LUKS Headers

If an encrypted device's LUKS header is backed up, then restore a backup to retrieve forgotten passwords or corrupted headers. Before restoring a header, always back up the existing header even if the header appears corrupted or is associated with an unknown password.

For multiple backups of an encrypted device, an administrator must identify which one to restore. Rather than restoring a header and attempting a decryption, an administrator can conduct a trial decryption with a header file.

```
[root@host ~]# cryptsetup luksOpen /path/to/encrypted/device DECRYPTED-NAME  
--header /path/to/backup/file  
Enter passphrase for /path/to/encrypted/device:
```

If the test succeeds, then you can restore the header with the `cryptsetup` command and the `luksHeaderRestore` subcommand.

```
[root@host ~]# 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](#)

Encrypting Block Devices Using LUKS

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html/security_hardening/encrypting-block-devices-using-luks_security-hardening

What is LUKS disk encryption and how can it be implemented?

<https://access.redhat.com/solutions/100463>

Cryptsetup project page

<https://gitlab.com/groups/cryptsetup>

► Guided Exercise

Resolving Storage Device Encryption Issues

In this exercise, you troubleshoot and resolve problems with mounting an encrypted file system.

Outcomes

You should be able to troubleshoot and recover from a failure to mount an encrypted volume.

Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

```
[student@workstation ~]$ lab start storage-luks
```

Instructions

A LUKS-encrypted volume that is mounted at `/mnt/secure` on `servera` was protected by the password `R3dH@t123`. After a routine change, an administrator changed the password to `newP@ss321`. Unfortunately, the administrator now reports that the new password does not work on the encrypted volume. A recent backup of the volume's LUKS header is available at `/root/luks_header_backup`. Assess the situation and restore the volume to working order. If needed, use the LUKS header backup to restore the encrypted volume's function.

- 1. Log in to `servera` and switch to the root user.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

- 2. Determine the system's encrypted volume configuration.

- 2.1. View `/etc/fstab` to determine the volume that is configured to mount at `/mnt/secure`.

```
[root@servera ~]# grep /mnt/secure /etc/fstab
/dev/mapper/secure      /mnt/secure      xfs      defaults      1 2
```

- 2.2. View `/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
```

- ▶ 3. Assess the current state of the encrypted volume. Use the `dmsetup` command to determine whether a mapped device exists for the encrypted block device. Use the `--target crypt` subcommand to filter for an encrypted block device.

```
[root@servera ~]# dmsetup ls --target crypt
No devices found
```

- ▶ 4. Because the mapped device does not exist, use the `cryptsetup luksOpen` command to manually open the encrypted volume and create the associated mapped device that is specified in `/etc/crypttab`. Try decrypting the encrypted volume with the new password that the previous administrator set. If that password does not work, then try the old password.

```
[root@servera ~]# cryptsetup luksOpen /dev/vdb1 secure
Enter passphrase for /dev/vdb1: newP@ss321
No key available with this passphrase.
Enter passphrase for /dev/vdb1: R3dH@t123
No key available with this passphrase.
Enter passphrase for /dev/vdb1: R3dH@t123
No key available with this passphrase.
```

- ▶ 5. Use the `cryptsetup luksDump` command to determine whether the encrypted volume has other key slots.

```
[root@servera ~]# cryptsetup luksDump /dev/vdb1
LUKS header information
Version:      2
Epoch:        5
Metadata area: 16384 [bytes]
Keyslots area: 16744448 [bytes]
UUID:         d132238a-6eb7-4025-9ea0-2a77837aa755
Label:        (no label)
Subsystem:    (no subsystem)
Flags:        (no flags)

Data segments:
  0: crypt
  offset: 16777216 [bytes]
  length: (whole device)
  cipher: aes-xts-plain64
  sector: 512 [bytes]

Keyslots:
  1: luks2
  Key:      512 bits
  Priority: normal
  Cipher:   aes-xts-plain64
  Cipher key: 512 bits
```

```
PBKDF:      argon2i
Time cost:  4
Memory:    930312
Threads:   2
Salt:       14 01 41 d5 1f de 0f 3c 3a 95 05 9e bc 61 00 d8
           c9 a3 e7 87 09 ab c1 dd 02 45 fc 28 a9 71 f3 0e
AF stripes: 4000
AF hash:    sha256
Area offset:290816 [bytes]
Area length:258048 [bytes]
Digest ID:  0
Tokens:
Digests:
 0: pbkdf2
Hash:      sha256
Iterations: 112604
Salt:       ff 16 a5 6c 70 4a 6f 51 ed 84 53 2a ef 24 5d 52
           cb 0a 51 18 3d c3 e4 95 54 01 10 b3 24 32 42 bb
Digest:    69 ba fe 4e 19 70 37 c4 9a ba 01 10 fe 35 18 f2
           9e b8 ed e6 5d e3 bd b5 3f 17 60 2e a1 67 51 d7
```

The volume does not have any other keys.

- ▶ 6. Because no known passwords or other keys are available in the volume, the remaining option is to reinstate the old password by restoring the LUKS header from the /root/luks_header_backup backup file.
 - 6.1. As a recommended step to reverse live mistakes, back up the current LUKS header before replacing it with the backup LUKS header.

```
[root@servera ~]# cryptsetup luksHeaderBackup /dev/vdb1
--header-backup-file /root/luks_header_new
```

- 6.2. Restore the old LUKS header from the /root/luks_header_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
```

- ▶ 7. Use the cryptsetup luksOpen command to manually open the encrypted volume and create the associated mapped device in the /etc/crypttab file. Try decrypting the encrypted volume with the old password.

```
[root@servera ~]# cryptsetup luksOpen /dev/vdb1 secure
Enter passphrase for /dev/vdb1: R3dH@t123
```

- 8. Use the `dmsetup` command to validate that the secure mapped device name now exists for the encrypted block device. Use the `--target crypt` subcommand to filter for encrypted block devices only.

```
[root@servera ~]# dmsetup ls --target crypt  
secure  (253, 0)
```

- 9. Verify that the mapped device for the encrypted volume can 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,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=32k,noquota)
```

- 10. Return to workstation as the student user.

```
[root@servera ~]# exit  
[student@servera ~]$ exit  
[student@workstation ~]$
```

Finish

On the `workstation` machine, use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish storage-luks
```

This concludes the guided exercise.

Resolving iSCSI Issues

Objectives

After completing this section, you should be able to identify and repair iSCSI issues.

Introducing iSCSI

Internet Small Computer System Interface (iSCSI) is a TCP/IP-based protocol for sending SCSI commands over IP networks. iSCSI emulates a high-performance local SCSI storage bus, enabling block-based storage devices on a server to appear as local devices on a client. 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.

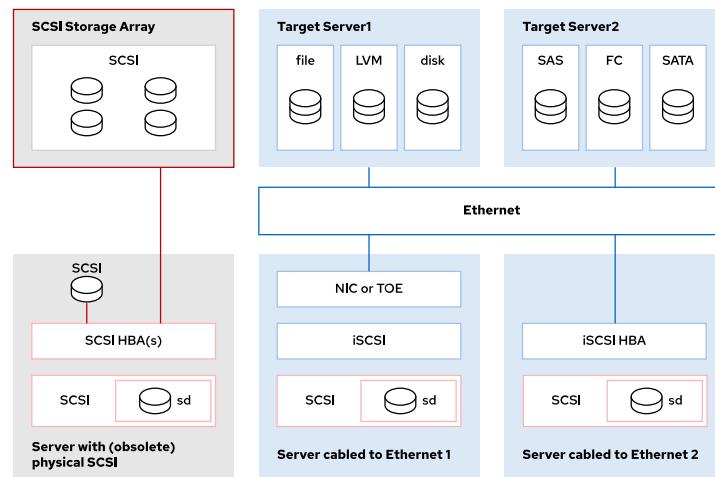


Figure 5.4: SCSI and iSCSI block storage topologies

iSCSI Component Terminology

Term	Description
initiator	An iSCSI client, typically available as software. Hardware initiators are also available in the form of iSCSI Host Bus Adapters (HBAs). Initiators must have unique names (see IQN).
target	An iSCSI storage resource on an iSCSI server. A target provides one or more numbered block devices, commonly referred to as <i>logical unit numbers</i> (see LUN). Typically, a target provides one device and a server provides multiple targets. Targets must have unique names (see IQN).

Term	Description
IQN	<p>A unique worldwide name to identify initiators and targets. IQNs have the following format: <code>iqn.YYYY-MM.com.reversed.domain:name_string</code></p> <p><code>iqn</code> – This name uses a domain name as its identifier.</p> <p><code>YYYY-MM</code> – The first year and month that ownership of the domain name was established.</p> <p><code>com.reversed.domain</code> – The domain name reversed.</p> <p><code>name_string</code> – An optional string that identifies a target in a naming space (<code>iqn.YYYY-MM.com.reversed.domain</code>).</p>
LUN	<p>A logical unit number that represents a block device provided by a target. Each target provides one or more LUNs.</p>
ACL	<p>An access control list that restricts access to LUNs based on an initiator's IQN.</p>
portal	<p>An IP address and port on a target or initiator to establish connections. Some iSCSI implementations use the terms <i>portal</i> and <i>node</i> interchangeably.</p>
TPG	<p>A target portal group that represents a complete configuration for a target, including portals, LUNs, and ACLs. While most targets use one TPG, advanced configurations might define multiple TPGs.</p>

Configuring iSCSI Initiators

The iSCSI protocol functions in a client-server configuration. The clients (initiators) send SCSI commands to servers (portals or nodes) that provide the clients with unformatted SCSI devices. These emulated devices appear identical to devices that are connected with physical SCSI or Fibre Channel cabling.

In Red Hat Enterprise Linux, iSCSI initiators are implemented in software; however, hardware initiators are also available. Hardware initiators can improve system performance by offloading Ethernet, TCP, and iSCSI processing to *iSCSI Host Bus Adapters* (HBAs).

Configuring the software iSCSI initiator requires installing the `iscsi-initiator-utils` package. This package includes the `iscsi` and `iscsid` services, and the `/etc/iscsi/iscsid.conf` and `/etc/iscsi/initiatorname.iscsi` configuration files.

iSCSI initiators require a unique iSCSI Qualified Name (IQN), with a default IQN provided in the `/etc/iscsi/initiatorname.iscsi` file. This file contains a randomly generated IQN with `redhat.com` as its domain name. However, administrators should change the IQN to their own domain and replace the randomly generated string with a specific name.

The `/etc/iscsi/iscsid.conf` file contains the default settings for iSCSI targets. These settings include iSCSI timeouts, retry parameters, and user names and passwords for authentication. Restart the `iscsid` service after changes to the `/etc/iscsi/iscsid.conf` file.

Connecting to iSCSI Targets

To discover targets, an iSCSI client must have network connectivity to the target server and be running the `iscsi` service. The `iscsiadm` utility, provided by the `iscsi-initiator-utils` package, is then used to discover, log in to, and list targets.



Note

These commands assume that targets are already configured on a remote server. iSCSI server configuration is outside the scope of this section.

Use `iscsiadm` for the following purposes:

To discover targets.

```
[root@host ~]# iscsiadm -m discovery -t st -p server.lab.example.com:3260  
172.25.250.X:3260,1 iqn.2021-10.com.example.lab:target1
```

To log in to targets.

```
[root@host ~]# iscsiadm -m node -T iqn.2021-10.com.example.lab:target1 -l  
Logging in to [iface: default, target: iqn.2021-10.com.example.lab:target1,  
portal: 172.25.250.X,3260]  
Login to [iface: default, target: iqn.2021-10.com.example.lab:target1, portal:  
172.25.250.X,3260] successful.
```

To list targets.

```
[root@host ~]# iscsiadm -m node  
172.25.250.X:3260,1 iqn.2021-10.com.example.lab:target1
```

When logged in to a target, the device appears locally on the client.

```
[root@host]# grep "Attached SCSI" /var/log/messages  
Oct  6 18:56:00 server kernel: sd 2:0:0:0: [sda] Attached SCSI disk  
  
[root@host]# lsblk  
NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT  
sda     8:0      0   1G  0 disk  
...output omitted...
```

During the discovery process, `iscsiadm` stores target information and settings in the `/var/lib/etc/iscsi/nodes/` directory. These additional configuration files complement the default configurations in the `/etc/iscsi/iscsid.conf` file.

Authentication Options

iSCSI uses ACLs to control which targets and LUNs initiators can access. Use Challenge-Handshake Authentication Protocol (CHAP) authentication to force initiators to provide a username and password before accessing LUNs. iSCSI ACLs are similar to Fibre Channel use of device worldwide numbers (WWNs) for soft zoning management restrictions. Although

Fibre Channel switch-level compulsory port restriction (hard zoning) has no comparable iSCSI mechanism, Ethernet VLANs can provide similar isolation security.

By default, the iSCSI initiator in RHEL 8 is not configured with CHAP authentication. If password authentication is enabled and configured on the target, then matching authentication must be configured on the initiator. Authentication configuration settings are located in the `/etc/iscsi/iscsid.conf` file. 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. In either case, the `node.session.auth.authmethod` value must be set to CHAP.

```
# Enable CHAP Authentication
node.session.auth.authmethod = CHAP

# One-way authentication:
node.session.auth.username = <username>
node.session.auth.password = <password>

# Add for two-way authentication:
node.session.auth.username_in = <username>
node.session.auth.password_in = <password>
```

For no authentication, clear all credential values and comment out the `node.session.auth` lines.

Troubleshooting iSCSI Initiator Issues

iSCSI initiators rely on an IP network connection to reach iSCSI targets. Troubleshooting an iSCSI target discovery issue must begin with verifying network access. When networking is eliminated as a potential issue, investigate iSCSI access control and authentication issues next.

Identifying and Resolving Network Issues

To ensure that connections are being made to the proper target host, use `dig` or a similar utility to verify that the domain name that `iscsiadm` requests correctly resolves to the target host's IP address.

```
[root@host ~]# dig server.lab.example.com
...output omitted...
;; ANSWER SECTION:
server.lab.example.com. 3600 IN A 172.25.250.X
...output omitted...
```

When DNS is eliminated as a potential issue, examine whether configuration in the `/etc/hosts` and `/etc/nsswitch.conf` files on the initiator host is providing conflicting domain names.

If domain name resolution is working properly, then use `ncat` to test the connection to the target host's iSCSI port (3260 by default).

```
[root@host ~]# ncat -v target_server_ip 3260
```

Chapter 5 | Troubleshooting Storage Issues

If the connection fails, then confirm that the target server is listening on the default iSCSI port and that `firewalld` is configured to allow iSCSI connections.

Confirm that the service is listening on port 3260.

```
[root@host ~]# ss -tln | grep 3260
LISTEN 0      256          0.0.0.0:3260        0.0.0.0:*
```

Confirm that `firewalld` is configured to allow port 3260.

```
[root@host ~]# firewall-cmd --list-all | grep 3260
ports: 3260/tcp
```

Identifying and Resolving Login Issues

If an initiator experiences an issue with logging in to an already discovered target, then the problem likely relates to access control or authentication. Access to iSCSI targets is granted via ACLs on the iSCSI server, which specify the names of initiators that are allowed access to specific LUNs. For a mismatch between the client's initiator name and the configured name on the server, a target login failure occurs. If the issue results from an incorrect initiator name on the client, then fix the name in the `/etc/iscsi/initiatorname.iscsi` file and restart the `iscsid` service to apply the change.

If the initiator name is correctly configured, then login failures can be attributed to an incorrect authentication configuration. To authenticate successfully, the initiator must be configured to use the authentication method that the target server requires. If the server is configured for CHAP authentication, then the initiator's CHAP configurations must be specified in `/etc/iscsi/iscsid.conf`.

To assess the cause for login failures, it can be useful to execute the login in debug mode with the `-d` option. Debug level can be set between 0 to 8, with level 8 being required to see authentication details.

```
[root@host ~]# iscsiadadm -m node -T iqn.2021-10.com.example.lab:target1 -l -d8
...
iscsiadadm: updated 'node.session.auth.authmethod', 'None' => 'CHAP'
...
```

If the issue is due to an incorrect CHAP configuration on the client, then apply fixes in `/etc/iscsi/iscsid.conf` and restart the `iscsid` service.

The discovery process stores target node information and settings in the `/var/lib/iscsi/nodes` directory. Because the same target can exist on multiple portals, node records are stored for each portal. On discovery, a node record is written to `/var/lib/iscsi/nodes` and is used for subsequent logins.

When `/etc/iscsi/iscsid.conf` is changed after the initial login attempt, the login process continues with saved settings in `/var/lib/iscsi/etc/nodes`. To log in with newly configured settings, administrators can either modify existing settings under `/var/lib/iscsi/nodes` or remove all saved information from that directory.

Use `iscsiadm` to display existing settings for a target.

```
[root@host ~]# iscsiadm -m node -T iqn.2021-01.com.example.lab:target1
# BEGIN RECORD 6.2.0.873-30
node.name = iqn.2021-01.com.example.lab:target1
node.tpgt = 1
... Output omitted ...
node.conn.iscsi.HeaderDigest = None
node.conn[0].iscsi.IFMarker = No
node.conn.iscsi.OFMarker += No
```

To modify existing settings, use the `iscsiadm` command with the `-o update` option.

Additionally, use the `-n` option to specify the parameter to modify, and the `-v` option to specify the value for the parameter. This example disables CHAP authentication.

```
[root@host ~]# iscsiadm -m node -T iqn.2021-01.com.example.lab:target1 -o update
-n node.session.authmethod -v None [-p target_server[:port]]
```

Alternatively, administrators can purge all node information from the cache. To purge existing node settings for a target, use the `iscsiadm` command with the `-o delete` option.

```
[root@host ~]# iscsiadm -m node -T iqn.2021-01.com.example.lab:iscsistorage -o
delete [-p target_server[:port]]
```

To purge settings for all known nodes, use the same command without specifying a target.

```
[root@host ~]# iscsiadm -m node -o delete [-p target_server[:port]]
```



References

`iscsiadm(8)` man page

Configuring an iSCSI Initiator

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html/managing_storage_devices/configuring-an-iscsi-initiator/managing-storage-devices

► Guided Exercise

Resolving iSCSI Issues

In this lab, you troubleshoot and resolve issues with an iSCSI initiator.

Outcomes

You should be able to identify and resolve a misconfiguration of an iSCSI initiator on a client system.

Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

```
[student@workstation ~]# lab start storage-iscsi
```

This command prepares an iSCSI storage device, ACL, and initiator.

Instructions

The serverb system is configured with a iSCSI target named `iqn.2021-10.com.example.lab:target1`. The target is configured with an ACL that grants access only to the `iqn.2021-10.com.example.lab:servera` IQN. The target is configured without authentication.

Another administrator configured the `servera` system to use `iqn.2021-10.com.example.lab:target1` as the target. The administrator can discover the target, but cannot log in to it. Troubleshoot, identify, and correct the issue so that the initiator can successfully log in to the target.

- 1. Log in to `servera` and switch to the `root` user.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

- 2. Verify that the client cannot log in to the target and that no active sessions exist.

- 2.1. Verify that the system has a successfully discovered target.

```
[root@servera ~]# iscsiadm -m node
172.25.250.11:3260,1 iqn.2021-10.com.example.lab:target1
```

- 2.2. Verify that the system has no active sessions.

```
[root@servera ~]# iscsiadadm -m session
iscsiadm: No active sessions.
```

- 2.3. Attempt to log in to the target. Verify that the login fails, and view the displayed errors.

```
[root@servera ~]# iscsiadadm -m node -T iqn.2021-10.com.example.lab:target1 -l
Logging in to [iface: default, target: iqn.2021-10.com.example.lab:target1,
portal: 172.25.250.11,3260]
iscsiadm: Could not login to [iface: default, target:
iqn.2021-10.com.example.lab:target1, portal: 172.25.250.11,3260].
iscsiadm: initiator reported error (24 - iSCSI login failed due to authorization
failure)
iscsiadm: Could not log into all portals
```

- ▶ 3. Using the -d diagnostic option, attempt to log in to the target again, and then view the extra debug information. Because the error messages indicated an authorization failure, look for debug messages with an "auth" keyword.

```
[root@servera ~]# iscsiadadm -m node -T iqn.2021-10.com.example.lab:target1 -l -d8
...output omitted...
iscsiadm: updated 'node.session.auth.authmethod', 'None' => 'CHAP'
...output omitted...
```

- ▶ 4. The debug messages indicate that the initiator used CHAP authentication. Because the target uses no authentication, the initiator's authorization method is incorrect.

- 4.1. Correct the configuration in /etc/iscsi/iscsid.conf by commenting out the following line:

```
# node.session.auth.authmethod = CHAP
```

- 4.2. Despite changing the initiator configuration, the previous target settings persist in the /var/lib/iscsi/nodes configuration cache. Clear the saved configuration with the iscsiadadm command.

```
[root@servera ~]# iscsiadadm -m node -T iqn.2021-10.com.example.lab:target1 -o
delete
```

- 4.3. Restart the iscsid service.

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

- 4.4. Rediscover the target.

```
[root@servera ~]# iscsiadadm -m discovery -t st -p serverb.lab.example.com
172.25.250.11:3260,1 iqn.2021-10.com.example.lab:target1
```

- ▶ 5. Attempt to log in to the target to verify that the login issue is resolved.

```
[root@servera ~]# iscsiadadm -m node -T iqn.2021-10.com.example.lab:target1 -l
Logging in to [iface: default, target: iqn.2021-10.com.example.lab:target1,
portal: 172.25.250.11,3260]
iscsiadm: Could not login to [iface: default, target:
iqn.2021-10.com.example.lab:target1, portal: 172.25.250.11,3260].
iscsiadm: initiator reported error (24 - iSCSI login failed due to authorization
failure)
iscsiadm: Could not log into all portals
```

- ▶ 6. Despite fixing the initiator's authentication configuration, the issue persists. Verify that the initiator IQN matches the `iqn.2021-10.com.example.lab:servera` IQN as defined in the ACL.

```
[root@servera ~]# cat /etc/iscsi/initiatorname.iscsi
InitiatorName=iqn.2021-10.lab.example.com:servera
```

- ▶ 7. The initiator name is incorrectly configured because the domain name is not reversed. Fix the name and restart the `iscsid` service to implement the change.

7.1. Fix the initiator name.

```
[root@servera ~]# echo InitiatorName=iqn.2021-10.com.example.lab:servera
> /etc/iscsi/initiatorname.iscsi
```

7.2. Restart the `iscsid` service.

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

- ▶ 8. Attempt to log in to the target to verify that the login issue is resolved.

```
[root@servera ~]# iscsiadadm -m node -T iqn.2021-10.com.example.lab:target1 -l
Logging in to [iface: default, target: iqn.2021-10.com.example.lab:target1,
portal: 172.25.250.11,3260]
Login to [iface: default, target: iqn.2021-10.com.example.lab:target1, portal:
172.25.250.11,3260] successful.
```

- ▶ 9. Return to `workstation` as the `student` user.

```
[root@servera ~]# exit
[student@servera ~]$ exit
[student@workstation ~]$
```

Finish

On the `workstation` machine, use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish storage-iscsi
```

This concludes the guided exercise.

► Lab

Troubleshooting Storage Issues

In this lab, you troubleshoot and resolve storage issues with encryption, file system, LVM, and iSCSI.

Outcomes

You should be able to troubleshoot and repair issues with corrupted file systems, LUKS2 headers, LVM administration, and iSCSI targets.

Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

```
[student@workstation ~]$ lab start storage-review
```

This command confirms that the required hosts for this exercise are accessible and configures them to perform the tasks.

Instructions

The CFO requests access to the company's financial data. The data resides in an encrypted volume on an iSCSI target, `iqn.2016-01.com.example.lab:iscsistorage`, provided by the `serverb` host. The target does not use authentication and is configured with an ACL to grant access to the `iqn.2016-01.com.example.lab:servera` initiator.

An administrator working on this request cannot access the encrypted volume. You are asked first to resolve this issue, and then to decrypt the volume with `RedHatR0cks!` as the last known password. Make the volume available as the `finance` mapped device mounted at `/mnt/finance` on `servera`.

If you have trouble decrypting the volume, then a backup of the encrypted volume's header exists on the `/dev/save/old` LVM logical volume on `servera`. Mount the logical volume at `/mnt/save` on `servera` and find the LUKS2 header backup file in `/mnt/save/luks/iscsistorage_luks_header`. If the file is not accessible, then 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, then discover targets on `serverb`. Resolve any issues in the discovery process.
3. When iSCSI target discovery for `serverb` succeeds, log in to the target. Resolve any issues in the login process.
4. When login to the target is successful, use the last known password to decrypt the encrypted volume.
5. If the decryption fails, then mount the `/dev/save/old` LVM logical volume to `/mnt/save` to access the LUKS2 header dump file. Resolve any issues with the restore process.

6. Locate and verify the LUKS2 header backup file at `/mnt/save/luks/iscsistorage_luks_header`. Resolve any issues with the restore process.
7. Using the LUKS2 header backup at `/mnt/save/luks/iscsistorage_luks_header`, restore the LUKS2 header to the encrypted volume on the `/dev/sda1` partition.
8. Use the last known password to decrypt the encrypted volume and map it to `/dev/mapper/finance`.
9. Make the contents of the decrypted volume accessible at the `/mnt/finance` mount point.
10. Return to `workstation` as the `student` user.

Evaluation

On the `workstation` machine, use the `lab` command to grade your work. Correct any reported failures and rerun the script until you receive a passing grade.

```
[student@workstation ~]$ lab grade storage-review
```

Finish

On the `workstation` machine, use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish storage-review
```

This concludes the lab.

► Solution

Troubleshooting Storage Issues

In this lab, you troubleshoot and resolve storage issues with encryption, file system, LVM, and iSCSI.

Outcomes

You should be able to troubleshoot and repair issues with corrupted file systems, LUKS2 headers, LVM administration, and iSCSI targets.

Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

```
[student@workstation ~]$ lab start storage-review
```

This command confirms that the required hosts for this exercise are accessible and configures them to perform the tasks.

Instructions

The CFO requests access to the company's financial data. The data resides in an encrypted volume on an iSCSI target, `iqn.2016-01.com.example.lab:iscsistorage`, provided by the `serverb` host. The target does not use authentication and is configured with an ACL to grant access to the `iqn.2016-01.com.example.lab:servera` initiator.

An administrator working on this request cannot access the encrypted volume. You are asked first to resolve this issue, and then to decrypt the volume with `RedHatR0cks!` as the last known password. Make the volume available as the `finance` mapped device mounted at `/mnt/finance` on `servera`.

If you have trouble decrypting the volume, then a backup of the encrypted volume's header exists on the `/dev/save/old` LVM logical volume on `servera`. Mount the logical volume at `/mnt/save` on `servera` and find the LUKS2 header backup file in `/mnt/save/luks/iscsistorage_luks_header`. If the file is not accessible, then 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.

- 1.1. Log in to `servera` and switch to the `root` user.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

- 1.2. List the 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, then discover targets on serverb. Resolve any issues in the discovery process.

- 2.1. Discover the targets on serverb.

```
[root@servera ~]# iscsiadm -m discovery -t st -p serverb.lab.example.com  
iscsiadm: cannot make connection to 172.25.252.11: No route to host  
iscsiadm: cannot make connection to 172.25.252.11: No route to host  
iscsiadm: cannot make connection to 172.25.252.11: No route to host  
iscsiadm: cannot make connection to 172.25.252.11: No route to host  
iscsiadm: cannot make connection to 172.25.252.11: No route to host  
iscsiadm: cannot make connection to 172.25.252.11: No route to host  
iscsiadm: connection login retries (reopen_max) 5 exceeded  
iscsiadm: Could not perform SendTargets discovery: iSCSI PDU timed out
```

- 2.2. Verify that the resolved address for serverb.lab.example.com during the discovery is correct.

```
[root@servera ~]# dig +short serverb.lab.example.com  
172.25.250.11
```

- 2.3. Because the IP address does not match with the address provided by DNS name resolution, validate the /etc/hosts file to determine whether it is the source of the problem.

```
[root@servera ~]# grep serverb /etc/hosts  
172.25.252.11 serverb.lab.example.com serverb
```

- 2.4. Fix the erroneous entries in /etc/hosts to correct the name resolution issue that causes the connectivity issues to serverb.lab.example.com.

```
[root@servera ~]# grep serverb /etc/hosts  
172.25.250.11 serverb.lab.example.com serverb
```

- 2.5. Rediscover the targets on serverb.

```
[root@servera ~]# iscsiadm -m discovery -t st -p serverb.lab.example.com  
172.25.250.11:3260,1 iqn.2016-01.com.example.lab:iscsistorage
```

3. When iSCSI target discovery for serverb succeeds, log in to the target. Resolve any issues in the login process.

- 3.1. Log in to the target on serverb.

```
[root@servera ~]# iscsiadadm -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: 172.25.250.11,3260]  
iscsiadm: Could not login to [iface: default, target:  
iqn.2016-01.com.example.lab:iscsistorage, portal: 172.25.250.11,3260].  
iscsiadm: initiator reported error (8 - connection timed out)  
iscsiadm: Could not log into all portals
```

- 3.2. Verify the configured authentication method on the initiator.

```
[root@servera ~]# iscsiadadm -m node -T iqn.2016-01.com.example.lab:iscsistorage |  
grep authmethod  
node.session.auth.authmethod = None
```

- 3.3. Check that the authentication failure is not caused by an ACL restriction. Confirm that the initiator name matches the access granted by the destination ACL.

```
[root@servera ~]# cat /etc/iscsi/initiatorname.iscsi  
InitiatorName=iqn.com.example.lab:servera
```

- 3.4. Fix the incorrect initiator name in /etc/iscsi/initiatorname.iscsi.

```
[root@servera ~]# cat /etc/iscsi/initiatorname.iscsi  
InitiatorName=iqn.2016-01.com.example.lab:servera
```

- 3.5. Restart iscsid to implement the new initiator name.

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

- 3.6. Log in to the target.

```
[root@servera ~]# iscsiadadm -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: 172.25.250.11,3260]  
Login to [iface: default, target: iqn.2016-01.com.example.lab:iscsistorage,  
portal: 172.25.250.11,3260] successful.
```

4. When login to the target is successful, use the last known password to decrypt the encrypted volume.

```
[root@servera ~]# cryptsetup luksOpen /dev/sda1 finance  
Enter passphrase for /dev/sda1: RedHatR0cks!  
No key available with this passphrase.
```

5. If the decryption fails, then mount the /dev/save/old LVM logical volume to /mnt/save to access the LUKS2 header dump file. Resolve any issues with the restore process.

- 5.1. Mount the /dev/save/old logical volume and find the LUKS2 header dump file.

```
[root@servera ~]# mkdir /mnt/save
[root@servera ~]# mount /dev/save/old /mnt/save
[root@servera ~]# ls /mnt/save
ls: cannot access '/mnt/save/luks': Structure needs cleaning
certs  keys  luks
```

5.2. Unmount /mnt/save and inspect the /dev/save/old logical volume.

```
[root@servera ~]# umount /mnt/save
[root@servera ~]# lvs
  LV   VG   Attr      LSize  Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert
  new  save -wi-a----  16.00m
```

5.3. Because the logical volume is missing, view the LVM archive log to determine the reason. Your archive log file names are expected to be different.

```
[root@servera ~]# vgcfgrestore -l save

File: /etc/lvm/archive/save_00000-242032145.vg
VG name: save
Description: Created *before* executing 'vgcreate save /dev/vdb1'
Backup Time: Wed Oct 13 01:44:43 2021

File: /etc/lvm/archive/save_00001-62014416.vg
VG name: save
Description: Created *before* executing 'lvcreate -W y -L 15M -n old save'
Backup Time: Wed Oct 13 01:44:43 2021

File: /etc/lvm/archive/save_00002-1184718629.vg
VG name: save
Description: Created *before* executing 'lvcreate -W y -L 15M -n new save'
Backup Time: Wed Oct 13 01:44:43 2021

File: /etc/lvm/archive/save_00003-184394976.vg
VG name: save
Description: Created *before* executing 'lvremove -f /dev/save/old'
Backup Time: Wed Oct 13 01:44:45 2021

File: /etc/lvm/backup/save
VG name: save
Description: Created *after* executing 'lvremove -f /dev/save/old'
Backup Time: Wed Oct 13 01:44:45 2021
```

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_00003-184394976.vg save
Volume group save has active volume: new.
WARNING: Found 1 active volume(s) in volume group "save".
```

```
Restoring VG with active LVs, may cause mismatch with its metadata.
Do you really want to proceed with restore of volume group "save", while 1
volume(s) are active? [y/n]: y
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 LUKS2 header backup file at /mnt/save/luks/iscsistorage_luks_header. Resolve any issues with the restore process.

- 6.1. Locate the /mnt/save/luks/iscsistorage_luks_header file.

```
[root@servera ~]# ls -la /mnt/save/luks
ls: cannot access '/mnt/save/luks': Structure needs cleaning
```

- 6.2. Determine the type of file system 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" BLOCK_SIZE="512"
  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
  - zero 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 ...
```

```
Invalid inode number 0x499602d2
xfs_dir_ino_validate: XFS_ERROR_REPORT
Metadata corruption detected at 0x55cf142addd0, inode 0x35e8 data fork
couldn't map inode 13800, err = 117
    - traversal finished ...
    - moving disconnected inodes to lost+found ...
disconnected inode 13801, would move to lost+found
Phase 7 - verify link counts...
Invalid inode number 0x499602d2
xfs_dir_ino_validate: XFS_ERROR_REPORT
Metadata corruption detected at 0x55cf142addd0, inode 0x35e8 data fork
couldn't map inode 13800, err = 117, can't compare 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
```

6.4. Mount the repaired file system.

Based on the file system repair report, you expect to find the `iscsistorage_luks_header` file in the `lost+found` directory. Restore the file to the `/mnt/save/luks` directory.

```
[root@servera ~]# mount /dev/save/old /mnt/save
[root@servera ~]# ls -la /mnt/save/lost+found/
total 1028
drwxr-xr-x. 2 root root      18 Oct 13 02:31 .
drwxr-xr-x. 6 root root     57 Jan 21 2016 ..
```

```
-rw-r--r-- 1 root root 1052672 Jan 21 2016 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
```

- Using the LUKS2 header backup at /mnt/save/luks/iscsistorage_luks_header, restore the LUKS2 header to the encrypted volume 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 'yes' in capital letters): YES
```

- Use the last known password to decrypt the encrypted volume and map it to /dev/mapper/finance.

```
[root@servera ~]# cryptsetup luksOpen /dev/sda1 finance
Enter passphrase for /dev/sda1: RedHatR0cks!
```

- 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. 8 root root 101 Jan 21 2016 .
drwxr-xr-x. 4 root root 33 Oct 12 23:48 ..
drwxr-xr-x. 2 root root 6 Jan 21 2016 accounts
drwxr-xr-x. 2 root root 6 Jan 21 2016 customers
drwxr-xr-x. 2 root root 6 Jan 21 2016 employees
drwxr-xr-x. 2 root root 6 Jan 21 2016 loans
drwxr-xr-x. 2 root root 6 Jan 21 2016 management
drwxr-xr-x. 2 root root 6 Jan 21 2016 shareholders
```

- Return to workstation as the student user.

```
[root@servera ~]# exit
[student@servera ~]$ exit
[student@workstation ~]$
```

Evaluation

On the workstation machine, use the lab command to grade your work. Correct any reported failures and rerun the script until you receive a passing grade.

```
[student@workstation ~]$ lab grade storage-review
```

Finish

On the **workstation** machine, use the **lab** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish storage-review
```

This concludes the lab.

Summary

In this chapter, you learned:

- The `stratis` command creates and manages Stratis storage pools.
- The `e2fsck` command checks and repairs `ext4` file systems.
- The `xfs_repair` command checks and repairs XFS file systems.
- The `vgcfgrestore` command restores the metadata of a volume group.
- The `cryptsetup` command restores and backs up LUKS headers.
- The `iscsiadm` command discovers, displays, and modifies iSCSI targets.

Chapter 6

Troubleshooting RPM Issues

Goal

Identify and resolve issues with the package management subsystem.

Objectives

- Identify and resolve package dependency issues.
- Identify and rebuild a corrupted RPM database.
- Identify and restore changed files with package management tools.
- Register a system with Red Hat and manage Red Hat subscriptions.

Sections

- Resolving Package Dependency Issues (and Guided Exercise)
- Recovering a Corrupted RPM Database (and Guided Exercise)
- Identifying and Recovering RPM Managed Files (and Guided Exercise)
- Managing Red Hat Subscriptions (and Quiz)

Lab

Troubleshooting RPM Issues

Resolving Package Dependency Issues

Objectives

After completing this section, you should be able to identify and resolve package dependency issues.

Package Management

Package management is the method of installing, updating, removing, and tracking software updates. The RPM Package Manager (RPM) and Yellowdog Updater Modified (YUM) are both Linux-based package managers. YUM v4, which is used in Red Hat Enterprise Linux 8, is based on Dandified YUM (DNF) technology. It is the primary package manager in RHEL, and is compatible with YUM v3 in RHEL 7. The `yum` command is a symbolic link to the `dnf` command. Most '`yum`' options work the same in RHEL 8 as in earlier RHEL versions.

RHEL 8 content is distributed through two main repositories:

BaseOS

Content in the BaseOS repository provides the core set of the OS functionality that is the underlying foundation for all installations. This content is available in the RPM format and is subject to similar support terms in previous releases of RHEL.

AppStream

Content in the AppStream repository includes additional user-space applications, runtime languages, and databases to support various workloads and use cases. Content in AppStream is available in one of two formats: the familiar RPM format and an extension to the RPM format called modules. Modules are collections of packages.

Package Dependencies

Package dependencies occur when an RPM package requires functionality from other RPM packages. Package requirements are defined in the metadata of each RPM package.

Installing Packages with Dependencies

Although the `rpm` command can install, upgrade, and remove individual RPM packages, it does not automatically install package dependencies. Alternatively, the `yum` command can install a package with dependencies; it can also install, upgrade, or downgrade dependency packages according to the defined requirements in the package metadata.

Diagnosing Dependency Issues

The `/var/log/dnf.log` file contains the history of installed and erased packages, but does not provide sufficient information when dependency issues occur. To display diagnostic information during package installation, use the `-v` option with the `yum install` command. Multiple `-v` options generate increasingly verbose details.

```
[root@host ~]# yum install -v yum
Loaded plugins: builddep, changelog, config-manager, copr, debug, debuginfo-
install, download, generate_completion_cache, groups-manager, kpatch, needs-
restarting, playground, repoclosure, repodiff, repograph, repomanage, reposync,
uploadprofile, versionlock
YUM version: 4.4.2
...output omitted...
Versionlock plugin: number of lock rules from file "/etc/dnf/plugins/
versionlock.list" applied: 1
Package yum-4.4.2-11.el8.noarch is already installed.
--> Starting dependency resolution
--> Finished dependency resolution
Dependencies resolved.
Nothing to do.
Complete!
```

Displaying Package Dependencies

The `yum deplist` command displays a list of dependencies for a specified package. The command prints the name and the latest available package version that satisfies each dependency.

```
[root@host ~]# yum deplist yum
Last metadata expiration check: 1 day, 2:06:36 ago on Tue 19 Oct 2021 09:26:43 PM
EDT.
package: yum-4.4.2-11.el8.noarch
dependency: dnf = 4.4.2-11.el8
provider: dnf-4.4.2-11.el8.noarch
```

Alternatively, use the `rpm` command to display package dependency information. The `--requires` option, or `-R`, displays the other packages that this package requires.

```
[root@host ~]# rpm -q --requires yum
config(yum) = 4.4.2-11.el8
dnf = 4.4.2-11.el8
rpmlib(CompressedFileNames) <= 3.0.4-1
rpmlib(FileDigests) <= 4.6.0-1
rpmlib(PayloadFilesHavePrefix) <= 4.0-1
rpmlib(PayloadIsXz) <= 5.2-1
```

Additionally, use the `rpm -q --provides` command to list capabilities that this package provides for other packages.

```
[root@host ~]# rpm -q --provides yum
config(yum) = 4.4.2-11.el8
yum = 4.4.2-11.el8
```

Resolving Package Dependencies

The `yum downgrade` command replaces the specified package with an earlier version of that package by removing the currently installed package and installing the earlier version in its place. YUM downgrades to a specific package version if a valid version is specified in the command.

Downgrading works for packages that allow only one version to be installed on the system at a time. Some packages, such as for the Linux kernel, C library, SELinux, or the system dbus, are restricted to update-only or install-only and cannot be downgraded.

**Important**

If the package to downgrade is a prerequisite for other installed packages, then the dependent packages might also require being specified and downgraded.

Listing All Versions of a Package

Use the `yum list --showduplicates` command option to list all versions of a package, both installed and available.

```
[root@host ~]# yum list --showduplicates python3-bind.noarch
Last metadata expiration check: 0:02:57 ago on Sat 23 Oct 2021 08:07:39 PM EDT.
Installed Packages
python3-bind.noarch      32:9.11.26-4.el8_4    @rhel-8-for-x86_64-appstream-rpms
Available Packages
python3-bind.noarch      32:9.11.26-3.el8      rhel-8.4-for-x86_64-appstream-rpms
```

Using YUM to Lock Package Versions

The YUM `versionlock` subcommand locks down a package or group of packages to a specific version. The `python3-dnf-plugin-versionlock.noarch` plug-in package provides the `versionlock` subcommand.

The following table lists common `versionlock` command options:

Command	Function
<code>yum versionlock list</code>	Display the list of locked package versions.
<code>yum versionlock add <i>WILDCARD</i></code>	Lock the current versions of the packages that the wildcard matched.
<code>yum versionlock delete <i>WILDCARD</i></code>	Delete the locks that the wildcard matched.
<code>yum versionlock clear</code>	Clear all package version locks.

When a specific version of a package is locked, YUM does not attempt to update the package when using `yum update`. Version locking typically maintains an interdependent set of packages at the required versions for the application that they belong to.

Viewing Transaction History

YUM history provides information about the timeline of YUM transactions, the dates and times when they occurred, the number of affected packages, whether these transactions succeeded or were aborted, and whether the RPM database was changed between transactions.

Use the `yum history` command to list, revert, and repeat transactions. To display a list of the latest YUM transactions, use the `yum history` command without subcommands:

```
[root@host ~]# yum history
ID      | Command line           | Date and time   | Action(s)    | Altered
-----
7 | install dovecot        | 2021-10-23 20:37 | Install       | 2
...output omitted...
2 | -y upgrade             | 2021-06-14 03:38 | I, O, U     | 33
1 | localinstall -y test.rpm | 2021-06-14 03:37 | Install       | 1 EE
```

To view specific transaction details, use the `info` subcommand with the transaction ID:

```
[root@host ~]# yum history info 7
Transaction ID : 7
Begin time     : Sat 23 Oct 2021 08:37:12 PM EDT
Begin rpmdb    : 639:4dc8e19f4fa090599874bee11c2dcf30b4e53bdd
End time       : Sat 23 Oct 2021 08:37:15 PM EDT (3 seconds)
End rpmdb      : 641:942bc72f7299bd957f1bb643f5c63bc5fdd0bbc6
User          : Student User <student>
Return-Code    : Success
Releasever    : 8
Command Line   : install dovecot
Comment        :
Packages Altered:
  Install clucene-core-2.3.3.4-31.20130812.e8e3d20git.el8.x86_64
    @rhel-8.4-for-x86_64-appstream-rpms
  Install dovecot-1:2.3.8-9.el8.x86_64
    @rhel-8.4-for-x86_64-appstream-rpms
```

To revert a specific transaction, use the `undo` subcommand with the transaction ID:

```
[root@host ~]# yum history undo 7
```

To repeat a specific transaction, use the `redo` subcommand with the transaction ID:

```
[root@host ~]# yum history redo 7
```



References

`rpm(8)`, `yum(8)`, and `yum-versionlock(1)` man pages

For further information, refer to *Chapter 12. Managing Software Packages* at https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/configuring_basic_system_settings/index#managing-software-packages_configuring-basic-system-settings

► Guided Exercise

Resolving Package Dependency Issues

In this exercise, you resolve a package dependency issue, and then lock the prerequisite package to the relevant version.

Outcomes

You should be able to resolve package dependency issues.

Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

```
[student@workstation ~]$ lab start rpm-dependency
```

Instructions

The security team recently updated the system requirements for hosts in the data center. The update requires installing a custom package, `rht-main`, on each host.

A colleague attempted unsuccessfully to install `rht-main` on `servera`. You are asked to resolve this issue and to verify that the `rht-main` package installs successfully.

- 1. Log in to `servera` and switch to the `root` user.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

- 2. Reproduce the issue, to collect and record troubleshooting information.

- 2.1. Use YUM to install the `rht-main` package. View the transaction messages.

```
[root@servera ~]# yum install rht-main
Last metadata expiration check: 2:29:27 ago on Wed 13 Oct 2021 05:41:46 PM EDT.
Error:
Problem: package rht-main-0.1-1.el8.noarch requires rht-prereq = 0.1-1.el8, but
none of the providers can be installed
- conflicting requests
- package rht-prereq-0.1-1.el8.noarch is filtered out by exclude filtering
(try to add '--skip-broken' to skip uninstallable packages or '--nobest' to use
not only best candidate packages)
```

The transaction message indicates that the `rht-main` package requires a specific version (0.1-1.el8) of the `rht-prereq` package.

- 2.2. Use the `yum install -v` command to generate verbose transaction details.

```
[root@servera ~]# yum -v install rht-main
Loaded plugins: builddep, changelog, config-manager, copr, debug, debuginfo-install, download, generate_completion_cache, groups-manager, kpatch, needs-restarting, playground, repoclosure, repodiff, repograph, repomanage, reposync, uploadprofile, versionlock
YUM version: 4.4.2
...output omitted...
Last metadata expiration check: 3:11:29 ago on Mon 18 Oct 2021 07:59:41 PM EDT.
Completion plugin: Generating completion cache...
Versionlock plugin: number of lock rules from file
"/etc/dnf/plugins/versionlock.list" applied: 1
--> Starting dependency resolution
--> Finished dependency resolution
Error:
Problem: package rht-main-0.1-1.el8.noarch requires rht-prereq = 0.1-1.el8, but
none of the providers can be installed
- conflicting requests
- package rht-prereq-0.1-1.el8.noarch is filtered out by exclude filtering
(try to add '--skip-broken' to skip un/installable packages or '--nobest' to use
not only best candidate packages)
```

The *Versionlock plugin* line indicates that a single package is present in the version lock list. Therefore, your next steps are to view dependencies and to verify whether a version lock is the root cause of the issue.

- 3. Use the `yum deplist` command to view the `rht-main` package dependencies.

```
[root@servera ~]# yum deplist rht-main
Last metadata expiration check: 2:01:55 ago on Wed 13 Oct 2021 05:41:46 PM EDT.
package: rht-main-0.1-1.el8.noarch
dependency: rht-prereq = 0.1-1.el8
provider: rht-prereq-0.1-1.el8.noarch
```

- 4. To list all versions, both installed and available, of the `rht-prereq` package that are known to the YUM configuration, use the `yum list --showduplicates` command.

```
[root@servera ~]# yum list --showduplicates rht-prereq
Last metadata expiration check: 2:01:24 ago on Wed 13 Oct 2021 05:41:46 PM EDT.
Installed Packages
rht-prereq.noarch          0.2-1.el8                               @rht_lab
Available Packages
rht-prereq.noarch          0.1-1.el8                               rht_lab
rht-prereq.noarch          0.2-1.el8                               rht_lab
```

The command output indicates that a later version of the `rht-prereq` package, `0.2-1.el8`, is already installed. However, the `rht-main` package requires an earlier version of the `rht-prereq` package. Because YUM automatically resolves package requirements even with version conflicts, additional troubleshooting is required.

- 5. To determine whether any packages have specific version locks, use the `yum versionlock` command.

```
[root@servera ~]# yum versionlock
Last metadata expiration check: 2:02:30 ago on Wed 13 Oct 2021 05:41:46 PM EDT.
rht-prereq-0:0.2-1.el8.*
```

The `rht-prereq` package is locked to a later version than the required version.

- 6. To resolve the version lock conflict, delete the version lock for the `rht-prereq-0:0.2-1.el8` package.

```
[root@servera ~]# yum versionlock delete rht-prereq-0:0.2-1.el8
Last metadata expiration check: 2:59:52 ago on Wed 13 Oct 2021 05:41:46 PM EDT.
Deleting versionlock for: rht-prereq-0:0.2-1.el8.*
```

6.1. Confirm the deletion.

```
[root@servera ~]# yum versionlock
Last metadata expiration check: 3:00:06 ago on Wed 13 Oct 2021 05:41:46 PM EDT.
[root@servera ~]#
```

- 7. Use the `yum` command to install the mandated `rht-main` package.

```
[root@servera ~]# yum install rht-main
Last metadata expiration check: 3:05:12 ago on Wed 13 Oct 2021 05:41:46 PM EDT.
Dependencies resolved.
=====
 Package           Architecture     Version       Repository      Size
 =====
 Installing:
  rht-main          noarch        0.1-1.el8   rht_lab        6.8 k
 Downgrading:
  rht-prereq        noarch        0.1-1.el8   rht_lab        6.7 k

Transaction Summary
=====
Install  1 Package
Downgrade 1 Package

Total size: 13 k
Is this ok [y/N]: y
Downloading Packages:
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing          :                                1/1
  Downgrading       : rht-prereq-0.1-1.el8.noarch    1/3
  Installing        : rht-main-0.1-1.el8.noarch      2/3
  Cleanup           : rht-prereq-0.2-1.el8.noarch      3/3
  Verifying          : rht-prereq-0.1-1.el8.noarch      1/3
```

Chapter 6 | Troubleshooting RPM Issues

Verifying	:	rht-prereq-0.2-1.el8.noarch	2/3
Verifying	:	rht-main-0.1-1.el8.noarch	3/3
Downgraded:			
rht-prereq-0.1-1.el8.noarch			
Installed:			
rht-main-0.1-1.el8.noarch			
Complete!			

As displayed in the transaction summary, the yum command downgrades the **rht-prereq** package from version **0.2-1** to **0.1-1**; installs the **rht-main** package; and then removes the **rht-prereq-0.2-1.el8.noarch** package.

- 8. Now that the **rht-main** package is installed, attempt to update the prerequisite **rht-prereq** package. Because of dependencies, YUM generates an error message and fails.

```
[root@servera ~]# yum update rht-prereq
Last metadata expiration check: 3:58:51 ago on Wed 13 Oct 2021 05:41:46 PM EDT.
Error:
Problem: problem with installed package rht-main-0.1-1.el8.noarch
- package rht-main-0.1-1.el8.noarch requires rht-prereq = 0.1-1.el8, but none of
the providers can be installed
- cannot install both rht-prereq-0.2-1.el8.noarch and rht-
prereq-0.1-1.el8.noarch
- cannot install the best update candidate for package rht-
prereq-0.1-1.el8.noarch
(try to add '--allowerasing' to command line to replace conflicting packages or
'--skip-broken' to skip un/installable packages or '--nobeast' to use not only best
candidate packages)
```

- 9. Create a version lock for the installed **rht-prereq** package. Adding a version lock ensures that YUM ignores any attempt to update the **rht-prereq-0.1-1.el8** package.
- 9.1. To lock the installed version of the **rht-prereq** package, use the **yum versionlock add** command.

```
[root@servera ~]# yum versionlock add rht-prereq
Last metadata expiration check: 0:32:06 ago on Wed 13 Oct 2021 09:52:41 PM EDT.
Adding versionlock on: rht-prereq-0:0.1-1.el8.*
```

- 9.2. Verify that the **versionlock** list includes the **rht-prereq-0:0.1-1.el8** package.

```
[root@servera ~]# yum versionlock
Last metadata expiration check: 0:34:09 ago on Wed 13 Oct 2021 09:52:41 PM EDT.
rht-prereq-0:0.1-1.el8.*
```

- 10. To verify that YUM no longer prints an error message when attempting to update the **rht-prereq** package, run the **yum update** command again.

```
[root@servera ~]# yum update rht-prereq
Last metadata expiration check: 0:36:01 ago on Wed 13 Oct 2021 09:52:41 PM EDT.
Dependencies resolved.
Nothing to do.
Complete!
```

- 11. Return to workstation as the student user.

```
[root@servera ~]# exit
[student@servera ~]$ exit
[student@workstation ~]$
```

Finish

On the **workstation** machine, use the **lab** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish rpm-dependency
```

This concludes the guided exercise.

Recovering a Corrupted RPM Database

Objectives

After completing this section, you should be able to identify and rebuild a corrupted RPM database.

Corrupted RPM Database

Red Hat Enterprise Linux 8 uses yum or rpm to install and remove packages. A Berkeley DB embedded database library manages the RPM package and transaction data that are stored in the /var/lib/rpm/ directory. This directory contains all the files that make up the RPM database, plus additional __db.* index files.

Although RPM database corruption is uncommon, a damaged database will block your ability to manage packages or to install RPM-based software. This section discusses procedures for repairing a corrupted RPM database.

Cleaning Stale Lock Files

The RPM database allows multiple processes to read and write to the database concurrently, which can make it difficult to identify and remove stale database locks. Typically, stale locks are left behind by abnormal process termination from a kernel crash, a kill command, or a loss of system power. Two methods are available to clean stale lock files.

1. The recommended method is to reboot the system. Rebooting removes the locks during the sysinit portion of the boot, when the system is assured that no other process has a lock on the RPM database.
2. If rebooting does not resolve the issue, then manually remove all lock files and the RPM database index files.
 - a. Delete all files in the /var/spool/up2date directory.

```
[root@host ~]# cd /var/spool/up2date  
[root@host ~]# rm *  
[root@host ~]# rm .*
```

- b. Verify that no processes with open RPM database files are running.

```
[root@host ~]# ps -aux | grep -e rpm -e yum -e up2date  
[root@host ~]# lsof | grep /var/lib/rpm
```

- c. If no RPM database process is running, then you can safely delete the database index files.

```
[root@host ~]# rm /var/lib/rpm/__db*  
...output omitted...
```

Recovering Corrupted RPM Database

RPM database corruption might persist after removing lock files. Recovering the RPM database might require rebuilding the package metadata files in the `/var/lib/rpm/` directory. Rebuilding the metadata files also requires reconstructing the database indexes. The method for recovering the corrupted RPM database starts with copying the files so that the recovery procedure can be repeated if necessary.

1. Back up the existing database.

```
[root@host ~]# tar cvf rpmdb-$(date +%Y%m%d-%H%M).tar.bz2 /var/lib/rpm
tar: Removing leading `/' from member names
/var/lib/rpm/
/var/lib/rpm/Requirename
/var/lib/rpm/Name
/var/lib/rpm/Basenames
/var/lib/rpm/Group
/var/lib/rpm/Packages
/var/lib/rpm/.dbenv.lock
/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/Filetriggername
/var/lib/rpm/Transfiletriggername
/var/lib/rpm/Recommendname
/var/lib/rpm/Suggestname
/var/lib/rpm/Supplementname
/var/lib/rpm/Enhancename
/var/lib/rpm/.rpm.lock
/var/lib/rpm/db.001
/var/lib/rpm/db.002
/var/lib/rpm/__db.003
```

2. Verify RPM database integrity by verifying the critical Packages file.

```
[root@host ~]# cd /var/lib/rpm
[root@host rpm]# /usr/lib/rpm/rpmdb_verify Packages
...output omitted...
BDB5105 Verification of Packages failed.
```

3. If the verification fails or displays errors, then rebuild the Packages file by dumping from the current file and reloading the good packages from the dump output stream into a new file. In this example, Packages is first renamed to Packages.bad, so that rpmdb_load can create a file with the default Packages name. Corrupted or incomplete packages are skipped and not reloaded.

```
[root@host rpm]# mv Packages Packages.bad  
[root@host rpm]# /usr/lib/rpm/rpmbuild --rebuilddb Packages  
[root@host rpm]# /usr/lib/rpm/rpmbuild --rebuilddb Packages  
BDB5105 Verification of Packages succeeded.
```

4. Use the package metadata from the newly built Packages file to rebuild the RPM database indexes.

```
[root@host rpm]# rpmbuild --rebuilddb  
...output omitted...
```

5. Verify that the RPM database is successfully rebuilt.

```
[root@host rpm]# /usr/lib/rpm/rpmbuild --rebuilddb Packages  
BDB5105 Verification of Packages succeeded.
```

Additional RPM operations should no longer cause error messages.

If the restored RPM database has fewer than all of the original packages, then view `/var/log/yum.log` to identify the missing packages. Then, use the `rpmbuild --justdb` command to add the missing packages to the RPM database without changing or reinstalling files on the host file system.



References

[rpmbuild\(8\) man page](#)

For further information, refer to *How to rebuild RPM database on a Red Hat Enterprise Linux system?* at
<https://access.redhat.com/solutions/6903>

► Guided Exercise

Recovering a Corrupted RPM Database

In this exercise, you repair a corrupted RPM package database.

Outcomes

You should be able to diagnose and repair a corrupted RPM package database.

Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

```
[student@workstation ~]$ lab start rpm-corruptdb
```

This command backs up the existing RPM database and then corrupts the database.

Instructions

The servera machine has a corrupted RPM database, causing YUM to fail to install the `httpd` package. Resolve the issue and verify that packages can install successfully.

- 1. Log in to the servera machine and switch to the root user.

```
[student@workstation ~]$ ssh student@servera  
...output omitted...  
[student@servera ~]$ sudo -i  
[sudo] password for student: student  
[root@servera ~]#
```

- 2. Use the `yum` command to attempt to install the `httpd` package, and view the resulting error. Abort the package installation.

```
[root@servera ~]# yum install httpd  
error: rpmdb: damaged header #2 retrieved -- skipping.  
...output omitted...  
Is this ok [y/N]: n  
Operation aborted.
```

- 3. Check the system and YUM logs to find related error messages. No error messages are logged for this issue in either location.

```
[root@servera ~]# tail /var/log/messages  
Oct 13 02:02:06 server systemd-logind[912]: New session 9 of user root.  
Oct 13 02:02:06 server systemd[1536]: Reached target Paths.  
Oct 13 02:02:06 server systemd[1536]: Starting D-Bus User Message Bus Socket.  
Oct 13 02:02:06 server systemd[1536]: Reached target Timers.
```

Chapter 6 | Troubleshooting RPM Issues

```
Oct 13 02:02:06 server systemd[1536]: Listening on D-Bus User Message Bus Socket.
Oct 13 02:02:06 server systemd[1536]: Reached target Sockets.
Oct 13 02:02:06 server systemd[1536]: Reached target Basic System.
Oct 13 02:02:06 server systemd[1536]: Reached target Default.
Oct 13 02:02:06 server systemd[1536]: Startup finished in 25ms.
Oct 13 02:02:06 server systemd[1]: Started User Manager for UID 0.
```

```
[root@servera ~]# tail /var/log/dnf.log
...output omitted...

2021-10-13T02:02:17-0400 INFO Total download size: 2.0 M
2021-10-13T02:02:17-0400 INFO Installed size: 5.4 M
2021-10-13T02:02:44-0400 ERROR Operation aborted.
2021-10-13T02:02:44-0400 DDEBUG Cleaning up.
```

- ▶ 4. Query an arbitrary package and check again for error messages. The output does not display any helpful information to troubleshoot the issue.

```
[root@servera ~]# rpm -q redhat-release
redhat-release-8.4-0.6.el8.x86_64
```

- ▶ 5. Query the RPM database to list the package metadata content. By default, both STDOUT and STDERR display on the screen. By redirecting STDOUT to the /dev/null file, only error messages are displayed.

```
[root@servera ~]# rpm -qa > /dev/null
error: rpmdbNextIterator: skipping h#      2 region trailer: BAD, tag 758394368
type 757093744 offset -1684108389 count 544501536
```

- ▶ 6. Use the lsof command to ensure that no rpmdb locks are present.

```
[root@servera ~]# lsof | grep /var/lib/rpm
```

- ▶ 7. Remove the RPM database index files.

```
[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
```

- ▶ 8. Back up the RPM database files to the root user's home directory before attempting a repair.

```
[root@servera ~]# tar cvjf rpmdb-$(date +%%Y%%m%%d-%H%M).tar.bz2 /var/lib/rpm
tar: Removing leading `/' from member names
/var/lib/rpm/
/var/lib/rpm/Requirename
/var/lib/rpm/Name
/var/lib/rpm/Basenames
/var/lib/rpm/Group
```

```
/var/lib/rpm/Packages  
/var/lib/rpm/.dbenv.lock  
/var/lib/rpm/Providename  
/var/lib/rpm/Conflictname  
/var/lib/rpm/Obsolename  
/var/lib/rpm/Triggername  
/var/lib/rpm/Dirnames  
/var/lib/rpm/Installtid  
/var/lib/rpm/Sigmd5  
/var/lib/rpm/Sha1header  
/var/lib/rpm/Filetriggername  
/var/lib/rpm/Transfiletriggername  
/var/lib/rpm/Recommendname  
/var/lib/rpm/Suggestname  
/var/lib/rpm/Supplementname  
/var/lib/rpm/Enhancename  
/var/lib/rpm/.rpm.lock
```

- 9. Verify the integrity of the RPM database Packages file. The verification fails.

```
[root@servera ~]# cd /var/lib/rpm  
[root@servera rpm]# /usr/lib/rpm/rpmdb_verify Packages  
rpmbuild: BDB0682 Page 32: bad prev_pgno 0 on overflow page (should be 31)  
rpmbuild: BDB0683 Page 32: overflow item incomplete  
rpmbuild: Packages: BDB0090 DB_VERIFY_BAD: Database verification failed  
BDB5105 Verification of Packages failed.
```

- 10. Rename the Packages file to Packages.broken. Use the rpmbuild_dump and rpmbuild_load commands to dump the valid packages and reload them into a new Packages file.

```
[root@servera rpm]# mv Packages Packages.broken  
[root@servera rpm]# /usr/lib/rpm/rpmdb_dump Packages.broken  
| /usr/lib/rpm/rpmdb_load Packages
```

- 11. Again, verify the integrity of the RPM database Packages file. Ensure that verification is successful.

```
[root@servera rpm]# /usr/lib/rpm/rpmdb_verify Packages  
BDB5105 Verification of Packages succeeded.
```

- 12. Rebuild the RPM database index files.

```
[root@servera rpm]# rpm --rebuilddb  
error: rpmbuildNextIterator: skipping h#          2 region trailer: BAD, tag 689992815  
type 544105843 offset -1952801887 count 1937141087
```

- 13. Query the RPM database again to check if any error messages appear. The output should not display any errors.

```
[root@servera rpm]# rpm -qa > /dev/null
```

- 14. Attempt to install the `httpd` package. Verify that no errors are displayed this time. Because this step is included only to test the RPM database for errors, abort the package installation.

```
[root@servera rpm]# yum install httpd
Last metadata expiration check: 0:08:13 ago on Thu 14 Oct 2021 10:28:03 AM EDT.
Dependencies resolved.

...output omitted...

Install 9 Packages

Total download size: 2.0 M
Installed size: 5.4 M
Is this ok [y/N]: n
Operation aborted.
```

- 15. Return to `workstation` as the `student` user.

```
[root@servera rpm]# exit
[student@servera ~]$ exit
Connection to servera closed.
[student@workstation ~]$
```

Finish

On the `workstation` machine, use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish rpm-corruptdb
```

This concludes the guided exercise.

Identifying and Recovering RPM Managed Files

Objectives

After completing this section, you should be able to identify and restore changed files with package management tools.

Verifying Installed Packages

A major advantage of RPM package management is the useful data that is stored in the system's RPM database. Every time that an RPM package is installed, including EPEL and third-party RPM packages, package and transaction information is recorded in the RPM database. This information includes file sizes, creation timestamps, content checksums, permissions, and user or group ownerships. Use RPM database content to verify existing files from RPM packages.

Verifying an installed package compares current file attributes with the stored information in the RPM database. The `rpm -V` command verifies a specified package, and `rpm -Va` verifies every installed package on the system.

The `rpm` command generates no output unless it finds discrepancies between the current files and the RPM database. When a difference is found, `rpm` prints the file name as a string to indicate which file attributes are different.

```
[root@host ~]# rpm -Va
S.....T. c /etc/ssh/sshd_config
.....L.... c /etc/rc.d/rc.local
S.5....T. c /etc/systemd/logind.conf
.M..... /var/lib/nfs/rpc_pipefs
```

The first character string is a mask of the file's attributes. Periods represent attributes that match file information in the database. The table lists common file attribute flags.

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 file location
U	User ownership
G	Group ownership
T	Modification time

The single character in front of the file name is the RPM file type that the RPM package builder specifies. The type designates files that need special RPM package handling, and that are not

related to Linux file types. If no character is displayed, then the file is a normal file without any additional package handling designation. The table lists common RPM file type characters.

Letter	File type
c	Configuration file
d	Documentation file
l	License file
r	Readme file

Recovering RPM Managed Files

Use the `rpm --setperms` option to restore file permissions to the recorded values in the RPM database. In this example, files from the `setup` package are compared, and permissions are modified when different.

```
[root@host ~]# rpm -V setup
.M....G.. c /etc/motd
[root@host ~]# rpm --setperms setup
[root@host ~]# rpm -V setup
.....G.. c /etc/motd
```

The group ownership setting is not modified by resetting the permissions. Use the `rpm --setugids` option to restore user and group ownership settings to the original values.

```
[root@host ~]# rpm --setugids setup
[root@host ~]# rpm -V setup
```

Use the `yum reinstall` command to recover a package's modified files, by replacing the current file with one from the original package. This command works for packages that support only one installed package version at a time, and does not work for *install-only* packages such as the Linux kernel.

In this example, `yum reinstall` restores a modified executable. The first `rpm -V` command shows that the `/usr/sbin/tuned` size, contents, and timestamps changed from original values. The later `rpm -V` command displays nothing, and verifies that the file matches the original attributes, because the file was reinstalled.

```
[root@host ~]# rpm -V tuned
S.5...T. /usr/sbin/tuned
[root@host ~]# yum reinstall tuned
...output omitted...
    Reinstalling      : tuned-2.15.0-2.el8.noarch
    Verifying        : tuned-2.15.0-2.el8.noarch
...output omitted...
[root@host ~]# rpm -V tuned
```



Note

Files from application deployment formats other than RPM cannot be restored by the `rpm` or `yum reinstall` commands.

Many data files are either not provided by RPM or YUM, or are empty when first installed. Typically, these data files are created when the application first runs. Such data files also cannot be restored by the `rpm` or `yum reinstall` commands, but can be recovered from backups.



References

`rpm(8)` and `yum(8)` man pages

► Guided Exercise

Identifying and Recovering RPM Managed Files

In this exercise, you diagnose and repair file permissions and corrupted files that belong to packages.

Outcomes

You should be able to diagnose and repair file permission issues and corrupted files with the `rpm` and `yum` commands.

Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

```
[student@workstation ~]$ lab start rpm-changedfiles
```

This command modifies RPM-managed files on your system.

Instructions

Users on `servera` are reporting that they cannot use the `sudo` or `ls` commands. Identify and resolve the issues that are preventing the commands from working.

- 1. Log in to `servera` as the `student` user and attempt to switch to the `root` user with the `sudo` command. An error is expected.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
bash: /usr/bin/sudo: Permission denied
```

- 2. Instead, run the `su` command to switch to the `root` user. Enter `redhat` as the password.

```
[student@servera ~]$ su - root
Password: redhat
[root@servera ~]#
```

- 3. Verify that the `sudo` and `ls` commands are not working and view the displayed errors.

- 3.1. Attempt to print a message with the `sudo` command.

```
[root@servera ~]# sudo echo test message
-bash: /usr/bin/sudo: cannot execute binary file: Exec format error
```

- 3.2. Attempt to list the contents of the `root` user's home directory with the `ls` command.

```
[root@servera ~]# ls -a  
-bash: /usr/bin/ls: Permission denied
```

- 4. Begin by investigating the `sudo` command. Run the `rpm` command to identify which package provides the `sudo` command and verify the contents of the package.

- 4.1. Determine which package provides the `sudo` command.

```
[root@servera ~]# rpm -qf /usr/bin/sudo  
sudo-1.8.29-7.el8.x86_64
```

- 4.2. Verify the contents of the `sudo` package.

```
[root@servera ~]# rpm -V sudo  
S.5....T. c /etc/sudoers  
..5....T. /usr/bin/sudo
```

The `/etc/sudoers` file is expected to be in the output because entries were added to the file since it was first installed. However, the `/usr/bin/sudo` file has content and timestamps that differ from when originally installed. The modified executable is expected to be the source of the issue.

- 5. Run the `yum reinstall` command to repair the `sudo` package. Verify the package to confirm that the executable file is restored and that the `sudo` command works again.

- 5.1. Run the `yum reinstall` command to repair the `sudo` package.

```
[root@servera ~]# yum reinstall sudo  
...output omitted...
```

- 5.2. Run the `rpm` command to verify the `sudo` package. The `/etc/sudoers` file is still expected in the output.

```
[root@servera ~]# rpm -V sudo  
S.5....T. c /etc/sudoers
```

- 5.3. Verify that the `sudo` command now works.

```
[root@servera ~]# sudo echo test message  
test message
```

- 6. Next, resolve the issue with the `ls` command. Run the `rpm` command to identify which package provides the `ls` command and verify the contents of the package..

- 6.1. Determine which package provides the `ls` command.

```
[root@servera ~]# rpm -qf /usr/bin/ls  
coreutils-8.30-8.el8.x86_64
```

6.2. Verify the contents of the `coreutils` package.

```
[root@servera ~]# rpm -V coreutils  
.M..... /usr/bin/ls
```

The M in the output indicates that the permissions, or mode, of the `/usr/bin/ls` executable file changed.

- 7. Run the `rpm` command to restore the permissions of the files in the `coreutils` package, verify the package, and confirm that the `ls` command works again.

- 7.1. Run the `rpm` command to restore the permissions of the files in the `coreutils` package.

```
[root@servera ~]# rpm --setperms coreutils
```

- 7.2. Verify the `coreutils` package. No output is expected if the verification succeeds.

```
[root@servera ~]# rpm -V coreutils
```

- 7.3. Verify that the `ls` command now works.

```
[root@servera ~]# ls -a  
. .. .bash_history .bash_logout .bash_profile .bashrc .cshrc .ssh .tcshrc
```

- 8. Exit from the root user shell and confirm that the `sudo` and `ls` commands now also work for the `student` user.

```
[root@servera ~]# exit  
[student@servera ~]$ sudo ls -a  
[sudo] password for student: student  
. .. .bash_history .bash_logout .bash_profile .bashrc
```

- 9. Return to `workstation` as the `student` user.

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

Finish

On the `workstation` machine, use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish rpm-changedfiles
```

This concludes the guided exercise.

Managing Red Hat Subscriptions

Objectives

After completing this section, you should be able to register a system with Red Hat and manage Red Hat subscriptions.

Red Hat Subscription Manager

Red Hat Enterprise Linux systems must be entitled to receive software packages. The Red Hat Subscription Manager (RHSM) service manages software entitlements and tracks subscription status with X.509 certificates. The system entitlement process consists of the following steps.

1. Add products to a system, which occurs when the system or additional software is installed.
2. Register the system with the Red Hat Customer Portal hosted server or an on-premise Red Hat Satellite server. When successful, the system stores an identity certificate.
3. Attach subscriptions to the system. When successful, the system stores one or more entitlement certificates.

The `subscription-manager` tool and `rhsmcertd` service are both integral to the preceding process. The `subscription-manager` tool registers systems to the RHSM service and can attach subscriptions to a system. The `rhsmcertd` service periodically validates current certificates and identifies which certificates require renewal.

RHSM client systems must be able to reach the Red Hat subscription management service and Content Delivery Network (CDN). The default RHSM and CDN servers are `subscription.rhsm.redhat.com` and `cdn.redhat.com`, at port 443.

Managing Subscriptions via a GUI

On systems where a GUI is installed, the **Red Hat Subscription Manager** application is available to configure and manage Red Hat subscriptions. Alternatively, the application is accessible from the web console in the **Subscriptions** menu.

When the **Red Hat Subscription Manager** application first launches, a window displays the system's RHSM status, purpose, and any installed products.

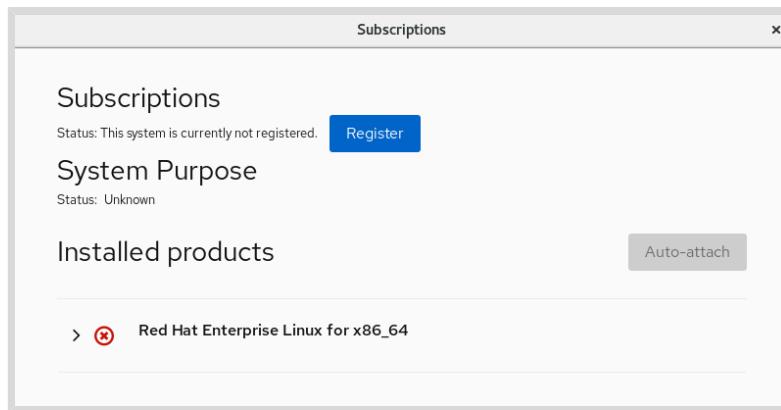


Figure 6.1: Red Hat Subscription Manager main menu

Click Register to start the RHSM registration process and display the **Register System** screen.

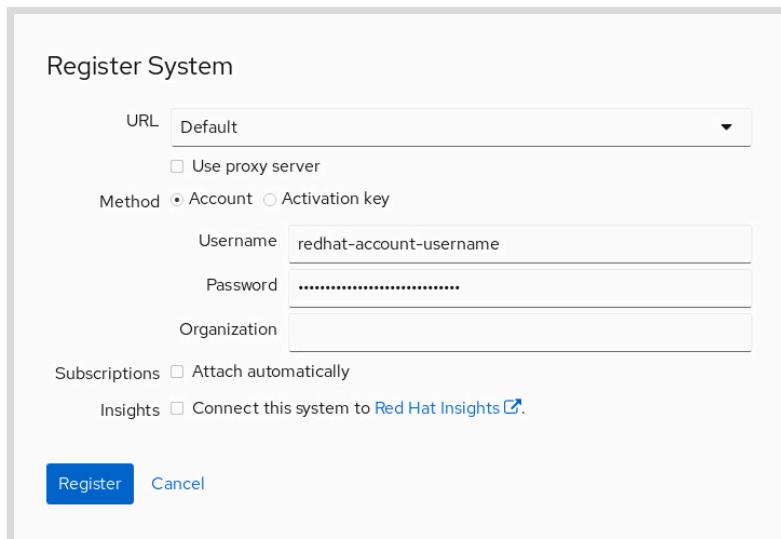


Figure 6.2: Red Hat Subscription Manager Register System screen

On the **Register System** screen, the **URL** field sets the RHSM server. Selecting **Default** uses `subscription.rhsm.redhat.com`. If your organization uses Red Hat Satellite, then provide your Satellite server location instead. If a proxy is required to reach the service, then select **Use proxy server** and enter the connection information in the additional fields.

With the **Method** option, systems can register with a Red Hat account or an Activation key. For the **Account** option, enter your Red Hat Customer Portal user name, password, and organization. Alternatively, for the **Activation key** option, create a key on the Red Hat Customer Portal, and copy it here.

The screen provides options to automatically attach subscriptions and to subscribe the system to Red Hat Insights. Finish the system registration by clicking **Register**. The system is now subscribed and ready to install Red Hat software and receive updates.

Managing Subscriptions via the CLI

The `subscription-manager` command provides the same functionality as the GUI application, and command-line options can override default values. For example, the `--serverurl` option can specify an RHSM or Satellite server other than `subscription.rhsm.redhat.com`.

The `subscription-manager register` command registers a system with the RHSM service. Like the GUI application, use either Red Hat Customer Portal credentials or an activation key to register. Specify usernames and passwords with the `--username` and `--password` options, or an activation key with the `--activation-key` option. By default, `subscription-manager` prompts for a username and password. You can also use options to specify proxy host and authentication credentials.

```
[root@host ~]# subscription-manager register
Registering to: subscription.rhsm.redhat.com:443/subscription
Username: redhat-account-username
Password: SecretPassw0rd!
The system has been registered with ID: 30675802-cb74-45b8-a5d6-620f195a031a
The registered system name is: host.lab.example.com
```

**Note**

For customized registration commands, the **Registration Assistant** Red Hat Customer Portal lab is available at <https://access.redhat.com/labs/registrationassistant>.

After a system is registered, its installed products must become entitled by attaching subscriptions to them.

The `--auto-attach` option attaches subscriptions during system registration, but might create unintended results by consuming high support service-level subscriptions on a non-production machine. To mitigate this behavior, use the `subscription-manager service-level` command to set a preferred service level for installed products on this system. For example, `subscription-manager service-level --set=Self-Support` selects only "Self-Support" subscriptions when using `--auto-attach`.

The `subscription-manager list --installed` command displays a list of installed products and their current status.

```
[root@host ~]# subscription-manager list --installed
-----
      Installed Product Status
-----
Product Name: Red Hat Enterprise Linux for x86_64
Product ID: 479
Version: 8.4
Arch: x86_64
Status: Not Subscribed
Status Details:
Starts:
Ends:
```

The `subscription-manager list --available` command lists the subscriptions that are available to attach to a system. The `Provides` entries show the supported products for this subscription.

```
[root@host ~]# subscription-manager list --available
-----
      Available Subscriptions
-----
Subscription Name: Red Hat Developer Subscription for Individuals
Provides: Red Hat Developer Tools (for RHEL Server for ARM)
           Red Hat Enterprise Linux for x86_64
           ...output omitted...
SKU: RH00798
Contract:
Pool ID: 2c9280817671500s0176dffde5147c39
Provides Management: No
Available: 3
Suggested: 1
Service Type:
Roles: Red Hat Enterprise Linux Server
           Red Hat Enterprise Linux Workstation
```

```
Red Hat Enterprise Linux Compute Node
Service Level: Self-Support
Usage: Development/Test
Add-ons:
Subscription Type: Standard
Starts: 01/07/2021
Ends: 01/06/2022
Entitlement Type: Virtual
... output omitted ...
```

The `subscription-manager attach` command attaches subscriptions to products. Use the `--pool` option to specify subscriptions via their Pool ID.

```
[root@host ~]# subscription-manager attach --pool 2c9280817671500s0176dffde5147c39
Successfully attached a subscription for: Red Hat Developer Subscription for
Individuals
```

The `subscription-manager repos` command manages software repositories. The `--list` option displays the available software repositories for this product and their enabled state. Alternatively, use `--list-enabled` and `--list-disabled` to restrict the list. Use the `--enable` and `--disable` options to include or exclude specified repositories. Wildcards can specify multiple repository IDs, and enable or disable more than one ID at a time. Disabling a repository disallows `rpm` or `yum` to locate software from those repositories.

```
[root@host ~]# subscription-manager repos --list-enabled
-----
Available Repositories in /etc/yum.repos.d/redhat.repo
-----
Repo ID: rhel-8-for-x86_64-baseos-rpms
Repo Name: Red Hat Enterprise Linux 8 for x86_64 - BaseOS (RPMs)
Repo URL: https://cdn.redhat.com/content/dist/rhel8/$releasever/x86_64/baseos/
os
Enabled: 1

Repo ID: rhel-8-for-x86_64-appstream-rpms
Repo Name: Red Hat Enterprise Linux 8 for x86_64 - AppStream (RPMs)
Repo URL: https://cdn.redhat.com/content/dist/rhel8/$releasever/x86_64/
appstream/os
Enabled: 1
```

The `subscription-manager remove` command removes attached subscriptions. Using `--all` matches all attached subscriptions, and using `--serial` or `--pool` options matches subscriptions based on their serial number or Pool ID.

View attached subscriptions with the `subscription-manager list --consumed` command.

```
[root@host ~]# subscription-manager list --consumed
-----
Consumed Subscriptions
-----
Subscription Name: Red Hat Developer Subscription for Individuals
Provides: Red Hat Developer Tools (for RHEL Server for ARM)
Red Hat Enterprise Linux for x86_64
```

```
...output omitted...
SKU: RH00798
Contract:
Pool ID: 2c9280817671500s0176dffde5147c39
Provides Management: No
Available: 3
Suggested: 1
Service Type:
Roles: Red Hat Enterprise Linux Server
       Red Hat Enterprise Linux Workstation
       Red Hat Enterprise Linux Compute Node
Service Level: Self-Support
Usage: Development/Test
Add-ons:
Subscription Type: Standard
Starts: 01/07/2021
Ends: 01/06/2022
Entitlement Type: Virutal
... output omitted ...

[root@host ~]# subscription-manager remove --pool 2c9280817671500s0176dffde5147c39
1 local certificate has been deleted.
The entitlement server successfully removed these pools:
  2c9280817671500s0176dffde5147c39
...output omitted...
```

The `subscription-manager unregister` command removes a system from the RHSM service. This command removes all attached subscriptions to this system and deletes the locally stored identity and entitlement certificates.

```
[root@host ~]# subscription-manager unregister
Unregistering from: subscription.rhsm.redhat.com:443/subscription
System has been unregistered.
```

Troubleshooting Red Hat Subscription Management

The Red Hat Subscription Manager (RHSM) service has two primary log files in the `/var/log/rhsm` directory: `rhsm.log` and `rhsmcertd.log`. Log messages generated by the `subscription-manager` command and **Red Hat Subscription Manager** GUI application are written to `rhsm.log`. Log messages generated by `rhsmcertd` are written to `rhsmcertd.log`.

Examining Configuration Files

The primary configuration file for RHSM is `/etc/rhsm/rhsm.conf`, and `/etc/rhsm/pluginconf.d` might contain additional configuration files for RHSM plug-ins.

In the top section of the `rhsm.conf` file, the `[server]` section includes RHSM server variables and the proxy server variables if needed.

```
[server]
# Server hostname:
hostname = subscription.rhsm.redhat.com

# Server prefix:
```

```
prefix = /subscription

# Server port:
port = 443

... output omitted ...

# an http proxy server to use
proxy_hostname =

# The scheme to use for the proxy when updating repo definitions, if needed
proxy_scheme = http

# 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 Red Hat Content Delivery Network (CDN). This section also defines the directories for storing X.509 certificates.

```
[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

... output omitted ...
```

The [rhsmcertd] section defines the runtime intervals for the rhsmcertd daemon.

```
# Interval to run cert check (in minutes):
certCheckInterval = 240
# Interval to run auto-attach (in minutes):
autoAttachInterval = 1440
```

The /etc/yum/pluginconf.d/subscription-manager.conf configuration file sets whether YUM uses RHSM repositories. The enabled setting is set to 1 by default.

```
[main]
enabled=1
```

Inspecting Certificates

X.509 certificates are stored in /etc/pki subdirectories. The consumer subdirectory stores the server identity key and certificate.

```
[root@host ~]# ls -l /etc/pki/consumer
total 8
-rw-r-----. 1 root root 2228 Oct 14 07:55 cert.pem
-rw-r-----. 1 root root 3243 Oct 14 07:55 key.pem
```

The product and product-default subdirectories contain product certificates.

```
[root@host ~]# ls -l /etc/pki/product-default
total 4
-rw-r--r--. 1 root root 2171 Mar 31 2021 479.pem
```

The entitlement subdirectory contains the subscription certificates.

```
[root@host ~]# ls /etc/pki/entitlement
total 340
-rw-r--r--. 1 root root 3243 Oct 14 07:55 7898548-key.pem
-rw-r--r--. 1 root root 343003 Oct 14 07:55 7898548.pem
```

The rct cat-cert command displays the fields and values that are embedded in RHSM X.509 certificates. Use rct cat-cert to compare entitlement certificate IDs with corresponding product certificate IDs.

```
[root@host ~]# rct cat-cert /etc/pki/product-default/479.pem | grep ID:
ID: 479
[root@host ~]# rct cat-cert /etc/pki/entitlement/7898548.pem | grep 'ID: 479'
ID: 479
```

Diagnosing Network Connectivity

The subscription-manager command must be able to reach RHSM servers. Use the curl command to diagnose network connectivity issues.

The following example shows a successful connection to Red Hat's CDN.

```
[root@host ~]# curl --head --key /etc/pki/entitlement/7898548-key.pem --cert /etc/pki/entitlement/7898548.pem --cacert /etc/rhsm/ca/redhat-uep.pem https://cdn.redhat.com
HTTP/1.1 403 Forbidden
Server: AkamaiGHost
Mime-Version: 1.0
Content-Type: text/html
Content-Length: 263
Expires: Thu, 14 Oct 2021 12:23:49 GMT
```

```
Date: Thu, 14 Oct 2021 12:23:49 GMT
X-Cache: TCP_DENIED from a204-2-243-134.deploy.akamaitechnologies.com
(AkamaiGHost/10.4.4-34529956) (-)
Connection: keep-alive
EJ-HOST: authorizer-prod-dc-iad2-4-9jw2j
X-Akamai-Request-ID: 3e7f845
```

Disregard the 403 Forbidden HTTP error. If any HTTP response is received, then you successfully verified the ability to locate and reach a CDN server.



References

Registration Assistant

<https://access.redhat.com/labs/registrationassistant/>
subscription-manager(8), rct(8), and rhsmcertd(8) man pages

► Quiz

Managing Red Hat Subscriptions

The following steps troubleshoot Red Hat Subscription Manager (RHSM) issues. Indicate the order to take the steps.



1. Confirm that the values in the `/etc/rhsm/rhsm.conf` configuration file are correct.



2. Use `curl` to identify possible network connectivity issues to RHSM servers.



3. Use the `rct cat-cert` command to view RHSM certificate details.



4. View the `/var/log/rhsm/rhsm.log` file for relevant error messages.

► Solution

Managing Red Hat Subscriptions

The following steps troubleshoot Red Hat Subscription Manager (RHSM) issues. Indicate the order to take the steps.

- 2 1. Confirm that the values in the `/etc/rhsm/rhsm.conf` configuration file are correct.
- 4 2. Use `curl` to identify possible network connectivity issues to RHSM servers.
- 3 3. Use the `rct cat-cert` command to view RHSM certificate details.
- 1 4. View the `/var/log/rhsm/rhsm.log` file for relevant error messages.

▶ Lab

Troubleshooting RPM Issues

In this lab, you fix a software package issue that is causing a recently installed network service to fail.

Outcomes

You should be able to diagnose and repair package issues that cause a service to fail.

Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

```
[student@workstation ~]$ lab start rpm-review
```

This command installs and modifies packages.

Instructions

The recently installed web server on the `servera` server displays an error message when it is accessed.

```
[student@workstation ~]$ curl -4 servera.lab.example.com
curl: (7) Failed to connect to servera port 80: No route to host
```

Resolve the issue and verify that the `servera` web server provides web content.

1. Connect to `servera` and assess the problem.
2. Permanently fix the issue.

Evaluation

On the `workstation` machine, use the `lab` command to grade your work. Correct any reported failures and rerun the script until you receive a passing grade.

```
[student@workstation ~]$ lab grade rpm-review
```

Finish

On the `workstation` machine, use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish rpm-review
```

This concludes the lab.

► Solution

Troubleshooting RPM Issues

In this lab, you fix a software package issue that is causing a recently installed network service to fail.

Outcomes

You should be able to diagnose and repair package issues that cause a service to fail.

Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

```
[student@workstation ~]$ lab start rpm-review
```

This command installs and modifies packages.

Instructions

The recently installed web server on the `servera` server displays an error message when it is accessed.

```
[student@workstation ~]$ curl -4 servera.lab.example.com
curl: (7) Failed to connect to servera port 80: No route to host
```

Resolve the issue and verify that the `servera` web server provides web content.

1. Connect to `servera` and assess the problem.

- 1.1. Log in to `servera` and switch to the `root` user.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

- 1.2. View the status of the `httpd` service.

The service is down, and error messages suggest that it could not be 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: failed (Result: exit-code) since Fri 2021-11-05 17:21:09 EDT; 6min ago
    Docs: man:httpd.service(8)
```

```
Process: 29854 ExecStart=/usr/sbin/httpd $OPTIONS -DFOREGROUND (code=exited,
status=203/EXEC)
Main PID: 29854 (code=exited, status=203/EXEC)

Nov 05 17:21:09 servera.lab.example.com systemd[1]: Starting The Apache HTTP
Server...
Nov 05 17:21:09 servera.lab.example.com systemd[1]: httpd.service: Main process
exited, code=exited, status=203/EXEC
Nov 05 17:21:09 servera.lab.example.com systemd[1]: httpd.service: Failed with
result 'exit-code'.
Nov 05 17:21:09 servera.lab.example.com systemd[1]: Failed to start The Apache
HTTP Server.
```

- 1.3. Because the service was recently installed, use the `yum history` command to identify recently installed packages. Your output might be different in your classroom environment, so assume that the following packages were recently installed on your system.

```
[root@servera ~]# yum history
ID      | Command line               | Date and time   | Action(s) | Altered
-----+
 8 | -y install httpd          | 2021-11-05 17:21 | Install    |   9
 7 | -y upgrade                | 2021-06-14 03:38 | I, O, U   |   33
...output omitted...
[root@servera ~]# yum history info 8
Transaction ID : 8
Begin time      : Fri 05 Nov 2021 05:21:03 PM EDT
Begin rpmdb     : 638:44ed0f50693e8e6bcb82e911467e7370b017ae90
End time        : Fri 05 Nov 2021 05:21:05 PM EDT (2 seconds)
End rpmdb       : 647:95b0a71b559ee29fad596689ead593d6de49c93b
User           : root <root>
Return-Code     : Success
Releasever     : 8
Command Line   : -y install httpd
Comment        :
Packages Altered:
    Install redhat-logos-httpd-84.4-1.el8.noarch
    @rhel-8.4-for-x86_64-baseos-rpms
        Install apr-1.6.3-11.el8.x86_64
    @rhel-8.4-for-x86_64-appstream-rpms
        Install apr-util-1.6.1-6.el8.x86_64
    @rhel-8.4-for-x86_64-appstream-rpms
        Install apr-util-bdb-1.6.1-6.el8.x86_64
    @rhel-8.4-for-x86_64-appstream-rpms
        Install apr-util-openssl-1.6.1-6.el8.x86_64
    @rhel-8.4-for-x86_64-appstream-rpms
        Install httpd-2.4.37-39.module+el8.4.0+9658+b87b2deb.x86_64
    @rhel-8.4-for-x86_64-appstream-rpms
        Install httpd-filesystem-2.4.37-39.module+el8.4.0+9658+b87b2deb.noarch
    @rhel-8.4-for-x86_64-appstream-rpms
        Install httpd-tools-2.4.37-39.module+el8.4.0+9658+b87b2deb.x86_64
    @rhel-8.4-for-x86_64-appstream-rpms
        Install mod_http2-1.15.7-3.module+el8.4.0+8625+d397f3da.x86_64
    @rhel-8.4-for-x86_64-appstream-rpms
```

- 1.4. Verify whether the recently installed packages are complete and unchanged.

The `rpm -V` command indicates that the contents and timestamp of `/usr/sbin/httpd` changed since it was installed.

```
[root@servera ~]# rpm -V apr apr-util apr-util-bdb apr-util-openssl httpd  
httpd-filesystem httpd-tools mod_http2  
.5....T. /usr/sbin/httpd
```

- 1.5. Locate the package that provides the changed file.

```
[root@servera ~]# rpm -qf /usr/sbin/httpd  
httpd-2.4.37-39.module+el8.4.0+9658+b87b2deb.x86_64
```

2. Permanently fix the issue.

- 2.1. Reinstall the located package.

```
[root@servera ~]# yum reinstall httpd
```

- 2.2. Restart the `httpd` service.

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

Verify that the service 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 2021-11-05 17:39:23 EDT; 2s ago  
    Docs: man:httpd.service(8)  
   Main PID: 30140 (httpd)  
     Status: "Started, listening on: port 80"  
       Tasks: 213 (limit: 11049)  
      Memory: 28.7M  
        CGroup: /system.slice/httpd.service  
                ├─30140 /usr/sbin/httpd -DFOREGROUND  
                ├─30141 /usr/sbin/httpd -DFOREGROUND  
                ├─30142 /usr/sbin/httpd -DFOREGROUND  
                ├─30143 /usr/sbin/httpd -DFOREGROUND  
                └─30144 /usr/sbin/httpd -DFOREGROUND  
  
Nov 05 17:39:22 servera.lab.example.com systemd[1]: Starting The Apache HTTP  
Server...  
Nov 05 17:39:23 servera.lab.example.com systemd[1]: Started The Apache HTTP  
Server.  
Nov 05 17:39:23 servera.lab.example.com httpd[30140]: Server configured, listening  
on: port 80
```

- 2.3. Verify that access is allowed to the service port.

```
[root@servera ~]# firewall-cmd --list-services
cockpit dhcpv6-client ssh
[root@servera ~]# firewall-cmd --add-service=http --permanent
success
[root@servera ~]# firewall-cmd --reload
success
[root@servera ~]# firewall-cmd --list-services
cockpit dhcpv6-client http ssh
```

2.4. Return to **workstation** as the **student** user.

```
[root@servera ~]# exit
[student@servera ~]$ exit
[student@workstation ~]$
```

2.5. Verify that the **servera** web server is providing web content.

```
[student@workstation ~]$ curl -4 servera.lab.example.com
Lab is fixed.
```

Evaluation

On the **workstation** machine, use the **lab** command to grade your work. Correct any reported failures and rerun the script until you receive a passing grade.

```
[student@workstation ~]$ lab grade rpm-review
```

Finish

On the **workstation** machine, use the **lab** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish rpm-review
```

This concludes the lab.

Summary

In this chapter, you learned:

- The `yum deplist`, `rpm -q --requires`, and `rpm -q --provides` commands display package dependencies.
- The `yum versionlock` command prevents a package from installing an earlier or a later version.
- The `rpmsdb_verify`, `rpmsdb_dump`, and `rpmsdb_load` commands repair a corrupted RPM database.
- The `rpm -V` command compares RPM-managed file attributes with the recorded values in the RPM database.
- The `rpm --setperms`, `rpm --setugids`, and `yum reinstall` commands recover RPM-managed files.
- The `subscription-manager` command and the Subscription Manager GUI application manage RHEL subscriptions.

Chapter 7

Troubleshooting Network Issues

Goal

Identify and resolve network connectivity issues.

Objectives

- Verify network connectivity with standard RHEL tools.
- Identify and repair network connectivity issues.
- Inspect network traffic to assist troubleshooting.

Sections

- Verifying Network Connectivity (and Guided Exercise)
- Resolving Connectivity Issues (and Guided Exercise)
- Inspecting Network Traffic (and Guided Exercise)

Lab

Troubleshooting Network Issues

Verifying Network Connectivity

Objectives

After completing this section, you should be able to verify network connectivity with standard RHEL tools.

Managing Network Connections with a Text-based User Interface

The `nmtui` application is a text user interface (TUI) for managing NetworkManager. The `NetworkManager-tui` package provides the `nmtui` application. To navigate, use the cursor keys or press `Tab` to advance through the options and press `Shift+Tab` to return to previous selections. Use `Enter` to select an option. Use the `Space` bar to toggle the status of checkboxes.

The `nmtui` command adds, modifies, activates, and deactivates connection profiles.

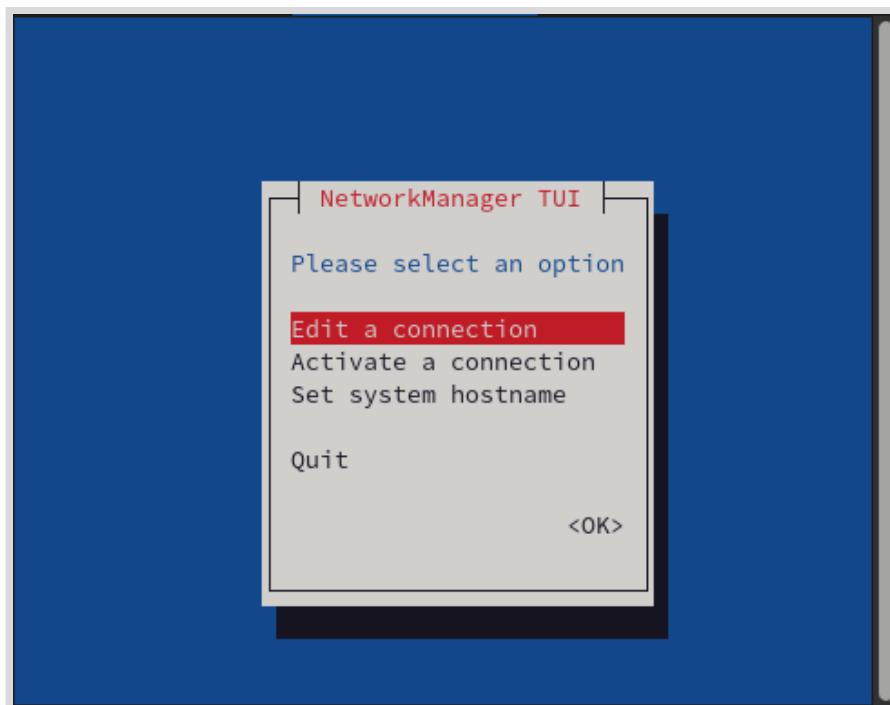


Figure 7.1: Main menu of `nmtui` NetworkManager

Managing Network Connections with the CLI

Manage NetworkManager configuration with the `nmcli` command-line tool. The `nmcli` utility controls the network by creating, displaying, editing, deleting, activating, and deactivating network connections, and managing and displaying network device status. The `nmcli` utility supports options to modify the output of `nmcli` commands. Use of the `nmcli` command simplifies processing the output in scripts. By default, the `nmcli` utility displays its output in a columnar format.

1. Display the list of connection profiles:

```
[root@host ~]# nmcli connection show
NAME           UUID                                  TYPE      DEVICE
Wired connection 2 d2c2b132-f573-3d49-9749-35163328ff0c  ethernet  eth1
Wired connection 1 924a129f-9360-3151-bf0b-1a15c73a699b  ethernet  eth0
```

2. View the information for a particular connection profile:

```
[root@host ~]# nmcli connection show "Wired connection 1"
connection.id:                           Wired connection 1
connection.uuid:                          924a129f-9360-3151-bf0b-1a15c73a699b
connection.stable-id:                     --
connection.type:                         802-3-ethernet
connection.interface-name:                eth0
...output omitted...
```

3. View the list of network devices:

```
[root@host ~]# nmcli device
DEVICE  TYPE      STATE       CONNECTION
eth0    ethernet  connected   Wired connection 1
eth1    ethernet  disconnected --
lo     loopback  unmanaged   --
```

Sending ICMP Echo Requests

The Internet Control Message Protocol (ICMP) is a low-level protocol to test host availability and to send error messages. A first step to test connectivity is to send ICMP echo requests to the remote host. By default, hosts send an ICMP echo reply to indicate that they are present and running. The `ping` command implements ICMP to test host network connectivity.

The `ping` command takes the hostname, or IP address, of the host of interest as an argument. With the `-b` option, the specified command argument is a broadcast address. By default, `ping` continuously sends ICMP echo requests once per second. All received responses are displayed with their packet sequence number and latency time. If the user interrupts the command with `Ctrl+C`, then `ping` displays a summary.

```
[user@host ~]$ ping serverb.lab.example.com
[student@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=2 ttl=64
time=0.267 ms
Ctrl+c
--- serverb.lab.example.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1005ms
rtt min/avg/max/mdev = 0.267/0.302/0.337/0.035 ms
```

Multiple options of `ping` are useful in shell programs. The `-c COUNT` option specifies a count that limits the number of echo requests to send. 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 the response.

```
[user@host ~]$ ping -c 1 -W 3 172.25.250.11
...output omitted...
[user@host ~]$ echo $?
0
```

The `ping` command 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`. Multiple interfaces can have an IPv6 link-local address (`fe80::`). The `-I INTERFACE` option specifies the interface to send the echo requests.

```
[user@host ~]$ ping6 -I eth0 fe80::4ee7:7805:6d16:16b5
...output omitted...
Ctrl+C
--- fe80::4ee7:7805:6d16:16b5 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 0.269/0.877/1.485/0.608 ms
```

The `-I` option is not needed when `ping6` is testing a routable IPv6 address.

```
[user@host ~]$ ping6 serverb.lab.example.com
...output omitted...
Ctrl+C
--- serverb.lab.example.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 0.434/0.693/0.953/0.260 ms
```

The following table lists the most commonly used `ping` and `ping6` options.

Option	Description
<code>-b</code>	Broadcast to the network specified as an argument.
<code>-n</code>	Display host information numerically.
<code>-i INTERVAL</code>	Specify the echo request interval in seconds (default is 1).
<code>-I INTERFACE</code>	Send echo requests through an INTERFACE.
<code>-c COUNT</code>	Send only COUNT echo requests.
<code>-W TIMEOUT</code>	Wait TIMEOUT seconds before quitting.

Blocking and Unblocking ICMP Requests

For security reasons, you can drop or reject ICMP requests, which blocks the request, although clients might still receive information about the blocked request. Blocking ICMP might cause communication problems, especially with IPv6 traffic. The `firewall-cmd` command can handle different ICMP request types in Red Hat Enterprise Linux 8.

The following example uses the `echo-reply` request type to explain the management of ICMP requests with the `firewall-cmd` command.

- List all the available ICMP types with the `firewall-cmd` command:

```
[root@host ~]# firewall-cmd --get-icmptypes  
...output omitted...
```

- The ICMP request works on IPv4, IPv6, or both protocols. Verify which protocol the ICMP request uses:

```
[root@host ~]# firewall-cmd --info-icmptype=echo-reply  
echo-reply  
destination: ipv4 ipv6
```

- Verify whether an ICMP request is blocked:

```
[root@host ~]# firewall-cmd --query-icmp-block=echo-reply  
no
```

- Block an ICMP request:

```
[root@host ~]# firewall-cmd --add-icmp-block=echo-reply  
success
```

- Verify whether an ICMP request is blocked:

```
[root@host ~]# firewall-cmd --query-icmp-block=echo-reply  
yes
```

- Unblock an ICMP request:

```
[root@host ~]# firewall-cmd --remove-icmp-block=echo-reply  
success
```

If you block ICMP requests, then clients can learn that you are blocking them. A potential attacker searching for live IP addresses can still see that your IP address is online. To hide this information entirely, you must drop all ICMP requests instead.

- Block and drop all ICMP requests, and set the target of your zone to DROP:

```
[root@host ~]# firewall-cmd --permanent --set-target=DROP  
success
```

All traffic is dropped, including ICMP requests, except traffic that you explicitly allow.

- To unblock all ICMP requests, set the target of your zone to default:

```
[root@host ~]# firewall-cmd --permanent --set-target=default  
success
```

Understanding Name Resolution

Applications use the `getaddrinfo()` function in the `glibc` library for resolving DNS requests. By default, `glibc` sends all DNS requests to the first DNS server that is specified in the `/etc/`

`resolv.conf` file. If this server does not reply, then the resolver uses the next server in this file. NetworkManager dynamically updates the `/etc/resolv.conf` file with the DNS settings from active NetworkManager connection profiles.

NetworkManager manages DNS servers with the `/etc/resolv.conf` file, by implementing the following rules:

- NetworkManager uses the specified IPv4 and IPv6 DNS servers in the network connection if only one connection profile exists.
- The NetworkManager behavior depends on the `dns` value when using DNS servers based on DNS priority value. The `/etc/NetworkManager/NetworkManager.conf` file stores the `dns` parameter under the `[main]` section.
 - NetworkManager manages the DNS servers from different connections based on `ipv4.dns-priority` and `ipv6.DNS-priority` parameters without `dns=default` or `dns` parameters.
 - When using the `dns=dnsmasq` or `dns=systemd-resolved` parameters, NetworkManager sets either `127.0.0.1` for `dnsmasq` or `127.0.0.53` as `nameserver` entries in the `/etc/resolv.conf` file.

NetworkManager uses the following default DNS priority values for connections:

- 50 for VPN connections
- 100 for other connections

You can set both the global default and connection-specific `ipv4.dns-priority` and `ipv6.dns-priority` parameters to a value between -2147483647 and 2147483647.

Scanning Network Ports

Nmap is an open source port scanner that is provided in Red Hat Enterprise Linux. The `nmap` command scans large networks and runs intensive port scans on individual hosts.



Warning

The use of network and port scanning tools can be a cause for immediate employment termination at many organizations. Unauthorized use of scanners can be interpreted as hacking. Always obtain proper permissions, preferably in writing, before using scanning tools on any organization's network.

Nmap uses raw IP packets to determine significant information, such as:

- The available hosts on the network.
- The service application names and versions that those hosts offer.
- The operating systems and versions that the hosts are running.
- The type of packet filters or firewalls that the hosts use.

The following example shows `nmap` scanning the `172.25.250.0/24` network. The `-n` option displays host information numerically, without using name resolution. As `nmap` discovers each host, it scans privileged TCP ports to look for services. The host MAC address and the corresponding network adapter manufacturer are displayed.

```
[root@host ~]# nmap -n 172.25.250.0/24
Starting Nmap 7.70 ( https://nmap.org ) at 2021-10-22 02:36 EDT
Nmap scan report for 172.25.250.9
Host is up (0.00029s latency).
Not shown: 998 filtered ports
PORT      STATE SERVICE
22/tcp    open  ssh
9090/tcp  closed zeus-admin
MAC Address: 52:54:00:00:FA:09 (QEMU virtual NIC)
...output omitted...

Nmap done: 256 IP addresses (4 hosts up) scanned in 216.29 seconds_
```

The `-sn` option disables port scans. Use this option to see which hosts are present on a network.

```
[root@host ~]# nmap -n -sn 172.25.250.0/24
Starting Nmap 7.70 ( https://nmap.org ) at 2021-10-22 02:46 EDT
Nmap scan report for 172.25.250.9
Host is up (0.00080s latency).
...output omitted...
Nmap done: 256 IP addresses (4 hosts up) scanned in 3.75 seconds
```

The `-sU` option scans a UDP port. The scanning time is much longer than for the default TCP port scan. Use this option to view detailed information about the services that a host exposes to the network.

```
[root@host ~]# nmap -n -sU 172.25.250.0/24
Starting Nmap 7.70 ( https://nmap.org ) at 2021-10-22 02:48 EDT
Nmap scan report for 172.25.250.9
Host is up (0.00040s latency).
All 1000 scanned ports on 172.25.250.9 are filtered
MAC Address: 52:54:00:00:FA:09 (QEMU virtual NIC)
...output omitted...
Nmap done: 256 IP addresses (4 hosts up) scanned in 1095.92 seconds
```

Communicating with a Remote Service

The Ncat troubleshooting tool, from the RHEL `nmap-ncat` package, communicates directly with a service port. The Ncat tool uses either TCP or UDP to interact with the network service, and supports SSL communication. The `-4` and `-6` options force Ncat to use either IPv4 or IPv6.

The Ncat tool has two modes of operation.

- By default, in *connect mode*, it acts as a network client. Specify the hostname and port as arguments to the `ncat` command.

```
[root@host ~]# ncat 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@host ~]#
```

- The Ncat tool acts as a server when invoked with the `-l` option, which is known as *listening mode*. With the `-k` option, the Ncat tool keeps the port open to listen for more connections when used in this mode.

```
[root@host ~]# ncat -l 2510
line of text 1
line of text 2
... output omitted ...
```

By default in listen mode, the Ncat tool displays on the screen any text that it receives over the network. With the `-e COMMAND` option, the Ncat tool passes incoming network traffic to the specified command. The following command launches the Ncat tool in listen mode on port 2510. It passes all network traffic to `/bin/bash`.

```
[root@remote ~]# ncat -l 2510 -e /bin/bash
```

The following output shows the Ncat tool communicating with the listener system.

```
[root@host ~]# ncat 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 that is sent to the server executes with `/bin/bash`. The command output is returned to the network client.



Important

Never directly connect a shell to a network port, for security reasons. The previous example demonstrates only the functionality of Ncat with the `-e` option.

Monitoring network traffic

The IPTraf network monitor software was initially developed in the mid-1990s. RHEL includes the next-generation version of IPTraf, from the `iptraf-ng` package.

The `iptraf-ng` command requires superuser privileges. The following screen capture shows the main menu when the tool runs.

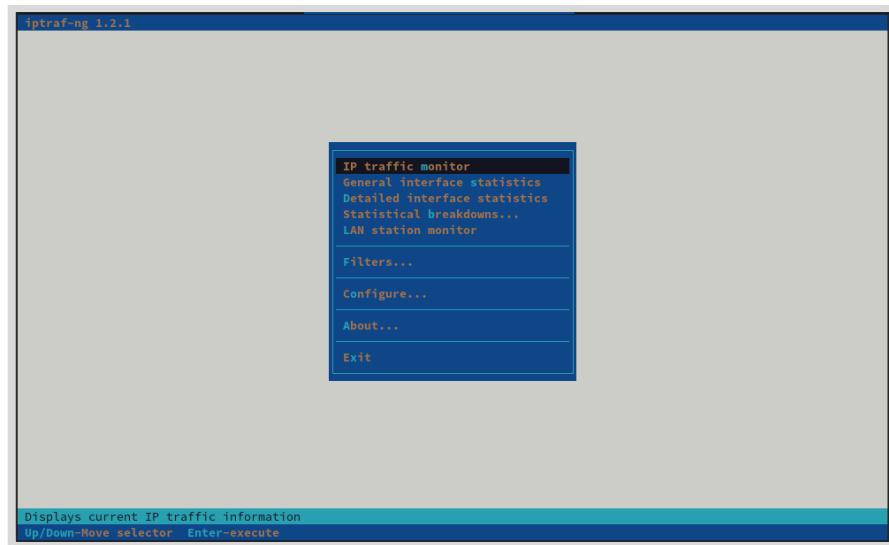


Figure 7.2: Main menu of IPTraf-ng network monitor

Use the up and down arrow keys to navigate the menu, and then press `Enter` to select an option. Alternatively, type the highlighted single character to select the option.

The `iptraf-ng` command monitors current network connections. It also shows UDP and ICMP packet information.

You can use the `iptraf-ng` command interface to view the network interface statistics.

Use the **Filters** main menu selection 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, destination, address, port, and IP protocol type.



References

`icmp(7)`, `iptraf-ng(8)`, `ncat(1)`, `nmap(1)`, `nmcli(1)`, `nmtui(1)` and `ping(8)` man pages

For further information, refer to *Chapter 4. Using nmtui to Manage Network*

Connections Using a Text-Based Interface at

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html/configuring_and_managing_networking/using-nmtui-to-manage-network-connections-using-a-text-based-interface_configuring-and-managing-networking

For further information, refer to *Chapter 5. Getting Started with nmtui* at

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html/configuring_and_managing_networking/getting-started-with-nmcli_configuring-and-managing-networking

For further information, refer to *Managing ICMP requests* at

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html/configuring_and_managing_networking/using-and-configuring-firewalld_configuring-and-managing-networking#managing-icmp-requests_using-and-configuring-firewalld

For further information, refer to *How to use iptraf to monitor network interface?* at

<https://access.redhat.com/solutions/30479>

► Guided Exercise

Verifying Network Connectivity

In this exercise, you use some software tools to test network connectivity.

Outcomes

You should be able to use software tools to test network connectivity.

Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

```
[student@workstation ~]$ lab start network-testing
```

This command installs the required packages and enables the network ports through a firewall.

Instructions

- 1. Log in to `servera` and switch to the `root` user.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

- 2. Test the network connectivity from the `servera` system to the `serverb` system with the `ping` command.

```
[root@servera ~]# ping -c 2 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=0.433 ms
64 bytes from serverb.lab.example.com (172.25.250.11): icmp_seq=2 ttl=64
time=0.457 ms

--- serverb.lab.example.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1036ms
rtt min/avg/max/mdev = 0.433/0.445/0.457/0.012 ms
```

- 3. Use the `nmap` command to scan network ports.

- 3.1. Install the `nmap` package if it is not already installed on the `servera` system.

```
[root@servera ~]# yum install nmap  
...output omitted...  
Complete!
```

- 3.2. Scan the 172.25.250.0/24 network to identify reachable systems in the network.

```
[root@servera ~]# nmap -sn 172.25.250.0/24  
Starting Nmap 7.70 ( https://nmap.org ) at 2021-10-25 10:14 EDT  
Nmap scan report for workstation.lab.example.com (172.25.250.9)  
Host is up (0.00078s latency).  
MAC Address: 52:54:00:00:FA:09 (QEMU virtual NIC)  
Nmap scan report for serverb.lab.example.com (172.25.250.11)  
Host is up (0.0022s latency).  
MAC Address: 52:54:00:00:FA:0B (QEMU virtual NIC)  
Nmap scan report for workstation.lab.example.com (172.25.250.254)  
Host is up (0.0010s latency).  
MAC Address: 52:54:00:00:FA:FE (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 3.77 seconds
```

- 3.3. Scan for IPv4 ports on the **serverb** system.

```
[root@servera ~]# nmap serverb.lab.example.com  
Starting Nmap 7.70 ( https://nmap.org ) at 2021-10-26 03:02 EDT  
Nmap scan report for serverb.lab.example.com (172.25.250.11)  
Host is up (0.00041s latency).  
Not shown: 996 filtered ports  
PORT      STATE SERVICE  
22/tcp    open  ssh  
80/tcp    open  http  
443/tcp   open  https  
9090/tcp  closed zeus-admin  
MAC Address: 52:54:00:00:FA:0B (QEMU virtual NIC)  
  
Nmap done: 1 IP address (1 host up) scanned in 16.12 seconds
```

- 4. Use the **ncat** command to send content from the **servera** system to the **serverb** system.

- 4.1. From the **workstation** system, open another terminal, log in to the **serverb** system, and switch to the **root** user.

```
[student@workstation ~]$ ssh student@serverb  
...output omitted...  
[student@serverb ~]$ sudo -i  
[sudo] password for student: student  
[root@serverb ~]#
```

- 4.2. The start script of this exercise uses the **firewall-cmd** command to open the 4231/tcp network port on the **serverb** system. Run the **ncat** command on the **serverb** system to listen for packets from the **servera** system on port 4231.

```
[root@serverb ~]# ncat -l -k 4231
```

- 4.3. Return to the **servera** system. Use the **nmap** command to scan for ports on the **serverb** system.

```
[root@servera ~]# nmap -p4000-4999 serverb
Starting Nmap 7.70 ( https://nmap.org ) at 2021-10-26 03:06 EDT
Nmap scan report for serverb (172.25.250.11)
Host is up (0.00050s latency).
rDNS record for 172.25.250.11: serverb.lab.example.com
Not shown: 999 filtered ports
PORT      STATE SERVICE
4231/tcp  open  vrml-multi-use
MAC Address: 52:54:00:00:FA:0B (QEMU virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 12.25 seconds
```

- 4.4. From the **servera** system, use the **ncat** command to connect to port 4231/tcp on the **serverb** system. Type a line of text, and then press Enter. Exit the prompt by typing **Ctrl+C**.

```
[root@servera ~]# ncat serverb 4231
This is a test
Ctrl+C
```

- 4.5. Return to the **serverb** system and verify that the text is displayed. Exit from the **ncat** prompt by typing **Ctrl+C**.

```
[root@serverb ~]# ncat -l -k 4231
This is a test
Ctrl+C
```

- 4.6. Close the additional terminal, and return to **workstation** as the **student** user.

```
[root@serverb ~]# exit
[student@serverb ~]$ exit
[student@workstation ~]$
```

- 5. On the **servera** system, install the **iptraf-ng** package if it is not already installed.

```
[root@servera ~]# yum install iptraf-ng
...output omitted...
Complete!
```

- 6. Start the **iptraf-ng** application for network monitoring.

```
[root@servera ~]# iptraf-ng
```

Use the arrow keys to navigate, and then press **Enter** to choose the highlighted menu item.

- ▶ 7. Select **IP traffic monitor** in the main menu. In the **Select Interface** dialog, select **All interfaces** for monitoring network traffic. The top frame shows all TCP connections. The bottom frame shows other types of packets.
- ▶ 8. Press X to return to the main menu in the **iptraf-ng** command interface.
- ▶ 9. Display network interface statistics with the **iptraf-ng** command.
- 9.1. Select **General interface statistics**, or press S to display interface statistics.
 - 9.2. On the **workstation** machine, open another terminal, log in to the **servera** system, and switch to the **root** user. Use the following command to generate some network traffic.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]# ls -R /usr
...output omitted...
```

Watch the traffic counters change. The packet count starts increasing, and activity for **eth0** rises.

- 9.3. Press X and return to the main menu.
- ▶ 10. Create a filter to exclude SSH traffic, but monitor all other network traffic:
- Select **Filters**, and then select **IP**. In this screen, you can create and manage IP network filters.
 - Select **Define new filter** and enter **Exclude SSH** in the description box, and then press **Enter**. An empty list of filtering rules appears.
 - Create a rule to exclude all incoming SSH traffic destined for port 22. Press A to add a new rule to the list of rules for this filter.
 - Use Tab to navigate and enter 22 in the first of the two **Port** fields of the **Destination** column. In the **TCP** field in the **Protocols to match** section, enter Y. In the **Include/Exclude** field, enter E.
 - Press **Enter** to accept the changes.
 - Create another rule to include all other network traffic. Press A to add another rule to the list of rules for this filter.
 - Enter Y in the **All IP** protocol field in the **Protocols to match** section. Press **Enter** to accept the changes.
 - Press X to exit the rule definition screen.
 - To apply the created filter, select **Apply filter**, and then select the **Exclude SSH** filter.
 - Press X to exit the **Filters** menu.
 - In the **Filter Status** frame, **IP filter active** confirms that the filter is in effect.
 - Press X to exit the **Filter Status** menu.

- To verify the newly created filter, select **IP traffic monitor**, and then select **All interfaces** for the interface.
- 10.1. From the newly created SSH session to the **servera** machine, log out and log in to the **servera** machine as the **root** user.

```
[root@servera ~]# exit  
logout  
[student@servera ~]$ exit  
logout  
Connection to servera closed.  
[student@workstation ~]$ ssh root@servera  
...output omitted...  
[root@servera ~]#
```

Return to the terminal that is running the **iptraf-ng** command and verify that the SSH in the TCP connection does not appear in the top frame. The **Packets captured** counter at the bottom of the screen still increases.

- 10.2. Return to the **servera** machine. Use the **curl** command to connect to **serverb.lab.example.com**.

```
[root@servera ~]# curl serverb.lab.example.com  
Welcome to serverb
```

Return to the terminal that is running the **iptraf-ng** command and check that a temporary connection **172.25.250.11:80** appears in the top frame. An HTTP connection uses a TCP connection. The HTTP connection is displayed because it did not match the first rule in the active filter.

- 10.3. Close the **servera** terminal.

```
[root@servera ~]# exit  
[student@workstation ~]$ exit
```

- 11. Return to the terminal where the **iptraf-ng** tool is running. Press X to go to the main menu.
- 12. In the **iptraf-ng** window, detach and delete the **Exclude SSH** filter:
- Select **Filters** and then select **IP**. Now select **Detach filter**. A message appears with **IP filter deactivated**.
 - Press X to exit.
 - Select **Delete Filter** and select **Exclude SSH**.
 - Press X twice and return to **Filter status**.
 - The **Filter status** changes to **No IP filter active**.
 - Press X and return to the main menu.
 - Select **IP traffic monitor**, and then select **All interfaces**. Monitoring now shows the active SSH connections.

- Press X and return to the main menu, and then press X again to exit the application.

► **13.** Return to workstation as the student user.

```
[root@servera ~]# exit  
[student@servera ~]$ exit  
[student@workstation ~]$
```

Finish

On the **workstation** machine, use the **lab** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish network-testing
```

This concludes the guided exercise.

Resolving Connectivity Issues

Objectives

After completing this section, you should be able to identify and repair network connectivity issues.

Network Interface Device Naming

In Red Hat Enterprise Linux 8, the udev device manager provides consistent and predictable device naming for network interfaces. By default, udev assigns names based on firmware, topology, and location information. This naming scheme ensures that device names remain consistent, even when hardware is added or removed. The udev device manager determines interface names based on the following scheme hierarchy.

Scheme	Description	Example
1	Device names incorporate firmware or BIOS-provided index numbers for onboard devices. If this information is not available or applicable, then udev uses scheme 2.	eno1
2	Device names incorporate firmware or BIOS-provided PCI Express (PCIe) hot plug slot index numbers. If this information is not available or applicable, then udev uses scheme 3.	ens1
3	Device names incorporate the physical location of the connector of the hardware. If this information is not available or applicable, then udev uses scheme 5.	enp2s0
4	Device names incorporate MAC addresses. Red Hat Enterprise Linux 8 does not use this scheme by default; however, administrators can optionally enable it.	enx52540...
5	The traditional unpredictable kernel naming scheme. If udev cannot apply any other scheme, then it uses this one.	eth0

The first two characters of a device name represent the type of detected interface.

- en for Ethernet
- wl for wireless LAN (WLAN)
- ww for wireless WAN (WWAN)

The subsequent characters in a device name are based on the schema that the udev device manager selected.

- o for onboard devices. Example: eno1
- s for PCIe devices. Example: ens1
- p and s for the bus and slot location. Example: enp2s0
- x for MAC address-based names. Example: enx525400d5e0fb

The `NamePolicy` setting in the `/usr/lib/systemd/network/99-default.link` file controls the naming hierarchy in the previous table.

```
[root@host ~]# cat /usr/lib/systemd/network/99-default.link | grep NamePolicy
NamePolicy=kernel database onboard slot path
```

The `kernel` and `database` entries indicate that the kernel and udev database are checked for existing device names before new ones are generated. If a persistent name does not already exist, then the system defaults to the `onboard`, `slot`, and `path` schemes. Alternatively, setting the `NamePolicy` value to `mac` incorporates MAC addresses in device names.

Custom interface prefixes are defined during the installation process by adding `net.ifnames.prefix=<prefix>` to the `Install Red Hat Enterprise Linux 8` boot entry kernel options. Custom prefixes take precedence over other naming schemes and must not conflict with existing prefixes, including `eno` and `ens`.



Note

The `biosdevname` package provides an alternative device naming scheme (for example, `p1p1` for PCIe slot 1 port 1) if none of the preceding options are suitable. In Red Hat Enterprise Linux 8, the `biosdevname` package is not installed or enabled by default, including on Dell systems.

The `/var/log/messages` file indicates which devices were detected at boot and which names they were assigned.

```
[root@host ~]# grep -n eth1 /var/log/messages
758:Jun 14 03:37:56 host NetworkManager[850]: <info> [1623656276.6346] manager:
  (eth1): new Ethernet device (/org/freedesktop/NetworkManager/Devices/3)
...output omitted...
761:Jun 14 03:37:56 host NetworkManager[850]: <info> [1623656276.6410] device
  (eth1): carrier: link connected
...output omitted...
789:Jun 14 03:37:56 host NetworkManager[850]: <info> [1623656276.6717] device
  (eth1): Activation: successful, device activated.
```

Manually Configuring Network Device Names

Network interface device names are manually set if a device is directly associated with a MAC address. The `ethername.mac-address` and `ifname` parameters associate a MAC address and name the device.

The following example demonstrates manually renaming a network interface to `net1`.

```
[root@host ~]# nmcli con mod wired1 ethernet.mac-address 52:54:00:00:fa:0a ifname
  net1
[root@host ~]# reboot
...output omitted...
[root@host ~]# ip addr
...output omitted...
2: net1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group
  default qlen 1000
    link/ether 52:54:00:00:fa:0a brd ff:ff:ff:ff:ff:ff
```

**Note**

You must reboot a system to apply manually configured network interface names.

Resolving Network Interface Configuration Issues

Red Hat Enterprise Linux 8 provides commands to collect high-level networking information. The `ip addr show` command provides an overview of network interfaces on a system. This command determines whether interfaces are down or configured with the wrong IP addresses.

```
[root@host ~]# ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    qlen 1000
        link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
        inet 127.0.0.1/8 scope host lo
            valid_lft forever preferred_lft forever
        inet6 ::1/128 scope host
            valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default
    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 noprefixroute eth0
            valid_lft forever preferred_lft forever
        inet6 fe80::c248:6dfe:ce2c:d5e5/64 scope link noprefixroute
            valid_lft forever preferred_lft forever
```

The `ip route` command displays the current routing table. Add the `-6` option to show the IPv6 routing table. Validate the `default` route to ensure that it is directing traffic to the correct gateway address.

```
[root@host ~]# ip route
default via 172.25.250.254 dev eth0 proto static metric 100
172.25.250.0/24 dev eth0 proto kernel scope link src 172.25.250.10 metric 100
```

Modifying NetworkManager Connections

Depending on the configuration issue, it might be necessary to modify or create a connection profile. The `nmcli con mod` command modifies an existing connection.

```
[root@host ~]# nmcli con mod con0 ipv4.addresses 172.25.250.13/24
```

Restart a connection to apply any changes.

```
[root@host ~]# nmcli con down con0
Connection 'con0' successfully deactivated (D-Bus active path: /org/freedesktop/
NetworkManager/ActiveConnection/4)
[root@host ~]# nmcli con up con0
Connection 'con0' successfully activated (D-Bus active path: /org/freedesktop/
NetworkManager/ActiveConnection/6)
```

Modifying a connection's `ifcfg-*` file and reloading the connection with `nmcli con load` also modifies a connection. The `nmcli con reload` command reloads all connections.

```
[root@host ~]# nmcli con load /etc/sysconfig/network-scripts/ifcfg-con0
```



Note

The `nmcli` command generates an `ifcfg-*` file when a connection is manually created. Newly attached interfaces have a default connection profile that is associated with them, but do not have an `ifcfg-*` file until their configurations are modified beyond the default settings.

Alternatively, create a new connection to replace an old one. Turn off the old connection before turning on the new connection. An interface can have only one active connection.

The following example assumes that the `con0` connection is initially paired with the `eth0` device.

```
[root@host ~]# nmcli con add con-name con2 type Ethernet ifname eth0
Connection 'con2' (cf946d5-f3a5-436e-9eea-cc833d807be9) successfully added.
[root@host ~]# nmcli con down con0
Connection 'con0' successfully deactivated (D-Bus active path: /org/freedesktop/
NetworkManager/ActiveConnection/6)
[root@host ~]# nmcli con up con2
Connection 'con2' successfully activated (D-Bus active path: /org/freedesktop/
NetworkManager/ActiveConnection/7)
```

Resolving DNS Issues

Network connectivity issues are notoriously caused by DNS. The `/etc/resolv.conf` configuration file contains a system's primary DNS configuration. By default, the `NetworkManager` service dynamically updates the `/etc/resolv.conf` file with the DNS settings from active connections.

```
[root@host ~]# cat /etc/resolv.conf
# Generated by NetworkManager
search lab.example.com example.com
nameserver 172.25.250.254
```

The output of the `host` command confirms that DNS servers are reachable and correctly resolving domain names.

```
[user@host ~]# host redhat.com 172.25.250.254
Using domain server:
Name: 172.25.250.254
Address: 172.25.250.254#53
...output omitted...
redhat.com has address 209.132.183.105
...output omitted...
```

**Note**

The `host` and `nslookup` commands ignore replies from name servers where the "Recursion Available" (ra) flag is not enabled, unless it is the last available server.

Alternatively, the `dig` or `mdig` commands generate a more detailed output than the `host` command. The `mdig` command sends multiple queries at once, rather than the single iterative queries from `dig`.

```
[root@host ~]# dig redhat.com
; <>> DiG 9.11.26-RedHat-9.11.26-3.el8 <>> redhat.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 18096
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

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

;; ANSWER SECTION:
redhat.com. 499 IN A 209.132.183.105

;; Query time: 1 msec
;; SERVER: 172.25.250.254#53(172.25.250.254)
;; WHEN: Thu Oct 21 15:40:28 EDT 2021
;; MSG SIZE rcvd: 55
```

The `status: NOERROR` value in the HEADER field confirms a successful domain resolution. A failed `dig` query likely includes the `NXDOMAIN` value, which indicates that the DNS server cannot resolve the provided domain name.

The `dig` command provides dozens of optional parameters to customize queries. The `+trace` option is particularly useful, because it displays which DNS servers were queried to resolve a domain, starting from the root name servers to every subsequent referenced server.

```
[root@host ~]# dig redhat.com +trace
...output omitted...
. 499674 IN NS b.root-servers.net.
...output omitted...
com. 172800 IN NS l.gtld-servers.net.
...output omitted...
redhat.com. 172800 IN NS a13-66.akam.net.
...output omitted...
redhat.com. 3600 IN A 209.132.183.105
```

While the `host` and `dig` commands are useful to diagnose DNS servers, they might not accurately reflect how a system resolves a particular domain name. A system might resolve a domain name with sources other than DNS, such as the `/etc/hosts` file. To mitigate this behavior, the `getent host` command resolves a domain name exactly how a system would, and does not exclusively check DNS servers.

```
[root@host ~]# cat /etc/hosts | grep redhat.com
127.0.0.1 redhat.com
[root@host ~]# getent hosts redhat.com
127.0.0.1      redhat.com
[root@host ~]# sed -i '/redhat.com/d' /etc/hosts
[root@host ~]# getent hosts redhat.com
209.132.183.105 redhat.com
```

In the preceding example, the `getent hosts` command resolves `redhat.com` to `127.0.0.1` until the `sed` command removes the entry from the `/etc/hosts` file.

Using NetworkManager for Manage DNS Configurations

If a connection is having DNS issues, then run the `nmcli con show` command to view its DNS configuration.

```
[root@host ~]# nmcli con show con0 | grep ipv4.dns
ipv4.dns:                           172.25.250.254
ipv4.dns-search:                     --
ipv4.dns-options:                   --
ipv4.dns-priority:                  0
```

If a fix is required, then the `nmcli con mod` command modifies a connection's DNS configuration. The changes appear in the `/etc/resolv.conf` file after the connection is restarted.

```
[root@host ~]# cat /etc/resolv.conf | grep nameserver
nameserver 172.25.250.254
[root@host ~]# nmcli con mod con0 ipv4.dns 192.168.0.254
[root@host ~]# nmcli con down con0
Connection 'con0' successfully deactivated (D-Bus active path: /org/freedesktop/
NetworkManager/ActiveConnection/6)
[root@host ~]# nmcli con up con0
Connection 'con0' successfully activated (D-Bus active path: /org/freedesktop/
NetworkManager/ActiveConnection/7)
[root@host ~]# cat /etc/resolv.conf | grep nameserver
nameserver 192.168.0.254
```

If DHCP is providing an incorrect or malfunctioning DNS server, then set a connection's `ignore-auto-dns` parameter equal to `yes` to ignore DNS servers from DHCP.

```
[root@host ~]# nmcli con mod con0 ipv4.ignore-auto-dns yes
```

Resolving Firewall Issues

The `firewalld` daemon might cause network issues by restricting access to network ports.

The `firewall-cmd` command displays and adjusts current firewall settings. Invoking the command with the `--list-all-zones` option provides a complete overview of the `firewalld` configuration.

```
[root@host ~]# firewall-cmd --list-all-zones
... output omitted ...
internal
  target: default
  icmp-block-inversion: no
  interfaces:
  sources:
  services: cockpit dhcpv6-client mdns samba-client ssh
  ports:
  protocols:
... output omitted ...
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: eth0
  sources:
  services: cockpit dhcpv6-client http https ssh
  ports:
  protocols:
... output omitted ...
```

Adding the `--permanent` option displays the persistent settings for `firewalld`. Comparing active `firewalld` settings with persistent settings can sometimes identify potential problems. The preceding output shows that the ports for the `http` and `https` services are not blocked. The following output shows that the port configuration is not persistent.

```
[root@host ~]# firewall-cmd --list-all-zones --permanent
... output omitted ...
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: eth0
  sources:
  services: cockpit dhcpv6-client ssh
  ports:
  protocols:
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
... output omitted ...
```

The `firewall-cmd --runtime-to-permanent` command converts runtime rules into permanent rules.

```
[root@host ~]# firewall-cmd --runtime-to-permanent
success
```

The `firewalld` daemon is built on top of the `nftables` framework. The `nft` command interacts with `nftables` and can access highly detailed firewall information that is unavailable with the `firewall-cmd` command.

For example, the `nft list ruleset` prints all of the active underlying firewall rules on a system.

```
[root@host ~]# nft list ruleset
...output omitted...
chain filter_IN_public_deny {
}

chain filter_IN_public_allow {
    tcp dport 22 ct state { new, untracked } accept
    ip6 daddr fe80::/64 udp dport 546 ct state { new, untracked } accept
    tcp dport 9090 ct state { new, untracked } accept
    tcp dport 443 ct state { new, untracked } accept
}
...output omitted...
```

The preceding output shows the `nftables` perspective of which ports are allowed in the "public" interface zone.



References

Consistent Network Interface Device Naming

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html/configuring_and_managing_networking/consistent-network-interface-device-naming_configuring-and-managing-networking
`firewall-cmd(1)`, `nft(8)`, `nmcli(1)`, and `resolv.conf(5)` man pages

► Guided Exercise

Resolving Connectivity Issues

In this exercise, you diagnose and resolve network connectivity issues.

Outcomes

You should be able to diagnose and resolve network connectivity issues on a Red Hat Enterprise Linux 8 host.

Before You Begin

As the student user on the **workstation** machine, use the **lab** command to prepare your system for this exercise.

```
[student@workstation ~]$ lab start network-resolving
```

This command configures network interfaces and modifies DNS servers.

Instructions

A second network interface, **eth1**, was added to the **servera** and **serverb** systems. The interfaces are connected to a private network and have the following IP addresses:

- **servera**: 172.24.250.10/24
- **serverb**: 172.24.250.11/24

Users report that the interfaces cannot communicate with each other. Additionally, DNS is not working on the **servera** system.

The **serverb** system is known to be correctly configured. Repair only the network configuration issues on the **servera** system. The network settings must persist when the machine is rebooted. The IP address of the working DNS server is 172.25.250.254.

When the network is functioning properly, the **servera** system should be able to ping the **eth1** interface on the **serverb** system and also ping the **redhat . com** website.

- 1. Log in to **servera** and switch to the **root** user.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

- 2. Verify the network connectivity issues.

- 2.1. Run the **ping** command against the IP address of the **eth1** interface on **serverb**. No reply is expected.

```
[root@servera ~]# ping -c 2 172.24.250.11
PING 172.24.250.11 (172.24.250.11) 56(84) bytes of data.

--- 172.24.250.11 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1059ms
```

- 2.2. Run the `ping` command with the `redhat.com` domain name. A "Name or service not known" error is expected.

```
[root@servera ~]# ping -c 2 redhat.com
ping: redhat.com: Name or service not known
```

► 3. Gather network information.

- 3.1. Display the current IP addresses.

```
[root@servera ~]# ip addr show
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
```

The `eth0` interface is up with an assigned IPv4 address and a link-local IPv6 address. The `eth1` interface is also up but does not have any IP addresses assigned to it.

- 3.2. Display the routing information.

```
[root@servera ~]# ip route
default via 172.25.250.254 dev eth0 proto static metric 100
172.25.250.0/24 dev eth0 proto kernel scope link src 172.25.250.10 metric 100
```

The 172.24.250.0/24 network does not have a route, because the `eth1` interface does not have an IP address in that subnet.

- 3.3. Display the connection and device information.

```
[root@servera ~]# nmcli con
NAME           UUID                                  TYPE      DEVICE
Wired connection 1  8c6fd7b1-ab62-a383-5b96-46e083e04bb1  802-3-ethernet  eth0
Wired connection 2  5fb06bd0-0bb0-7ffb-45f1-d6edd65f3e03  802-3-ethernet  --
```

```
[root@servera ~]# nmcli dev
DEVICE  TYPE      STATE       CONNECTION
eth0    ethernet  connected   Wired connection 1
eth1    ethernet  disconnected --
lo     loopback  unmanaged   --
```

NetworkManager has two connections: "Wired connection 1" and "Wired connection 2". The former is assigned to `eth0`, while no devices are assigned to the latter.

- 3.4. Run the `dig` command to look for any information regarding the DNS issue.

```
[root@servera ~]# dig redhat.com
; <>> DiG 9.11.26-RedHat-9.11.26-4.el8_4 <>> redhat.com
;; global options: +cmd
;; connection timed out; no servers could be reached
```

The output indicates that an incorrect DNS server might be configured.

- 3.5. View the configuration of the `/etc/resolv.conf` file.

```
[root@servera ~]# cat /etc/resolv.conf
# Generated by NetworkManager
search lab.example.com example.com
nameserver 172.24.250.11
```

The IP address of the DNS server is incorrect.

- 3.6. Inspect the connection profiles. Filter the output to the parameters that are expected to be relevant.

```
[root@servera ~]# nmcli con show "Wired connection 1" | grep dns
connection.mdns:                         -1 (default)
ipv4.dns:                                172.24.250.11
ipv4.dns-search:                          lab.example.com,example.com
ipv4.dns-options:                         --
ipv4.dns-priority:                        0
ipv4.ignore-auto-dns:                     yes
ipv6.dns:                                --
ipv6.dns-search:                          --
ipv6.dns-options:                         --
ipv6.dns-priority:                        0
ipv6.ignore-auto-dns:                     yes
[root@servera ~]# nmcli con show "Wired connection 2" | grep interface
connection.interface-name:                ens1
```

The "Wired connection 1" profile is configured with an incorrect IP address for its DNS server. The "Wired connection 2" profile is configured with the `ens1` interface, which does not exist. Modifying the `ipv4.dns` parameter of the "Wired connection 1" profile and changing the `ifname` parameter of "Wired connection 2" should resolve the network connectivity issues.

- 4. Repair the `eth1` interface issue, and then verify that the systems can communicate.

- 4.1. Modify the `ifname` parameter of the "Wired connection 2" profile to reflect the correct `eth1` interface.

```
[root@servera ~]# nmcli con mod "Wired connection 2" iface eth1
```

- 4.2. Run the `nmcli con up` command to activate the connection.

```
[root@servera ~]# nmcli con up "Wired connection 2"
Connection successfully activated (D-Bus active path: /org/freedesktop/
NetworkManager/ActiveConnection/3)
```

- 4.3. Verify that the connection is now active with the correct interface.

```
[root@servera ~]# nmcli con show
NAME           UUID                                  TYPE      DEVICE
Wired connection 1  06d487b0-4ef5-3dcb-8baa-260c6f435574  ethernet  eth0
Wired connection 2  97d45f09-316b-3678-b77d-9a3ef019d835  ethernet  eth1
```

- 4.4. Confirm that the `eth1` interface now has an IP address.

```
[root@servera ~]# ip addr show eth1
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 8942 qdisc fq_codel state UP group
default 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 noprefixroute eth1
        valid_lft forever preferred_lft forever
    inet6 fe80::243:e4c0:38fb:f19e/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
```

- 4.5. Verify that the `ping` command now works.

```
[root@servera ~]# ping -c 2 172.24.250.11
PING 172.24.250.11 (172.24.250.11) 56(84) bytes of data.
64 bytes from 172.24.250.11: icmp_seq=1 ttl=64 time=1.48 ms
64 bytes from 172.24.250.11: icmp_seq=2 ttl=64 time=0.311 ms

--- 172.24.250.11 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 0.311/0.897/1.483/0.586 ms
```

▶ 5. Repair the DNS issue and verify that the domain names resolve.

- 5.1. Modify the `ipv4.dns` parameter of the "Wired connection 1" profile to reflect the correct IP address of the DNS server.

```
[root@servera ~]# nmcli con mod "Wired connection 1" ipv4.dns 172.25.250.254
```

- 5.2. Attempt to ping the `redhat.com` website.

```
[root@servera ~]# ping -c 2 redhat.com
ping: redhat.com: Name or service not known
```

The error persists because the `/etc/resolv.conf` file does not change until the interface is restarted.

5.3. Restart the NetworkManager service to apply the changes.

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

**Note**

Normally, an interface is restarted to apply changes. However, in this case the SSH session drops if the interface goes down. Whenever possible, the NetworkManager service keeps connections active as it restarts and applies changes.

5.4. Verify that the `/etc/resolv.conf` file now shows the correct server.

```
[root@servera ~]# cat /etc/resolv.conf
# Generated by NetworkManager
search lab.example.com example.com
nameserver 172.25.250.254
```

5.5. Observe the output of a successful `dig` command. The NOERROR value in the header is expected.

```
[root@servera ~]# dig redhat.com
; <>> DiG 9.11.26-RedHat-9.11.26-4.el8_4 <>> redhat.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 64384
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

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

;; ANSWER SECTION:
redhat.com. 2397 IN A 209.132.183.105

;; Query time: 1 msec
;; SERVER: 172.25.250.254#53(172.25.250.254)
;; WHEN: Tue Oct 26 19:17:40 EDT 2021
;; MSG SIZE rcvd: 55
```

5.6. Verify that a ping to `redhat.com` now works.

```
[root@servera ~]# ping -c 2 redhat.com
PING redhat.com (209.132.183.105) 56(84) bytes of data.
64 bytes from redirect.redhat.com (209.132.183.105): icmp_seq=1 ttl=234 time=37.0
ms
64 bytes from redirect.redhat.com (209.132.183.105): icmp_seq=2 ttl=234 time=35.8
ms

--- redhat.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 35.762/36.394/37.026/0.632 ms
```

► **6.** Return to workstation as the student user.

```
[root@servera ~]# exit
[student@servera ~]$ exit
[student@workstation ~]$
```

Finish

On the **workstation** machine, use the **lab** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish network-resolving
```

This concludes the guided exercise.

Inspecting Network Traffic

Objectives

After completing this section, you should be able to inspect network traffic to assist troubleshooting.

Inspecting Network Traffic with Wireshark

Wireshark, formerly Ethereal, is an open source, graphical application for capturing, filtering, and inspecting network packets. Wireshark can perform promiscuous packet sniffing when network interface controllers support it. Packets are colorized for easy identification.

Protocol	Color
HTTP	Light green
TCP	Gray
UDP	Light blue
ARP	Light yellow
ICMP	Pink
Errors	Black

Red Hat Enterprise Linux 8 includes the `wireshark` package. This package provides Wireshark functionality on a system where X Windows is installed.

```
[root@host ~]# yum install wireshark
```

When Wireshark is installed, start it by selecting **Activities > Show Applications > Wireshark** from the GNOME desktop. You can also start Wireshark from the command line.

```
[root@host ~]# wireshark
```

Capturing Packets with Wireshark

Wireshark can capture network packets. Wireshark requires privileged user access to capture packets, because direct access to the network interfaces requires `root` privilege. On the **Capture** top-level menu, you can start and stop packet captures. Administrators select the interfaces to capture packets on. The `any` interface matches and captures on all of the network interfaces.

Wireshark can write captured packets to a file for sharing or later analysis. Use the **File > Save** or **File > Save as** menu items to specify a file to save the packets to. Wireshark supports multiple file formats.

Wireshark can read from a previously saved file to analyze captured packets. Analyzing packets from existing capture files does not require `root` privilege. The **File > Open** menu item selects

the file to open, or the user can specify the file as an option when starting Wireshark from the command line.

```
[user@host ~]$ wireshark -r yesterday-eth0.pcapng &
```

Inspecting Packets with Wireshark

Use the **Filter** field to enter expressions to limit which packets display in Wireshark. Wireshark recognizes and parses numerous network protocols. Enter `http` to filter the captured packets to display only HTTP TCP packets. Enter `ip` or `ipv6` to filter packets to display only IPv4 or IPv6 packets.

Use the **Expression** button to open a window to create more robust filtering expressions. The `ip.src == 192.168.10.23` expression filters the packets so that only packets that originate from the 192.168.10.23 IP address are displayed.

Use the panes in the Wireshark main display to inspect packet contents. The top pane displays the list of captured packet headers that are selected with the current filter. The highlighted packet is currently displayed in the middle and bottom panes. The bottom pane displays the whole packet, both headers and data, in hexadecimal and ASCII format.

The middle pane displays the packet as Wireshark parsed it. Each network layer header is displayed in a brief human-readable format. You can expand each header to inspect more detailed information about that network layer data. Each line starts with the header field name followed by its value. Wireshark translates recognized values to strings. 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 pane.

Wireshark can display related packets between a protocol client and server in a more readable format. To view this format, right-click a packet, and then select **Follow > TCP Stream**. Client messages are displayed in one color and server responses are displayed in a different color.

Capturing Packets with the CLI

The `tcpdump` command captures and displays packets in a nongraphical environment. It is in the `tcpdump` package and is installed by default on RHEL 8 systems that are purposed as servers.

When started without arguments, the `tcpdump` command captures and displays brief information about all of the departing and arriving packets on the primary network interface. Packet capture continues until the user interrupts the capture by typing `Ctrl+C`. Add the `-c COUNT` option to stop the capture after `COUNT` packets are captured. When the program exits, the packet capture summary for this session is displayed.

```
[root@host ~]# tcpdump -c 3
dropped privs to tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
22:40:57.581470 IP workstation.lab.example.com.ssh >
    bastion.lab.example.com.52100: Flags [P.], seq 3763069044:3763069256, ack
    272296579, win 316, options [nop,nop,TS val 3466052426 ecr 3699108528], length
    212
22:40:57.581671 IP bastion.lab.example.com.52100 >
    workstation.lab.example.com.ssh: Flags [.], ack 212, win 1424, options
    [nop,nop,TS val 3699108558 ecr 3466052426], length 0
```

```
22:40:57.581766 IP workstation.lab.example.com.ssh >
bastion.lab.example.com.52100: Flags [P.], seq 212:648, ack 1, win 316, options
[nop,nop,TS val 3466052426 ecr 3699108558], length 436
3 packets captured
4 packets received by filter
0 packets dropped by kernel
```

The `tcpdump` command sends packet information to standard output, unless the `-w FILE` option is used. Use the `-w` option to write packet details to the specified file as raw data for later analysis. Recommended practice is to name packet capture files with a `.pcap` extension for human recognition, although the `tcpdump` command does not require it. Start the `tcpdump` command with the `-r FILE` option to read from a capture file, instead of listening to network interfaces. Using the `tcpdump` command to process existing capture files does not require `root` privileges.

```
[user@host ~]$ 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 ...
```

The `tcpdump` command can filter captured packets based on an expression that is passed as the argument. The following example captures only packets to or from the `ntpserver` host.

```
[root@host ~]# tcpdump 'host ntpserver'
```

The following example captures ICMP packets to and from the host at the 192.168.32.7 IP address.

```
[root@host ~]# tcpdump 'icmp and host 192.168.32.7'
```

Logical operators support complex expressions. The following example filters all IP packets between the `matrix` host and any other host except `server175`.

```
[root@host ~]# tcpdump 'ip host matrix and not server175'
```

More syntax details and examples for `tcpdump` filter expressions can be found in the `pcap-filter(7)` man page.

The `tcpdump` command has options to control the information that it displays about each captured packet. Use the lowercase `-x` option to display all packet header and data as hexadecimal values. Use the uppercase `-X` option to display data as hexadecimal and ASCII values. The uppercase `-X` option is useful if the network protocol includes text commands.

For analysis, Wireshark can read and parse packet capture files that the `tcpdump` command created.



Note

Wireshark also has a text-based version, named `tshark`. This text version is included in the `wireshark` package.



References

pcap-filter(7), tcpdump(8), tcpslice(8), tshark(1), wireshark (1), and wireshark-filter(4) man pages

► Guided Exercise

Inspecting Network Traffic

In this exercise, you install Wireshark and use it to capture, filter, and inspect network packets. You also use `tcpdump` to perform similar functions in a textual environment.

Outcomes

You should be able to capture, filter, and inspect network packets with Wireshark and `tcpdump`.

Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

```
[student@workstation ~]$ lab start network-traffic
```

Instructions

- 1. As the `student` user on the `workstation` system, switch to the `root` user.

```
[student@workstation ~]$ sudo -i  
[sudo] password for student: student  
[root@workstation ~]#
```

Install the Wireshark package.

```
[root@workstation ~]# yum install wireshark  
...output omitted...  
Is this ok [y/N]: y  
...output omitted...  
Complete!
```

- 2. Open Wireshark and configure it to capture packets on all network interfaces.

- 2.1. As the `root` user, open Wireshark to run as a background process.

```
[root@workstation ~]# wireshark &
```

- 2.2. On the `Welcome to Wireshark` page, in the upper right of the `Capture` section, select `All interfaces` shown from the list.
- 2.3. In the center of the `Capture` section, click any from the interfaces list.
- 2.4. From the `Capture` menu, click `Start`, or type `Ctrl+E` to start the packet capture.

- 3. As the `root` user on the `workstation` machine, open a terminal window and generate network traffic.

- 3.1. Use the `ping` command to send ICMP requests to `servera`.

```
[root@workstation ~]# ping -c 5 servera.lab.example.com
```

- 3.2. Use the `chronyc ntpdata` command to generate NTP traffic that is specific to the `classroom` server.

```
[root@workstation ~]# chronyc ntpdata classroom.example.com
```

- 3.3. Open Firefox and browse `http://materials.example.com` to generate HTTP network traffic.
- 3.4. From the **Capture** menu, click **Stop**, or type `Ctrl+E`, to stop the packet capture.

► **4.** Perform simple filter and inspection routines on the captured network traffic.

- 4.1. Filter the packets for ICMP traffic. In the **Apply a display filter** field, type `icmp`. Press **Enter**. From the middle pane, click any of the right-pointing arrows to inspect the different packet header and data values.
- 4.2. Filter the packets for NTP traffic. In the **Apply a display filter** field, type `ntp`. Press **Enter**.
- 4.3. Filter the packets for HTTP traffic. In the **Apply a display filter** field, type `http`. Press **Enter**. From the top pane, right-click any of the HTTP packets, and then select **Follow > TCP Stream** to display the stream content in a more readable format. Messages from the HTTP client are displayed in red. Responses from the HTTP server are displayed in blue.

► **5.** Save the captured packet data for later analysis.

- 5.1. From the **File** menu, click **Save**, or type `Ctrl+S` to open the **Save Capture File As** window. From the **Look in** list, select the `root` user's home directory. In the **File name** field, type `practice`. Leave `Wireshark/... - pcapng` as the selected type in the **Save As** field. Click **Save**.

Confirm that the `practice.pcapng` data file exists.

```
[root@workstation ~]# ls practice*
practice.pcapng
[root@workstation ~]# file practice*
practice.pcapng: pcap-ng capture file - version 1.0
```

- 5.2. Close Wireshark. From the **File** menu, click **Quit**.

► **6.** As the `student` user on the `workstation` machine, open another terminal window. Download the preconfigured `rh342-practice.tcpdump` capture file and open it in Wireshark.

- 6.1. Download the capture file.

```
[student@workstation ~]$ curl -o http://materials.example.com/labs/network-traffic/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
```

- 6.2. As the **student** user on the **workstation** machine, open Wireshark from the desktop. Click **Activities** in the upper left of the desktop. In the **Type to search** field, in the top center of the desktop, type **wireshark**. Click **Wireshark**.
 - 6.3. Open the capture file. From the **File** menu, click **Open**, or type **Ctrl+O** to open the **Open Capture File** window. Browse and select the **rh342-practice.tcpdump** file. Click **Open**.
 - 6.4. Filter the captured data so that only SMTP packets are displayed. In the **Apply a display filter** field, type **smtp**. Press **Enter**.
 - 6.5. Inspect the content of the network packets. In the middle pane, click any of the right-pointing triangles to expand different parts of the network packet.
 - 6.6. In the top pane, right-click any packet in the SMTP exchange, and then select **Follow > TCP Stream** to display the stream content in a more readable format. Messages from the SMTP server are displayed in blue and messages from the SMTP client are displayed in red.
You should see text that reads "GOLD RING HAS BEEN CAUGHT", which is the first line in the body of the mail message.
 - 6.7. Close Wireshark. From the **File** menu, click **Quit**.
- 7. Capture and display network traffic, which requires **root** user access.

- 7.1. As the **student** user on the **workstation** machine, log in to the **serverb** machine and switch to the **root** user.

```
[student@workstation ~]$ ssh student@serverb
...output omitted...
[student@serverb ~]$ sudo -i
[sudo] password for student: student
[root@serverb ~]#
```

- 7.2. Verify that the **tcpdump** package is installed.

```
[root@serverb ~]# rpm -q tcpdump
tcpdump-4.9.3-1.el8.x86_64
```

- 7.3. Start the **tcpdump** utility on the **serverb** machine so that it captures network traffic for the **servera** machine.

```
[root@serverb ~]# tcpdump 'ip host servera'
dropped privs to tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
```

- 7.4. From a separate terminal window, log in as the student user on the servera machine.

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

- 7.5. Generate ICMP network traffic that is addressed to the serverb machine.

```
[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=0.060 ms
64 bytes from serverb.lab.example.com (172.25.250.11): icmp_seq=2 ttl=64
time=0.052 ms
64 bytes from serverb.lab.example.com (172.25.250.11): icmp_seq=3 ttl=64
time=0.055 ms

--- serverb.lab.example.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2047ms
rtt min/avg/max/mdev = 0.052/0.055/0.060/0.009 ms
```

- 7.6. From the serverb terminal window, view the tcpdump output that is displayed on the serverb machine.

```
22:53:20.038855 IP servera.lab.example.com > serverb.lab.example.com: ICMP echo
request, id 1610, seq 1, length 64
22:53:20.038963 IP serverb.lab.example.com > servera.lab.example.com: ICMP echo
reply, id 1610, seq 1, length 64
22:53:21.045671 IP servera.lab.example.com > serverb.lab.example.com: ICMP echo
request, id 1610, seq 2, length 64
22:53:21.045736 IP serverb.lab.example.com > servera.lab.example.com: ICMP echo
reply, id 1610, seq 2, length 64
22:53:22.069693 IP servera.lab.example.com > serverb.lab.example.com: ICMP echo
request, id 1610, seq 3, length 64
22:53:22.069752 IP serverb.lab.example.com > servera.lab.example.com: ICMP echo
reply, id 1610, seq 3, length 64
```

- 7.7. End the tcpdump packet capture on the serverb machine.

```
Ctrl+C
6 packets captured
6 packets received by filter
0 packets dropped by kernel
```

- 8. Capture incoming network traffic for a specific port. Save the captured packets to a file for later analysis.

- 8.1. Start the `tcpdump` utility on the `serverb` machine so that it captures HTTP traffic. Save the captured network packets to the `http-test.tcpdump` file.

```
[root@serverb ~]# tcpdump -w http-test.tcpdump 'port 80'  
dropped privs to tcpdump  
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
```

- 8.2. In the `servera` terminal window, display the web content from the `serverb` machine.

```
[student@servera ~]$ curl http://serverb.lab.example.com  
SERVERB is providing web content.
```

- 8.3. In the `serverb` terminal window, interrupt the `tcpdump` utility to stop the packet capture.

```
Ctrl+C  
12 packets captured  
12 packets received by filter  
0 packets dropped by kernel
```

► 9. Display the captured network traffic.

- 9.1. In the `serverb` terminal window, use the `tcpdump` utility to display the captured network traffic. If `-r FILENAME` is the only option, then only packet header information is displayed by default.

```
[root@serverb ~]# tcpdump -r http-test.tcpdump  
reading from file http-test.tcpdump, link-type EN10MB (Ethernet)  
dropped privs to tcpdump  
20:42:16.374181 IP servera.lab.example.com.60408 > serverb.lab.example.com.http:  
Flags [S], seq 2599241697, win 29200, options [mss 1460,sackOK,TS val 3400465832  
ecr 0,nop,wscale 7], length 0  
20:42:16.374284 IP serverb.lab.example.com.http > servera.lab.example.com.60408:  
Flags [R.], seq 0, ack 2599241698, win 0, length 0  
20:43:31.505131 IP servera.lab.example.com.60412 > serverb.lab.example.com.http:  
Flags [S], seq 3593487441, win 29200, options [mss 1460,sackOK,TS val 3400540964  
ecr 0,nop,wscale 7], length 0  
20:43:31.505208 IP serverb.lab.example.com.http > servera.lab.example.com.60412:  
Flags [S.], seq 1455608366, ack 3593487442, win 28960, options [mss  
1460,sackOK,TS val 1722155038 ecr 3400540964,nop,wscale 7], length 0  
...output omitted...
```

- 9.2. Display incoming packets that originate only from the `servera` machine. Display the packets in hexadecimal format.

```
[root@serverb ~]# tcpdump -x -r http-test.tcpdump 'src servera'  
reading from file http-test.tcpdump, link-type EN10MB (Ethernet)  
dropped privs to tcpdump  
20:42:16.374181 IP servera.lab.example.com.60408 > serverb.lab.example.com.http:  
Flags [S], seq 2599241697, win 29200, options [mss 1460,sackOK,TS val 3400465832  
ecr 0,nop,wscale 7], length 0
```

```
0x0000: 4500 003c 8b66 4000 4006 630c ac19 fa0a
0x0010: ac19 fa0b ebff 0050 9aed 47e1 0000 0000
0x0020: a002 7210 f236 0000 0204 05b4 0402 080a
0x0030: caae fda8 0000 0000 0103 0307
20:43:31.505131 IP servera.lab.example.com.60412 > serverb.lab.example.com.http:
Flags [S], seq 3593487441, win 29200, options [mss 1460,sackOK,TS val 3400540964
ecr 0,nop,wscale 7], length 0
0x0000: 4500 003c d7e6 4000 4006 168c ac19 fa0a
0x0010: ac19 fa0b ebfc 0050 d630 4451 0000 0000
0x0020: a002 7210 4c78 0000 0204 05b4 0402 080a
0x0030: cab0 2324 0000 0000 0103 0307
...output omitted...
```

- 9.3. Display incoming packets that originate only from the servera machine. Display the packets in hexadecimal and ASCII formats.

```
[root@serverb ~]# tcpdump -X -r http-test.tcpdump 'src servera'
reading from file http-test.tcpdump, link-type EN10MB (Ethernet)
dropped privs to tcpdump
20:42:16.374181 IP servera.lab.example.com.60408 > serverb.lab.example.com.http:
Flags [S], seq 2599241697, win 29200, options [mss 1460,sackOK,TS val 3400465832
ecr 0,nop,wscale 7], length 0
0x0000: 4500 003c 8b66 4000 4006 630c ac19 fa0a E..<.f@.@@.c....
0x0010: ac19 fa0b ebff 0050 9aed 47e1 0000 0000 .....P..G....
0x0020: a002 7210 f236 0000 0204 05b4 0402 080a ..r..6.....
0x0030: caae fda8 0000 0000 0103 0307 .....
20:43:31.505131 IP servera.lab.example.com.60412 > serverb.lab.example.com.http:
Flags [S], seq 3593487441, win 29200, options [mss 1460,sackOK,TS val 3400540964
ecr 0,nop,wscale 7], length 0
0x0000: 4500 003c d7e6 4000 4006 168c ac19 fa0a E..<..@.@@.....
0x0010: ac19 fa0b ebfc 0050 d630 4451 0000 0000 .....P.0DQ....
0x0020: a002 7210 4c78 0000 0204 05b4 0402 080a ..r.Lx.....
0x0030: cab0 2324 0000 0000 0103 0307 ...$.....
20:43:31.505357 IP servera.lab.example.com.60412 > serverb.lab.example.com.http:
Flags [.], ack 1455608367, win 229, options [nop,nop,TS val 3400540964 ecr
1722155038], length 0
0x0000: 4500 0034 d7e7 4000 4006 1693 ac19 fa0a E..4..@.@@.....
0x0010: ac19 fa0b ebfc 0050 d630 4452 56c2 d22f .....P.0DRV../
0x0020: 8010 00e5 4c70 0000 0101 080a cab0 2324 ....Lp.....$#
0x0030: 66a6 001e .....f...
20:43:31.505391 IP servera.lab.example.com.60412 > serverb.lab.example.com.http:
Flags [P.], seq 0:87, ack 1, win 229, options [nop,nop,TS val 3400540964 ecr
1722155038], length 87: HTTP: GET / HTTP/1.1
0x0000: 4500 008b d7e8 4000 4006 163b ac19 fa0a E.....@.@@.;....
0x0010: ac19 fa0b ebfc 0050 d630 4452 56c2 d22f .....P.0DRV../
0x0020: 8018 00e5 4cc7 0000 0101 080a cab0 2324 ....L.....#$#
0x0030: 66a6 001e 4745 5420 2f20 4854 5450 2f31 f...GET./.HTTP/1
0x0040: 2e31 0d0a 486f 7374 3a20 7365 7276 6572 .1..Host:.server
0x0050: 622e 6c61 622e 6578 616d 706c 652e 636f b.lab.example.co
0x0060: 6d0d 0a55 7365 722d 4167 656e 743a 2063 m..User-Agent:.c
0x0070: 7572 6c2f 372e 3631 2e31 0d0a 4163 6365 url/7.61.1..Acce
0x0080: 7074 3a20 2a2f 2a0d 0a0d 0a pt:./.....
...output omitted...
```

- 9.4. Copy the captured HTTP data to the **workstation** machine for further analysis. Use Wireshark on the **workstation** machine to view the captured HTTP network traffic. Follow the TCP stream to view the HTTP exchange in a more readable format.

```
[root@serverb ~]# scp http-test.tcpdump student@workstation:  
...output omitted...  
student@workstation's password: student  
http-test.tcpdump                                         100% 1395      1.5MB/s   00:00
```

- 9.5. Return to **workstation** as the **student** user.

```
[root@serverb ~]# exit  
[student@serverb ~]$ exit  
[student@workstation ~]$
```

Finish

On the **workstation** machine, use the **lab** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish network-traffic
```

This concludes the guided exercise.

▶ Lab

Troubleshooting Network Issues

In this lab, you identify and correct the misconfigured network settings of a server.

Outcomes

You should be able to diagnose and repair network configuration issues that cause a network service to be unavailable.

Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

```
[student@workstation ~]$ lab start network-review
```

This command creates the necessary files to perform this lab.

Instructions

A web server on `serverb` publishes content at the URL: `http://serverb.lab.example.com`.

The original network specifications for the `serverb` web server are not available. The network team properly configured the DNS entry for `serverb` on their name servers. Diagnose and fix the network configuration on `serverb` to ensure that the web content is available at the given URL. Verify that the network fixes are persistent.



Important

Because this exercise introduces network problems, you might lose ssh connectivity to the virtual machine that you are fixing. Therefore, this lab instructs you to log in directly to the system console for the `serverb` system when creating and resolving this problem. All Red Hat Training delivery modes provide direct access to classroom virtual machine consoles.

1. As the root user on the `serverb` system console, run the `net-trouble-issue` command to create the problem to troubleshoot.
2. As the student user on the `servera` system, discover which public IP network settings are expected for the `serverb` system.
3. Diagnose and fix the network issue on the `serverb` machine.
4. Take any further needed steps to ensure access to the web service content, including to verify the firewall and the web server's `systemd` service.
5. Return to the `workstation` machine as the student user.

Evaluation

On the **workstation** machine, use the **lab** command to grade your work. Correct any reported failures and rerun the script until you receive a passing grade.

```
[student@workstation ~]$ lab grade network-review
```



Important

After successful grading, you must run the **net-restore-exercise** command on the **serverb** machine to ensure that the **finish** script runs successfully. Use the direct **serverb** console, not **ssh**, to access the **serverb** system.

As the **root** user on the **serverb** system console, run the **net-restore-exercise** command, before running the **lab finish** command on the **workstation** system.

```
[root@serverb ~]# net-restore-exercise
```

Finish

On the **workstation** machine, use the **lab** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish network-review
```

This concludes the lab.

► Solution

Troubleshooting Network Issues

In this lab, you identify and correct the misconfigured network settings of a server.

Outcomes

You should be able to diagnose and repair network configuration issues that cause a network service to be unavailable.

Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

```
[student@workstation ~]$ lab start network-review
```

This command creates the necessary files to perform this lab.

Instructions

A web server on `serverb` publishes content at the URL: `http://serverb.lab.example.com`.

The original network specifications for the `serverb` web server are not available. The network team properly configured the DNS entry for `serverb` on their name servers. Diagnose and fix the network configuration on `serverb` to ensure that the web content is available at the given URL. Verify that the network fixes are persistent.



Important

Because this exercise introduces network problems, you might lose ssh connectivity to the virtual machine that you are fixing. Therefore, this lab instructs you to log in directly to the system console for the `serverb` system when creating and resolving this problem. All Red Hat Training delivery modes provide direct access to classroom virtual machine consoles.

1. As the `root` user on the `serverb` system console, run the `net-trouble-issue` command to create the problem to troubleshoot.
 - 1.1. Using the correct method for your classroom environment, access the `serverb` system's text console. Log in on the `serverb` console as the `root` user with `redhat` as the password.

```
serverb login: root
password: redhat
...output omitted...
[root@serverb ~]#
```

- 1.2. As the **root** user on the **serverb** system console, run the **net-trouble-issue** command.

```
[root@serverb ~]# net-trouble-issue
```

2. As the **student** user on the **servera** system, discover which public IP network settings are expected for the **serverb** system.
 - 2.1. In a new terminal window on the **workstation** system, log in to the **servera** system as the **student** user.

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

- 2.2. Discover the expected public IPv4 address of the **serverb** machine.

```
[student@servera ~]$ host serverb.lab.example.com  
serverb.lab.example.com has address 172.25.250.11
```

- 2.3. Identify the expected IPv4 netmask, knowing that **workstation**, **servera**, and **serverb** are on the same subnet.

```
[student@servera ~]$ ip addr show dev eth0  
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group  
default 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 noprefixroute eth0  
        valid_lft forever preferred_lft forever  
    inet6 fe80::2c1:1c11:1485:3f4c/64 scope link noprefixroute  
        valid_lft forever preferred_lft forever
```

The netmask is /24.

- 2.4. Identify the gateway, knowing that **workstation**, **servera**, and **serverb** must use the same gateway.

```
[student@servera ~]$ ip route  
default via 172.25.250.254 dev eth0 proto static metric 100  
172.25.250.0/24 dev eth0 proto kernel scope link src 172.25.250.10 metric 100
```

The default gateway is 172.25.250.254.

- 2.5. Determine the DNS name server that the **serverb** machine can use, knowing that **workstation**, **servera**, and **serverb** use the same name server.

```
[student@servera ~]$ cat /etc/resolv.conf  
# Generated by NetworkManager  
search lab.example.com example.com  
nameserver 172.25.250.254
```

3. Diagnose and fix the network issue on the **serverb** machine.

- 3.1. You should already be logged in to the serverb system console as the root user. Display the current network settings to identify the number of network interfaces and their configuration.
- The eth0 interface has an incorrect IP address.

```
[root@serverb ~]# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    qlen 1000
        link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
        inet 127.0.0.1/8 scope host lo
            valid_lft forever preferred_lft forever
        inet6 ::1/128 scope host
            valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default
    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 noprefixroute eth0
            valid_lft forever preferred_lft forever
        inet6 fe80::8e80:72dc:3b82:55e7/64 scope link noprefixroute
            valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 8942 qdisc fq_codel state UP group default
    qlen 1000
        link/ether 52:54:00:01:fa:0b brd ff:ff:ff:ff:ff:ff
```

- 3.2. Determine the NetworkManager connection names for the two physical network interfaces.

```
[root@serverb ~]# nmcli conn
NAME           UUID                                  TYPE      DEVICE
Wired connection 1  df120ee2-aded-3ef7-9ff9-9f33c48c9043  ethernet  eth0
Wired connection 2  433d347a-dceb-3549-a86e-981b863bd5d2  ethernet  --
```

Depending on your system configuration, the output might display both eth0 and eth1 device names.

- 3.3. Display the IPv4 NetworkManager settings for `Wired connection 1`.

The DNS server settings look correct, but the network interface is configured for the wrong subnet. It must be configured for the network information that you discovered previously.

```
[root@serverb ~]# nmcli conn show 'Wired connection 1' | grep ipv4
ipv4.method:                      manual
ipv4.dns:                          172.25.250.254
ipv4.dns-search:                   lab.example.com,example.com
ipv4.dns-options:                  --
ipv4.dns-priority:                 0
ipv4.addresses:                    172.25.251.11/24
ipv4.gateway:                      172.25.251.254
ipv4.routes:                       --
ipv4.route-metric:                 -1
ipv4.route-table:                  0 (unspec)
ipv4.routing-rules:                --
ipv4.ignore-auto-routes:           no
```

```
ipv4.ignore-auto-dns: yes
ipv4.dhcp-client-id: --
ipv4.dhcp-iaid: --
ipv4.dhcp-timeout: 0 (default)
ipv4.dhcp-send-hostname: yes
ipv4.dhcp-hostname: --
ipv4.dhcp-fqdn: --
ipv4.dhcp-hostname-flags: 0x0 (none)
ipv4.never-default: no
ipv4.may-fail: yes
ipv4.dad-timeout: -1 (default)
ipv4.dhcp-vendor-class-identifier: --
ipv4.dhcp-reject-servers: --
```

- 3.4. Copy the interface configuration files into the `root` home directory as a backup before making changes.

```
[root@serverb ~]# cp /etc/sysconfig/network-scripts/ifcfg-* .
[root@serverb ~]#
```

- 3.5. Modify the network connection for the `eth0` device.

```
[root@serverb ~]# nmcli conn mod 'Wired connection 1' ipv4.addresses
'172.25.250.11/24' ipv4.gateway 172.25.250.254
[root@serverb ~]#
```

- 3.6. Verify the new settings.

```
[root@serverb ~]# nmcli conn show 'Wired connection 1' | grep ipv4
ipv4.method: manual
ipv4.dns: 172.25.250.254
ipv4.dns-search: lab.example.com,example.com
ipv4.dns-options: --
ipv4.dns-priority: 0
ipv4.addresses: 172.25.250.11/24
ipv4.gateway: 172.25.250.254
ipv4.routes: --
ipv4.route-metric: -1
ipv4.route-table: 0 (unspec)
ipv4.routing-rules: --
ipv4.ignore-auto-routes: no
ipv4.ignore-auto-dns: yes
ipv4.dhcp-client-id: --
ipv4.dhcp-iaid: --
ipv4.dhcp-timeout: 0 (default)
ipv4.dhcp-send-hostname: yes
ipv4.dhcp-hostname: --
ipv4.dhcp-fqdn: --
ipv4.dhcp-hostname-flags: 0x0 (none)
ipv4.never-default: no
ipv4.may-fail: yes
```

```
ipv4.dad-timeout: -1 (default)
ipv4.dhcp-vendor-class-identifier: --
ipv4.dhcp-reject-servers: --
```

- 3.7. To implement the new configuration, deactivate the 'Wired connection 1' profile, and then reactivate it. The active configuration number will change.

```
[root@serverb ~]# nmcli conn down 'Wired connection 1'
Connection 'Wired connection 1' successfully deactivated (D-Bus active path:
/org/freedesktop/NetworkManager/ActiveConnection/645)
[root@serverb ~]# nmcli conn up 'Wired connection 1'
Connection successfully activated (D-Bus active path:
/org/freedesktop/NetworkManager/ActiveConnection/676)
```

- 3.8. Verify that the network configuration settings are correct.

```
[root@serverb ~]# ip addr show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group
    default 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 noprefixroute eth0
            valid_lft forever preferred_lft forever
        inet6 fe80::8e80:72dc:3b82:55e7/64 scope link noprefixroute
            valid_lft forever preferred_lft forever
```

4. Take any further needed steps to ensure access to the web service content, including to verify the firewall and the web server's `systemd` service.

- 4.1. Verify that the HTTP and HTTPS services are present in the firewall services list.

```
[root@serverb ~]# firewall-cmd --list-services
cockpit dhcpcv6-client http https ssh
```

- 4.2. Restart the `httpd` service so that it binds to the newly configured address.

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

- 4.3. As the `student` user on the `servera` machine, verify that web content is available from the `serverb` machine.

```
[student@servera ~]$ curl http://serverb.lab.example.com
SERVERB is providing web content.
```

5. Return to the `workstation` machine as the `student` user.

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

Evaluation

On the `workstation` machine, use the `lab` command to grade your work. Correct any reported failures and rerun the script until you receive a passing grade.

```
[student@workstation ~]$ lab grade network-review
```



Important

After successful grading, you must run the `net-restore-exercise` command on the `serverb` machine to ensure that the `finish` script runs successfully. Use the direct `serverb` console, not `ssh`, to access the `serverb` system.

As the `root` user on the `serverb` system console, run the `net-restore-exercise` command, before running the `lab finish` command on the `workstation` system.

```
[root@serverb ~]# net-restore-exercise
```

Finish

On the `workstation` machine, use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish network-review
```

This concludes the lab.

Summary

In this chapter, you learned:

- The `ping` and `ping6` commands are useful for sending ICMP requests to verify network connections.
- The `nmap` command can quickly scan a network to look for hosts, or can run longer to perform exhaustive port scans.
- The `ncat`, or `nc`, command can act as a network client, or as a simple server for testing network connectivity.
- The `iptraf-ng` tool monitors network traffic and provides statistics.
- The `dig`, `host`, and `getent` commands help to troubleshoot DNS-related network issues.
- The `nmcli` command diagnoses and repairs network configuration problems with NetworkManager-managed interfaces.
- To activate new settings on NetworkManager interfaces, deactivate and then reactivate the connection profile.
- Wireshark is an effective tool for capturing, filtering, and inspecting network packets.
- The `tcpdump` command can capture and display packets in text-only environments.

Chapter 8

Troubleshooting Application Issues

Goal

Identify and resolve application issues with debugging tools.

Objectives

- Identify library dependencies for installed software.
- Identify applications that have memory leaks.
- Debug an application with standard RHEL tools.
- Troubleshoot a containerized application or service.

Sections

- Resolving Library Dependencies (and Guided Exercise)
- Debugging Memory Leaks (and Guided Exercise)
- Debugging Application Execution (and Guided Exercise)
- Troubleshooting Containerized Applications (and Guided Exercise)

Lab

Troubleshooting Application Issues

Resolving Library Dependencies

Objectives

After completing this section, you should be able to identify library dependencies for installed software.

Using Shared Libraries in Applications

Most Linux applications are dynamically linked against the shared libraries that they use. Shared libraries are built so that their functions can be mapped into an application's memory when the application is executed. Because the required code is provided by the library at runtime, it is not copied into the application itself.

The dynamic linking of applications with shared libraries has many benefits:

- Application binaries are smaller due to the external shared code.
- Libraries can be shared simultaneously with many running programs.
- Shared libraries make it easier to fix bugs in the library code, without requiring application rebuilds.

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 verifies that they exist.

Identifying information about each required library is embedded in the executable. This information 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 field includes the name and version of the shared library, so that the application uses the correct version at runtime. The `ln` linker with the `-soname` option sets this field when creating the shared library. The `objdump` command includes this field when displaying shared library information.

```
[user@host ~]$ objdump -p /usr/lib64/libpthread-2.28.so | grep SONAME
SONAME           libpthread.so.0
```

The shared library uses a symbolic link that is named with the DT_SONAME field.

```
[user@host ~]$ ls -l /usr/lib64/libpthread*
-rwxr-xr-x. 1 root root 320704 Mar  5 2021 /usr/lib64/libpthread-2.28.so
lrwxrwxrwx. 1 root root      18 Mar  5 2021 /usr/lib64/libpthread.so.0 ->
libpthread-2.28.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 8, the `/lib64/ld-linux-x86-64.so.2` library is the default, 64-bit version of the runtime linker. The `glibc.x86_64` package provides the runtime linker. The 32-bit version of the `glibc` library provides the 32-bit version of the runtime linker when it is installed.

```
[user@host ~]$ rpm -qlp glibc-2.28-151.el8.i686.rpm | grep ld-linux  
/lib/ld-linux.so.2
```

The runtime linker uses the embedded DT_SONAME names in the application to determine which library versions to use. The runtime linker searches the following items, in order, to determine which library version to use:

- The specified directories 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.
- The /etc/ld.so.cache cache file. This file contains a compiled list of previously found candidate libraries. The `ldconfig -p` command prints the list of libraries that are mapped in /etc/ld.so.cache.

```
[user@host ~]$ ldconfig -p  
566 libs found in cache '/etc/ld.so.cache'  
p11-kit-trust.so (libc6,x86-64) => /lib64/p11-kit-trust.so  
libzstd.so.1 (libc6,x86-64) => /lib64/libzstd.so.1  
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.12 (libc6,x86-64) => /lib64/libxtables.so.12  
...output omitted...
```

- The directories in the default library paths. The 64-bit shared library loader points to 64-bit versions of the default locations, /lib64 and then /usr/lib64. The 32-bit runtime linker searches in turn the /lib and the /usr/lib directories.

```
[user@host ~]$ strings /lib64/ld-linux-x86-64.so.2 | grep '^/'  
/usr/libH9  
/t E  
/t E  
/usr/libI9  
/uzH  
/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.

```
[user@host ~]$ ldd /usr/sbin/httpd
linux-vdso.so.1 (0x00007ffe4f7bd000)
libpcre.so.1 => /lib64/libpcre.so.1 (0x00007f5994c55000)
libselinux.so.1 => /lib64/libselinux.so.1 (0x00007f5994a2b000)
libsystemd.so.0 => /lib64/libsystemd.so.0 (0x00007f59946e4000)
libaprutil-1.so.0 => /lib64/libaprutil-1.so.0 (0x00007f59944b6000)
libcrypt.so.1 => /lib64/libcrypt.so.1 (0x00007f599428d000)
libexpat.so.1 => /lib64/libexpat.so.1 (0x00007f5994052000)
libapr-1.so.0 => /lib64/libapr-1.so.0 (0x00007f5993e18000)
libpthread.so.0 => /lib64/libpthread.so.0 (0x00007f5993bf8000)
libdl.so.2 => /lib64/libdl.so.2 (0x00007f59939f4000)
libc.so.6 => /lib64/libc.so.6 (0x00007f599362f000)
libpcre2-8.so.0 => /lib64/libpcre2-8.so.0 (0x00007f59933ab000)
/lib64/ld-linux-x86-64.so.2 (0x00007f5995154000)
librt.so.1 => /lib64/librt.so.1 (0x00007f59931a3000)
liblzma.so.5 => /lib64/liblzma.so.5 (0x00007f5992f7c000)
liblz4.so.1 => /lib64/liblz4.so.1 (0x00007f5992d5f000)
libcap.so.2 => /lib64/libcap.so.2 (0x00007f5992b59000)
libmount.so.1 => /lib64/libmount.so.1 (0x00007f59928ff000)
libgcrypt.so.20 => /lib64/libgcrypt.so.20 (0x00007f59925e1000)
libgcc_s.so.1 => /lib64/libgcc_s.so.1 (0x00007f59923c9000)
libuuid.so.1 => /lib64/libuuid.so.1 (0x00007f59921c1000)
libblkid.so.1 => /lib64/libblkid.so.1 (0x00007f5991f6e000)
libgpg-error.so.0 => /lib64/libgpg-error.so.0 (0x00007f5991d4d000)
```



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 RHEL system when third-party software is installed with a method other than yum or rpm. Also, using the --force or --nodeps options with rpm can cause library dependency issues.

Library dependency issues are easy to identify. When an application references an unavailable shared library, the runtime linker displays a distinctive error message. The runtime linker displays the name of the first library that it cannot find, and then exits the application with an exit status of 127.

```
[user@host ~]$ application
application: error while loading shared libraries: library.so.X: cannot
open shared object file: No such file or directory
[user@host ~]$ echo $?
127
```

An application might be missing more libraries than the runtime linker indicates. The ldd command displays all of the shared libraries that an application uses. If a library is missing, then the ldd command displays it as *not found*.

```
[user@host ~]$ which application
/usr/bin/application
[student@host ~]$ 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)
```

If shared libraries are missing, then install them. RPM packages include shared library dependency information in the package's metadata.

```
[user@host ~]$ rpm -q --requires httpd | grep pthread
libpthread.so.0()(64bit)
libpthread.so.0(GLIBC_2.2.5)(64bit)
```

The `yum provides` command identifies the package that provides the specified shared library.

```
[user@host ~]$ yum provides '*/lib/libpthread.so.0'
Updating Subscription Management repositories.
glibc-2.28-151.el8.i686 : The GNU libc libraries
Repo        : rhel-8-for-x86_64-baseos-rpms
Matched from:
Filename    : /lib/libpthread.so.0
...output omitted...
```

If the shared library is not packaged and distributed by Red Hat, then contact the third-party vendor to obtain it. When a shared library is installed or removed, run the `ldconfig` command to update the runtime linker cache, `/etc/ld.so.cache`, with the changed information.

```
[user@host ~]$ rpm -q --scripts libnls
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 exercise, you install third-party software and resolve library dependencies.

Outcomes

You should be able to identify which libraries an application requires to run on a system.

Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

```
[student@workstation ~]$ lab start application-dependencies
```

This command confirms that the required hosts for this exercise are accessible and installs the application package.

Instructions

Another system administrator installed the `genisoimage` application and cannot run it. You are asked to solve the issue.

- 1. Log in as `student` on `servera`. Attempt to execute the newly installed program, and observe how it behaves. An error message is displayed. The program generates the same error message when failing to resolve a library as when the shell cannot locate a command.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[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
```

- 2. Determine which shared libraries the program requires. The error message states that `libusal.so.0` is required, but that other required libraries might also be missing. All but two libraries are resolved: `libusal.so.0` and `libr01s.so.0`.

```
[student@servera ~]$ which genisoimage
/usr/bin/genisoimage
[student@servera ~]$ ldd /usr/bin/genisoimage
linux-vdso.so.1 (0x00007ffe333e9000)
libmagic.so.1 => /lib64/libmagic.so.1 (0x00007fab7d738000)
libc.so.6 => /lib64/libc.so.6 (0x00007fab7d373000)
libz.so.1 => /lib64/libz.so.1 (0x00007fab7d15c000)
```

```
libbz2.so.1 => /lib64/libbz2.so.1 (0x00007fab7cf4b000)
libusal.so.0 => not found
librols.so.0 => not found
...output omitted...
```

- 3. Locate the packages that provide those two libraries.

```
[student@servera ~]$ yum provides libusal.so.0
libusal-1.1.11-39.el8.i686 : Library to communicate with SCSI devices
Repo       : rhel-8.4-for-x86_64-appstream-rpms
Matched from:
Provide   : libusal.so.0
[student@servera ~]$ yum provides librols.so.0
libusal-1.1.11-39.el8.i686 : Library to communicate with SCSI devices
Repo       : rhel-8.4-for-x86_64-appstream-rpms
Matched from:
Provide   : librols.so.0
```

- 4. The **libusal** package provides both libraries. Use the **sudo** command to run **yum** to install it. Use **student** as the password if prompted.

```
[student@servera ~]$ sudo yum install libusal
[sudo] password for student: student
...output omitted...
```

- 5. As the **student** user, attempt to run the original program again.

The executable appears to run, but displays a new message because of invalid arguments.

```
[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 .
```

- 6. Red Hat Enterprise Linux provides a package that includes a **genisoimage** command. Use the **yum** command to locate which RPM provides it and inspect it locally.

```
[student@servera ~]$ yum provides '*bin/genisoimage'
genisoimage-1.1.11-39.el8.x86_64 : Creates an image of an ISO9660 file-system
Repo       : rhel-8.4-for-x86_64-appstream-rpms
Matched from:
Other      : *bin/genisoimage
```

- 7. Use the `--downloadonly` option to download the package into your home directory, without installing it. If prompted, use **student** as the password.

```
[student@servera ~]$ sudo yum reinstall genisoimage --downloadonly  
--destdir /home/student/  
[sudo] password for student: student  
...output omitted...  
Downloading Packages:  
genisoimage-1.1.11-39.el8.x86_64.rpm           60 MB/s | 316 kB   00:00  
-----  
Total                                         31 MB/s | 316 kB   00:00  
Complete!  
The downloaded packages were saved in cache until the next successful transaction.  
You can remove cached packages by executing 'yum clean packages'.
```

- 8. Using RPM package management simplifies system administration. The library dependency would have been detected when the software was originally installed. Runtime libraries are identified and stored in the package metadata when a package is built.

```
[student@servera ~]$ rpm -q --requires -p genisoimage-* .rpm  
...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)  
librals.so.0()(64bit)  
libusal = 1.1.11-39.el8  
libusal.so.0()(64bit)  
libz.so.1()(64bit)  
...output omitted...
```

- 9. **Optional:** Remove the package that provides the needed libraries.

- 9.1. Remove the **libusal** package. When prompted, use **student** as the password.

```
[student@servera ~]$ sudo rpm -e $(rpm -qa | grep libusal) --nodeps  
[sudo] password for student: student
```

- 9.2. Attempt to execute the application. The program fails to run again. If a package provides the application, then use the **rpm** and **yum** commands to display the missing library dependencies.

```
[student@servera ~]$ genisoimage  
genisoimage: error while loading shared libraries: libusal.so.0: cannot open  
shared object file: No such file or directory
```

- **10.** Return to workstation as the student user.

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

Finish

On the workstation machine, use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish application-dependencies
```

This concludes the guided exercise.

Debugging Memory Leaks

Objectives

After completing this section, you should be able to identify applications that have memory leaks.

Memory Management in RHEL 8

System memory management is one of the most complex kernel tasks. Computer systems organize memory into fixed-size chunks called pages. The processor architecture defines the page size; for x86_64, the standard page size is 4 KiB. A system's physical RAM is divided into page frames; one page frame holds one page of data. The virtual address space size depends on the processor architecture, not on the amount of installed RAM. On a 64-bit x86-64 system, the addressable space is 2^{64} bytes or 16 EiB.

Processes do not access physical memory directly. Instead, each process has a virtual address space. When a process requests memory, the kernel maps the physical address of a page frame to a virtual address in the address space. The process views a private memory space and accesses only the physical page frames that the kernel mapped into its virtual space. This method enforces security restrictions and provides process isolation.

A single process generally does not use its entire addressable space. Much of the space remains unallocated and unmapped to any physical memory.

Diagnosing Memory Leaks

A memory leak is not a significant issue for a short-lived process, such as `ls` or `netstat`, because the kernel frees process memory when it exits. However, a process might not free all of its memory when exiting. Memory leaks can become severe when processes run for an extended time without opportunities to free memory.

Tools such as the `ps`, `top`, `free`, `sar -r`, and `sar -R` commands help to identify memory leaks. Dedicated tools, such as the `memcheck` tool from the `valgrind` framework, can identify application memory leaks.

Using valgrind to Diagnose Memory Leaks

The `valgrind` framework helps to detect memory errors such as uninitialized memory use and improper memory allocation or deallocation. The `valgrind` framework provides various debugging and profiling tools.

cachegrind

The `cachegrind` tool simulates application interaction with the system cache hierarchy and branch predictor. It gathers statistics for the duration of the application's execution and displays a summary as output to the console.

massif

The `massif` tool measures the heap space that a specified application uses. It measures both usable space and any further allocated space for bookkeeping and alignment purposes.

memcheck

The memcheck tool is the default tool in the valgrind framework. It detects and reports memory-related errors such as memory leaks, disallowed memory access that must not occur, use of an undefined or uninitialized value, incorrect release of heap memory, and pointer overlaps.

Profiling Application Memory Usage

A memory leak happens due to two different situations. In the first case, a program requests memory with a `malloc` system, but it does not use the requested memory. This requested memory causes the virtual size of the program to increase (`VIRT` in `top`). The `Committed_AS` line in `/proc/meminfo` increases but not actual physical memory. The resident size (`RSS` in `top`) stays the same.

In the second case, when a program uses the allocated memory, the resident size increases with the virtual size, and a physical memory shortage occurs. Although leaking virtual memory is not good, leaking resident memory impacts the system more.

Use the `memcheck` tool with the process that is leaking memory. Use it on processes that run for extended periods, such as daemons or desktop applications such as web browsers. Processes that leak memory show a gradual increase in memory or create out-of-memory errors that cause some processes to exit immediately. Some processes might generate a log event and continue to work. The `memcheck` tool reports these errors but cannot prevent them from happening. The `memcheck` tool logs an error message immediately before the error occurs.

The following example uses the `bigmem` application to help to diagnose memory leaks. The `bigmem` application was developed for use in Red Hat Training courses.

Install the `valgrind` package to use the available tools in the framework.

```
[root@host ~]# yum install valgrind
...output omitted...
Complete!
```

Run the `bigmem` application with the `memcheck` tool.

```
[root@host ~]# valgrind --tool=memcheck bigmem 128
==26545== Memcheck, a memory error detector
==26545== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==26545== Using Valgrind-3.16.0 and LibVEX; rerun with -h for copyright info
==26545== Command: bigmem 128
==26545==
Attempting to allocate 128 Mebibytes of resident memory...
Press <Enter> to exit Enter
==26545==
==26545== HEAP SUMMARY:
==26545==     in use at exit: 134,217,728 bytes in 128 blocks
==26545==    total heap usage: 130 allocs, 2 frees, 134,219,776 bytes allocated
==26545==
==26545== LEAK SUMMARY:
==26545==    definitely lost: 128,974,848 bytes in 123 blocks
==26545==    indirectly lost: 0 bytes in 0 blocks
==26545==    possibly lost: 5,242,880 bytes in 5 blocks
==26545==    still reachable: 0 bytes in 0 blocks
==26545==            suppressed: 0 bytes in 0 blocks
```

Chapter 8 | Troubleshooting Application Issues

```
==26545== Rerun with --leak-check=full to see details of leaked memory
==26545==
==26545== For lists of detected and suppressed errors, rerun with: -s
==26545== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

Use the `--leak-check=full` option to obtain detailed information to diagnose which function is leaking in the `bigmem` application.

```
[root@host ~]# valgrind --tool=memcheck --leak-check=full bigmem 128
==26546== Memcheck, a memory error detector
==26546== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==26546== Using Valgrind-3.16.0 and LibVEX; rerun with -h for copyright info
==26546== Command: bigmem 128
==26546==
Attempting to allocate 128 Mebibytes of resident memory...
Press <Enter> to exit Enter
==26546==
==26546== HEAP SUMMARY:
==26546==     in use at exit: 134,217,728 bytes in 128 blocks
==26546== total heap usage: 130 allocs, 2 frees, 134,219,776 bytes allocated
==26546==
==26546== 5,242,880 bytes in 5 blocks are possibly lost in loss record 1 of 2
==26546==     at 0x4C34F0B: malloc (vg_replace_malloc.c:307)
==26546==     by 0x400961: ??? (in /usr/local/bin/bigmem)
==26546==     by 0x4011BB: ??? (in /usr/local/bin/bigmem)
==26546==     by 0x4E65492: (below main) (in /usr/lib64/libc-2.28.so)
==26546==
==26546== 128,974,848 bytes in 123 blocks are definitely lost in loss record 2 of
2
==26546==     at 0x4C34F0B: malloc (vg_replace_malloc.c:307)
==26546==     by 0x400961: ??? (in /usr/local/bin/bigmem)
==26546==     by 0x4011BB: ??? (in /usr/local/bin/bigmem)
==26546==     by 0x4E65492: (below main) (in /usr/lib64/libc-2.28.so)
==26546==
==26546== LEAK SUMMARY:
==26546==     definitely lost: 128,974,848 bytes in 123 blocks
==26546==     indirectly lost: 0 bytes in 0 blocks
==26546==     possibly lost: 5,242,880 bytes in 5 blocks
==26546==     still reachable: 0 bytes in 0 blocks
==26546==             suppressed: 0 bytes in 0 blocks
==26546==
==26546== For lists of detected and suppressed errors, rerun with: -s
==26546== ERROR SUMMARY: 2 errors from 2 contexts (suppressed: 0 from 0)
```

Run the `bigmem` application with the `valgrind` default tool and redirect the output to the `bigmem.memchk` file.

```
[root@host ~]# valgrind -v --leak-check=full --show-reachable=yes
--log-file=bigmem.memchk bigmem 128
Attempting to allocate 128 Mebibytes of resident memory...
Press <Enter> to exit Enter
```

View the contents of the `bigmem.memchk` file to verify the leak information in the LEAK SUMMARY section.

```
[root@host ~]# cat bigmem.memchk
==26588== Memcheck, a memory error detector
==26588== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==26588== Using Valgrind-3.16.0-bf5e647edb-20200519X and LibVEX; rerun with -h for
==26588== copyright info
==26588== Command: bigmem 128
==26588== Parent PID: 1426

...output omitted...

==26588== HEAP SUMMARY:
==26588==     in use at exit: 134,217,728 bytes in 128 blocks
==26588==   total heap usage: 130 allocs, 2 frees, 134,219,776 bytes allocated
==26588==
==26588== Searching for pointers to 128 not-freed blocks
==26588== Checked 5,311,904 bytes
==26588==
==26588== 5,242,880 bytes in 5 blocks are possibly lost in loss record 1 of 2
==26588==   at 0x4C34F0B: malloc (vg_replace_malloc.c:307)
==26588==     by 0x400961: ??? (in /usr/local/bin/bigmem)
==26588==     by 0x4011BB: ??? (in /usr/local/bin/bigmem)
==26588==     by 0xE65492: (below main) (in /usr/lib64/libc-2.28.so)
==26588==
==26588== 128,974,848 bytes in 123 blocks are definitely lost in loss record 2 of
2
==26588==   at 0x4C34F0B: malloc (vg_replace_malloc.c:307)
==26588==     by 0x400961: ??? (in /usr/local/bin/bigmem)
==26588==     by 0x4011BB: ??? (in /usr/local/bin/bigmem)
==26588==     by 0xE65492: (below main) (in /usr/lib64/libc-2.28.so)
==26588==
==26588== LEAK SUMMARY:
==26588==   definitely lost: 128,974,848 bytes in 123 blocks
==26588==   indirectly lost: 0 bytes in 0 blocks
==26588==   possibly lost: 5,242,880 bytes in 5 blocks
==26588==   still reachable: 0 bytes in 0 blocks
==26588==       suppressed: 0 bytes in 0 blocks
==26588==
==26588== ERROR SUMMARY: 2 errors from 2 contexts (suppressed: 0 from 0)
```



References

`valgrind(1)` man pages

For further information, refer to *Understanding Memory Leaks - Using Valgrind (Memcheck)* at
<https://access.redhat.com/articles/17774>

► Guided Exercise

Debugging Memory Leaks

In this exercise, you identify memory leaks.

Outcomes

You should be able to use the `valgrind` framework to identify a virtual memory leak and a resident memory leak.

Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

```
[student@workstation ~]$ lab start application-memory
```

This command installs the `bigmem` application that causes a memory leak.

Instructions

The `bigmem` application in this exercise causes a memory leak. To verify the memory leak, run the `bigmem` application with the `memcheck` tool in the `valgrind` framework.

- 1. Log in to `servera` and switch to the root user.

```
[student@workstation ~]$ ssh student@servera  
...output omitted...  
[student@servera ~]$ sudo -i  
[sudo] password for student: student  
[root@servera ~]#
```

- 2. Install the `valgrind` package on the `servera` system.

```
[root@servera ~]# yum install valgrind  
...output omitted...  
Complete!
```

- 3. Use the `valgrind` tool to run the `bigmem` command, allocating 256 MiB of physical memory.

The `valgrind` summary indicates a memory leak.

```
[root@servera ~]# valgrind --tool=memcheck bigmem 256  
==26907== Memcheck, a memory error detector  
==26907== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.  
==26907== Using Valgrind-3.16.0 and LibVEX; rerun with -h for copyright info  
==26907== Command: bigmem 256  
==26907==
```

```
Attempting to allocate 256 Mebibytes of resident memory...
Press <Enter> to exit Enter
==26907==
==26907== HEAP SUMMARY:
==26907==     in use at exit: 268,435,456 bytes in 256 blocks
==26907==   total heap usage: 258 allocs, 2 frees, 268,437,504 bytes allocated
==26907==
==26907== LEAK SUMMARY:
==26907==   definitely lost: 263,192,576 bytes in 251 blocks
==26907==   indirectly lost: 0 bytes in 0 blocks
==26907==   possibly lost: 5,242,880 bytes in 5 blocks
==26907==   still reachable: 0 bytes in 0 blocks
==26907==   suppressed: 0 bytes in 0 blocks
==26907== Rerun with --leak-check=full to see details of leaked memory
==26907==
==26907== For lists of detected and suppressed errors, rerun with: -s
==26907== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

- 4. Use `valgrind` to run the `bigmem` application again, with 256 MiB of virtual memory.

The summary output matches the output for physical memory.

```
[root@servera ~]# valgrind --tool=memcheck bigmem -v 256
==26910== Memcheck, a memory error detector
==26910== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==26910== Using Valgrind-3.16.0 and LibVEX; rerun with -h for copyright info
==26910== Command: bigmem -v 256
==26910==
Attempting to allocate 256 MebiBytes of virtual memory...
Press <Enter> to exit Enter
==26910==
==26910== HEAP SUMMARY:
==26910==     in use at exit: 268,435,456 bytes in 256 blocks
==26910==   total heap usage: 258 allocs, 2 frees, 268,437,504 bytes allocated
==26910==
==26910== LEAK SUMMARY:
==26910==   definitely lost: 264,241,152 bytes in 252 blocks
==26910==   indirectly lost: 0 bytes in 0 blocks
==26910==   possibly lost: 4,194,304 bytes in 4 blocks
==26910==   still reachable: 0 bytes in 0 blocks
==26910==   suppressed: 0 bytes in 0 blocks
==26910== Rerun with --leak-check=full to see details of leaked memory
==26910==
==26910== For lists of detected and suppressed errors, rerun with: -s
==26910== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

- 5. The `valgrind` command detected the memory leak but provides insufficient information to know the type of memory leak from the `bigmem` application.

- 5.1. In a second terminal, log in to the `servera` system and switch to the `root` user.

```
[student@workstation ~]$ ssh student@servera  
...output omitted...  
[student@servera ~]$ sudo -i  
[sudo] password for student: student
```

- 5.2. Continuously display an overview of the system's allocated and committed memory.

```
[root@servera ~]# watch -d -n1 'free -m; grep -i commit /proc/meminfo'
```

- 5.3. In the first terminal window, run the **bigmem** application twice to allocate physical memory and virtual memory. This time, do not use the **valgrind** tool. Before you press **Enter** for both of the commands, navigate to the terminal running the **watch** command and verify the output.

```
[root@servera ~]# bigmem 256  
Attempting to allocate 256 Mebibytes of resident memory...  
Press <Enter> to exit  
[root@servera ~]# bigmem -v 256  
Attempting to allocate 256 Mebibytes of virtual memory...  
Press <Enter> to exit
```

- 5.4. View the system's allocated and committed memory in the second terminal.

The **watch** command output shows the **bigmem** application using the requested memory when run without the **-v** option. In this case, both committed memory (**Committed_As** field) and resident memory (**used** column) rise by the requested amount of memory.

When run with the **-v** option, the committed memory (**Committed_As** field) increases with the requested amount, but resident memory (**used** column) does not increase significantly. The resident memory (**used** column) increase slightly due to memory for executing itself and its page tables.

- 5.5. Use **Ctrl+C** to exit the **watch** command in the second terminal.

► 6. Exit the second terminal.

```
[root@servera ~]# exit  
[student@servera ~]$ exit  
[student@workstation ~]$ exit
```

Return to the **workstation** machine as the **student** user on first terminal.

```
[root@servera ~]# exit  
[student@servera ~]$ exit  
[student@workstation ~]$
```

Finish

On the **workstation** machine, use the **lab** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish application-memory
```

This concludes the guided exercise.

Debugging Application Execution

Objectives

After completing this section, you should be able to debug an application with standard RHEL tools.

Displaying Application System and Library Calls

Application error messages are not always helpful for diagnosing their issues. Occasionally, applications silently hang if they are stuck in a loop, or are waiting for an external event to occur. To troubleshoot such applications, an administrator must obtain more application state information.

The `strace` and `ltrace` commands display runtime information while a program executes. These utilities intercept application system calls and signals, and also process shared library calls with `ltrace`. These utilities send the trace information to the `STDERR` channel.

Displaying System Calls

The `strace` command, from the `strace` package, displays application-related system calls and signals. To use the `strace` command, provide an application as an argument.

In this example, the command displays the system calls from the `pwd` command.

```
[user@host ~]$ strace pwd
execve("/usr/bin/pwd", ["pwd"], 0x7ffcd1c467b0 /* 26 vars */) = 0 ①
brk(NULL) = 0x56403534d000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffef5fd8a40) = -1 EINVAL (Invalid argument)
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
    0x7f5f1b970000
access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory) ②
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=20755, ...}) = 0
mmap(NULL, 20755, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f5f1b96a000
...output omitted...
openat(AT_FDCWD, "/usr/lib/locale/locale-archive", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=217796128, ...}) = 0
mmap(NULL, 217796128, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f5f0e3cc000
close(3) = 0
getcwd("/home/user", 4096) = 14 ③
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x1), ...}) = 0
write(1, "/home/user\n", 14/home/user) = 14 ④
close(1) = 0
close(2) = 0
exit_group(0) = ?
exited with 0
```

- ① In the first line, `strace` executes the `pwd` command with the `execve` system call.

- ❷ The subsequent `access`, `openat`, `fstat`, `mmap`, and `close` system calls are the runtime linker to map the required shared libraries.
- ❸ The `getcwd` system call returns the current working directory.
- ❹ The `write` system call displays the value to STDOUT.

Each system call is displayed with its arguments and subsequent return values. When a system call fails, the numeric and symbolic value of the `errno` variable is displayed.

Alternatively, use the `-p PID` option to trace a current, running process via its process ID (PID).

```
[user@host ~]$ strace -p 1552
```

You might display only system call counts, instead of detailed function call information. Add the `-c` option to view how often a system call is executed, and the average time spent on it during program execution.

```
[user@host ~]$ strace -c pwd
/home/user
% time      seconds   usecs/call    calls    errors syscall
-----
0.00  0.000000       0          2        0      read
0.00  0.000000       0          1        0      write
0.00  0.000000       0          5        0      close
0.00  0.000000       0          4        0      fstat
0.00  0.000000       0          1        0      lseek
0.00  0.000000       0          7        0      mmap
0.00  0.000000       0          4        0      mprotect
0.00  0.000000       0          1        0      munmap
0.00  0.000000       0          4        0      brk
0.00  0.000000       0          1        1      access
0.00  0.000000       0          1        0      execve
0.00  0.000000       0          1        0      getcwd
0.00  0.000000       0          2        1      arch_prctl
0.00  0.000000       0          3        0      openat
-----
100.00 0.000000       0         37        2  total
```

When used with the `-f` option, `strace` also follows and profiles the child processes from the `fork()` system call. By default, child processes are not followed. The following `strace` command displays system calls and follows process forks.

```
[user@host ~]$ strace -f elinks -dump http://classroom.example.com
```

The `-o FILENAME` option saves output to a file, rather than displaying it on the terminal.

The `-e` option traces only specific system calls. For example, the following command traces only the `openat` and `fstat` system calls, and sends the output to the `/tmp/mytrace` file.

```
[user@host ~]$ strace -o /tmp/mytrace -e openat,fstat mycommand
```

Displaying Library Calls

The `ltrace` command, from the `ltrace` package, is similar to the `strace` command. The `ltrace` command also traces calls to shared libraries, rather than tracing system calls. Add the `-S` option to the `ltrace` command to also trace system calls.

In this example, the command displays the library calls from the `pwd` command.

```
[user@host ~]$ ltrace pwd
getenv("POSIXLY_CORRECT") = nil
strchr("pwd", '/') = nil
setlocale(LC_ALL, "") = "en_US.UTF-8"
textdomain("coreutils") = "coreutils"
...output omitted...
getcwd(nil, 0) = ""
puts("/home/user"/home/user) = 14
free(0x55af41767440) = <void>
_fpending(0x7f82f1e926e0, 0, 0x55af4038c9b0, 1) = 0
fileno(0x7f82f1e926e0) = 1
_freading(0x7f82f1e926e0, 0, 0x55af4038c9b0, 1) = 0
_freading(0x7f82f1e926e0, 0, 2052, 1) = 0
fflush(0x7f82f1e926e0) = 0
fclose(0x7f82f1e926e0) = 0
_fpending(0x7f82f1e92600, 0, 0x7f82f1e8d880, 2880) = 0
fileno(0x7f82f1e92600) = 2
_freading(0x7f82f1e92600, 0, 0x7f82f1e8d880, 2880) = 0
_freading(0x7f82f1e92600, 0, 4, 2880) = 0
fflush(0x7f82f1e92600) = 0
fclose(0x7f82f1e92600) = 0
_cxa_finalize(0x55af40591ae0, 0x55af40591ad8, 1, 2) = 0
exited (status 0)
```

The `ltrace` command traces only binaries that are compiled with the position-independent executables (PIE) option disabled. To check whether a binary is compiled with this option, run the `file` command. The `file` command returns "ELF 64-bit LSB executable" as output for binaries that are compiled with the PIE option disabled.

```
[user@host ~]$ file /usr/bin/pwd
/usr/bin/pwd: ELF 64-bit LSB pie executable ...output omitted...
```

Similar to the `strace` command, the `ltrace` command can trace a current, running process with the `-p PID` option.

```
[user@host ~]$ ltrace -p 1729
```

The `-o`, `-c`, and `-f` options of the `ltrace` command perform the same function as the corresponding `strace` command options.

Some binary programs are only executable, but not readable, by unprivileged users. The `ltrace` command requires the user to have read access to the executable file to trace calls.

```
[user@host ~]# ls -l /usr/bin/testprog
-rwx--x--x. 1 root root 57903 Feb 15 15:01 /usr/bin/testprog
[user@host ~]# ltrace testprog
Can't open /bin/testprog: Permission denied
```

The `strace` command can trace whether an executable file is executable but not readable, but the output from the command is limited.

In the following example, the `testprog` application tries to open and read a file. The file exists, but the program cannot read it. Because `strace` cannot read the executable file, it cannot display the name of the targeted file.

```
[root@host ~]# 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, 35)Unable to open configuration file.) = 35
exit_group(1)                           = ?
exited with 1
```

Auditing Program Execution

The Linux audit daemon, `auditd`, provides useful information to troubleshoot application runtime issues. The `auditd` daemon stores logs in the `/var/log/audit/audit.log` file, based on rules from the `auditctl` command. These rules might be triggered when a system call fails, or when an application executes.

The following commands demonstrate the use of `auditd` to troubleshoot an application that fails at runtime.

```
[root@host ~]# example
Error: configuration file not found
[root@host ~]# auditctl -a always,exit -F arch=b64 -S openat -F success=0
[root@host ~]# example
Error: configuration file not found
[root@host ~]# 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 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 initial error message in the output suggests that the command fails to open a file. The `auditctl` command creates a rule that audits `openat()` system calls that fail (`-F success=0`). With the new rule, the application failure generates an error message in the `audit.log` file. The entry indicates that the application cannot open the `/etc/example.conf` file.

The `auditctl -d` command removes rules that are no longer needed.

```
[root@host ~]# auditctl -d always,exit -F arch=b64 -S openat -F success=0
```



References

`auditctl(8)`, `auditd(8)`, `ausearch(8)`, `ltrace(1)`, and `strace(1)` man pages

► Guided Exercise

Debugging Application Execution

In this exercise, you troubleshoot an application with runtime problems.

Outcomes

You should be able to run `ltrace` and `strace` to diagnose runtime application issues.

Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

```
[student@workstation ~]$ lab start application-debugging
```

This command installs a custom application called `program`.

Instructions

A user reports that their application named `program` is not working. The application runs with the `program` command but displays an error that a configuration file is missing.

Troubleshoot the issue to discover which configuration file the `program` application requires.

- 1. Log in to `servera` and switch to the `root` user.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

- 2. Run the `program` command.

It displays an error message that the application cannot load its configuration file.

```
[root@servera ~]# program
Unable to load configuration file.
```

- 3. View the /var/log/messages file for recent errors.

No relevant errors are expected.

```
[root@servera ~]# tail -n 5 /var/log/messages
Oct 29 12:37:56 server NetworkManager[923]: <info>  [1635525476.9586] device
(eth1): state change: ip-config -> failed (reason 'ip-config-unavailable', sys-
iface-state: 'managed')
Oct 29 12:37:56 server NetworkManager[923]: <warn>  [1635525476.9595] device
(eth1): Activation: failed for connection 'Wired connection 2'
Oct 29 12:37:56 server NetworkManager[923]: <info>  [1635525476.9597] device
(eth1): state change: failed -> disconnected (reason 'none', sys-iface-state:
'managed')
Oct 29 12:37:56 server NetworkManager[923]: <info>  [1635525476.9637] dhcpc4
(eth1): canceled DHCP transaction
Oct 29 12:37:56 server NetworkManager[923]: <info>  [1635525476.9637] dhcpc4
(eth1): state changed timeout -> done
```

- 4. Run the strace command to display the system calls from the program application.

The program tries to open the /etc/program.conf file but fails because the file does not exist.

```
[root@servera ~]# strace program
...output omitted...
openat(AT_FDCWD, "/etc/program.conf", O_RDONLY) = -1 ENOENT (No such file or
directory)
dup(2)                                = 3
fcntl(3, F_GETFL)                      = 0x402 (flags O_RDWR|O_APPEND)
fstat(3, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}) = 0
write(3, "Unable to load configuration fil"..., 35)Unable to load configuration
file.) = 35
write(3, ": No such file or directory\n", 28: No such file or directory) = 28
close(3)                               = 0
exit_group(1)                          = ?
+++ exited with 1 +++
```

- 5. For comparison, run the ltrace command to display the library calls from the application.

The output displays similar information to the strace command in the form of C library calls.

```
[root@servera ~]# ltrace program
fopen("/etc/program.conf", "r")
= nil
puts("Unable to load configuration fil"...)Unable to load configuration file.
)
= 35
exit(1 <no return ...>
+++ exited (status 1) +++
```

- 6. Create the /etc/program.conf file.

```
[root@servera ~]# touch /etc/program.conf
```

- 7. Run the program command again.

The command runs successfully and returns a zero exit status.

```
[root@servera ~]# program  
Program successfully executed.  
[root@servera ~]# echo $?  
0
```

- 8. View the output of the strace and ltrace commands when the /etc/program.conf file is present but the read permissions are removed from the program executable.

- 8.1. Change the permissions of the program executable so that unprivileged users cannot read it.

```
[root@servera ~]# chmod 711 /usr/bin/program  
[root@servera ~]# ls -l /usr/bin/program  
-rwx--x--x. 1 root root 17536 Nov 1 10:57 /usr/bin/program
```

- 8.2. Exit the root shell and run the strace program command as the student user.

The strace command indicates that the application uses the openat system call, but does not print the name of the file.

```
[root@servera ~]# exit  
logout  
[student@servera ~]$ strace program  
...output omitted...  
openat(AT_FDCWD, 0x40006fa, O_RDONLY) = 3  
fstat(1, 0x7ffe2d759340) = 0  
write(1, 0x1816490, 31) = 31  
Program successfully executed.  
) = 31  
exit_group(0) = ?  
+++ exited with 0 +++
```

- 8.3. Run the ltrace program command. A "permission denied" error is expected, because the ltrace command requires read access to the executable file.

```
[student@servera ~]$ ltrace program  
Can't open /usr/bin/program: Permission denied
```

- 9. Return to workstation as the student user.

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

Finish

On the `workstation` machine, use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish application-debugging
```

This concludes the guided exercise.

Troubleshooting Containerized Applications

Objectives

After completing this section, you should be able to troubleshoot a containerized application or service.

Overview of Containerized Applications

Software applications generally depend on other libraries, configuration files, or services that the runtime environment provides. The traditional runtime environment for a software application is a physical host or virtual machine, and application dependencies get installed as part of the host.

A container is a set of one or more processes that are isolated from the rest of the system. This figure describes the difference between applications that run as containers and applications that run on the host operating system.

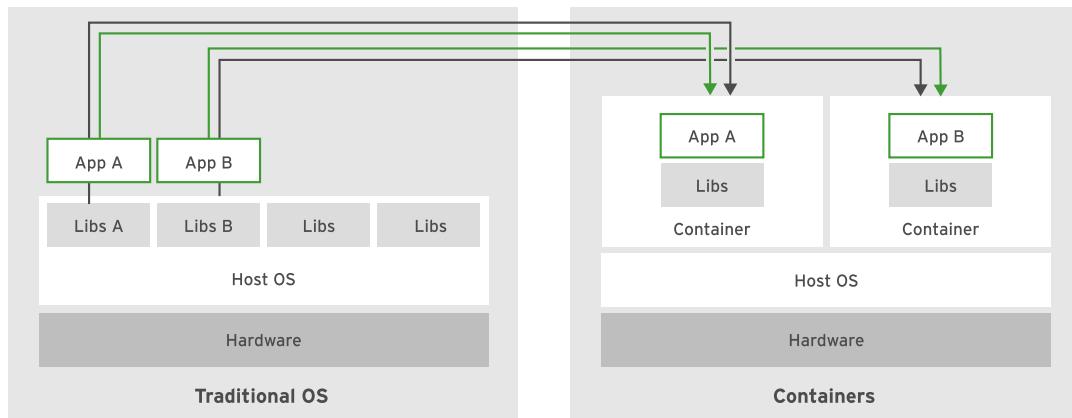


Figure 8.1: Container versus operating system differences

The isolation process takes advantage of certain features of the Linux kernel:

Namespaces

A namespace isolates specific system resources that are usually visible to all processes. Inside a namespace, only processes that are members of that namespace can see those resources. Namespaces can include resources such as network interfaces, the process ID list, mount points, IPC resources, and the system's host name information.

Control groups (cgroups)

Control groups partition sets of processes and their children into groups to manage and limit the resources that they consume. Control groups place restrictions on the amount of system resources that processes might use. Those restrictions keep one process from using too many resources on the host.

SELinux

Security-Enhanced Linux (SELinux) is a mandatory access control system for processes. Linux kernel uses SELinux to protect processes from each other and to protect the host system from its running processes. Processes run as a confined SELinux type that has limited access to host system resources.

From the Linux kernel perspective, a container is a process with restrictions. Instead of running a single binary file, a container runs an *image*. An image is a file-system bundle that contains all the required dependencies to execute a process, including files in the file system, installed packages, available resources, running processes, and kernel modules.

Running containers use an immutable view of the image, so that multiple containers can reuse the same image simultaneously.

Container images must be locally available for the container runtime to execute them, but the images are usually stored and maintained in an image repository. An image repository is a public or private service to store, search, and retrieve the images. Image repositories provide remote access, image metadata, image authorization, and image version control.

Using Podman to Manage Containers

Podman is an open source tool for managing containers and container images and interacting with image registries. It offers the following key features:

- It uses the specified image format of the **Open Container Initiative** (OCI) at <https://opencontainers.org/>. Those specifications define a standard, community-driven, non-proprietary image format.
- Podman stores local images in a local file system. Doing so avoids unnecessary client/server architecture or running daemons on a local machine.
- Podman follows the same command patterns as the Docker CLI.

The `podman` command provides subcommands to create, manage, and update containers, and to get information about both running and stopped containers for use in container debugging.

Reviewing Registry Authentication

Red Hat distributes container images primarily from two locations:

`registry.access.redhat.com`, with no authentication required, and `registry.redhat.io`, which requires an account for authentication. Although both registries contain essentially the same container images, images that require a subscription are available only from `registry.redhat.io`.

To retrieve content from an authenticated registry, log in to the registry with a Customer Portal, Red Hat Developer, or Registry Service Account credentials. Use `podman login` to authenticate to `registry.redhat.io`.

```
[root@host ~]# podman login registry.redhat.io
Username: redhat-account-username
Password: *****
Login Succeeded!
```

To test whether you have functional basic authentication, query the API with the `curl` command. A successful authentication attempt results in HTTP 200 OK.

```
[root@host ~]# curl -Lv -u redhat-account-username:***** "https://
sso.redhat.com/auth/realms/rhcc/protocol/redhat-docker-v2/auth?service=docker-
registry&client_id=curl&scope=repository:rhel:pull"
* Trying 104.75.112.205...
* TCP_NODELAY set
* Connected to sso.redhat.com (104.75.112.205) port 443 (#0)
...output omitted...
```

```
* SSL certificate verify ok.  
* Server auth using Basic with user 'redhat-account-username'  
> GET /auth/realms/rhcc/protocol/redhat-docker-v2/auth?service=docker-  
registry&client_id=curl&scope=repository:rhel:pull HTTP/1.1  
> Host: sso.redhat.com  
> Authorization: Basic cmhuLXN1cHBvcnQtYWNhbGxlamE6TXVsDF2NGMh  
> User-Agent: curl/7.61.1  
> Accept: */*  
>  
< HTTP/1.1 200 OK  
...output omitted...
```

To consume container images from `registry.redhat.io` in shared environments, it is recommended to create and use a Registry Service Account, also referred to as an *authentication token*, instead of using an organization's or an individual's Customer Portal credentials.



Note

To create an authentication token, refer to the **Creating Registry Service Accounts** section of the *Red Hat Container Registry Authentication* article at <https://access.redhat.com/RegistryAuthentication>.

To test basic authentication with an authentication token, store the token as a system variable and reference it with a `curl` command. A successful authentication attempt results in `HTTP 200 OK` and a JSON object.

```
[root@host ~]# export TOKEN="account_number|name"  
[root@host ~]# export SECRET="token_value"  
[root@host ~]# curl -u $TOKENID:$SECRET "https://sso.redhat.com/  
auth/realms/rhcc/protocol/redhat-docker-v2/auth?service=docker-  
registry&client_id=curl&scope=repository:rhel:pull"  
{"token": "_tokenvalue_","access_token": "_tokenvalue_","  
"expires_in":300,"issued_at": "2021-10-19T04:18:16Z"}
```

When the basic authentication test is successful, use the access token to test connectivity with the registry itself.

```
[root@host ~]# curl -Lv -u $TOKENID:$SECRET "https://sso.redhat.com/  
auth/realms/rhcc/protocol/redhat-docker-v2/auth?service=docker-  
registry&client_id=curl&scope=repository:rhel:pull" | python3 -m json.tool  
% Total    % Received % Xferd  Average Speed   Time     Time     Time  Current  
          Dload  Upload Total   Spent    Left  Speed  
0       0      0      0      0      0      0      0 --:--:-- --:--:-- --:--:-- 0  
* Trying 104.75.112.205...  
* TCP_NODELAY set  
* Connected to sso.redhat.com (104.75.112.205) port 443 (#0)  
...output omitted...  
< HTTP/1.1 200 OK  
...output omitted...  
100 2168 100 2168 0 0 3269 0 --:--:-- --:--:-- --:--:-- 3269  
* Connection #0 to host sso.redhat.com left intact  
{  
  "token": "eyJhbGciOiJSUzI1NiJ9...",
```

```
"access_token": "eyJhbGciOiJSUzI1NiJ9...",
"expires_in": 300,
"issued_at": "2021-10-19T04:39:52Z"
}
```

Managing Persistent Containers

Podman can mount a host directory inside a running container, as in a standard *volume mount*. In this way, the containerized application uses the host directories as the container storage, like a normal application's use of network shared storage. When the container exits and is removed, the container's directory is unmounted, and the source directory and contents remain on the host, allowing new containers to mount the directory and access the contained data.

Directory configuration involves:

- Configuring the ownership and permissions of the directory.
- Setting the appropriate SELinux context.

Podman uses the SELinux `container_file_t` context type to control which files on the host system a container is allowed to access.

To mount a host directory to a container, add the `--volume` (or `-v`) option, and specify the host directory path and the container storage path, separated by a colon.

```
[root@host ~]# podman run -d --name mydb -v /home/user/dbfiles:/var/lib/mysql:z
-e MYSQL_USER=user -e MYSQL_PASSWORD=redhat -e MYSQL_DATABASE=inventory
registry.redhat.io/rhel8/mariadb-103:1-102
```

Monitoring Container Health

Podman subcommands manage a container environment, determining the container health, viewing system information, and monitoring Podman events.

The `healthcheck` subcommand verifies the health or readiness of the process that runs inside the container. A health check consists of the following components, of which only the first one is required and the others are optional scheduling parameters:

Command

Executes a command inside the target container and waits for the exit code to determine the container health.

Retries

Defines the number of consecutive failed health checks that must occur before the container is marked as *unhealthy*. A successful health check resets the retry counter.

Interval

Specifies the time interval between each run of the health check command. With small intervals, the system spends much time with running health checks. With large intervals, it is difficult to catch timeouts.

Start-period

Describes the elapsed time between the container starting and when it ignores health check failures.

Timeout

Specifies the time interval in which the check must complete before it is considered as failed.

To define a container health check, add the command that confirms the health of the container to run inside the container along with the other processes in the image.

```
[user@host ~]$ podman run --name test-httdp --health-cmd='curl http://localhost:8080 || exit 1' --health-interval=0 -dt -p 8080:8080 registry.redhat.io/rhel8/httpd-24  
8a60a90f0149ee6229164f539995f18702d6fbaf34e5cf6de47212b5bfbc20fa
```

Verify the health of the running container with the `podman healthcheck run container_name` command, which invokes the health command that you added to the container.

```
[user@host ~]$ podman healthcheck run test-httdp  
healthy
```

Podman subcommands can monitor and display the events that occur in a container environment. Each event includes a timestamp, a type, a status, and when applicable, a name and an image.

The event logging mechanism is configured in the `/usr/share/containers/containers.conf` file. The default mechanism is `file`, and can be changed to `journald` to send events to the system journal. Using a value of `none` disables Podman event reporting.

To view all events, verify `journald` as the logging mechanism and run the `podman events` command as a privileged user.

```
[root@host ~]# grep logger /usr/share/containers/containers.conf  
events_logger = "journald"  
[root@host ~]# podman events  
Oct 19 12:45:24 host.example.com podman[3415]: 2021-10-19  
12:45:24.108408331 -0400 EDT m=+0.063967269 container create  
9af5be974dbf4762850a0c799cba85c5f0b9940e0c8a7c0e04bce5604796d8ef  
(image=registry.redhat.io/rhel8/httpd-24:latest, name=test-httdp,  
...output omitted...  
Oct 19 12:45:24 host.example.com podman[3415]: 2021-10-19 12:45:24.080958781 -0400  
EDT m=+0.036517732 image pull registry.redhat.io/rhel8/httpd-24  
Oct 19 12:45:24 host.example.com podman[3415]: 2021-10-19  
12:45:24.251704702 -0400 EDT m=+0.207263649 container init  
9af5be974dbf4762850a0c799cba85c5f0b9940e0c8a7c0e04bce5604796d8ef  
(image=registry.redhat.io/rhel8/httpd-24:latest, name=test-httdp,  
...output omitted...
```

This example adds a filter to show only `create` events that occurred in the last five minutes.

```
[root@host ~]# podman events --since 5m --filter event=create  
2021-10-19 12:45:24.108408331 -0400 EDT container create  
9af5be974dbf4762850a0c799cba85c5f0b9940e0c8a7c0e04bce5604796d8ef  
(image=registry.redhat.io/rhel8/httpd-24:latest,  
...output omitted...
```

**Note**

The event output length varies depending on the type of event and the container image's configuration.

Enabling Rootless Containers

A *rootless container* runs as a non-root user, but runs processes and interacts with files as a privileged user with the concept of a *user namespace*. A user namespace is a Linux facility where a non-root user can map a range of UIDs and GIDs on the host system to a different range of UIDs and GIDs within the namespace.

With rootless containers, the running user must have the mapped ID ranges that are listed in the /etc/subuid and /etc/subgid files.

```
[root@host ~]# cat /etc/subuid
user:100000:65536
[root@host ~]# cat /etc/subgid
user:100000:65536
```

The `unshare` subcommand creates a namespace to run a specified program, instead of sharing the current namespace. With `podman unshare`, you are leaving your normal namespace, and working in a new Podman user namespace.

View the mappings by displaying the /proc/self/uid_map and /proc/self/gid_map files from inside the user namespace, by using the `podman unshare` command as the non-root user.

```
[user@host ~]$ podman unshare cat /proc/self/uid_map
0          3267          1
1          100000        65536
[user@host ~]$ podman unshare cat /proc/self/gid_map
0          3267          1
1          100000        65536
```

Using a systemd Unit to Manage a Containerized Service

Use the `podman generate systemd` command to create a `systemd` unit file for a specified service container. The `systemd` unit automatically starts, stops, enables, or disables a containerized service on the host system.

```
[user@host ~]$ podman generate systemd --name web --files --new
/home/user/container-web.service
```

Move the unit file to the user's `systemd` configuration directory.

```
[user@host ~]$ mv container-web.service /home/user/.config/systemd/user/
```

Inspect the unit file to verify the service's `systemd` structure and parameter settings.

```
[user@host ~]$ cat /home/user/.config/systemd/user/container-web.service
# container-web.service
# autogenerated by Podman 3.3.1
# Wed Nov 17 20:41:44 CEST 2021

[Unit]
Description=Podman container-web.service
```

```
Documentation=man:podman-generate-systemd(1)
Wants=network-online.target
After=network-online.target
RequiresMountsFor=%t/containers

[Service]
Environment=PODMAN_SYSTEMD_UNIT=%n
Restart=on-failure
TimeoutStopSec=70
ExecStartPre=/bin/rm -f %t/%n.ctr-id
ExecStart=/usr/bin/podman run --cidfile=%t/%n.ctr-id --sdnotify=common
--cgroups=no-common --rm -d --replace --name web -p 8080:8080
registry.access.redhat.com/ubi8/httpd-24
ExecStop=/usr/bin/podman stop --ignore --cidfile=%t/%n.ctr-id
ExecStopPost=/usr/bin/podman rm -f --ignore --cidfile=%t/%n.ctr-id
Type=notify
NotifyAccess=all

[Install]
WantedBy=multi-user.target default.target
```

To manage the containerized service, use the `systemctl` command.

```
[user@host ~]$ systemctl --user [start, stop, status, enable,
 disable] container_name
```

After modifying the unit file to change the service configuration, implement the change by requesting systemd to reload its configuration.

```
[user@host ~]$ systemctl daemon-reload
```

Inspecting Container Logs

Podman can inspect logs in running containers to assist with troubleshooting. Podman ignores custom application-specific logs and assumes that containerized applications send all log output to STDOUT.

A container is a process tree from the perspective of the host operating system. When Podman starts a container, it redirects the container's STDOUT and STDERR, saving them to disk in the container's ephemeral storage. The container's log files can be displayed with the `podman logs` command. Until deleting the container, the container logs remain, because the container instance resides in memory and its ephemeral storage still exists. Deleting a container also deletes its ephemeral storage, making the instance logs unavailable.

```
[user@host ~]$ podman logs test-httdp
[Tue Oct 19 16:45:24.626604 2021] [:notice] [pid 1:tid 140604915785152]
ModSecurity: Status engine is currently disabled, enable it by set
SecStatusEngine to On.
AH00558: httpd: Could not reliably determine the server's fully qualified domain
name, using 10.0.2.100. Set the 'ServerName' directive globally to suppress this
message
```

Chapter 8 | Troubleshooting Application Issues

```
[Tue Oct 19 16:45:24.709771 2021] [ssl:warn] [pid 1:tid 140604915785152] AH01909:  
10.0.2.100:8443:0 server certificate does NOT include an ID which matches the  
server name  
  
[Tue Oct 19 16:45:24.715306 2021] [core:notice] [pid 1:tid 140604915785152]  
AH00094: Command line: 'httpd -D FOREGROUND'  
10.0.2.100 - - [19/Oct/2021:18:22:18 +0000] "GET / HTTP/1.1" 403 4481 "-"  
"curl/7.61.1"  
[Tue Oct 19 18:22:24.120915 2021] [autoindex:error] [pid 59:tid 140603983611648]  
[client 127.0.0.1:50552] AH01276: Cannot serve directory /var/www/html/: No  
matching DirectoryIndex (index.html) found, and server-generated directory index  
forbidden by Options directive  
...output omitted...
```

Podman writes all output to STDOUT, and by default uses an `error` log level. Podman supports the `debug`, `info`, `warn`, `error`, `fatal`, and `panic` standard logging levels.

```
[user@host ~]$ podman logs --log-level info test-httdp  
INFO[0000] podman filtering at log level info  
INFO[0000] Setting parallel job count to 7  
=> sourcing 10-set-mpm.sh ...  
=> sourcing 20-copy-config.sh ...  
=> sourcing 40-ssl-certs.sh ...  
---> Generating SSL key pair for httpd...  
AH00558: httpd: Could not reliably determine the server's fully qualified domain  
name, using 10.0.2.100. Set the 'ServerName' directive globally to suppress this  
message  
[Tue Oct 19 16:45:24.625273 2021] [ssl:warn] [pid 1:tid 140604915785152] AH01909:  
10.0.2.100:8443:0 server certificate does NOT include an ID which matches the  
server name  
[Tue Oct 19 16:45:24.626589 2021] [:notice] [pid 1:tid 140604915785152]+  
ModSecurity for Apache/2.9.2 ( http://www.modsecurity.org/ ) configured.  
...output omitted...
```



References

Podman

<https://docs.podman.io/>

Use of Podman in a Rootless Environment

https://github.com/containers/podman/blob/master/docs/tutorials/rootless_tutorial.md

For further information, refer to the **Red Hat Enterprise Linux 8 Building, running, and managing containers Guide** at

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/building_running_and_managing_containers/index

► Guided Exercise

Troubleshooting Containerized Applications

In this exercise, you configure container health checks, and inspect the container logs. You also configure a container host to record the container events.

Outcomes

You should be able to configure the health check option on a container and inspect the container logs for the source of an unexpected behavior. You should also be able to configure the container host to record container events.

Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

```
[student@workstation ~]$ lab start application-container
```

This command confirms that the required hosts for this exercise are accessible and creates the containerized service.

Instructions

A new containerized website is not displaying the service test page. You are requested to resolve the issue. Additionally, you install mechanisms that monitor the behavior of the container.

- 1. Log in to `servera` as the `student` user.

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

- 2. Verify the container and its status.

```
[student@servera ~]$ podman ps -a  
CONTAINER ID IMAGE COMMAND  
CREATED STATUS PORTS NAMES  
97ae356eb822 registry.access.redhat.com/ubi8/httpd-24:latest /usr/bin/run-  
http... 52 minutes ago Up 52 minutes ago 0.0.0.0:40195->8080/tcp website
```

- 3. Verify that container events are logged.

- 3.1. Discover the container event logger mechanism.

```
[student@servera ~]$ grep logger /usr/share/containers/containers.conf  
events_logger = "file"
```

3.2. Change the logger mechanism to journald.

```
[student@servera ~]$ sudo vim /usr/share/containers/containers.conf
[sudo] password for student: student
...output omitted...
events_logger = "journald"
```

3.3. Restart the container and verify that container events are logged to the system journal.

```
[student@servera ~]$ podman stop website
...output omitted...
[student@servera ~]$ podman start website
website
[student@servera ~]$ podman events --since 5m --until 0m
2021-10-21 18:33:52.624255505 -0400 EDT container cleanup
97ae356eb8227edce0041af12f26d87371cddf54ae658bf9f2d9639dd4bda673
(image=registry.access.redhat.com/ubi8/httpd-24:latest, name=website,
...output omitted...
2021-10-21 18:33:56.054608729 -0400 EDT container init
97ae356eb8227edce0041af12f26d87371cddf54ae658bf9f2d9639dd4bda673
(image=registry.access.redhat.com/ubi8/httpd-24:latest, name=website,
...output omitted...
```

3.4. Filter the events by container name. Format the output as JSON to read the events more easily.

```
[student@servera ~]$ podman events --since 5m --filter container=website --format
json --until 0m
{"ID":"97ae356eb8227edce0041af12f26d87371cddf54ae658bf9f2d9639dd4bda673", "Image":
"registry.access.redhat.com/ubi8/httpd-24:latest", "Name":"website", "Status":
"died", "Time":"2021-10-21T18:33:52.576507917-04:00", "Type":"container",
"Attributes":null}
{"ID":"97ae356eb8227edce0041af12f26d87371cddf54ae658bf9f2d9639dd4bda673", "Image":
"registry.access.redhat.com/ubi8/httpd-24:latest", "Name":"website", "Status":
"cleanup", "Time":"2021-10-21T18:33:52.624255505-04:00", "Type":"container",
...output omitted...
```

► 4. View information about the containerized service.

4.1. Verify that the containerized service process is running.

```
[student@servera ~]$ ps auxf | grep httpd
101000 4827 0.0 1.1 391232 21800 pts/0 Ss+ 18:33 0:00 \_ httpd -D FOREGROUND
101000 4868 0.0 0.2 369144 5500 pts/0 S+ 18:33 0:00 \_ httpd -D FOREGROUND
101000 4869 0.0 1.0 2038556 20336 pts/0 Sl+ 18:33 0:00 \_ httpd -D FOREGROUND
101000 4871 0.0 1.2 1907420 22432 pts/0 Sl+ 18:33 0:00 \_ httpd -D FOREGROUND
101000 4873 0.0 0.9 1907420 18336 pts/0 Sl+ 18:33 0:00 \_ httpd -D FOREGROUND
```

4.2. Verify the rootless container configuration.

```
[student@servera ~]$ cat /etc/subuid
student:100000:65536
devops:165536:65536
[student@servera ~]$ cat /etc/subgid
student:100000:65536
devops:165536:65536
[student@servera ~]$ podman unshare cat /proc/self/uid_map
      0          1000          1
      1        100000      65536
[student@servera ~]$ podman unshare cat /proc/self/gid_map
      0          1000          1
      1        100000      65536
```

► 5. Manage the containerized service.

5.1. View the running container's attributes.

```
[student@servera ~]$ podman inspect website
...output omitted...
"CreateCommand": [
    "podman",
    "run",
    "--name",
    "website",
    "-dt",
    "-p",
    "8080",
    "registry.access.redhat.com/ubi8/httpd-24"
],
...output omitted...
```

5.2. Attempt to verify that the container is showing the test page.

The connection fails.

```
[student@servera ~]$ curl localhost:8080
curl: (7) Failed to connect to localhost port 8080: Connection refused
```

5.3. Verify the ports that the container image exposes.

```
[student@servera ~]$ podman inspect -f "{{.Config.ExposedPorts}}"
registry.access.redhat.com/ubi8/httpd-24
map[8080/tcp:{} 8443/tcp:{}]
```

5.4. List the ports that the containerized service exposes.

```
[student@servera ~]$ podman ps -a --format "{{.Names}} {{.Ports}}"
website 0.0.0.0:40195->8080/tcp
```

5.5. Stop the container. Run the container again with port forwarding of container port 8080 to host port 8080.

```
[student@servera ~]$ podman stop website
...output omitted...
[student@servera ~]$ podman rm website
...output omitted...
[student@servera ~]$ podman run --name website -dt -p 8080:8080
registry.access.redhat.com/ubi8/httpd-24
...output omitted...
```

- 5.6. Attempt again to verify that the container is showing the test page.

The connection succeeds. Ignore any "403 Forbidden" errors, the purpose is to confirm a response appears.

```
[student@servera ~]$ curl -Iv localhost:8080
* Rebuilt URL to: localhost:8080/
*   Trying ::1...
* TCP_NODELAY set
* Connected to localhost (::1) port 8080 (#0)
> HEAD / HTTP/1.1
> Host: localhost:8080
> User-Agent: curl/7.61.1
> Accept: /
...output omitted...
```

- ▶ 6. Add a `healthcheck` option to the container to monitor the containerized service.

- 6.1. Stop the container. Run the container with a `curl` command as the health check to validate the container's health.

```
[student@servera ~]$ podman stop website
...output omitted...
[student@servera ~]$ podman rm website
...output omitted...
[student@servera ~]$ podman run --name website --health-cmd='curl
http://localhost:8080 || exit 1' --health-interval=0 -dt -p 8080:8080
registry.access.redhat.com/ubi8/httpd-24
...output omitted...
```

- 6.2. Use the `podman healthcheck run` command to verify the health of the container.

```
[student@servera ~]$ podman healthcheck run website
healthy
```

- ▶ 7. View the container logs with the `podman logs` command.

```
[student@servera ~]$ podman logs website
...output omitted...
AH00558: httpd: Could not reliably determine the server's fully qualified domain
name, using 10.0.2.100. Set the 'ServerName' directive globally to suppress this
message
[Fri Oct 22 00:08:27.724749 2021] [mpm_event:notice] [pid 1:tid 140441417752000]
AH00489: Apache/2.4.37 (Red Hat Enterprise Linux) OpenSSL/1.1.1g configured --
resuming normal operations
[Fri Oct 22 00:08:27.724772 2021] [core:notice] [pid 1:tid 140441417752000]
AH00094: Command line: 'httpd -D FOREGROUND'
[Fri Oct 22 00:12:00.474978 2021] [autoindex:error] [pid 44:tid 140440489633536]
[client 127.0.0.1:41236] AH01276: Cannot serve directory /var/www/html/: No
matching DirectoryIndex (index.html) found, and server-generated directory index
forbidden by Options directive
127.0.0.1 - - [22/Oct/2021:00:12:00 +0000] "GET / HTTP/1.1" 403 4481 "-"
"curl/7.61.1"
```

- 8. Return to workstation as the student user.

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

Finish

On the `workstation` machine, use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish application-container
```

This concludes the guided exercise.

► Lab

Troubleshooting Application Issues

In this lab, you debug a third-party containerized application.

Outcomes

You should be able to diagnose and troubleshoot library dependency and application runtime issues.

Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

```
[student@workstation ~]$ lab start application-review
```

This command confirms that the required hosts for this exercise are accessible and deploys the containerized application.

Instructions

Another system administrator deployed the `showview` application, but it is not working. The application uses a web service with a containerized database service. Resolve the issue and verify that the containerized service is working.

1. Verify the newly deployed application and observe its behavior.
2. View the dynamic libraries of the web service for missing libraries. Install the packages that provide the shared library requirements.
3. Resolve the issues that prevent the application from obtaining the database information.
4. Verify that the application runs correctly and displays data for the views and visits.

Evaluation

On the `workstation` machine, use the `lab` command to grade your work. Correct any reported failures and rerun the script until you receive a passing grade.

```
[student@workstation ~]$ lab grade application-review
```

Finish

On the `workstation` machine, use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish application-review
```

This concludes the lab.

► Solution

Troubleshooting Application Issues

In this lab, you debug a third-party containerized application.

Outcomes

You should be able to diagnose and troubleshoot library dependency and application runtime issues.

Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

```
[student@workstation ~]$ lab start application-review
```

This command confirms that the required hosts for this exercise are accessible and deploys the containerized application.

Instructions

Another system administrator deployed the `showview` application, but it is not working. The application uses a web service with a containerized database service. Resolve the issue and verify that the containerized service is working.

1. Verify the newly deployed application and observe its behavior.

- 1.1. Log in to `servera` and switch to the `root` user.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

- 1.2. View the web service status.

The service status shows an error in a missing library dependency, `libaprutil-1.so.0`.

```
[root@servera ~]# systemctl status httpd
● httpd.service - The Apache HTTP Server
  Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; vendor preset: disabled)
  Drop-In: /usr/lib/systemd/system/httpd.service.d
            └─php-fpm.conf
  Active: failed (Result: exit-code) since Thu 2021-11-04 20:51:43 EDT; 43s ago
    Docs: man:httpd.service(8)
   Process: 7455 ExecStart=/usr/sbin/httpd $OPTIONS -DFOREGROUND (code=exited,
              status=127)
```

```
Main PID: 7455 (code=exited, status=127)

Nov 04 20:51:43 servera.lab.example.com systemd[1]: Starting The Apache HTTP
Server...
Nov 04 20:51:43 servera.lab.example.com httpd[7455]: /usr/sbin/httpd: error while
loading shared libraries: libaprutil-1.so.0: cannot open shared object file: No
such file>
Nov 04 20:51:43 servera.lab.example.com systemd[1]: httpd.service: Main process
exited, code=exited, status=127/n/a
Nov 04 20:51:43 servera.lab.example.com systemd[1]: httpd.service: Failed with
result 'exit-code'.
Nov 04 20:51:43 servera.lab.example.com systemd[1]: Failed to start The Apache
HTTP Server.
```

2. View the dynamic libraries of the web service for missing libraries. Install the packages that provide the shared library requirements.

- 2.1. View the shared libraries for the `httpd` binary.

All of the shared libraries resolved except for one: `libaprutil-1.so.0`.

```
[root@servera ~]# which httpd
/usr/sbin/httpd
[root@servera ~]# ldd /usr/sbin/httpd
libpcre.so.1 => /lib64/libpcre.so.1 (0x00007f3b9539a000)
libselinux.so.1 => /lib64/libselinux.so.1 (0x00007f3b95170000)
libsystemd.so.0 => /lib64/libsystemd.so.0 (0x00007f3b94e29000)
libaprutil-1.so.0 => not found
...output omitted...
```

- 2.2. Locate the packages to install that provide the library.

```
[root@servera ~]# yum provides libaprutil-1.so.0
apr-util-1.6.1-6.el8.i686 : Apache Portable Runtime Utility library
Repo        : rhel-8.4-for-x86_64-appstream-rpms
Matched from:
Provide    : libaprutil-1.so.0
```

- 2.3. Install the package that provides the `libaprutil-1.so.0` library.

```
[root@servera ~]# yum install apr-util
...output omitted...
```

- 2.4. Restart the web service and verify the service status.

```
[root@servera ~]# systemctl restart httpd
[root@servera ~]# systemctl status httpd
● httpd.service - The Apache HTTP Server
  Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; vendor preset:
  disabled)
  Drop-In: /usr/lib/systemd/system/httpd.service.d
            └─php-fpm.conf
  Active: active (running) since Thu 2021-11-04 21:03:16 EDT; 3s ago
```

```
Docs: man:httdp.service(8)
Main PID: 7582 (httdp)
Status: "Started, listening on: port 80"
...output omitted...
```

- 2.5. Query the web service with the `localhost` default.

The application exposes the `showviews.php` URL.

```
[root@servera ~]# curl localhost
<meta http-equiv="refresh" content="0; URL=showviews.php">
```

- 2.6. Query the web service at the `showviews.php` URL.

The service displays a page about views and visits, which verifies that the service is functioning.

However, the application is not obtaining data from the database. You must resolve the database connection issues.

```
[root@servera ~]# curl localhost/showviews.php
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<body>






```

3. Resolve the issues that prevent the application from obtaining the database information.

- 3.1. View the `showviews.php` file and verify the connection to the database.

```
[root@servera ~]# cat /var/www/html/showviews.php
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<body>
<?php

$hostname = "10.89.0.7";
$username = "site_monitor";
$password = "reviews";
$db = "sitemonitor";

$dbconnect=mysqli_connect($hostname,$username,$password,$db);
...output omitted...
```

- 3.2. View the status of the containerized database.

```
[root@servera ~]# podman ps -a
CONTAINER ID IMAGE COMMAND
CREATED STATUS PORTS NAMES
da31cd1291c1 registry.access.redhat.com/rhscl/mariadb-101-rhel7:latest run-
mysqld 7 minutes ago Up 7 minutes ago mariadb-service
```

3.3. Verify the IP address of the database container.

```
[root@servera ~]# podman inspect mariadb-service | jq '.[].NetworkSettings.Networks' | grep IPAddress
"IPAddress": "10.89.0.2",
```

3.4. Modify the showviews.php file to use the correct IP address.

```
[root@servera ~]# cat /var/www/html/showviews.php
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<body>
<?php

$hostname = "10.89.0.2";
...output omitted...
```

3.5. View the image to discover the exposed port and verify whether the port is open.

The port is not open.

```
[root@servera ~]# podman inspect -f "{{.Config.ExposedPorts}}"
registry.access.redhat.com/rhscl/mariadb-101-rhel7
map[3306/tcp:{}]
[root@servera ~]# ss -n -p -t -a | grep -w 3306
```

3.6. View the container's run command options.

```
[root@servera ~]# podman inspect mariadb-service
...output omitted...
"CreateCommand": [
    "podman",
    "run",
    "-d",
    "--name",
    "mariadb-service",
    "-v",
    "/opt/var/lib/mysql/data:/var/lib/mysql/data:Z",
    "-e",
    "MYSQL_USER=site_monitor",
    "-e",
    "MYSQL_PASSWORD=reviews",
    "-e",
    "MYSQL_DATABASE=sitemonitor",
    "--net",
    "maria-bridge",
```

```
"registry.access.redhat.com/rhscl/mariadb-101-rhel7"  
],  
...output omitted...
```

- 3.7. Run the container again with the exposed port.

```
[root@servera ~]# podman stop mariadb-service  
...output omitted...  
[root@servera ~]# podman rm mariadb-service  
...output omitted...  
[root@servera ~]# podman run -d --name mariadb-service -p 3306:3306  
-v /opt/var/lib/mysql/data:/var/lib/mysql/data:Z -e MYSQL_USER=site_monitor  
-e MYSQL_PASSWORD=reviews -e MYSQL_DATABASE=sitemonitor --net maria-bridge  
registry.access.redhat.com/rhscl/mariadb-101-rhel7  
...output omitted...
```

- 3.8. Verify the container is running and the port is open.

```
[root@servera ~]# podman ps  
CONTAINER ID IMAGE COMMAND  
CREATED STATUS PORTS NAMES  
ad91cccc1bd2e registry.access.redhat.com/rhscl/mariadb-101-rhel7 run-mysqld 35  
seconds ago Up 35 seconds ago 0.0.0.0:3306->3306/tcp mariadb-service  
[root@servera ~]# ss -n -p -t -a | grep -w 3306  
LISTEN 0 128 0.0.0.0:3306 0.0.0.0:* users:  
"common", pid=31787, fd=5
```

4. Verify that the application runs correctly and displays data for the views and visits.

- 4.1. Verify that the application obtains data from the database.

```
[root@servera ~]# curl localhost/showviews.php  
...output omitted...  
<table border="1" align="center">  
<tr>  
 <td>Site Name</td>  
 <td>Views</td>  
 <td>Details</td>  
</tr>  
  
<tr>  
 <td>example.com</td>  
 <td>15</td>  
 <td>Example site</td>  
</tr>  
<tr>  
 <td>site-test.com</td>  
 <td>18</td>  
 <td>Test site</td>  
</tr>  
</table>  
...output omitted...
```

- 4.2. Add a record to the database and confirm that the application is displaying it. Use the access data in the `showviews.php` file.

```
[root@servera ~]# mysql -u site_monitor -p -h 10.89.0.2 sitemonitor
Enter password: reviews
...output omitted...
MariaDB [sitemonitor]> INSERT INTO site_monitor (site_name, views, details) VALUES
  ('mysite.com', '1', 'My test site');
Query OK, 1 row affected (0.009 sec)
MariaDB [sitemonitor]> exit
Bye
```

- 4.3. Return to workstation as the student user.

```
[root@servera ~]# exit
[student@servera ~]$ exit
[student@workstation ~]$
```

- 4.4. On the `workstation` machine, open a web browser and navigate to `http://servera/showviews.php`.

Look for the new `mysite.com` data in the table.

Evaluation

On the `workstation` machine, use the `lab` command to grade your work. Correct any reported failures and rerun the script until you receive a passing grade.

```
[student@workstation ~]$ lab grade application-review
```

Finish

On the `workstation` machine, use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish application-review
```

This concludes the lab.

Summary

In this chapter, you learned:

- The `ldd` command displays the shared libraries that an application requires.
- The `valgrind --tool=memcheck` command identifies memory leaks.
- The `strace` command traces application system calls.
- The `ltrace` command traces application library calls.
- The `podman healthcheck` command determines the health or readiness of a process inside a container, with a command that you define.
- The `podman logs` command displays a container's log files.

Chapter 9

Troubleshooting Security Issues

Goal

Identify and resolve issues related to security subsystems.

Objectives

- Identify and repair SELinux issues.
- Identify and repair user authentication and authorization issues.
- Identify and repair LDAP, Kerberos, and SSSD identity management issues.

Sections

- Repairing SELinux Issues (and Guided Exercise)
- Identifying Authentication Issues
- Identifying Authentication Issues (Guided Exercise)
- Resolving Identity Management Issues (and Guided Exercise)

Lab

Troubleshooting Security Issues

Repairing SELinux Issues

Objectives

After completing this section, you should be able to identify and repair SELinux issues.

SELinux Logging

Security Enhanced Linux (SELinux) implements *Mandatory Access Control* (MAC). Every process and system resource has a security label called an *SELinux context*. An SELinux context, sometimes known as an SELinux label, is an identifier that abstracts the system-level details and focuses on the security properties of the entity. The SELinux policy uses these contexts in a series of rules that define how processes can interact with each other and the various system resources. By default, the policy does not allow any interaction unless a rule explicitly grants access.

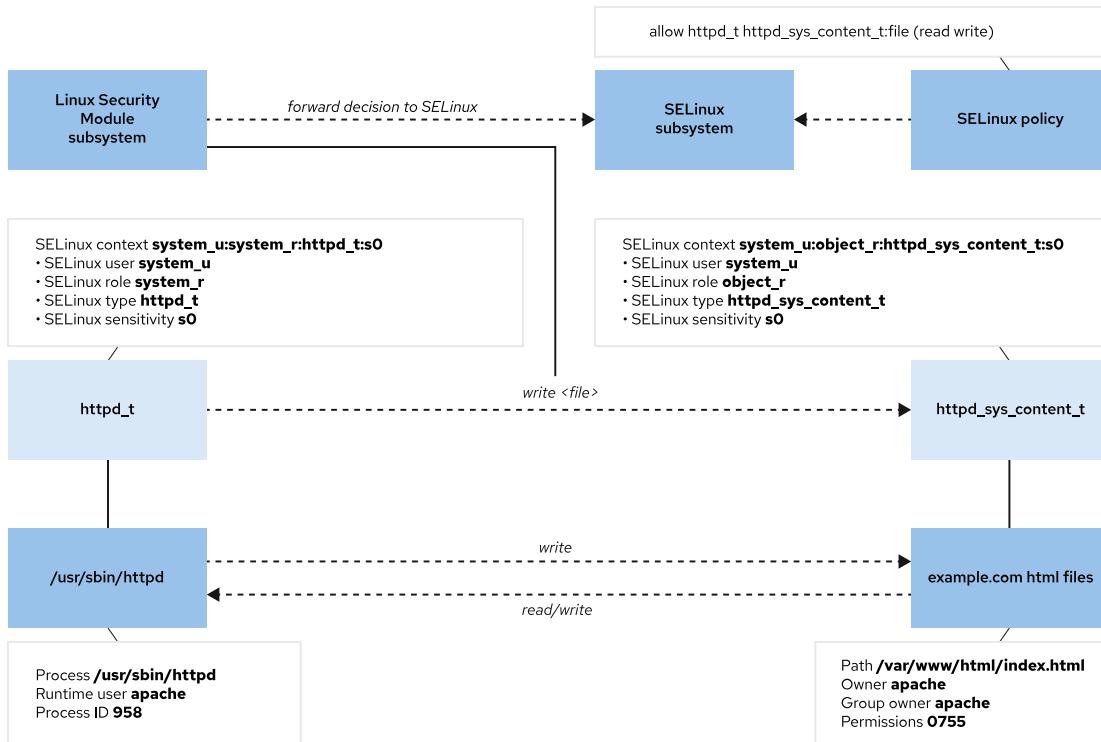


Figure 9.1: SELinux decision making flow

When SELinux blocks an action from happening, such as when a web server tries to write to `/etc/shadow`, the action is logged with the `auditd` daemon. The audit messages are in `/var/log/audit/audit.log` where they are searched with the `ausearch` command. Administrators can use the `ausearch` command to focus only on messages of interest.

```
[root@host ~]# ausearch -m avc -ts recent ① ②
---
time->Sun Nov  7 21:01:00 2021
type=PROCTITLE msg=audit(1636336860.921:117):
  procname=2F7573722F7362696E2F6874747064002D44464F524547524F554E44 ③
type=SYSCALL msg=audit(1636336860.921:117): arch=c000003e syscall=4 success=no
  exit=-13 a0=7fd8e400e898 a1=7fd8ebffe810 a2=7fd8ebffe810 a3=1 items=0 ppid=5497
  pid=5501 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) ④
type=AVC msg=audit(1636336860.921:117): avc: denied { getattr } for
  pid=5501 comm="httpd" path="/etc/shadow" dev="vda3" ino=16803119
  scontext=system_u:system_r:httpd_t:s0 tcontext=system_u:object_r:shadow_t:s0
  tclass=file permissive=0 ⑤
```

- ① The `-m avc` option directs `ausearch` to display only Access Vector Cache (AVC) messages, the message type that is associated with SELinux denials.
- ② The `-ts recent` option specifies to show messages starting from 10 minutes ago. Other indications, such as `today`, `yesterday`, and `this-week` are recognized, as well as actual times.
- ③ Typically, three lines are shown for every SELinux denial. The `type=PROCTITLE` line displays the full command line that triggered the Audit event. The field is encoded in hexadecimal notation so that the user cannot influence the Audit log parser. Use the `ausearch` command with the `--interpreter` option to convert hexadecimal values into human-readable equivalents. The `2F7573722F7362696E2F6874747064002D44464F524547524F554E44` value is interpreted as `/usr/sbin/httpd -DFOREGROUND`.
- ④ The `type=SYSCALL` line has useful event information, such as the actual, original, and effective user ID of the calling process and the action that caused the denial, typically a system call, and the actual denial.
- ⑤ The `type=AVC` line is the actual denial. The `audit(1636336860.921:117)` part lists the time when this message was passed to the audit subsystem, in seconds since the epoch. Use the `date --date=@timestamp` command to convert the time to the local time zone. The remainder of the line is 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 relative path, the file system's device, the inode of the *target* file, and the full SELinux contexts of both the process that attempts access and the file, process, or network port that is accessed.

Using `dontaudit` Rules to Limit Audit Events

Some frequent actions are always denied and do not need to be tracked. The SELinux policy has a `dontaudit` rule, so that the action is still blocked, but not logged.

Use the `sestatus --dontaudit` command from the `setools-console` package to list the active `dontaudit` rules.

If a `dontaudit` rule is suspected as the cause of an issue, then all `dontaudit` rules can be temporarily disabled with the `semanage dontaudit off` command. The `dontaudit` rules remain disabled until the `semanage dontaudit on` command is run.

**Important**

The `dontaudit` rules exist for frequent actions that are always blocked and do not impact system behavior. However, keeping the `dontaudit` feature disabled fills the audit log rapidly, and impedes locating more important denial events.

SELinux Troubleshooting Tools

When SELinux blocks an action, first look in the `/var/log/audit/audit.log` file for information about a denial. Use the `ausearch` command to query the audit logs.

If `auditd` is running, but the output of the `ausearch` command does not have any matches, then use the `journalctl` command to inspect the messages from the `systemd` journal.

```
[root@host ~]# journalctl -t setroubleshoot
-- Logs begin at Sun 2021-11-07 19:19:56 EST, end at Sun 2021-11-07 22:34:13 EST.
--
Nov 07 21:01:05 host.example.com setroubleshoot[6414]: AnalyzeThread.run(): Cancel pending alarm
Nov 07 21:01:06 host.example.com setroubleshoot[6414]: SELinux is preventing /usr/sbin/httpd from getattr access on the file /etc/shadow. For complete SELinux messages run: sealert -l 2e7830e4-aed5-4ff6-b242-47d104ebc9ec
Nov 07 21:01:06 host.example.com setroubleshoot[6414]: SELinux is preventing /usr/sbin/httpd from getattr access on the file /etc/shadow.

***** Plugin catchall (100. confidence) suggests *****

If you believe that httpd should be allowed getattr access on the shadow file by default.
Then you should report this as a bug.
You can generate a local policy module to allow this access.
Do allow this access for now by executing:
# ausearch -c 'httpd' --raw | audit2allow -M my-httpd
# semodule -X 300 -i my-httpd.pp
...output omitted...
```

You can diagnose and fix many simple SELinux denials, such as mislabeled files in `/var/www/html`, by inspecting `/var/log/audit/audit.log` and running the `restorecon` command in the files and directories to correct the labels. Other denials are more difficult to diagnose and fix. The `sealert` command works in two ways:

- Passing it the UUID of a denial found in the system logs, such as `sealert -l fdfef9d90-557b-4ad0-a1a3-a6d09e065523`.
- Parsing all denial messages in a file, such as `sealert -a /var/log/audit/audit.log`.

In both cases, the output is the same as `setroubleshootd` writes to the system logs.

**Important**

The `setroubleshoot` daemon might suggest creating a policy module with the `audit2allow` command. In most cases, it is not the preferred solution. Build custom policy modules only when the impact of the new policy module rules is fully understood.

Common SELinux issues

Most SELinux issues fall into the following scenarios:

- Using nonstandard locations for service data.
- Switching from `disabled` to `enforcing` mode.
- Setting Booleans incorrectly.
- Using nonstandard network ports for services.

Each scenario has common characteristics that assist troubleshooting.

Using nonstandard locations for service data

The targeted policy provides default file contexts for all RHEL services, including secondary data locations. For example, the default locations for web server data at `/var/www` and secondary locations under `/srv/*/www` are recognized and set with the correct SELinux contexts when using the `restorecon` command.

Use the `semanage fcontext` command to list and manage the stored mappings between file names and standard contexts.

When a nonstandard data location is used, configure SELinux to recognize the location. For example, to use `/mysites/sitea/www` as a document root for a web server, first create a mapping in the SELinux database. This example configures the `httpd_sys_content_t` context on all files under the `/mysites/sitea/www` directory.

```
[root@host ~]# semanage fcontext -a -t httpd_sys_content_t
'/mysites/sitea/www(.*)?'
```

Then, restore correct settings recursively to the `/mysites/sitea/www` directory, with the proper contexts obtained from the database.

```
[root@host ~]# restorecon -Rv /mysites/sitea/www
```

Switching from disabled to enforcing mode.

When a system is running with SELinux disabled, any program that creates files does not set the file's SELinux context, because the SELinux policy module is not loaded and therefore cannot provide or verify the context. Programs such as `sed`, other text editors, and NetworkManager do not edit files in place, but create a temporary file and then move that file to the original file name.

When SELinux is enabled after the system is run in disabled mode, SELinux does not trust that the system is secure, because it does not immediately know if files exist without their required SELinux context. To resolve this problem, a system must be relabeled. Relabeling is the process of checking and updating every file with the proper SELinux file context as stored in the `semanage` database.

If a system with SELinux enabled has unlabeled files, then normal, confined services cannot access those files with SELinux policies. Unlabeled files display the `unlabeled_t` type. If critical files, such as for user authentication, are unlabeled, then RHEL might not boot, because the files are insecure and untrusted.

To force a relabel, create an empty file called `/.autorelabel`, and then reboot. When detected during the boot process, the system relabels all files, deletes the `/.autorelabel` file, and then restarts the boot process. This process confirms that all files on a running system with SELinux enabled are properly protected by labels.

Setting Booleans incorrectly

The confined services in RHEL 8 are limited by their SELinux policy, but have optional features that can be enabled with SELinux Booleans. For example, by default the `httpd` services can serve content only from their main web server directory. Moreover, the `httpd` services cannot create outgoing network connections, by default.

Specific Booleans exist for `httpd` use cases such as connecting to various databases, `memcached` use, FTP servers, and LDAP services. One Boolean allows the `httpd` service to permit all network connections, if another Boolean does not cover a specific use case. Another Boolean allows the `httpd` service to permit users to access their home directory from a web browser with `http`.

Use the `semanage boolean --list` command to query the list of Booleans, including their current state, default state, and description. Use the `getsebool` and `setsebool` commands to query and set the Booleans individually.



Important

Use the `setsebool -P` flag to set Boolean values that persist across reboots. Otherwise, non-default Boolean settings are lost when SELinux or the system is stopped.

Using nonstandard network ports for services

By default, the `targeted` policy allows confined services to listen only on a predefined set of network ports. For example, the `httpd` daemon is allowed to bind only to ports that are labeled as either `http_port_t` or `http_cache_port_t`.

Use the `semanage` command to label an unlabeled port.

```
[root@host ~]# semanage port -a -t http_port_t -p tcp 8001
```

With a port label policy, services cannot listen on ports with a conflicting type. For example, `sshd` cannot listen on port 443/tcp, which is labeled for use by the `httpd` service. Although not normally recommended, you can create policies that allow non-standard port bindings.

Understanding SELinux Rules

The `targeted` SELinux policy is the default policy on RHEL 8 and includes with many types, rules, and Booleans to cover all of the services that are distributed in RHEL. To learn about the supported types, policy rules, and available Booleans, install the `setools-console` package. This package provides various tools to help with SELinux troubleshooting, including `seinfo` and `sesearch`.

The `seinfo -t` and `seinfo -b` commands list all types and Booleans. Passing a type name with the `-t` option displays that type's details and configured aliases.

```
[root@host ~]# seinfo -t httpd_sys_content_t
Types: 1
    httpd_sys_content_t
```

The `seinfo` command determines which port types are associated with a specific network port:

```
[root@host ~]# seinfo --portcon=443

Portcon: 4
  portcon sctp 1-511 system_u:object_r:reserved_port_t:s0
  portcon tcp 1-511 system_u:object_r:reserved_port_t:s0
  portcon tcp 443 system_u:object_r:http_port_t:s0
  portcon udp 1-511 system_u:object_r:reserved_port_t:s0
```

The `sesearch` command searches within the defined policy rules, to indicate which rules a Boolean enables. For example, use `sesearch -b` to view all `allow` rules enabled by the `httpd_can_connect_ldap` Boolean:

```
[root@host ~]# sesearch --allow -b httpd_can_connect_ldap
allow httpd_t ldap_port_t:tcp_socket name_connect; [ httpd_can_connect_ldap ]:True
```

This rule states that processes of the type `httpd_t` can connect to TCP sockets of the type `ldap_port_t`.



References

Further information is available in the *Using SELinux* chapter in the *System Design Guide* at

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/system_design_guide/index#using_selinux

For more information, refer to the *Auditing the system* chapter in the *Security Hardening Guide* at

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/security_hardening/index#auditing-the-system_security-hardening

`semanage(8)`, `sesearch(1)`, `getsebool(8)`, `setsebool(8)`, `sealert(8)`, `setroubleshootd(8)`, `seinfo(1)`, and `sesearch(1)` man pages.

► Guided Exercise

Repairing SELinux Issues

In this lab, you resolve a database service running issue.

Outcomes

You should be able to resolve SELinux issues.

Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

```
[student@workstation ~]$ lab start security-selinux
```

This command confirms that the required hosts for this exercise are accessible and installs the database service.

Instructions

A coworker recently installed a custom database service for an application, which used a non-default configuration in the service. Your coworker failed to get the service running. Your manager assigned you to solve the issue and set `redhat` as the password for the `root` user on the database.

- 1. Log in to `servera` and switch to the `root` user.

```
[student@workstation ~]$ ssh student@servera
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

- 2. Diagnose the failed service.

- 2.1. View the status of the `mariadb` service.

```
[root@servera ~]# systemctl status mariadb.service
● mariadb.service - MariaDB 10.3 database server
   Loaded: loaded (/usr/lib/systemd/system/mariadb.service; enabled; vendor
   preset: disabled)
     Active: failed (Result: exit-code) since Wed 2021-11-10 00:10:05 EST; 22s ago
       Docs: man:mysqld(8)
              https://mariadb.com/kb/en/library/systemd/
     Process: 8362 ExecStart=/usr/libexec/mysqld --basedir=/usr $MYSQLD_OPTS
$_WSREP_NEW_CLUSTER (code=exited, status=1/FAILURE)
     Process: 8323 ExecStartPre=/usr/libexec/mysql-prepare-db-dir mariadb.service
               (code=exited, status=0/SUCCESS)
```

```
Process: 8298 ExecStartPre=/usr/libexec/mysql-check-socket (code=exited,
status=0/SUCCESS)
Main PID: 8362 (code=exited, status=1/FAILURE)
    Status: "MariaDB server is down"
...output omitted...
```

2.2. Restart the mariadb service to verify that the service fails.

```
[root@servera ~]# systemctl restart mariadb.service
Job for mariadb.service failed because the control process exited with error code.
See "systemctl status mariadb.service" and "journalctl -xe" for details.
```

2.3. View the last log entry for the failed service.

The output reports several files with SELinux access failures, possibly because of customization of the service.

```
[root@servera ~]# journalctl -r
-- Logs begin at Tue 2021-11-09 23:22:48 EST, end at Wed 2021-11-10 00:15:55 EST.
--
Nov 10 00:15:55 servera.lab.example.com setroubleshoot[8566]: AnalyzeThread.run():
Set alarm timeout to 10
Nov 10 00:15:55 servera.lab.example.com setroubleshoot[8566]: SELinux is
preventing /usr/libexec/mysqld from write access on the file /opt/database/
aria_log_control.

***** Plugin catchall_labels (83.8 confidence) suggests *****

If you want to allow mysqld to have write access on the aria_log_control
file
Then you need to change the label on /opt/database/aria_log_control
Do
# semanage fcontext -a -t FILE_TYPE '/opt/database/aria_log_control'
where FILE_TYPE is one of the following: afs_cache_t, faillog_t,
hugetlbfs_t, initrc_tmp_t, krb5_host_rcache_t, lastlog_t, mysqld_db_t,
mysqld_log_t, mysqld_tmp_t, mysqld_var_run_t, puppet_tmp_t, security_t,
user_cron_spool_t.
Then execute:
restorecon -v '/opt/database/aria_log_control'
...output omitted...
```

► 3. Troubleshoot and resolve the SELinux file context issue.

3.1. Search for all AVC denials in the audit.log file.

Look for a message that correlates to the previously viewed error in the journal.

```
[root@servera ~]# ausearch -m avc -ts recent
---
time->Wed Nov 10 00:29:39 2021
type=PROCTITLE msg=audit(1636522179.002:1164):
    proctitle=2F7573722F6C6962657865632F6D7973716C64002D2D626173656469723D2F757372
type=SYSCALL msg=audit(1636522179.002:1164): arch=c000003e syscall=257 success=no
    exit=-13 a0=ffffff9c a1=7ffe36a29310 a2=80042 a3=1b6 items=0 ppid=1 pid=8691
    auid=4294967295 uid=27 gid=27 euid=27 suid=27 egid=27 sgid=27
    fsgid=27 tty=(none) ses=4294967295 comm="mysqld" exe="/usr/libexec/mysqld"
    subj=system_u:system_r:mysqld_t:s0 key=(null)
type=AVC msg=audit(1636522179.002:1164): avc: denied { write }
for pid=8691 comm="mysqld" name="database" dev="vda3" ino=25793706
scontext=system_u:system_r:mysqld_t:s0 tcontext=unconfined_u:object_r:object_t:s0
tclass=dir permissive=0 ①
---
...output omitted...
```

- ① The message indicates denied access to `mysqld` for writing to the `database` directory.

3.2. View the database configuration file to discover the service customization.

```
[root@servera ~]# less /etc/my.cnf.d/mariadb-server.cnf
...output omitted...
[mysqld]
datadir=/opt/database/
socket=/var/lib/mysql/mysql.sock
log-error=/var/log/database/mariadb.log
pid-file=/run/mariadb/mariadb.pid
...output omitted...
```

3.3. Verify the context of the directory that is configured as `datadir`.

```
[root@servera ~]# ls -dz /opt/database
unconfined_u:object_r:object_t:s0 /opt/database
```

3.4. Review the default `datadir` directory context.

```
[root@servera ~]# semanage fcontext --list | grep '/var/lib/mysql'
/var/lib/mysql(-files|-keyring)?(/.*)? all files
    system_u:object_r:mysqld_db_t:s0
/var/lib/mysql/mysql.sock socket
    system_u:object_r:mysqld_var_run_t:s0
```

3.5. Set the persistent file context for the customized `datadir` directory.

```
[root@servera ~]# semanage fcontext -a -t mysqld_db_t "/opt/database(/.*)?"  
[root@servera ~]# restorecon -Rv /opt/database  
Relabeled /opt/database from unconfined_u:object_r:usr_t:s0 to  
unconfined_u:object_r:mysqld_db_t:s0  
Relabeled /opt/database/ibdata1 from system_u:object_r:usr_t:s0 to  
system_u:object_r:mysqld_db_t:s0  
Relabeled /opt/database/ib_logfile1 from system_u:object_r:usr_t:s0 to  
system_u:object_r:mysqld_db_t:s0  
...output omitted...
```

3.6. Restart the mariadb service to verify the fix.

```
[root@servera ~]# systemctl restart mariadb.service  
Job for mariadb.service failed because the control process exited with error code.  
See "systemctl status mariadb.service" and "journalctl -xe" for details.
```

▶ 4. Troubleshoot and resolve the SELinux port issue.

4.1. View the last log entry for the failed service.

```
[root@servera ~]# journalctl -r  
-- Logs begin at Tue 2021-11-09 23:22:48 EST, end at Wed 2021-11-10 00:55:37 EST.  
--  
Nov 10 00:55:37 servera.lab.example.com setroubleshoot[8870]: AnalyzeThread.run():  
Set alarm timeout to 10  
Nov 10 00:55:37 servera.lab.example.com setroubleshoot[8870]: SELinux is  
preventing mysqld from name_bind access on the tcp_socket port 3307.  
  
***** Plugin bind_ports (92.2 confidence) suggests *****  
  
If you want to allow mysqld to bind to network port 3307  
Then you need to modify the port type.  
Do  
# semanage port -a -t PORT_TYPE -p tcp 3307  
where PORT_TYPE is one of the following: mysqld_port_t, tram_port_t.  
...output omitted...
```

4.2. View the database configuration file to discover the service port customization.

```
[root@servera ~]# grep port /etc/my.cnf.d/mariadb-server.cnf  
port = 3307
```

4.3. Set the port context for the customized service port.

```
[root@servera ~]# semanage port --list | grep mysqld  
mysqld_port_t    tcp    1186, 3306, 63132-63164  
[root@servera ~]# semanage port -a -t mysqld_port_t -p tcp 3307
```

4.4. Restart the mariadb service to verify the fix.

```
[root@servera ~]# systemctl restart mariadb.service
[root@servera ~]# systemctl status mariadb.service
● mariadb.service - MariaDB 10.3 database server
   Loaded: loaded (/usr/lib/systemd/system/mariadb.service; enabled; vendor
preset: disabled)
     Active: active (running) since Wed 2021-11-10 01:05:25 EST; 3s ago
       Docs: man:mysqld(8)
              https://mariadb.com/kb/en/library/systemd/
   Process: 9020 ExecStartPost=/usr/libexec/mysql-check-upgrade (code=exited,
status=0/SUCCESS)
   Process: 8950 ExecStartPre=/usr/libexec/mysql-prepare-db-dir mariadb.service
(code=exited, status=0/SUCCESS)
   Process: 8926 ExecStartPre=/usr/libexec/mysql-check-socket (code=exited,
status=0/SUCCESS)
 Main PID: 8989 (mysqld)
    Status: "Taking your SQL requests now..."
...output omitted...
```

► 5. Complete the database configuration and verify that the service is functional.

5.1. Stop the database service and set the password of the database root user.

```
[root@servera ~]# systemctl stop mariadb.service
[root@servera ~]# mysqld_safe --skip-grant-tables --skip-networking &
[1] 9082
211110 01:10:06 mysqld_safe Logging to '/var/log/database/mariadb.log'.
211110 01:10:06 mysqld_safe Starting mysqld daemon with databases from /opt/
database/
[root@servera ~]# mysql -u root
...output omitted...
MariaDB [(none)]> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.001 sec)

MariaDB [(none)]> ALTER USER 'root'@'localhost' IDENTIFIED BY 'redhat';
Query OK, 0 rows affected (0.001 sec)

MariaDB [(none)]> exit
Bye
[root@servera ~]#
```

5.2. Restart the database service and confirm the password setting of the database root user.

```
[root@servera ~]# pkill mysql
[root@servera ~]# systemctl restart mariadb.service
[1]+  Done    mysqld_safe --skip-grant-tables --skip-networking
[root@servera ~]# mysql -u root -p
Enter password: redhat
...output omitted...
MariaDB [(none)]> exit
Bye
[root@servera ~]#
```

- 5.3. View the database service log to verify that event entries are being logged.

The log is not recording database event entries.

```
[root@servera ~]# tail /var/log/database/mariadb.log
...output omitted...
2021-11-10 1:17:02 0 [Note] InnoDB: Buffer pool(s) dump completed at 211110
1:17:02
2021-11-10 1:17:03 0 [Note] InnoDB: Shutdown completed; log sequence number
1625470; transaction id 21
2021-11-10 1:17:03 0 [Note] InnoDB: Removed temporary tablespace data file:
"ibtmp1"
2021-11-10 1:17:03 0 [Note] /usr/libexec/mysqld: Shutdown complete
```

► 6. Troubleshoot and resolve the SELinux context issue on the database log.

- 6.1. View the last log entry for the failed service.

```
[root@servera ~]# journalctl -r
...output omitted...
Nov 10 01:18:05 servera.lab.example.com setroubleshoot[9395]: SELinux is
preventing /usr/libexec/mysqld from open access on the file /var/log/database/
mariadb.log.

***** Plugin catchall (100. confidence) suggests *****

If you believe that mysqld should be allowed open access on the mariadb.log
file by default.
Then you should report this as a bug.
You can generate a local policy module to allow this access.
Do
allow this access for now by executing:
# ausearch -c 'mysqld' --raw | audit2allow -M my-mysqld
# semodule -X 300 -i my-mysqld.pp
...output omitted...
```

- 6.2. Search for all AVC denials in the `audit.log` file.

Look for a message that correlates to the previously viewed error in the journal.

```
[root@servera ~]# ausearch -m avc -ts recent
...output omitted...
---
time-->Wed Nov 10 01:18:03 2021
type=PROCTITLE msg=audit(1636525083.284:1234):
    procitle=2F7573722F6C6962657865632F6D7973716C64002D2D626173656469723D2F757372
type=SYSCALL msg=audit(1636525083.284:1234): arch=c000003e syscall=257 success=no
    exit=-13 a0=fffffff9c a1=55a350340180 a2=441 a3=1b6 items=0 ppid=1 pid=9330
    auid=4294967295 uid=27 gid=27 euid=27 suid=27 fsuid=27 egid=27 sgid=27
    fsgid=27 tty=(none) ses=4294967295 comm="mysqld" exe="/usr/libexec/mysqld"
    subj=system_u:system_r:mysqld_t:s0 key=(null)
type=AVC msg=audit(1636525083.284:1234): avc: denied { open }
    for pid=9330 comm="mysqld" path="/var/log/database/mariadb.log"
    dev="vda3" ino=435268 scontext=system_u:system_r:mysqld_t:s0
    tcontext=unconfined_u:object_r:var_log_t:s0 tclass=file permissive=0 ①
```

- ① The message indicates denied access to mysqld for writing the /var/log/database/mariadb.log file.

6.3. Set the persistent file context for the customized log file.

```
[root@servera ~]# semanage fcontext --list | grep mysqld_log_t
/var/log/mariadb(/.*)?    all files      system_u:object_r:mysqld_log_t:s0
/var/log/mysql(/.*)?      all files      system_u:object_r:mysqld_log_t:s0
/var/log/mysql.*          regular file   system_u:object_r:mysqld_log_t:s0
[root@servera ~]# semanage fcontext -a -t mysqld_log_t "/var/log/database(/.*)?"
[root@servera ~]# restorecon -Rv /var/log/database
Relabeled /var/log/database from unconfined_u:object_r:var_log_t:s0 to
unconfined_u:object_r:mysqld_log_t:s0
Relabeled /var/log/database/mariadb.log from unconfined_u:object_r:var_log_t:s0 to
unconfined_u:object_r:mysqld_log_t:s0
```

6.4. Restart the mariadb service to verify the fix.

```
[root@servera ~]# systemctl restart mariadb.service
[root@servera ~]# tail /var/log/database/mariadb.log
...output omitted...
2021-11-10  1:44:04 0 [Note] Reading of all Master_info entries succeeded
2021-11-10  1:44:04 0 [Note] Added new Master_info '' to hash table
2021-11-10  1:44:04 0 [Note] /usr/libexec/mysqld: ready for connections.
Version: '10.3.27-MariaDB'  socket: '/var/lib/mysql/mysql.sock'  port: 3307
  MariaDB Server
```

► 7. Return to workstation as the student user.

```
[root@servera ~]# exit
[student@servera ~]$ exit
[student@workstation ~]$
```

Finish

On the **workstation** machine, use the **lab** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish security-selinux
```

This concludes the guided exercise.

Identifying Authentication Issues

Objectives

After completing this section, you should be able to identify and repair user authentication and authorization issues.

Review of Pluggable Authentication Modules

Pluggable Authentication Modules (PAM) provide a common interface for applications to authenticate users. Applications call the `libpam` library to authenticate users with one or more PAM modules. Administrators can use the *pluggable* aspect of PAM to choose how individual applications authenticate users.

Modules provide different authentication mechanisms. For example, the `pam_unix` module authenticates users with their passwords. Alternatively, the `pam_group` module authenticates users with their group memberships.

PAM-enabled applications store configuration files in the `/etc/pam.d/` directory. The `libpam` library uses configuration files to determine which modules to implement. Generally, configuration files share the same name as their corresponding application. For example, the `/etc/pam.d/sshd` file applies to the `sshd` service.

Rules in PAM configuration files determine how PAM handles failed authentication checks.

PAM Configuration Files

Rules in PAM configuration files use the following format:

```
type control module-path [module-arguments]
```

- The `type` field is the management group. Possible values include `account`, `auth`, `password`, and `session`.
- The `control` field determines how PAM handles a failed authentication. Possible values include `required`, `requisite`, `sufficient`, `optional`, `include`, and `substack`.
- The `module-path` field is the absolute or relative path to the module. By default, modules are expected in the `/lib64/security/` directory.
- The `module-arguments` field is a space-delimited list of arguments for the specified module. Not all modules require arguments.



Note

The range of PAM configuration file choices is beyond the scope of this course. Consult the `pam` man page for more information.

PAM applies the rules in configuration files, in order, from top to bottom.

Consider the `/etc/pam.d/sshd` configuration file:

```
%%PAM-1.0
auth      substack    password-auth
auth      include     postlogin
account   required    pam_sepermit.so
account   required    pam_nologin.so
account   include     password-auth
password  include     password-auth
session   required    pam_selinux.so close
session   required    pam_loginuid.so
session   required    pam_selinux.so open env_params
session   required    pam_namespace.so
session   optional    pam_keyinit.so force revoke
session   optional    pam_motd.so
session   include     password-auth
session   include     postlogin
```

Many configuration files, including `/etc/pam.d/sshd`, incorporate rules from other files with the `substack` or `include` control values. In the previous output, the `password-auth` configuration file provides password authentication for users who log in with SSH.



Note

As an administrator, avoid manually editing PAM configuration files. Instead, use the `authselect` tool to configure system identity and authentication sources by selecting a specific PAM profile.

Troubleshooting PAM Issues

By default, PAM sends events to the `/var/log/secure` file. Alternatively, PAM-related entries might appear in application-specific log files.

```
[root@host ~]# tail /var/log/secure | grep pam
Nov  4 12:47:09 server sshd[1455]: pam_unix(sshd:auth): check pass; user unknown
Nov  4 12:47:09 server sshd[1455]: pam_unix(sshd:auth): authentication failure
```

Many modules support the `debug` argument to generate more verbose logs.

```
auth      sufficient  pam_unix.so debug
```

The `pam_echo` module prints messages and helps to determine which modules are causing authentication issues.

```
auth      required    pam_env.so
auth      optional    pam_echo.so "Passed pam_env!"
auth      sufficient  pam_unix.so
auth      optional    pam_echo.so "Passed pam_unix!"
```

Administrators can use the `pam_debug` module to manually set a pass or fail response. This module is useful to troubleshoot how certain configurations affect the authentication process.

```
auth      sufficient      pam_debug.so auth=success cred=success
```

Restoring PAM Configuration Files

The `authselect` command modifies PAM configuration files with *profiles*. A profile is a set of PAM configuration files that use a specific authentication scheme. For example, the `sssd` (System Security Services Daemon) profile incorporates the `pam_sss` module in the `/etc/pam.d/system-auth` file.

The `authselect select` command configures a profile. The `--backup` option backs up the current PAM configuration files.

```
[root@host ~]# authselect select sssd --backup pamfiles.bak
```

The `backup-restore` subcommand restores a backup.

```
[root@host ~]# authselect restore-backup /var/lib/authselect/backups/pamfiles.bak
```

The `authselect` command does not modify application-specific PAM files. Instead, it modifies general-purpose files including `/etc/pam.d/system-auth` and `/etc/pam.d/password-auth`. Restore application-specific PAM configuration files by reinstalling the application.

```
[root@host ~]# cat /etc/pam.d/sshd
# Accidentally deleted the contents!
[root@host ~]# yum reinstall openssh-server
...output omitted...
[root@host ~]# cat /etc/pam.d/sshd
 #%PAM-1.0
auth      substack    password-auth
auth      include     postlogin
account   required    pam_sepermit.so
account   required    pam_nologin.so
account   include     password-auth
password  include     password-auth
...output omitted...
```



References

`pam(8)`, `pam.d(5)`, and `authselect(8)` man pages.

► Guided Exercise

Identifying Authentication Issues

In this exercise, you diagnose and correct an authentication-related issue.

Outcomes

You should be able to resolve authentication-related issues.

Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

```
[student@workstation ~]$ lab start security-authentication
```

This command configures an FTP server and modifies PAM configuration files.

Instructions

Despite the security risks, your company provides authenticated FTP access to home directories on the `serverb` system. After a recent change, users report that they cannot log in to the FTP server. You are tasked to restore authenticated FTP access. Use the `ftpuser` account with the `redhat` password to troubleshoot the issue.

- 1. Log in to the `servera` system and switch to the `root` user.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

- 2. Attempt to log in to the FTP server.

- 2.1. Run the `lftp` command to log in as the `ftpuser` user.

```
[root@servera ~]# lftp ftpuser@serverb.lab.example.com
Password: redhat
lftp ftpuser@serverb.lab.example.com :~>
```

- 2.2. FTP servers authenticate users only when they execute a command. Run an `ls` command to cause the authentication attempt and error. Do not exit the `lftp` prompt.

```
lftp ftpuser@serverb.lab.example.com :~> ls
ls: Login failed: 530 Login incorrect.
```

▶ 3. Gather more troubleshooting information.

- 3.1. From the **workstation** system, open a second terminal. Log in to the **serverb** system and switch to the **root** user.

```
[student@workstation ~]$ ssh student@serverb
...output omitted...
[student@serverb ~]$ sudo -i
[sudo] password for student: student
[root@serverb ~]#
```

- 3.2. View the **/var/log/secure** file to locate FTP-related authentication errors.

The log file indicates a problem with the **vsftpd** PAM configuration file.

```
[root@serverb ~]# tail -n 10 /var/log/secure
...output omitted...
Nov  8 12:03:52 server vsftpd[2010]: PAM unable to resolve symbol:
pam_sm_acct_mgmt
...output omitted...
```

- 3.3. Verify if any files that belong to the **vsftpd** package changed since installation.

The output indicates that the **vsftpd** PAM configuration changed.

```
[root@serverb ~]# rpm -V vsftpd
S.5....T. c /etc/pam.d/vsftpd
```

▶ 4. Restore the **vsftpd** PAM configuration file. Keep a backup of the broken file.

- 4.1. Rename the broken file.

```
[root@serverb ~]# mv /etc/pam.d/vsftpd{, .broken}
```

- 4.2. Reinstall the **vsftpd** package.

```
[root@serverb ~]# yum reinstall vsftpd
```

**Note**

The broken file is *moved*, not copied, because the **yum** command by default does not overwrite configuration files when reinstalling. The broken file would remain if the **cp** command was used.

▶ 5. Test whether authenticated FTP access is restored, and then analyze why the modified PAM configuration file did not work.

- 5.1. On the **servera** system, test whether authenticated FTP access is restored.

```
lftp ftpuser@serverb.lab.example.com :~> ls
-rw-r--r--    1 0          0           12 Feb 17 12:05 README.txt
```

**Note**

If the authentication error persists, then exit the `lftp` prompt and reconnect to the FTP server.

- 5.2. On the `serverb` system, compare the `/etc/pam.d/vsftpd` and `/etc/pam.d/vsftpd.broken` files.

```
[root@serverb ~]# diff -u /etc/pam.d/vsftpd{,.broken}
...output omitted...
+account    required    pam_ftp.so
...output omitted...
```

The broken file has an extra requirement to use the `pam_ftp.so` module.

- 5.3. Read the `pam_ftp` module's documentation to determine why it caused an authentication error.

```
[root@serverb ~]# man pam_ftp
```

According to the man page, the `pam_ftp.so` module provides anonymous FTP access. However, the module is available for use only in `auth` rules and not in `account` rules.

- 6. Exit the second terminal. Return to `workstation` as the `student` user.

```
[root@serverb ~]# exit
[student@serverb ~]$ exit
[student@workstation ~]$ exit
```

```
lftp ftpuser@serverb.lab.example.com:~> exit
[root@servera ~]# exit
[student@servera ~]$ exit
[student@workstation ~]$
```

Finish

On the `workstation` machine, use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish security-authentication
```

This concludes the guided exercise.

Resolving Identity Management Issues

Objectives

After completing this section, you should be able to identify and repair LDAP, Kerberos, and SSSD identity management issues.

What Is Identity Management?

Enterprise computing environments typically consist of hundreds or thousands of interconnected computer systems. These systems run many applications and services, and have many local and remote users who access these systems daily.

Historically, the user accounts, application permissions, and security policies were configured with a diverse set of protocols and utilities, such as the Network Information Service (NIS), Kerberos, Lightweight Directory Access Protocol (LDAP), sudo, and TCP Wrappers. Each of these tools was locally managed and separately administered.

Identity management solutions address this administrative ordeal by creating and configuring users, systems, services, and policies with centralized storage and a single set of tools that is built on existing, native Linux protocols and technologies.

Red Hat Identity Management (IdM) implements identity and access management with central authentication management of identities and security mechanisms, fine-grained access control policies, integrated Public Key Infrastructure (PKI) services, two-factor authentication (2FA) support, cross-realm Kerberos trusts with Active Directory (AD), and a feature for direct client connections to AD.

Although the IdM toolset includes all of these services, you can select to install only the services that you require. Some services can be distributed to additional, external servers for scaling, and be integrated with the IdM domain.

The Kerberos and LDAP services are always installed and are integral to every IdM server. The `ipa` command uses Kerberos and LDAP to search for user information. The `ipa` command is an integrated security information management solution that includes a web interface and command-line administration tools for managing identity data.

For example, to search for user information, use the `kinit` command to obtain a Ticket Granting Ticket (TGT), and then use the `ipa` command to gather user information.

Obtain a TGT:

```
[user@host ~]$ kinit idmuser
Password for idmuser@LAB.EXAMPLE.NET :
[user@host ~]$ klist
Ticket cache: KCM:1000
Default principal: idmuser@LAB.EXAMPLE.NET

Valid starting     Expires            Service principal
11/21/2021 23:17:35 11/22/2021 23:17:33  krbtgt/LAB.EXAMPLE.NET@LAB.EXAMPLE.NET
```

Gather user information:

```
[user@host ~]$ ipa user-find idmuser
-----
1 user matched
-----
User login: idmuser
First name: idmuser
Last name: idm
Home directory: /home/idmuser
Login shell: /bin/sh
Principal name: idmuser@LAB.EXAMPLE.NET
Principal alias: idmuser@LAB.EXAMPLE.NET
Email address: idmuser@example.com
UID: 546600020
GID: 546600020
Account disabled: False
-----
Number of entries returned 1
-----
```

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 directory services protocol and related services.

LDAP organizes objects, such as user accounts, and their related attributes, such as passwords and full name, in a tree. The attributes that are used on an object depend on the associated *object classes* with that object. In a directory, the same object classes and attributes are associated with all objects that represent the same thing, such as a user or a computer.

LDAP is an essential part of common identity management solutions, such as Red Hat Identity Management, Red Hat Directory Server, and Microsoft Active Directory.

What Is Kerberos?

Kerberos provides authentication services and a *Single Sign-On* (SSO) service to users and services. Kerberos assumes that individual machines on the network are trusted but that the network itself is insecure. Therefore, never transmit passwords across the network.

A typical Kerberos transaction involves three parties, where two parties who want to conduct transactions, but do not have a trusted relationship, use a third-party intermediary who is trusted by both parties to provide and verify a secure introduction. The *Kerberos Key Distribution Center* is the trusted third party for a user who wants to begin transactions with a service.

- The *Kerberos Key Distribution Center* (KDC). The KDC server knows the passwords for all users and services. A principal (a unique identifier for tickets), name, and password identify each user and service.
- The user who wants to authenticate. This user must know their password.
- The service that the user wants to authenticate with. Because services cannot interactively type their passwords, a file named *keytab* stores the service's password.

Initial authentication

When a user with a Kerberos principal configuration logs in, they are authenticated and provided a Ticket Granting Ticket (TGT). This TGT permits the user to obtain service tickets without having to enter their password again, for the duration of the TGT (typically a few hours). The TGT is obtained automatically when interactively logging in to a machine that uses Kerberos, or by using the `kinit` command-line tool.

Obtaining a TGT follows these steps:

1. At login, the system requests from the KDC a TGT for the user.
2. The KDC responds with a new TGT, which is encrypted with the user's password.
3. If the TGT can be decrypted with the user's typed password, then the login succeeds and the TGT is stored.

Authentication with a Ticket

While a user has a valid TGT, they can connect to other Kerberos-enabled services without typing a password.

Authentication happens according to these steps:

1. The client requests from the KDC the service ticket for the service that they want to access.
2. The KDC responds with two copies of the same service ticket: one is encrypted with the user's TGT, and the other is encrypted with the password for the service.
3. The client decrypts the TGT-encrypted version, and uses the decrypted ticket to encrypt a timestamp.
4. The client sends to the service the encrypted timestamp and the encrypted ticket with the service's password.
5. The service decrypts the service ticket, knowing that it can come only from the KDC, and uses it to decrypt the timestamp.

If the timestamp decryption succeeds and is less than two minutes old, then the user is considered to be authenticated.

A note on principals

In the simplest form, a user's principal name uses the `username@REALM` format, for example `vimes@example.com`. User names typically match an LDAP or local system user, and the realm typically maps to DNS domains as the organization defines.

Computer and service principals take a slightly different format, such as `service/hostname@REALM`. For example: `host/host.example.com@example.com`, `nfs/host.example.com@example.com`, or `http/www.example.com@example.com`. The service type determines the name of the service. Authentication services such as SSH and telnet use `host/`; NFS services use `nfs/`; and web services use `http/`. Other services often use unique prefixes.

The SSSD Client Access Daemon

The System Security Services Daemon (SSSD) is a system service to access remote directories and authentication mechanisms. It connects a local SSSD client system to an external back-end provider system for remote identity and authentication services. Example remote services include an LDAP directory, an Identity Management (IdM) or Active Directory (AD) domain, or a Kerberos realm.

When a user authenticates to a service, the SSSD daemon handles the authentication process:

- SSSD connects the client to an identity store to retrieve authentication information.
- SSSD uses the obtained authentication information to create a local cache of users and credentials on the client.

Users on the local system can then authenticate with the stored user accounts in the external back-end system. SSSD does not create user accounts on the local system. Instead, it uses the identities from the external data store and lets the users access the local system.



Figure 9.2: How SSSD works

SSSD can provide caches for several system services, such as Name Service Switch (NSS) or Pluggable Authentication Modules (PAM).

SSSD capabilities:

Reduce the load on identity and authentication servers

When requesting information, SSSD clients contact SSSD, which checks its cache. SSSD contacts the servers only if the information is not available in the cache.

Utilize offline authentication

SSSD optionally keeps a cache of user identities and credentials that are retrieved from remote services. In this setup, users can successfully authenticate to resources even if the remote server or the SSSD client is offline.

Implement single user accounts to improve consistency of the authentication process

With SSSD, you do not need to maintain a central account and a local user account for offline authentication.

Remote users often have multiple user accounts. For example, to connect to a virtual private network (VPN), remote users have one account for the local system and another for the VPN system.

Remote users can connect to network resources simply by authenticating to their local machine, thanks to caching and offline authentication. SSSD then maintains their network credentials.

Troubleshooting LDAP Issues

The most common LDAP issues can be categorized:

Network connectivity

LDAP uses TCP port 389 for both clear text and TLS-encrypted connections. Older LDAP versions supported TCP port 636 for SSL-encrypted connections. If a firewall is blocking port access, or if an older client attempts to connect to port 636/TCP on a server that supports only STARTTLS on port 389/TCP, then connections fail.

Mismatched security settings

LDAP servers can set restrictions on how clients connect and authenticate. Some servers might refuse to connect with clients that do not use TLS encryption, or might require either simple or SASL (Simple Authentication and Security Layer) authentication before responding to queries. You can use combinations of these requirements on individual actions. For example, a server can allow only TLS-encrypted connections for queries and require authentication for updates.

Wrong search base

Clients typically avoid searching the entire LDAP data tree, and limit queries to the branches that store the requested data. For example, a login program would query the part of the tree with user accounts, not the part with computer accounts. If the search base is incorrectly configured, then searches might be slow or not return the needed information.

Administrators can use the `getent password` command to troubleshoot user information issues.



Note

When using SSSD for user information and authentication, the `getent passwd` command lists only local users, not all available users. For testing, add the name of a known network account to test whether network users resolve correctly. For example, `getent passwd ldapuser`.

Troubleshooting Kerberos Issues

The two most common Kerberos issues are that users cannot obtain a TGT and fail to log in, and that users or services cannot authenticate to services.

To troubleshoot login issues, an administrator can verify basic configuration:

- Are the times synchronized between all parties? A time drift of more than two minutes causes all Kerberos validations to fail.
- Does a `kinit username` for that user work from the same machine? If `kinit` does work, then verify that the PAM configuration files are correctly configured. PAM issues are typically fixed by using the `authselect` family of tools. If `kinit` does not work, then verify that the KDC service is working, and the Kerberos configuration files on the machine.
- If DNS is used to store the Kerberos records, then are the SRV records configured correctly?
- The `/var/log/secure` log file and the `journalctl` command can provide messages regarding failed logins.

When users can log in, but cannot authenticate to certain services, or service-to-service authentication such as NFSv4 fails, additional configuration must be verified.

- Can the services read their keytab file? Services such as SSH and NFS use `/etc/krb5.keytab`, while other services might use custom keytab file names.
- 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 stored password. Whenever a principal is added to a keytab, a new, random password is generated automatically.

Users with access to the `kadmin` Kerberos KDC management tool (or `kadmin.local` for root on the KDC) can use the `getprinc <PRINCIPAL>` command from within `kadmin` to view the password version as recognized by the KDC.

- Are the correct principals present in the keytab?
- Do services agree on how to use Kerberos? For example, an NFS export that specifies `sec=krb5p` can be mounted only when the client also specifies the same security level.

- Are all needed helper services running? For example, an NFS server needs the `nfs-secure-server` service, and an NFS client needs the `nfs-secure` service when using Kerberos-enabled NFSv4.

Troubleshooting SSSD Issues

On an IdM client machine, SSSD stores a cache of user identities and credentials that are retrieved from the IdM server. SSSD also authenticates online against LDAP or Kerberos and applies an access and password policy to the user who attempts to log in to services. Troubleshooting SSSD issues might involve the IdM server, or the IdM client, or both.

Common command-line tools for troubleshooting SSSD issues include the `ssctl` command to obtain information about SSSD status, active servers, domains, and cached objects. Another useful tool is the `sss_cache` command, which invalidates records in SSSD cache to target specific trouble areas more easily.

Troubleshooting SSSD Issues on an IdM Server

If you experience issues when attempting to authenticate as an IdM user to an IdM server, then enable detailed debug logging in the SSSD service on the server. Then, gather logs of attempts to retrieve information about the user.

You need the `root` password to run the `ssctl` command and restart the SSSD service.

- Enable detailed SSSD debug logging on the IdM server.

```
[root@idmserver ~]# sssctl debug-level 6
```

- Invalidate objects in the SSSD cache for the user who is experiencing authentication issues. Invalidating the cache ensures that you do not bypass the LDAP server and retrieve information that SSSD already cached.

```
[root@idmserver ~]# sss_cache -u idmuser
```

- Remove older SSSD logs to minimize the troubleshooting data set.

```
[root@idmserver ~]# sssctl logs-remove
```

- Attempt to switch to a user with authentication problems. Use the `date` command to generate timestamps before and after the attempt. These timestamps narrow the scope of the data set.

```
[root@idmserver ~]# date; su idmuser; date
```

- Review SSSD logs for information about the failed request.

```
[root@idmserver ~]# cat /var/log/sssd/sssd_example.com.log
```

Troubleshooting SSSD Issues on an IdM Client

If you experience issues when attempting to authenticate as an IdM user to an IdM client:

- Verify that you can retrieve user information on the IdM server.

- Ensure that you have `root` user access to run the `ssctl` command and restart the SSSD service.
- On the IdM client, open the `/etc/sssd/sssd.conf` file in a text editor. Add the `ipa_server` option to the `[domain]` section of the file and set it to an IdM server. This step avoids the IdM client autodiscovering other IdM servers, thus limiting this test to one client and one server.

```
[domain/example.com]
ipa_server = server.example.com
```

- Restart the SSSD service to load the configuration changes.

```
[root@idmclient ~]# systemctl restart sssd
```

- Enable detailed SSSD debug logging.

```
[root@idmclient ~]# sssctl debug-level 6
```

- Invalidate objects in the SSSD cache for the user with authentication issues, so that you do not bypass the LDAP database and retrieve information that SSSD already cached.

```
[root@idmclient ~]# sss_cache -u idmuser
```

- Remove older SSSD logs to minimize the troubleshooting data set.

```
[root@idmclient ~]# sssctl logs-remove
```

- Attempt to switch to a user with authentication problems. Use the `date` command to generate timestamps before and after the attempt. These timestamps narrow the scope of the data set.

```
[root@idmclient ~]# date; su idmuser; date
```

- Review SSSD logs for information about the failed request.

```
[root@idmclient ~]# cat /var/log/sssd/sssd_example.com.log
```



References

`ssctl(8)`, `sss_cache(8)`, `sssd.conf(5)`, `krb5.conf(5)`, `ipa(1)`, and `kadmin(1)` manual pages.

For further information, refer to *Configuring and Managing Identity Management* guide at

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/configuring_and_managing_identity_management/index

► Guided Exercise

Resolving Identity Management Issues

In this lab, you resolve an issue related to LDAP user information and Kerberos authentication.

Outcomes

You should be able to diagnose and resolve issues with LDAP user information and Kerberos authentication.

Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

This command installs and configures the IdM client on servera for use in this exercise.

```
[student@workstation ~]$ lab start security-ldap
```

Instructions

The servera machine is a client to the IdM server for LDAP and Kerberos KDC services for user authentication. At your organization, remote users log in to the servera client machine with SSH keys instead of passwords. User home directories automatically mount at login.

The operator1 user reported that when remotely logging in to the servera machine, the connection prompts for a password and either times out after entering a password or immediately disconnects.

The system administrator provided you with the necessary information to troubleshoot the issue. Use RedHat123^ as the password for the operator1 and admin user accounts. Client systems and accounts are in the `lab.example.net` domain that the IdM server supports.

► 1. Re-create the issue.

- 1.1. From the workstation machine, attempt to log in to the operator1 account on the servera machine. The password prompt means that authentication is not working as expected and further investigation is necessary. Press `Ctrl+C` at any time to exit the authentication process.

The expected failure should look like this example:

```
[student@workstation ~]$ ssh operator1@servera
operator@servera's password: RedHat123^
Connection to servera closed by remote host.
Connection to servera closed.
[student@workstation ~]$
```

The failure might also appear as multiple password attempts until disconnection.

```
[student@workstation ~]$ ssh operator1@servera
Password: RedHat123^
Password: RedHat123^
Password: RedHat123^
operator1@servera's password: RedHat123^
Permission denied, please try again.
operator1@servera's password: RedHat123^
Received disconnect from 172.25.250.10 port 22:2: Too many authentication failures
Disconnected from 172.25.250.10 port 22
[student@workstation ~]$
```

- 1.2. Log in to the **servera** machine and switch to the **root** user.

The **student** account succeeds, which indicates that the IdM server is functioning, and that the issue might be limited to the **operator1** user.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

- 2. Start the troubleshooting process by verifying the IdM server's Kerberos functions.

- 2.1. Verify that the IdM service is providing Kerberos tickets.

Use the **kinit admin** command to authenticate and obtain Kerberos credentials. Use **RedHat123^** as the password.

```
[root@servera ~]# kinit admin
Password for admin@LAB.EXAMPLE.NET : RedHat123^
[root@servera ~]#
```

- 2.2. Verify that the Kerberos ticket is created.

The server's LDAP and Kerberos mechanisms appear to be functioning normally.

```
[root@servera ~]# klist
Default principal: admin@LAB.EXAMPLE.NET

Valid starting     Expires            Service principal
11/19/2021 00:12:27 11/19/2021 23:18:24  krbtgt/LAB.EXAMPLE.NET@LAB.EXAMPLE.NET
```

- 3. Next, diagnose SSSD operations.

Clear the SSSD log files on the **servera** machine and attempt to log in as the **operator1** user to generate fresh logs. Search the log files for possible errors.

- 3.1. Enable detailed SSSD debug logging.

```
[root@servera ~]# sssctl debug-level 6
```

- 3.2. Invalidate all objects in the SSSD cache. Invalidating cache objects ensures that you do not bypass the LDAP database and retrieve SSSD already cached information.

```
[root@servera ~]# sss_cache -E
```

- 3.3. Clear previous SSSD logs to minimize the troubleshooting data set.

```
[root@servera ~]# sssctl logs-remove
Truncating log files...
[root@servera ~]#
```

- 3.4. Confirm that the `/var/log/sssd/sssd_lab.example.net.log` file is empty, or nearly empty. Run the `sssctl logs-remove` command again if the file is larger than 1000 bytes.

```
[root@servera ~]# ls -l /var/log/sssd/
total 4
...output omitted...
-rw----- 1 root root 850 Nov 18 23:12 sssd_lab.example.net.log
...output omitted...
```

- 3.5. Attempt to switch to the `operator1` user account while gathering timestamps before and after the attempt. These timestamps narrow the scope of the data set.

```
[root@servera ~]# date; su operator1; date
Thu Nov 18 23:13:19 EST 2021
su: user operator1 does not exist
Thu Nov 18 23:13:19 EST 2021
```

- 3.6. Review the SSSD logs for information about a failed request.

In the following log file, a request is made for the `operator1` user with a data provider (DP) request type of `Account`. The search is based on a top-level domain component of `.com`, which is incorrect. The information from the system administrator indicates that the domain should be `lab.example.net`.

```
[root@servera ~]# cat /var/log/sssd/sssd_lab.example.net.log
...output omitted...
(2021-11-18 23:13:19): [be[lab.example.net]] [dp_get_account_info_send] (0x0200):
Got request for [0x1][BE_REQ_USER][name= operator1@lab.example.net ]
(2021-11-18 23:13:19): [be[lab.example.net]] [dp_attach_req] (0x0400): DP Request
[Account #8]: New request. Flags [0x0001].
(2021-11-18 23:13:19): [be[lab.example.net]] [dp_attach_req] (0x0400): Number of
active DP request: 1
(2021-11-18 23:13:19): [be[lab.example.net]] [sdap_search_user_next_base]
(0x0400): Searching for users with base [cn=accounts,dc=lab,dc=example,dc=com]
(2021-11-18 23:13:19): [be[lab.example.net]] [sdap_get_generic_ext_step] (0x0400):
calling ldap_search_ext with [(&(uid=operator1)(objectclass=posixAccount)(uid=*)
(&(uidNumber=\*)(!(uidNumber=0))))][cn=accounts,dc=lab,dc=example,dc=com].
(2021-11-18 23:13:19): [be[lab.example.net]] [sdap_get_generic_op_finished]
(0x0400): Search result: No such object(32), no errmsg set
```

```
(2021-11-18 23:13:19): [be[lab.example.net]] [sdap_search_user_process] (0x0400):  
  Search for users, returned 0 results.  
(2021-11-18 23:13:19): [be[lab.example.net]] [sysdb_search_by_name] (0x0400): No  
  such entry  
  ...output omitted...
```

▶ 4. Locate and fix the incorrect configuration.

4.1. Review the SSSD configuration.

Check the contents of the `/etc/sssd/sssd.conf` file for a possible error in data provider entries. All domain entries should be in the `lab.example.net` domain.

```
[root@servera ~]# cat /etc/sssd/sssd.conf  
[domain/lab.example.net]  
  
id_provider = ipa  
dns_discovery_domain = lab.example.com  
ipa_server = srv, utility.lab.example.com  
ipa_domain = lab.example.com  
ipa_hostname = servera.lab.example.com  
auth_provider = ipa  
chpass_provider = ipa  
access_provider = ipa  
cache_credentials = True  
ldap_tls_cacert = /etc/ipa/ca.crt  
krb5_store_password_if_offline = True  
[sssd]  
services = nss, pam, ssh, sudo  
  
domains = lab.example.net  
...output omitted...
```

4.2. Edit the incorrect values.

The `dns_discovery_domain` and `ipa_domain` key values are incorrect. These key values should represent the `.net` top-level domain component. Correct both entries and restart the `sssd` service.

The key values should match the following:

```
[root@servera ~]# cat /etc/sssd/sssd.conf  
...output omitted...  
dns_discovery_domain = lab.example.net  
...output omitted...  
ipa_domain = lab.example.net  
...output omitted...
```

4.3. Restart the `sssd` service to implement the new configuration.

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

▶ 5. Verify the `operator1` user by attempting to log in again.

When functioning correctly, the user is not prompted for a password and has an automatically mounted home directory.

- 5.1. Return to **workstation** as the student user.

```
[root@servera ~]# exit  
[student@servera ~]$ exit  
[student@workstation ~]$
```

- 5.2. Log in to the **servera** system as the **operator1** user.

```
[student@workstation ~]$ ssh operator1@servera  
...output omitted...  
[operator1@servera ~]$
```

- 5.3. Verify that the **operator1** user has a home directory on the **servera** machine.

If the working directory returned is the system's root directory, then the automatic home directory mount failed. The home directory appears correct.

```
[operator1@servera ~]$ pwd  
/home/operator1  
[operator1@servera ~]$
```

- 5.4. Return to **workstation** as the student user.

```
[operator1@servera ~]$ exit  
[student@workstation ~]$
```

Finish

On the **workstation** machine, use the **lab** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish security-ldap
```

This concludes the guided exercise.

► Lab

Troubleshooting Security Issues

In this lab, you resolve issues with one or more security subsystems.

Outcomes

You should be able to diagnose and fix issues with security subsystems.

Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

```
[student@workstation ~]$ lab start security-review
```

This command provides the necessary resources to perform this lab.

Instructions

The IdM server on the `utility` machine provides LDAP and Kerberos services for the `lab.example.net` domain. Users report that they cannot acquire a Ticket Granting Ticket (TGT) to access services. An administrator reports that the `servera` system was configured manually without using configuration management tools.

You are asked to locate the root cause and resolve this issue. When resolved, users on the `servera` system can acquire a TGT and use the `ipa` command to gather user information.

1. Log in to the `servera` machine and re-create the issue.
2. Investigate the issue to determine the root cause.
3. Resolve the issue and verify that the `kinit` and `ipa` commands work as expected.
4. Return to `workstation` as the student user.

Evaluation

On the `workstation` machine, use the `lab` command to grade your work. Correct any reported failures and rerun the script until you receive a passing grade.

```
[student@workstation ~]$ lab grade security-review
```

Finish

On the `workstation` machine, use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish security-review
```

This concludes the lab.

► Solution

Troubleshooting Security Issues

In this lab, you resolve issues with one or more security subsystems.

Outcomes

You should be able to diagnose and fix issues with security subsystems.

Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

```
[student@workstation ~]$ lab start security-review
```

This command provides the necessary resources to perform this lab.

Instructions

The IdM server on the utility machine provides LDAP and Kerberos services for the `lab.example.net` domain. Users report that they cannot acquire a Ticket Granting Ticket (TGT) to access services. An administrator reports that the `servera` system was configured manually without using configuration management tools.

You are asked to locate the root cause and resolve this issue. When resolved, users on the `servera` system can acquire a TGT and use the `ipa` command to gather user information.

1. Log in to the `servera` machine and re-create the issue.

- 1.1. Log in to `servera` and switch to the `root` user.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

- 1.2. Run the `kinit admin` command, with `RedHat123^` as the password, to attempt to acquire a TGT for the `admin` user.

```
[root@servera ~]# kinit admin
kinit: Cannot contact any KDC for realm 'LAB.EXAMPLE.NET' while getting initial
credentials
```

2. Investigate the issue to determine the root cause.

- 2.1. Check the `sssd` service logs for any failed request messages.

```
[root@servera ~]# grep failed /var/log/sssd/sssd_lab.example.net.log
...output omitted...
(2021-11-27 17:57:06): [be[lab.example.net]] [resolv_discover_srv_done] (0x0040):
SRV query failed [4]: Domain name not found
(2021-11-27 17:57:09): [be[lab.example.net]] [sssd_async_connect_done] (0x0020):
connect failed [113][No route to host].
(2021-11-27 17:57:09): [be[lab.example.net]] [sssd_async_socket_init_done]
(0x0020): sdap_async_sys_connect request failed: [113]: No route to host.
(2021-11-27 17:57:09): [be[lab.example.net]] [sss_ldap_init_sys_connect_done]
(0x0020): sssd_async_socket_init request failed: [113]: No route to host.
(2021-11-27 17:57:09): [be[lab.example.net]] [sdap_sys_connect_done] (0x0020):
sdap_async_connect_call request failed: [113]: No route to host.
```

- 2.2. The **utility** system hosts the IdM service that provides the LAB.EXAMPLE.NET realm. Verify that a DNS record for the **utility.lab.example.com** system exists. The DNS record appears to be correct. The **utility.lab.example.com** server is assigned 172.25.250.17 as an IP address.

```
[root@servera ~]# dig utility.lab.example.com
; <>> DiG 9.11.26-RedHat-9.11.26-4.el8_4 <>> utility.lab.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 40518
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;utility.lab.example.com. IN A

;; ANSWER SECTION:
utility.lab.example.com. 3600 IN A 172.25.250.17

;; Query time: 1 msec
;; SERVER: 172.25.250.254#53(172.25.250.254)
;; WHEN: Sat Nov 27 18:14:43 EST 2021
;; MSG SIZE rcvd: 68
```

- 2.3. Verify that the network interface on the **utility.lab.example.com** system is active. The **ping** command attempts to access a 172.25.250.177 IP address, which is incorrect.

```
[root@servera ~]# ping -c3 utility.lab.example.com
PING utility.lab.example.com (172.25.250.177) 56(84) bytes of data.
From servera.lab.example.com (172.25.250.10) icmp_seq=1 Destination Host
Unreachable
From servera.lab.example.com (172.25.250.10) icmp_seq=2 Destination Host
Unreachable
From servera.lab.example.com (172.25.250.10) icmp_seq=3 Destination Host
Unreachable
```

```
-- utility.lab.example.com ping statistics --
3 packets transmitted, 0 received, +3 errors, 100% packet loss, time 2024ms
pipe 2
```

- 2.4. View the `/etc/hosts` file on the `servera` machine for a possible conflicting record.
It appears that the `utility` IP address record was manually entered incorrectly.

```
[root@servera ~]# cat /etc/hosts
...output omitted...
172.25.250.254 bastion.lab.example.com bastion
172.25.250.9 workstation.lab.example.com workstation
172.25.250.10 servera.lab.example.com servera
172.25.250.11 serverb.lab.example.com serverb
172.25.250.177 utility.lab.example.com utility
```

3. Resolve the issue and verify that the `kinit` and `ipa` commands work as expected.
- 3.1. Edit the `/etc/hosts` file to add `172.25.250.17` as the correct IP address for the `utility.lab.example.com` machine.
The entry for the `utility.lab.example.com` machine should match the following output:

```
[root@servera ~]# cat /etc/hosts
...output omitted...
172.25.250.17 utility.lab.example.com utility
```

- 3.2. Restart the `sssd` service.

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

- 3.3. Run the `kinit admin` command, with `RedHat123^` as the password, to acquire a TGT for the `admin` user.

```
[root@servera ~]# kinit admin
Password for admin@LAB.EXAMPLE.NET : RedHat123^
```

- 3.4. Run the `ipa user-find admin` command to verify the `admin` user record in IdM.

```
[root@servera ~]# ipa user-find admin
-----
1 user matched
-----
User login: admin
Last name: Administrator
Home directory: /home/admin
Login shell: /bin/bash
Principal alias: admin@LAB.EXAMPLE.NET , root@LAB.EXAMPLE.NET
UID: 546600000
GID: 546600000
Account disabled: False
```

```
-----  
Number of entries returned 1  
-----
```

4. Return to workstation as the student user.

```
[root@servera ~]# exit  
[student@servera ~]$ exit  
[student@workstation ~]$
```

Evaluation

On the **workstation** machine, use the **lab** command to grade your work. Correct any reported failures and rerun the script until you receive a passing grade.

```
[student@workstation ~]$ lab grade security-review
```

Finish

On the **workstation** machine, use the **lab** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish security-review
```

This concludes the lab.

Summary

In this chapter, you learned:

- The `ausearch` command searches audit logs for specific events.
- The `semanage` command updates SELinux policies, including file and port context mappings.
- The `/etc/pam.d/` directory contains PAM configuration files.
- The `ssctl` command manages the SSSD service, including clearing logs and setting debug levels.
- The `/etc/sssd/sssd.conf` file is the primary configuration file for the SSSD service.
- Red Hat Identity Management (IdM) is a centralized framework that includes the LDAP and Kerberos services.

Chapter 10

Troubleshooting Kernel Issues

Goal

Identify kernel issues and assist Red Hat Support in resolving kernel issues.

Objectives

- Configure systems to enable kernel crash dumps.
- Compile SystemTap scripts into kernel modules for deployment on remote systems.

Sections

- Configuring Kernel Crash Dumps (and Guided Exercise)
- Kernel Debugging with SystemTap (and Guided Exercise)

Lab

Troubleshooting Kernel Issues

Configuring Kernel Crash Dumps

Objectives

After completing this section, you should be able to configure systems to enable kernel crash dumps.

Generating Kernel Crash Dumps

When an application crashes, the Linux kernel captures its memory image in a *core dump*. Core dumps contain the application's memory at the moment that it stopped working. Application vendors analyze core dumps to determine why an application crashed.

Similarly, when an operating system crashes, it captures the kernel's memory image in a *crash dump*. Operating system vendors analyze crash dumps to determine why a system crashed.

kdump and kexec

In Red Hat Enterprise Linux, the `kdump` service captures kernel crash dumps. The `kdump` service uses the `kexec` system call to boot a secondary Linux kernel. The secondary kernel is also known as the capture kernel. Without restarting the system, the capture kernel boots from a reserved memory area in the primary kernel. After booting, the capture kernel copies the primary kernel's memory image to a crash dump file.

Configuring kdump

By default, RHEL 8 installs the `kexec-tools` package, which provides the `kdump` service. The package provides command-line utilities to manage the `kdump` service. Alternatively, navigate to the **Crash Dump** tab of the web console to manage the `kdump` service in a graphical interface.

Memory Reservation

The capture kernel's reserved memory size depends on a system's architecture and on the total available physical memory. For `x86_64` architectures, the minimum reserved memory to capture dumps is 160 MB.

On most systems, the `kdump` service automatically calculates the required memory. To enable this feature, add the `crashkernel=auto` setting in the `GRUB_CMDLINE_LINUX` parameter of the `/etc/default/grub` configuration file.

```
GRUB_CMDLINE_LINUX="console=tty0 crashkernel=auto no_timer_check net.ifnames=0  
console=ttyS0,115200n8"
```



Note

The `crashkernel=auto` setting requires `x86_64` systems to have at least 1 GB of memory installed. Size requirements for ARM and IBM Power architectures vary. For more information, consult the references at the end of this section.

Chapter 10 | Troubleshooting Kernel Issues

If you modify the `/etc/default/grub` file, you must regenerate the GRUB2 configuration.

For systems that use BIOS firmware, use the following command.

```
[root@host ~]# grub2-mkconfig -o /boot/grub2/grub.cfg
```

For systems that use UEFI firmware, use the following command.

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

Reboot the system to implement the new amount of reserved memory.

The kdump Service

Enable and start the kdump service to generate crash dumps.

```
[root@host ~]# systemctl enable kdump
Created symlink from /etc/systemd/system/multi-user.target.wants/kdump.service
to /usr/lib/systemd/system/kdump.service.
[root@host ~]# systemctl start kdump
[root@host ~]# 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 2021-11-10 12:38:14 EST; 3s ago
    Process: 3482 ExecStart=/usr/bin/kdumpctl start (code=exited, status=0/SUCCESS)
  Main PID: 3482 (code=exited, status=0/SUCCESS)

Nov 10 12:38:13 host.lab.example.com systemd[1]: Starting Crash recovery kernel
arming...
Nov 10 12:38:14 host.lab.example.com kdumpctl[3482]: kdump: kexec: loaded kdump
kernel
Nov 10 12:38:14 host.lab.example.com kdumpctl[3482]: kdump: Starting kdump: [OK]
Nov 10 12:38:14 host.lab.example.com systemd[1]: Started Crash recovery kernel
arming.
```

Modify the `/etc/kdump.conf` configuration file to alter the behavior and collection settings of kernel crash dumps.

Designating Crash Dump Targets

By default, the kdump service stores crash dump files in the `/var/crash` directory.

```
[root@host ~]# 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 01:28 127.0.0.1-2021-11-09-13:11:30
[root@host ~]$ ls -la /var/crash/127.0.0.1-2021-11-09-13\:11\:30/
total 117964
drwxr-xr-x. 2 root root 67 Nov 9 13:11 .
drwxr-xr-x. 3 root root 43 Nov 9 13:11 ..
```

```
-rw-r--r--. 1 root root    41567 Nov  9 13:11 kexec-dmesg.log  
-rw-----. 1 root root 120707664 Nov  9 13:11 vmcore  
-rw-r--r--. 1 root root     39868 Nov  9 13:11 vmcore-dmesg.txt
```

The `vmcore` file contains the crash dump. The `vmcore-dmesg.txt` file contains the kernel log at the time of the crash.

Large crash dump files can be difficult to send quickly to Red Hat Support. To expedite the crash dump analysis, send the smaller `vmcore-dmesg.txt` file first for a preliminary assessment.

Modify the `path` option in the `/etc/kdump.conf` configuration file to change the crash dump directory.

```
path /var/crash
```

The `kdump` service offers crash dump targets other than local files. The following options are available in the `/etc/kdump.conf` configuration file.

/etc/kdump.conf Options for Configuring Dump Target

Option	Description
<code>raw [partition]</code>	Run the <code>dd</code> command to copy the crash dump to the specified partition.
<code>nfs [nfs share]</code>	Mount and copy the crash dump to the specified location in the <code>path</code> option on the NFS share.
<code>ssh [user@server]</code>	Run the <code>scp</code> command to transfer the crash dump to the location in the <code>path</code> option on the remote server with the specified user account for authentication.
<code>sshkey [sshkeypath]</code>	Used with the <code>ssh</code> crash dump type to specify the location of the SSH key to use for authentication.
<code>[fs type] [partition]</code>	Mount the specified partition with the specified file system type to the <code>/mnt</code> directory and write the crash dump to the specified path location in the <code>path</code> option.
<code>path [path]</code>	Specifies the path to save the crash dump to on the 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 `makedumpfile` utility generates kernel core dumps. The `core_collector` option in the `/etc/kdump.conf` configuration file modifies the collection parameters.

```
core_collector makedumpfile -l --message-level 1 -d 31
```

The `-c`, `-l`, and `-p` options change the compression algorithm of the core dump.

makedumpfile Compression Options

Option	Use for crash dump data compression
-c	zlib
-l	lzo
-p	snappy

Message levels filter message types in crash dumps. The previous example uses message level 1, which includes only a progress indicator in the crash output message. The following table lists some of the message levels.

makedumpfile Message Levels

Message level	Description
0	Do not include any messages.
1	Include only progress indicator.
4	Include only error messages.
31	Include all messages.

Dump levels filter page types in crash dumps. Dump levels can filter out zero pages, cached pages, user data pages, and free pages. Dump level filtering decreases 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 generates the smallest crash dump.

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 SSH dump targets, specify the scp utility in place of makedumpfile.

```
core_collector scp
```



Note

The full list of message and dump levels is in the `makedumpfile(8)` man page.

Using kdumpctl

The `kdumpctl` command, from the `kexec-tools` package, performs common `kdump` administration tasks.

```
[root@host ~]# kdumpctl -h
kdump: Usage: /bin/kdumpctl {start|stop|status|restart|reload|rebuild|propagate|
showmem}
```

The `kdumpctl status` command verifies the status of the `kdump` service.

```
[root@host ~]# kdumpctl status
kdump: Kdump is operational
```

The `kdumpctl showmem` command displays the current reserved memory for the capture kernel.

```
[root@host ~]# kdumpctl showmem
kdump: Reserved 192MB memory for crash kernel
```

The `kdumpctl propagate` command simplifies the setup of SSH crash dump targets. It determines from the `sshkey` parameter in the `/etc/kdump.conf` file which SSH key to use. If the key does not exist, then the `kdumpctl` utility automatically creates it. The `ssh-copy-id` command is then automatically invoked to copy the key to the target SSH server.

```
[root@host ~]# 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 'server.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@server.lab.example.com 's password: redhat

Number of key(s) added: 1

Now try logging into the machine, with: "ssh ' root@server.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
server.lab.example.com
```



Note

The Kdump Helper lab generates a script to automatically configure `kdump` targets. The lab is available on the Red Hat Customer Portal at <https://access.redhat.com/labs/kdumphelper/>.

Kernel Crash Dump Triggers

Systems generate crash dumps when their kernel encounters an unrecoverable error. By default, any other type of error does not generate a crash dump. Administrators can enable crash dumps to troubleshoot specific errors.

OOM Events

When a system runs out of memory, the `oom killer` process kills other processes to free up system memory and to keep the system operational. In many use cases, using the `oom killer` is preferred to triggering a panic because killing targeted processes attempts to kill the process that causes the memory issue while protecting other critical or long-running processes from crashing.

However, configuring a system to panic instead is appropriate when confidence in the overall system memory stability might be uncertain, and the recommended recovery method is to immediately halt possible memory data corruption and reboot the system cleanly.

The following command temporarily configures a system to panic on OOM-killer events:

```
[root@host ~]# echo 1 > /proc/sys/vm/panic_on_oom
```

To make the configuration permanent, use the following commands:

```
[root@host ~]# echo "vm.panic_on_oom=1" >> /etc/sysctl.conf  
[root@host ~]# sysctl -p
```

Hung Processes

Applications sometimes experience bugs and their processes appear to hang.

The following command temporarily configures a system to panic when processes hang for longer than a specific timeout value:

```
[root@host ~]# echo 1 > /proc/sys/kernel/hung_task_panic
```

To make the configuration permanent, use the following commands:

```
[root@host ~]# echo "kernel.hung_task_panic=1" >> /etc/sysctl.conf  
[root@host ~]# sysctl -p
```

The default timeout value is 120 seconds. The timeout value is configured in the `/proc/sys/kernel/hung_task_timeout_secs` file.

```
[root@host ~]# cat /proc/sys/kernel/hung_task_timeout_secs  
120
```

Soft Lockup

Soft lockups occur when a task is executing in kernel space on a CPU without rescheduling.

The following command temporarily configures a system to panic when soft lockups occur:

```
[root@host ~]# echo 1 > /proc/sys/kernel/softlockup_panic
```

To make the configuration permanent, use the following commands:

```
[root@host ~]# echo "kernel.softlockup_panic=1" >> /etc/sysctl.conf  
[root@host ~]# sysctl -p
```

**Note**

Do not enable the `softlockup_panic` or `nmi_watchdog` kernel parameters on a virtualized RHEL 8 machine. The virtualized environment might trigger inauthentic soft lockups that rarely require a system panic.

Non-Maskable Interrupts

A non-maskable interrupt (NMI) usually occurs when a system detects a critical hardware error. NMIs are automatically generated by the NMI Watchdog program, if it is enabled. NMIs are manually generated by pressing the physical NMI button on system hardware or a virtual NMI button from the system's out-of-band management interface, such as HP's iLO or Dell's iDRAC.

The following command temporarily configures a system to panic when NMIs are detected:

```
[root@host ~]# echo 1 > /proc/sys/kernel/panic_on_io_nmi
```

To make the configuration permanent, use the following commands:

```
[root@host ~]# echo "kernel.panic_on_io_nmi=1" >> /etc/sysctl.conf  
[root@host ~]# sysctl -p
```

Magic SysRq

The "Magic" SysRq key is a key sequence to diagnose an unresponsive system. The following command temporarily enables the SysRq key:

```
[root@host ~]# echo 1 > /proc/sys/kernel/sysrq
```

To make the configuration permanent, use the following commands:

```
[root@host ~]# echo "kernel.sysrq=1" >> /etc/sysctl.conf  
[root@host ~]# sysctl -p
```

When enabled, certain SysRq commands trigger system events. Use the **Alt +PrintScreen+_[CommandKey]** key sequence to enter SysRq commands. 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.

SysRq command	Event
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.

Alternatively, issue SysRq commands by writing their associated key characters to the `/proc/sysrq-trigger` file. For example, the following command initiates a system crash.

```
[root@host ~]# echo 'c' > /proc/sysrq-trigger
```

The `c` character is often used to test system crash dumps.

The early kdump Feature

`early kdump` is a feature of the kdump mechanism to capture the core dump of a booting kernel. In earlier versions than Red Hat Enterprise Linux 8, the kdump service starts later in the boot sequence, typically alongside system services such as `sshd`. This delayed start prevents kdump from capturing core dumps if a system crashes before system services start. In Red Hat Enterprise Linux 8, `early kdump` starts the kdump service earlier in the boot sequence, and creates core dumps even if the system crashes before system services start.

The follow commands enable the `early kdump` feature.

1. Ensure that a `kdump initramfs` image exists for the current kernel.

```
[root@host ~]# ls /boot/initramfs-`uname -r`kdump.img
/boot/initramfs-4.18.0-305.el8.x86_64kdump.img
```

2. Rebuild the `initramfs` of the booting kernel with `early kdump` support.

```
[root@host ~]# dracut -f --add earlykdump
```

3. Append the `rd.earlykdump` kernel boot parameter to the `kernelopts` line in `grub`.

```
[root@host ~]# grub2-editenv - set "kernelopts=root=/dev/mapper/rhel-root
ro crashkernel=auto r esume=/dev/mapper/rhel-swap rd.lvm.lv=rhel/root
rd.lvm.lv=rhel/swap console=tty0 console=ttyS0,115200 rd.earlykdump
```

4. To implement the changes, reboot the system.

```
[root@host ~]# reboot
```

Analyzing Kernel Crash Dumps

Analyzing a crash dump is complex and requires knowledge of the Linux kernel. Specific tools help administrators to analyze high-level crash dump information.

Preparing a System for Crash Dump Analysis

To analyze a crash dump, install the following packages:

- The `kernel-debuginfo` package that matches the version of the kernel where the dump was created. This information is in the `vmcore-dmesg.txt` file that is stored alongside the kernel crash dump, or by running the `strings` command against the `vmcore` file.

```
[root@host]# strings vmcore | head
KDUMP
Linux
host.example.com
4.18.0-305.el8.x86_64
#1 SMP Thu Apr 29 08:54:30 EDT 2021
...output omitted...
```

- The `crash` package.

Analyzing Crash Dumps

The `crash` command requires two parameters: the debug version of the kernel image (from the `kernel-debuginfo` package), and the kernel crash dump `vmcore` file. If the `vmcore` file is omitted, then the `crash` session runs against the currently running kernel.

```
[root@host ~]# crash /usr/lib/debug/lib/modules/4.18.0-305.el8.x86_64/vmlinu
/var/crash/127.0.0.1-2021-11-09-13:11:30/vmcore
```

The `crash` prompt offers various useful commands.

- `files <PID>`: Shows the open files for the specified process.
- `ps`: Lists every processes that was running at the time of the crash.
- `fuser <PATHNAME>`: Displays which processes were using a certain file or directory.

The `help` command displays command usage information. The `exit` command quits the `crash` prompt.



References

Installing and Configuring kdump

[https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html/system_design_guide/installing-and-configuring-kdump_system-design-guide_kdump\(8\)_kexec\(8\)_grub-mkconfig\(1\)_kdump.conf\(5\)_and_makedumpfile\(8\)_manual_pages](https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html/system_design_guide/installing-and-configuring-kdump_system-design-guide_kdump(8)_kexec(8)_grub-mkconfig(1)_kdump.conf(5)_and_makedumpfile(8)_manual_pages)

► Guided Exercise

Configuring Kernel Crash Dumps

In this exercise, you configure and test kernel crash dumps.

Outcomes

You should be able to configure the kdump service to send a crash dump to a remote server.

Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

```
[student@workstation ~]# lab start kernel-crashdump
```

This command confirms that your systems are reachable.

Instructions

You are tasked to configure the `servera` system to send crash dumps to `serverb`. Use the `scp` command to send crash dumps. Store crash dumps in the `/var/crash` directory on `serverb`.

- 1. Log in to `servera` and switch to the `root` user.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

- 2. Enable and start the `kdump` service.

```
[root@servera ~]# systemctl enable --now kdump
Created symlink from /etc/systemd/system/multi-user.target.wants/kdump.service
to /usr/lib/systemd/system/kdump.service.
[root@servera ~]# systemctl status kdump
● kdump.service - Crash recovery kernel arming
  Loaded: loaded (/usr/lib/systemd/system/kdump.service; disabled; vendor preset:
    enabled)
  Active: active (exited) since Wed 2021-11-10 16:26:46 EST; 3s ago
    Process: 1439 ExecStart=/usr/bin/kdumpctl start (code=exited, status=0/SUCCESS)
   Main PID: 1439 (code=exited, status=0/SUCCESS)
```

```
Nov 10 16:26:45 servera.lab.example.com systemd[1]: Starting Crash recovery kernel arming...
Nov 10 16:26:46 servera.lab.example.com kdumpctl[1439]: kdump: kexec: loaded kdump kernel
Nov 10 16:26:46 servera.lab.example.com kdumpctl[1439]: kdump: Starting kdump: [OK]
Nov 10 16:26:46 servera.lab.example.com systemd[1]: Started Crash recovery kernel arming.
```

- 3. Modify the `/etc/kdump.conf` file to send crash dumps to the `/var/kdump` directory on `serverb`.

- 3.1. Set `serverb.lab.example.com` as the SSH target.

```
ssh root@serverb.lab.example.com
```

- 3.2. Set `/root/.ssh/kdump_id_rsa` as the SSH key.

```
sshkey /root/.ssh/kdump_id_rsa
```

- 3.3. Modify the `core_collector` line to use `scp`. Use this line to replace any existing `core_collector` lines.

```
core_collector scp
```

- 3.4. Set the `path` parameter to the `/var/crash` directory.

```
path /var/crash
```

- 4. Generate and remotely install the SSH key on `serverb`. If prompted, enter `redhat` as the password.

```
[root@servera ~]# kdumpctl propagate
kdump: WARNING: '/root/.ssh/kdump_id_rsa' doesn't exist, using default value '/root/.ssh/kdump_id_rsa'
kdump: Generating new ssh keys...
kdump: done.
/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/root/.ssh/kdump_id_rsa.pub"
The authenticity of host 'serverb.lab.example.com (172.25.250.11)' can't be established.
ECDSA key fingerprint is SHA256:NJAYJMx8B2AeIYHRnVLAUJ1XZwbloomyOKowyfTwGrTY.
Are you sure you want to continue connecting (yes/no/[fingerprint])? 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@serverb.lab.example.com 's password: redhat

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'root@serverb.lab.example.com'"
```

and check to make sure that only the key(s) you wanted were added.

```
kdump: /root/.ssh/kdump_id_rsa has been added to ~root/.ssh/authorized_keys on serverb.lab.example.com
```

- 5. Restart the kdump service to implement the configuration file changes.

```
[root@servera ~]# systemctl restart kdump  
[root@servera ~]# kdumpctl status  
Kdump: Kdump is operational
```

- 6. Trigger a system crash to test the kdump configuration.

- 6.1. Enable SysRq functions by setting the value in the /proc/sys/kernel/sysrq file to 1.

```
[root@servera ~]# echo 1 > /proc/sys/kernel/sysrq
```

- 6.2. Trigger a system crash by setting the value in the /proc/sysrq-trigger file to c.

```
[root@servera ~]# echo c > /proc/sysrq-trigger
```



Note

The SSH session hangs when the system crash is triggered. Press any key to return to the workstation system.

- 7. Log in to serverb and switch to the root user.

```
[student@workstation ~]$ ssh student@serverb  
...output omitted...  
[student@serverb ~]$ sudo -i  
[sudo] password for student: student  
[root@serverb ~]#
```

- 8. Verify that the crash dump is present in the /var/kdump directory and view its contents.

- 8.1. List the contents of the /var/crash directory.

```
[root@serverb ~]# ll /var/crash  
total 0  
drwxr-xr-x. 2 root root 67 Nov 10 16:44 172.25.250.10-2021-11-10-16:43:54
```

- 8.2. View the contents of the crash dump.

```
[root@serverb ~]# ll /var/crash/172.25.250.10-2021-11-10-16\:43\:54/  
total 1977448  
-rw-r--r--. 1 root root 52465 Nov 10 16:44 kexec-dmesg.log  
-r-----. 1 root root 2024812544 Nov 10 16:44 vmcore  
-rw-r--r--. 1 root root 39176 Nov 10 16:44 vmcore-dmesg.txt
```

- 9. Return to workstation as the student user.

```
[root@serverb ~]# exit  
[student@serverb ~]$ exit  
[student@workstation ~]$
```

Finish

On the **workstation** machine, use the **lab** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises. Wait for **servera** to finish rebooting before executing the command.

```
[student@workstation ~]$ lab finish kernel-crashdump
```

This concludes the guided exercise.

Kernel Debugging with SystemTap

Objectives

After completing this section, you should be able to compile SystemTap scripts into kernel modules for deployment on remote systems.

Introduction to SystemTap Framework

The SystemTap tool gathers information for tracing and probing to monitor and analyze activities on a running Linux kernel. SystemTap collects information on performance or functional issues. SystemTap helps developers and system administrators to gather debug, profile, and performance data without requiring the installation or use of instrumented kernels and other specialized tool packages.

SystemTap provides options to analyze and filter collected data similar to capabilities of the `netstat`, `ps`, `top`, and `iostat` tools. The `systemtap` package includes numerous example scripts, and supports custom, user-developed scripts.

The `stap` command processes a SystemTap script, which is referred to as a *probe* file, by translating the probe into C language and compiling it as a kernel module. SystemTap inserts this kernel module into the running kernel for analysis and performs the probe functions as defined in the script. You can view the probe output when the probe runs or redirects to a file.

Installing the SystemTap Tool

The `systemtap` package provides the SystemTap framework. Probes must be compiled with the identical RHEL kernel version to the system where the probe is inserted to monitor and analyze kernel activity. A probe that is compiled on a different kernel is not guaranteed to run properly on other kernel versions, similar to any other C application.

Red Hat recommends that you do not normally compile SystemTap probes on the production systems to be analyzed, because these probes are a security vulnerability and can affect performance. Instead, use a development system for compiling SystemTap modules, and then transfer the compiled module to the target system. To use a SystemTap probe on more than one kernel, the probe must be recompiled for each target kernel version.

The compiling machine must be running the same kernel version as the system to be analyzed. Although it is possible to install multiple kernels on a single system, and boot into the chosen kernel version before compiling, the recommended practice is to create virtual machines on your development system, each with one of the target kernel versions. Install SystemTap and the version-specific kernel packages into each of these unique virtual machines.

The SystemTap framework requires the `kernel-debuginfo`, `kernel-debuginfo-common`, and `kernel-devel` packages that match the kernel version to be used for compiling. The C compiler and related developer tools are required and are installed as SystemTap dependencies if they are not already available on the development system. Most of the required packages are distributed in the RHEL 8 base distribution. The `kernel-debuginfo` and `kernel-debuginfo-common` packages are provided by the debug and source repositories.

To install all of these packages, enable the debug repositories with the following commands:

```
[root@host ~]# subscription-manager repos --enable rhel-8-for-$uname
-i)-baseos-debug-rpms
[root@host ~]# subscription-manager repos --enable rhel-8-for-$uname
-i)-baseos-source-rpms
[root@host ~]# subscription-manager repos --enable rhel-8-for-$uname
-i)-appstream-debug-rpms
[root@host ~]# subscription-manager repos --enable rhel-8-for-$uname
-i)-appstream-source-rpms
```

The `stap-prep` command from the `systemtap` package simplifies the tool installation process. Because the `stap-prep` command installs the matching `kernel-devel` and `kernel-debuginfo` packages for the currently running kernel, verify that you are booted into the needed target kernel version before running the `stap-prep` command. The SystemTap installation installs the necessary C compiler, libraries, build tools, kernel source code, and Perl tools if not already installed.

Running SystemTap Scripts

The `systemtap` package provides various useful example scripts for gathering data on the running kernel. These scripts are stored in `/usr/share/systemtap/examples` and are organized by the type of probed information.

By convention, SystemTap scripts use an `.stp` extension.

The `stap` command compiles and runs SystemTap scripts, including the provided examples. The `stap` command performs the following actions:

1. Parse the SystemTap script to validate syntax.
2. Elaborate the script to resolve and include symbols, variables, functions, and aliases.
3. Translate the script into C and save it to a temporary file.
4. Compile the C code into a kernel module object file.
5. Load and run the kernel module, trace the output, and unload when exited.

When run without options, the `stap` command runs the `syscalls_by_proc.stp` SystemTap example script, gathering data until the probe is interrupted and the summary results are displayed.

```
[root@host ~]# stap /usr/share/systemtap/examples/process/syscalls_by_proc.stp
Collecting data... Type Ctrl-C to exit and display results
#SysCalls Process Name
...output omitted...
12        crond
7         rpcbind
5         stap
```

Running `stap` with the `-v` option displays the multiple passes that the `stap` command makes to run the probe.

```
[root@host ~]# stap -v /usr/share/systemtap/examples/process/syscalls_by_proc.stp
Pass 1: parsed user script and 482 library scripts using
456140virt/88756res/12612shr/75596data kb, in 180usr/50sys/239real ms. ①
Pass 2: analyzed script: 4 probes, 2 functions, 95 embeds, 5 globals using
464380virt/98008res/13668shr/83836data kb, in 220usr/200sys/432real ms. ②
Pass 3: using cached /root/.systemtap/cache/5a/
stap_5adeba4dc66f80ef54ba94400fc572d_133889.c ③
Pass 4: using cached /root/.systemtap/cache/5a/
stap_5adeba4dc66f80ef54ba94400fc572d_133889.ko ④
Pass 5: starting run. ⑤
Collecting data... Type Ctrl-C to exit and display results
#SysCalls Process Name
...output omitted...
2           lsmd
2           rs:main Q:Reg
1           sssd
Pass 5: run completed in 10usr/30sys/12362real ms.
```

- ① Pass 1 parses the script and verifies that the code is syntactically consistent.
- ② Pass 2 elaborates the script to resolve symbols, variables, functions, and aliases.
- ③ Pass 3 translates the program to C code, creates a build Makefile, and places these files in a temporary cache directory. If this probe was run previously and the script was not modified since, then the previously translated C object and Makefile are retrieved from the cache and reused, as in this example.
- ④ Pass 4 builds the kernel object file that uses the C object, the Makefile, and the system's C compiler tools. If the kernel object exists in the cache and the script is not modified, then the previously built kernel object is retrieved from the cache and reused, as in this example.
- ⑤ Pass 5 invokes the `staprund` program with the kernel object to load the kernel module, communicate with it, and copy the generated trace data until the user sends an interrupt signal.

Compiling SystemTap Programs to Portable Kernel Modules

Pass 4 of the `stap` command compiles a kernel module with a unique and random name, and stores it in the hidden cache directory. Unless an absolute path is specified, the named module is created in the user's current working directory. Use the `-m` option to name the module, which is preferred when the module will be used repetitively and copied to target systems.

Kernel module names must contain only the following characters:

- ASCII lowercase alphabetic characters (a-z)
- Digits (0-9)
- Underscores (_)

When compiling on a developer system or virtual machine SystemTap scripts that are intended to be run on a different target system, use the `-p 4` option to stop after completing pass 4 and before the kernel module is loaded to be run.

This example runs the `syscalls_by_proc.stp` example script, and stops after compiling the kernel module. The manually named `syscalls_by_proc.ko` file is created in the `/root/` directory, which is the current working directory.

```
[root@host ~]# stap -p 4 -v -m
  syscalls_by_proc /usr/share/systemtap/examples/process/syscalls_by_proc.stp
Pass 1: parsed user script and 482 library scripts using
  456012virt/88844res/12928shr/75468data kb, in 190usr/40sys/232real ms.
Pass 2: analyzed script: 4 probes, 2 functions, 95 embeds, 5 globals using
  464368virt/98252res/13924shr/83824data kb, in 73080usr/13570sys/44902real ms.
Pass 3: translated to C into "/tmp/stapl1x3aE/syscalls_by_proc_src.c" using
  464368virt/98252res/13924shr/83824data kb, in 20usr/10sys/24real ms.
  syscalls_by_proc.ko
Pass 4: compiled C into "syscalls_by_proc.ko" in 18520usr/3340sys/14244real ms.
```

Running SystemTap Programs as Non-Root User

On target systems, install the `systemtap-runtime` package to obtain the `staprund` command without the additional compile and build tools. Use the `staprund` command to run previously compiled modules. Copy the kernel module to the target system, placing the module into the system-wide kernel modules directory for access by general users with 'stapusr' privileges.

Developers with the `stapdev` privileges, and the `root` user, can run the module from any directory location.

SystemTap group	Description
<code>stapusr</code>	User can run only the SystemTap instrumentation modules that are stored in the default <code>/lib/modules/\$(uname -r)/systemtap</code> kernel module directory. The module files must have <code>root</code> ownership, and <code>root</code> must have write permissions on the <code>/lib/modules/\$(uname -r)/systemtap</code> directory. Users who are assigned to the <code>stapusr</code> group can use only the <code>staprund</code> command, and do not have permission to use the <code>stap</code> command for compiling SystemTap scripts.
<code>stapdev</code>	Developers with <code>stapdev</code> group membership can compile their own SystemTap kernel modules with the <code>stap</code> command. If the user also has <code>stapusr</code> group membership, then they can use the <code>staprund</code> command to load a module from any directory.

In this example, a SystemTap kernel module that is created as `/root/syscalls_by_proc.ko` on the target system is prepared for use by copying the file to the default kernel modules directory. Files that are stored in the default location can be run by the module name, which matches the file name without the `.ko` extension.

```
[root@host ~]# mkdir -p /lib/modules/$(uname -r)/systemtap
[root@host ~]# ls -ld /lib/modules/$(uname -r)/systemtap
drwxr-xr-x. 2 root root 33 Nov 20 22:33 /lib/modules/4.18.0-305.el8.x86_64/
systemtap
[root@host ~]# cp /root/syscalls_by_proc.ko /lib/modules/$(uname -r)/systemtap
```

Any user with `stapusr`, `stapdev`, or `root` privileges can run the module by using the `staprund` command with the module name.

```
[root@host ~]# staprun syscalls_by_proc
Collecting data... Type Ctrl-C to exit and display results
#SysCalls Process Name
...output omitted...
2 gmain
2 lsmd
1 rpcbind
```

Alternatively, a developer with `stapdev` or `root` privileges can run a module by using the absolute or relative path to the target module file in any writable and accessible directory. However, modules that are run this way are not located by module name, but by the file name, which includes the `.ko` extension. The user must have write permission in the directory with the module file for SystemTap to capture the trace data in temporary files.

Kernel Debugging

The `ftrace` framework uses virtual files in the `debugfs` file system, and enables specific tracers. The `ftrace` function tracer displays each function that is called by the kernel in real-time; other tracers within the `ftrace` framework can analyze wakeup latency, task switches, and kernel events. You can also add new tracers for `ftrace`, to make it a flexible solution for analyzing kernel events.

The `ftrace` framework helps to debug or analyze latencies and performance issues outside userspace. The interface for `ftrace` resides in the `debugfs` file system in the '`tracing`' directory.

To view the kernel config file for the `ftrace` command:

```
[root@host ~]# grep TRACER /boot/config-4.18.0-305.el8.x86_64
CONFIG_NOP_TRACER=y
CONFIG_HAVE_FUNCTION_TRACER=y
CONFIG_HAVE_FUNCTION_GRAPH_TRACER=y
CONFIG_TRACER_MAX_TRACE=y
CONFIG_CONTEXT_SWITCH_TRACER=y
CONFIG_GENERIC_TRACER=y
CONFIG_FUNCTION_TRACER=y
CONFIG_FUNCTION_GRAPH_TRACER=y
# CONFIG_IRQSOFF_TRACER is not set
CONFIG_SCHED_TRACER=y
CONFIG_HWLAT_TRACER=y
CONFIG_TRACER_SNAPSHOT=y
# CONFIG_TRACER_SNAPSHOT_PER_CPU_SWAP is not set
CONFIG_STACK_TRACER=y
```

To verify whether the `debugfs` file system is mounted on the system:

```
[root@host ~]# findmnt -t debugfs
TARGET SOURCE FSTYPE OPTIONS
/sys/kernel/debug debugfs debugfs rw,relatime,seclabel
[root@host ~]# mount | grep ^debugfs
debugfs on /sys/kernel/debug type debugfs (rw,relatime,seclabel)
```

To view the `available_tracers` file and check all the available tracers:

```
[root@host ~]# cat /sys/kernel/debug/tracing/available_tracers
hwlat blk function_graph wakeup_dl wakeup_rt wakeup function nop
```

To change the current tracer and use the function tracer:

```
[root@host ~]# echo function > /sys/kernel/debug/tracing/current_tracer
```

To view the current_tracer file and check the current tracer:

```
[root@host ~]# cat /sys/kernel/debug/tracing/current_tracer
function
```

To enable tracing:

```
[root@host ~]# echo 1 > /sys/kernel/debug/tracing/tracing_on
[root@host ~]# cat /sys/kernel/debug/tracing/tracing_on
1
```

To check the trace file and check the trace buffer:

```
[root@host ~]# head -n 20 /sys/kernel/debug/tracing/trace
# tracer: function
#
#                                     _----=> irqs-off
#                                     / _----=> need-resched
#                                     | / _---=> hardirq/softirq
#                                     || / _--=> preempt-depth
#                                     ||| /     delay
#           TASK-PID    CPU#  ||||   TIMESTAMP  FUNCTION
#           | |        |  ||||   |       |
kworker/0:1-25346 [000] .... 3031.749852: unlock_page <-
shmem_read_mapping_page_gfp
kworker/0:1-25346 [000] .... 3031.749852: shmem_read_mapping_page_gfp <-
drm_gem_get_pages
kworker/0:1-25346 [000] .... 3031.749852: shmem_getpage_gfp <-
shmem_read_mapping_page_gfp
kworker/0:1-25346 [000] .... 3031.749852: find_lock_entry <-
shmem_getpage_gfp
kworker/0:1-25346 [000] .... 3031.749852: find_get_entry <-find_lock_entry
kworker/0:1-25346 [000] .... 3031.749852: PageHuge <-find_get_entry
kworker/0:1-25346 [000] .... 3031.749852: __cond_resched <-find_lock_entry
kworker/0:1-25346 [000] .... 3031.749852: rcu_all_qs <-__cond_resched
kworker/0:1-25346 [000] .... 3031.749852: page_mapping <-find_lock_entry
kworker/0:1-25346 [000] .... 3031.749852: unlock_page <-
shmem_read_mapping_page_gfp
kworker/0:1-25346 [000] .... 3031.749852: shmem_read_mapping_page_gfp <-
drm_gem_get_pages
```

Describing the trace-cmd Kernel Debugging Tool

The `trace-cmd` tool is a front-end tool to the `ftrace` tool, from the `trace-cmd` package. The `trace-cmd` tool enables the `ftrace` tool interactions without writing to the `/sys/kernel/debug/tracing/` directory.



References

`stap(1)`, `staprun(8)`, `gdb(1)`, `strace(1)`, `ltrace(1)`, and `trace-cmd(1)` man pages

For further information, refer to *Chapter 37. Getting Started with Systemtap* at
https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html/monitoring_and_managing_system_status_and_performance/getting-started-with-systemtap_monitoring-and-managing-system-status-and-performance

For further information, refer to *What Is Ftrace and How Do I Use It?* at
<https://access.redhat.com/solutions/221323>

For further information, refer to *Chapter 7. GNU Debugger (GDB)* at
https://access.redhat.com/documentation/en-us/red_hat_developer_toolset/8/html/user_guide/chap-gdb

For futher information, refer to *Chapter 8. strace* at
https://access.redhat.com/documentation/en-us/red_hat_developer_toolset/8/html/user_guide/chap-strace

► Guided Exercise

Kernel Debugging with SystemTap

In this lab, you compile and distribute a System Tap module.

Outcomes

You should be able to compile a SystemTap program and execute it on another system.

Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

```
[student@workstation ~]$ lab start kernel-debug
```

This command installs kernel debug packages and executes a script to increase disk I/O traffic on the `/dev/vdb` disk on the `serverb` system.

Instructions

A developer reports poor system performance on the `serverb` system. An administrator adds that recent diagnostics point to high system disk activities. The `iostat` utility reports heavy activity on the system's `/dev/vdb` secondary disk.

You are asked to determine which process is responsible for the heavy disk activity. Create and run a SystemTap probe that locates the root cause. Use the `servera` system to build the `Systemtap` probe. Enable the `student` user to reuse this probe in the future.

- 1. Log in to the `servera` system and switch to the `root` user.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

- 2. Install the necessary SystemTap packages, compile tools, and debug dependencies on the `servera` system. Locate a suitable example script to use for this issue.

- 2.1. Install the `systemtap` package on the `servera` system.

The C compiler, libraries, build tools, kernel source code, and Perl tools are also pulled in as dependencies and installed.

```
[root@servera ~]# yum install systemtap
...output omitted...
Complete!
```

- 2.2. Verify that the `kernel-debuginfo` and `kernel-devel` packages are installed.

```
[root@servera ~]# rpm -q kernel-debuginfo kernel-devel
kernel-debuginfo-4.18.0-305.el8.x86_64
kernel-devel-4.18.0-305.el8.x86_64
```

- 2.3. List the available I/O example scripts from the `systemtap` package.

```
[root@servera ~]# ls -la /usr/share/systemtap/examples/io/
...output omitted...
-rw-r--r-- 1 root root 410 Mar 11 2021 iotop.meta
-rwxr-xr-x 1 root root 624 Mar 11 2021 iotop.stp
...output omitted...
```

- 2.4. View the contents of the `iotop.stp` script file. This script displays the top 10 I/O processes every 5 seconds.

```
[root@servera ~]# cat /usr/share/systemtap/examples/io/iotop.stp
#!/usr/bin/stap

global reads, writes, total_io

...output omitted...

# print top 10 IO processes every 5 seconds
probe timer.s(5) {
    printf ("%16s\t%10s\t%10s\n", "Process", "KB Read", "KB Written")
    foreach (name in total_io @sum- limit 10)
        printf("%16s\t%10d\t%10d\n", name,
               @sum(reads[name])/1024, @sum(writes[name])/1024)
    delete reads
    delete writes
    delete total_io
    print("\n")
}
```

- 3. Prepare the `iotop.stp` script to analyze the `serverb` system, and copy the kernel module to the `serverb` system.
- 3.1. Compile the `iotop.stp` script file to generate the `iotop.ko` portable kernel module.
Ignore any warning messages after the `iotop` kernel module completes the Pass 1 step.

```
root@servera ~]# stap -v -p 4 -m iotop /usr/share/systemtap/examples/io/iotop.stp
Pass 1: parsed user script and 482 library scripts using
456012virt/88412res/12504shr/75468data kb, in 200usr/40sys/237real ms.
...output omitted...
Pass 2: analyzed script: 5 probes, 7 functions, 7 embeds, 35 globals using
688652virt/322008res/13676shr/308108data kb, in 3420usr/980sys/5467real ms.
Pass 3: translated to C into "/tmp/stapXQNYv1/iotop_src.c" using
688652virt/322136res/13804shr/308108data kb, in 30usr/60sys/79real ms.
iotop.ko
Pass 4: compiled C into "iotop.ko" in 15500usr/3160sys/11595real ms.
```

- 3.2. Open a second terminal, log in to the **serverb** system, and switch to the **root** user.

```
[student@workstation ~]$ ssh student@serverb
...output omitted...
[student@serverb ~]$ sudo -i
[sudo] password for student: student
[root@serverb ~]#
```

- 3.3. On the **serverb** system, create the SystemTap kernel module directory if it does not exist. Use the running kernel's version when naming the module directory.

```
[root@serverb ~]# ls -la /lib/modules/$(uname -r)/systemtap
ls: cannot access '/lib/modules/4.18.0-305.el8.x86_64/systemtap': No such file or
directory
[root@serverb ~]# mkdir -p /lib/modules/$(uname -r)/systemtap
```

- 3.4. Copy the **iotop.ko** kernel module from the **servera** system to the SystemTap kernel module directory on the **serverb** system.

```
[root@serverb ~]# rsync -avz -e ssh
root@servera:/root/iotop.ko /lib/modules/$(uname -r)/systemtap/
...output omitted...
root@servera's password: redhat
receiving incremental file list
iotop.ko

sent 43 bytes received 610,648 bytes 135,709.11 bytes/sec
total size is 2,729,880 speedup is 4.47
```

► 4. Prepare the **serverb** system to use the SystemTap probe.

- 4.1. Install the **systemtap-runtime** package on the **serverb** system.

```
[root@serverb ~]# yum install systemtap-runtime
...output omitted...
Complete!
```

- 4.2. On the **serverb** system, configure the **student** user account to access the newly installed SystemTap kernel module. Add the **student** user to the **stapusr** group to run approved SystemTap kernel modules.

```
[root@serverb ~]# usermod -a -G stapusr student
```

- 4.3. On the **serverb** system, verify that the **student** user has the necessary memberships to run the SystemTap probe.

```
[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
```

▶ 5. Troubleshoot the disk activity issue.

- 5.1. Run the **iotop** module. Interrupt the command after a few iterations.

The output of the SystemTap program shows an increase in the I/O traffic by the **dd** command.

```
[student@serverb ~]$ staprun iotop
      Process      KB Read   KB Written
          dd        91060        91060
          irqbalance     3          0
          rs:main Q:Reg     0          0
          in:imjournal    0          0
          NetworkManager   0          0
          stadio          0          0
          systemd-journal 0          0

      Process      KB Read   KB Written
          dd        84680        84680
          sshd          0          0
          stadio          0          0
          systemd          0          0
          systemd-logind    0          0
          in:imjournal    0          0

      Process      KB Read   KB Written
          dd        86784        86784
          irqbalance     3          0
          sshd          0          0
          stadio          0          0
          in:imjournal    0          0
Ctrl+c
```

- 5.2. View additional details of the **dd** process to discover the command that spawned this process.

```
[student@serverb ~]$ ps -lef | grep -w dd
...output omitted...
0 D root      1787    1708  4  82   2 - 54281 -          08:56 ?        00:00:04 dd
  if=/dev/zero of=/dev/vdb bs=1024 count=1000000
0 D root      1788    1688  4  82   2 - 54281 -          08:56 ?        00:00:04 dd
  if=/dev/zero of=/dev/vdb bs=1024 count=1000000
0 S student   44848   25068  0  80   0 - 55482 -          08:58 pts/0        00:00:00
grep --color=auto dd
```

**Note**

The lab `finish` command removes the offending processes.

- ▶ 6. Log out of the `serverb` system and close the second terminal. Return to the `workstation` system as the `student` user.

```
[student@serverb ~]$ exit
[root@serverb ~]# exit
[student@serverb ~]$ exit
[student@workstation ~]$ exit
```

```
[root@servera rpm]# exit
[student@servera ~]$ exit
[student@workstation ~]$
```

Finish

On the `workstation` machine, use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish kernel-debug
```

This concludes the guided exercise.

▶ Lab

Troubleshooting Kernel Issues

In this lab, you compile a SystemTap module and configure a system for crash dump.

Outcomes

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

As the student user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

```
[student@workstation ~]$ lab start kernel-review
```

This command installs the kernel debug packages.

Instructions

An administrator is experiencing issues on one of the systems in the organization during times of heavy network activity. The system's performance worsens and eventually the system hangs. To determine the cause of the issue, the administrator suggested to use the `/usr/share/systemtap/examples/network/nettop.stp` example SystemTap script.

Provide the administrator with the compiled kernel module for the `nettop.stp` script. The administrator requires that the kernel module is built for the `4.18.0-305.el8.x86_64` version of the kernel and debug packages, which matches the affected system's kernel version. Compile the SystemTap script on the `servera` system as a portable kernel module and prepare it for sending to the affected system.

The administrator also suggested to enable kernel crash dumps on the `serverb` system to help to identify any future issues. Configure the `serverb` system to save a crash dump when the system hangs. Configure the dumps to write locally to the `/var/crash` directory with the `lzo` compression algorithm to reduce the crash dump size and to save only user data pages. Configure the crash dump output to provide maximum diagnostic information with all message types. Verify that the kernel crash dumps are recorded correctly and stored in the `/var/crash` directory.

1. Log in to the `servera` system and switch to the `root` user.
2. On the `servera` system, compile the `nettop.stp` SystemTap script into a portable kernel module.
The SystemTap script shows an error message while compiling. The error messages show that the necessary `debuginfo` dependencies are missing.
3. Troubleshoot and fix the compilation issue.
4. Finish building the SystemTap probe. Store the compiled modules as `/root/nettop.ko` on the `servera` system.
5. Enable the `kdump` service on the `serverb` system.

6. Configure the kdump service to store crash dumps in the `/var/crash` directory; use the lzo compression algorithm; and generate all message types.
7. Access the `serverb` system with its console. Test the kdump configuration by triggering a system crash.
8. Return to the `workstation` system as the `student` user.

Evaluation

On the `workstation` machine, use the `lab` command to grade your work. Correct any reported failures and rerun the script until you receive a passing grade.

```
[student@workstation ~]$ lab grade kernel-review
```

Finish

On the `workstation` machine, use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish kernel-review
```

This concludes the lab.

► Solution

Troubleshooting Kernel Issues

In this lab, you compile a SystemTap module and configure a system for crash dump.

Outcomes

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

As the student user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

```
[student@workstation ~]$ lab start kernel-review
```

This command installs the kernel debug packages.

Instructions

An administrator is experiencing issues on one of the systems in the organization during times of heavy network activity. The system's performance worsens and eventually the system hangs. To determine the cause of the issue, the administrator suggested to use the `/usr/share/systemtap/examples/network/nettop.stp` example SystemTap script.

Provide the administrator with the compiled kernel module for the `nettop.stp` script. The administrator requires that the kernel module is built for the `4.18.0-305.el8.x86_64` version of the kernel and debug packages, which matches the affected system's kernel version. Compile the SystemTap script on the `servera` system as a portable kernel module and prepare it for sending to the affected system.

The administrator also suggested to enable kernel crash dumps on the `serverb` system to help to identify any future issues. Configure the `serverb` system to save a crash dump when the system hangs. Configure the dumps to write locally to the `/var/crash` directory with the `lzo` compression algorithm to reduce the crash dump size and to save only user data pages. Configure the crash dump output to provide maximum diagnostic information with all message types. Verify that the kernel crash dumps are recorded correctly and stored in the `/var/crash` directory.

1. Log in to the `servera` system and switch to the `root` user.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

2. On the servera system, compile the `nettop.stp` SystemTap script into a portable kernel module.

The SystemTap script shows an error message while compiling. The error messages show that the necessary `debuginfo` dependencies are missing.

```
[root@servera ~]# stap -v -p 4 -m
  nettop /usr/share/systemtap/examples/network/nettop.stp
Pass 1: parsed user script and 482 library scripts using
  456012virt/88784res/12872shr/75468data kb, in 230usr/60sys/629real ms.
WARNING: cannot find module kernel debuginfo: No DWARF information found [man
warning::debuginfo]
WARNING: cannot find module kernel debuginfo: No DWARF information found [man
warning::debuginfo]
semantic error: resolution failed in DWARF builder

semantic error: resolution failed in DWARF builder

semantic error: while resolving probe point: identifier 'kernel' at /usr/share/
systemtap/tapset/linux/networking.stp:140:4
    source:      kernel.function("dev_queue_xmit")
                  ^
semantic error: no match

semantic error: resolution failed in alias expansion builder

WARNING: cannot find module kernel debuginfo: No DWARF information found [man
warning::debuginfo]
WARNING: cannot find module kernel debuginfo: No DWARF information found [man
warning::debuginfo]
semantic error: resolution failed in DWARF builder

semantic error: resolution failed in alias expansion builder

Pass 2: analyzed script: 3 probes, 1 function, 0 embeds, 3 globals using
  487412virt/103992res/16656shr/86864data kb, in 260usr/230sys/1192real ms.
Pass 2: analysis failed. [man error::pass2]
```

3. Troubleshoot and fix the compilation issue.

3.1. Verify the installed kernel debug packages.

```
[root@servera ~]# rpm -qa | grep ^kernel
kernel-tools-4.18.0-305.el8.x86_64
kernel-core-4.18.0-305.el8.x86_64
kernel-modules-4.18.0-305.el8.x86_64
kernel-tools-libs-4.18.0-305.el8.x86_64
kernel-4.18.0-305.el8.x86_64
kernel-devel-4.18.0-305.el8.x86_64
kernel-headers-4.18.0-305.el8.x86_64
kernel-debuginfo-common-x86_64-4.18.0-348.el8.x86_64
kernel-debuginfo-4.18.0-348.el8.x86_64
```

- 3.2. Uninstall the incorrect kernel-debuginfo and kernel-debuginfo-common-x86_64 packages.

```
[root@servera ~]# yum remove -y kernel-debuginfo kernel-debuginfo-common-x86_64
...output omitted...
Removed:
  kernel-debuginfo-4.18.0-348.el8.x86_64          kernel-debuginfo-common-
x86_64-4.18.0-348.el8.x86_64

Complete!
```

- 3.3. Verify that the kernel debug packages match the installed kernel version.

```
[root@servera ~]# yum list kernel-debuginfo-$(uname -r)
kernel-debuginfo-common-x86_64-$(uname -r)
Last metadata expiration check: 0:22:47 ago on Sun 21 Nov 2021 05:24:43 AM EST.
Available Packages
kernel-debuginfo.x86_64                      4.18.0-305.el8
  rhel-8.4-for-x86_64-baseos-debug-rpms
kernel-debuginfo-common-x86_64.x86_64          4.18.0-305.el8
  rhel-8.4-for-x86_64-baseos-debug-rpms
```

- 3.4. Install the package version that matches the installed kernel version.

```
[root@servera ~]# yum install kernel-debuginfo-$(uname -r)
kernel-debuginfo-common-x86_64-$(uname -r)
...output omitted...
Installed:
  kernel-debuginfo-4.18.0-305.el8.x86_64          kernel-debuginfo-common-
x86_64-4.18.0-305.el8.x86_64

Complete!
```

4. Finish building the SystemTap probe. Store the compiled modules as /root/nettop.ko on the servera system.

- 4.1. Compile the SystemTap script into a portable kernel module. Store the compiled modules as /root/nettop.ko on the servera system.

```
[root@servera ~]# stap -v -p 4 -m
nettop /usr/share/systemtap/examples/network/nettop.stp
Pass 1: parsed user script and 482 library scripts using
456012virt/88608res/12692shr/75468data kb, in 230usr/60sys/616real ms.
Pass 2: analyzed script: 6 probes, 11 functions, 0 embeds, 3 globals using
685864virt/319488res/13720shr/305320data kb, in 2680usr/330sys/3714real ms.
Pass 3: translated to C into "/tmp/stap0jpxAs/nettop_src.c" using
685864virt/319680res/13912shr/305320data kb, in 20usr/60sys/84real ms.
nettop.ko
Pass 4: compiled C into "nettop.ko" in 16670usr/3740sys/12838real ms.
```

- 4.2. Return to workstation as the student user.

```
[root@servera ~]# exit
[student@servera ~]$ exit
[student@workstation ~]$
```

5. Enable the kdump service on the serverb system.

5.1. Log in to the serverb system and switch to the root user.

```
[student@workstation ~]$ ssh student@serverb
...output omitted...
[student@serverb ~]$ sudo -i
[sudo] password for student: student
[root@serverb ~]#
```

5.2. Verify whether the kdump service is enabled and active.

```
[root@serverb ~]# systemctl is-enabled kdump.service
disabled
[root@serverb ~]# systemctl is-active kdump.service
inactive
```

5.3. Start and enable the kdump service if it is inactive or disabled.

```
[root@serverb ~]# systemctl enable kdump.service
Created symlink /etc/systemd/system/multi-user.target.wants/kdump.service → /usr/
lib/systemd/system/kdump.service.
[root@serverb ~]# systemctl start kdump.service
[root@serverb ~]# systemctl status kdump.service
● kdump.service - Crash recovery kernel arming
   Loaded: loaded (/usr/lib/systemd/system/kdump.service; enabled; vendor preset:
   enabled)
   Active: active (exited) since Sun 2021-11-21 05:36:10 EST; 7s ago
     Process: 1655 ExecStart=/usr/bin/kdumpctl start (code=exited, status=0/SUCCESS)
   Main PID: 1655 (code=exited, status=0/SUCCESS)

Nov 21 05:36:09 serverb.lab.example.com systemd[1]: Starting Crash recovery kernel
arming...
Nov 21 05:36:10 serverb.lab.example.com kdumpctl[1655]: kdump: kexec: loaded kdump
kernel
Nov 21 05:36:10 serverb.lab.example.com kdumpctl[1655]: kdump: Starting kdump:
[OK]
Nov 21 05:36:10 serverb.lab.example.com systemd[1]: Started Crash recovery kernel
arming.
```

6. Configure the kdump service to store crash dumps in the /var/crash directory; use the lzo compression algorithm; and generate all message types.

6.1. Edit the /etc/kdump.conf file and modify the core_collector entry to use the lzo compression algorithm. Also, ensure that the dump stores only user data pages.

```
core_collector makedumpfile -l --message-level 31 -d 23
```

- 6.2. Restart the kdump service to implement the changes.

```
[root@serverb ~]# systemctl restart kdump.service
[root@serverb ~]# systemctl status kdump.service
● kdump.service - Crash recovery kernel arming
  Loaded: loaded (/usr/lib/systemd/system/kdump.service; enabled; vendor preset: enabled)
    Active: active (exited) since Sun 2021-11-21 05:38:48 EST; 16s ago
      Process: 2031 ExecStop=/usr/bin/kdumpctl stop (code=exited, status=0/SUCCESS)
      Process: 2040 ExecStart=/usr/bin/kdumpctl start (code=exited, status=0/SUCCESS)
        Main PID: 2040 (code=exited, status=0/SUCCESS)

Nov 21 05:38:40 serverb.lab.example.com dracut[2335]: *** Install squash loader
*** 
Nov 21 05:38:41 serverb.lab.example.com dracut[2335]: *** Stripping files ***
Nov 21 05:38:41 serverb.lab.example.com dracut[2335]: *** Stripping files done ***
Nov 21 05:38:41 serverb.lab.example.com dracut[2335]: *** Squashing the files
inside the initramfs ***
Nov 21 05:38:47 serverb.lab.example.com dracut[2335]: *** Squashing the files
inside the initramfs done ***
Nov 21 05:38:47 serverb.lab.example.com dracut[2335]: *** Creating image file '/
boot/initramfs-4.18.0-305.el8.x86_64k'
Nov 21 05:38:48 serverb.lab.example.com dracut[2335]: *** Creating initramfs image
file '/boot/initramfs-4.18.0-305.e>
Nov 21 05:38:48 serverb.lab.example.com kdumpctl[2040]: kdump: kexec: loaded kdump
kernel
Nov 21 05:38:48 serverb.lab.example.com kdumpctl[2040]: kdump: Starting kdump:
[OK]
Nov 21 05:38:48 serverb.lab.example.com systemd[1]: Started Crash recovery kernel
arming.
```

7. Access the serverb system with its console. Test the kdump configuration by triggering a system crash.

- 7.1. Log in to the serverb system console and switch to the root user.

```
...output omitted...
serverb login: student
Password: student
[student@serverb ~]$ sudo -i
[sudo] password for student: student
[root@serverb ~]#
```

- 7.2. Enable all kernel SysRq functions by setting the value in the /proc/sys/kernel/sysrq file to 1.

```
[root@serverb ~]# echo 1 > /proc/sys/kernel/sysrq
```

- 7.3. Trigger a system crash by setting the value in the /proc/sysrq-trigger file to c. Wait for the serverb system to restart and then proceed to the next step.

```
[root@serverb ~]# echo c > /proc/sysrq-trigger
```

- 7.4. Return to the **workstation** system. Log in to the **serverb** system and switch to the **root** user.

```
[student@workstation ~]$ ssh student@serverb  
...output omitted...  
[student@serverb ~]$ sudo -i  
[sudo] password for student: student  
[root@serverb ~]#
```

- 7.5. Verify that the kernel crash dump was generated and stored in the **/var/crash** directory.

```
[root@serverb ~]# ll /var/crash/  
total 0  
drwxr-xr-x. 2 root root 67 Nov 21 05:40 127.0.0.1-2021-11-21-05:40:49  
[root@serverb ~]# ll /var/crash/127.0.0.1-2021-11-21-05\:40\:49/  
total 83760  
-rw-r--r--. 1 root root 56051 Nov 21 05:40 kexec-dmesg.log  
-rw----- 1 root root 85671410 Nov 21 05:40 vmcore  
-rw-r--r--. 1 root root 38269 Nov 21 05:40 vmcore-dmesg.txt
```

8. Return to the **workstation** system as the **student** user.

```
[root@serverb ~]# exit  
[student@serverb ~]$ exit  
[student@workstation ~]$
```

Evaluation

On the **workstation** machine, use the **lab** command to grade your work. Correct any reported failures and rerun the script until you receive a passing grade.

```
[student@workstation ~]$ lab grade kernel-review
```

Finish

On the **workstation** machine, use the **lab** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish kernel-review
```

This concludes the lab.

Summary

In this chapter, you learned:

- The `/etc/kdump.conf` file configures the kdump service.
- The `kdumpctl` command performs common kdump administration tasks.
- The `sysctl` command can configure kernel crashes to occur on OOM, hung process, soft lookup, and NMI events.
- The `stap` command runs SystemTap scripts and compiles them into portable kernel modules.
- The `staprun` command runs SystemTap-compiled probes on target systems to gather diagnostic traces.
- The non-root users in the `stapusr` and `stapdev` groups can run and compile SystemTap modules.
- The `ftrace` framework helps to debug and analyze latency and performance issues.

Chapter 11

Comprehensive Review

Goal

Review tasks from *Red Hat Enterprise Linux Diagnostics and Troubleshooting*

Objectives

- Review tasks from *Red Hat Enterprise Linux Diagnostics and Troubleshooting*

Sections

- Comprehensive Review of Red Hat Enterprise Linux Diagnostics and Troubleshooting

Labs

- Repairing a Non-running Application
- Resolving a Console Login Issue
- Resolving Authentication Issues
- Repairing a Web Server Issue
- Resolving Network Delay Issues
- Resolving Container Issues

Comprehensive Review

Objectives

After completing this section, you should be able to demonstrate knowledge and skills learned in *Red Hat Enterprise Linux Diagnostics and Troubleshooting*.

Reviewing Red Hat Enterprise Linux Diagnostics and Troubleshooting

Before beginning the comprehensive review for this course, you should be comfortable with the topics covered in each chapter.

You can refer to earlier sections in the textbook for extra study.

Chapter 1, Introducing Troubleshooting Strategy

Describe effective troubleshooting methods and data collection strategies.

- Use a systematic approach to troubleshooting with the scientific method.
- Collect system information to support troubleshooting.
- Use Red Hat resources to support troubleshooting.

Chapter 2, Configuring Baseline Data

Configure baseline data collection with monitoring, logging, and change tracking.

- Monitor systems to gather information.
- Configure systems for remote logging to a central log host.
- Describe configuration management with Red Hat Satellite and Red Hat Ansible Automation Platform.
- Implement change tracking to monitor system modifications.

Chapter 3, Troubleshooting Boot Issues

Identify and resolve issues that can affect a system's ability to boot.

- Repair boot issues on BIOS systems.
- Repair boot issues on UEFI systems.
- Identify and resolve service failures that affect the boot process.
- Reset the root password.

Chapter 4, Identifying Hardware Issues

Identify hardware issues that can affect a system's ability to operate normally.

- Identify system hardware devices and hardware issues.
- Manage kernel modules and parameters.
- Identify and resolve virtualization issues.

Chapter 5, Troubleshooting Storage Issues

Identify and resolve issues related to storage.

- Describe the Linux storage layers and their function, and identify tools to examine activity at different layers.
- Detect and recover from file system corruption.
- Revert an LVM storage misconfiguration.
- Recover data from an encrypted device.
- Identify and repair iSCSI issues.

Chapter 6, Troubleshooting RPM Issues

Identify and resolve issues with the package management subsystem.

- Identify and resolve package dependency issues.
- Identify and rebuild a corrupted RPM database.
- Identify and restore changed files with package management tools.
- Register a system with Red Hat and manage Red Hat subscriptions.

Chapter 7, Troubleshooting Network Issues

Identify and resolve network connectivity issues.

- Verify network connectivity with standard RHEL tools.
- Identify and repair network connectivity issues.
- Inspect network traffic to assist troubleshooting.

Chapter 8, Troubleshooting Application Issues

Identify and resolve application issues with debugging tools.

- Identify library dependencies for installed software.
- Identify applications that have memory leaks.
- Debug an application with standard RHEL tools.
- Troubleshoot a containerized application or service.

Chapter 9, Troubleshooting Security Issues

Identify and resolve issues related to security subsystems.

- Identify and repair SELinux issues.

- Identify and repair user authentication and authorization issues.
- Identify and repair LDAP, Kerberos, and SSSD identity management issues.

Chapter 10, Troubleshooting Kernel Issues

Identify kernel issues and assist Red Hat Support in resolving kernel issues.

- Configure systems to enable kernel crash dumps.
- Compile SystemTap scripts into kernel modules for deployment on remote systems.

► Lab

Repairing a Non-running Application

In this review, you repair a non-running application.

Outcomes

You should be able to successfully repair the non-running application on your machine.

Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command installs a custom application on your system.

```
[student@workstation ~]$ lab start compreview-review1
```

Specifications

- The custom program application is not functioning properly on the `serverA` system.
 - Troubleshoot and repair the issue, resulting in a functioning application.
 - The application prints the text "Success!" when it is functioning properly.
 - The `program` command runs the application.
 - The `program` application relies on a locally running web service.

Evaluation

As the student user on the `workstation` machine, use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab grade compreview-review1
```

Finish

As the student user on the `workstation` machine, use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish compreview-review1
```

This concludes the lab.

► Solution

Repairing a Non-running Application

In this review, you repair a non-running application.

Outcomes

You should be able to successfully repair the non-running application on your machine.

Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

This command installs a custom application on your system.

```
[student@workstation ~]$ lab start comprevew-review1
```

1. Troubleshoot and resolve the `program` application issue.

- 1.1. Log in to `servera` and switch to the `root` user.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

- 1.2. Attempt to run the `program` command, and view any messages.

The runtime linker cannot open the `libreadline.so.7` library.

```
[root@servera ~]# program
program: error while loading shared libraries: libreadline.so.7: cannot open
shared object file: No such file or directory
```

- 1.3. Discover whether any other shared libraries cannot be resolved.

No other libraries are missing.

```
[root@servera ~]# which program
/usr/bin/program
[root@servera ~]# ldd /usr/bin/program
linux-vdso.so.1 (0x00007ffc55d92000)
libreadline.so.7 => not found
libc.so.6 => /lib64/libc.so.6 (0x00007f193603f000)
/lib64/ld-linux-x86-64.so.2 (0x00007f1936404000)
```

- 1.4. Identify the package that provides the missing shared library.

```
[root@servera ~]# yum provides libreadline.so.7
Last metadata expiration check: 2:05:19 ago on Fri 12 Nov 2021 02:41:41 PM EST.
readline-7.0-10.el8.i686 : A library for editing typed command lines
Repo       : rhel-8.4-for-x86_64-baseos-rpms
Matched from:
Provide   : libreadline.so.7
```

- 1.5. Install the required packages.

The package is present, but the library that it provides is missing.

```
[root@servera ~]# yum install readline
Last metadata expiration check: 2:06:19 ago on Fri 12 Nov 2021 02:41:41 PM EST.
Package readline-7.0-10.el8.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
```

- 1.6. Verify the package's files.

The library file is missing.

```
[root@servera ~]# rpm -V readline
missing      /usr/lib64/libreadline.so.7
```

- 1.7. Reinstall the package to restore the missing file.

```
[root@servera ~]# yum reinstall readline
...output omitted...
Reinstalled:
readline-7.0-10.el8.x86_64

Complete!
```

- 1.8. Attempt to run the `program` command, and view any messages. When prompted, enter a string of characters.

The application is starting, but presents a new error.

```
[root@servera ~]# program
Enter value: anystring
Error: application failed to run!
```

2. Troubleshoot and resolve the additional `program` application issue.

- 2.1. View the application's system calls.

The `strace` output will pause when it reaches the string input prompt. Press `Enter` to view the full output.

The application tries to connect to port 80 on the local system, but the connection is refused.

```
[root@servera ~]# strace program
execve("/usr/bin/program", ["program"], 0x7ffcb6e91dd0 /* 25 vars */) = 0
...output omitted...
connect(3, {sa_family=AF_INET, sin_port=htons(80),
    sin_addr=inet_addr("127.0.0.1")}, 16) = -1 ECONNREFUSED (Connection refused)
write(1, "Error: application failed to run"..., 34) = 34
Error: application failed to run!
exit_group(1) = ?
exited with 1
```

- 2.2. The application relies on a local web service to be running. Verify whether one is installed.

```
[root@servera ~]# systemctl status httpd
Unit httpd.service could not be found.
```

- 2.3. Install and start a web server to listen on port 80.

```
[root@servera ~]# yum install httpd
...output omitted...
[root@servera ~]# systemctl start httpd
```

- 2.4. Attempt to run the `program` command, and view any messages.

The "Success!" output indicates that the `program` application is functioning properly.

```
[root@servera ~]# program
Enter value: anystring
Success!
```

- 2.5. Return to `workstation` as the `student` user.

```
[root@servera ~]# exit
[student@servera ~]$ exit
[student@workstation ~]$
```

Evaluation

As the `student` user on the `workstation` machine, use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab grade comprevew-review1
```

Finish

As the `student` user on the `workstation` machine, use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish compreview-review1
```

This concludes the lab.

► Lab

Resolving a Console Login Issue

In this review, you reset a `root` password and repair a boot issue.

Outcomes

You should be able to successfully reset a root password and repair a boot issue.

Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command modifies the root password and boot parameters.

```
[student@workstation ~]$ lab start compreview-review2
```



Important

This lab is intended to practice the technique to use on a physical system that might not be network-accessible. You are asked to assume that you do not have ssh access to `serverb`, and that the student account on `serverb` does not have full sudo access. Access a direct virtual console to `serverb` by using the appropriate method for your classroom environment.

Specifications

- Users report that the `serverb` system displays an error and hangs during the boot process. While troubleshooting, your colleague accidentally modified the `root` password of the `serverb` system to an unknown value.
 - Reset the `root` password to `redhat`.
 - Repair the boot parameters so that the system boots without manual intervention.

Evaluation

As the student user on the `workstation` machine, use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab grade compreview-review2
```

Finish

As the student user on the `workstation` machine, use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish compreview-review2
```

This concludes the lab.

► Solution

Resolving a Console Login Issue

In this review, you reset a `root` password and repair a boot issue.

Outcomes

You should be able to successfully reset a root password and repair a boot issue.

Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command modifies the root password and boot parameters.

```
[student@workstation ~]$ lab start compreview-review2
```



Important

This lab is intended to practice the technique to use on a physical system that might not be network-accessible. You are asked to assume that you do not have `ssh` access to `serverb`, and that the student account on `serverb` does not have full `sudo` access. Access a direct virtual console to `serverb` by using the appropriate method for your classroom environment.

1. Reset the `root` password. Ignore the malfunctioning default boot entry for now.
 - 1.1. If the `grub2` boot menu is not visible, then reboot the system.
 - 1.2. Highlight the `Red Hat Enterprise Linux 8.4 (0otpa)` boot entry, and press `e` to edit it.
 - 1.3. Scroll down to the line that starts with `linux`, press `Ctrl+e` to jump to the end of the line, remove all `console=` settings (if any), and append `rd.break`.
 - 1.4. Press `Ctrl+x` to boot with these modified settings.
 - 1.5. Remount the file system on `/sysroot` with read-write capabilities.

```
switch_root:/# mount -o remount,rw /sysroot
```

- 1.6. Change the root directory to `/sysroot`.

```
switch_root:/# chroot /sysroot
```

- 1.7. Set the `root` password to `redhat`.

```
sh-4.2# echo "root:redhat" | chpasswd
```

- 1.8. Force SELinux to relabel all files during the next boot.

```
sh-4.2# touch /.autorelabel
```

- 1.9. Reboot your system by typing **exit** twice. On rebooting, select the Red Hat Enterprise Linux 8.4 (Ootpa) boot entry.

```
sh-4.2# exit  
exit  
switch_root:/# exit  
exit
```

- 1.10. Log in to the console as the **root** user with the password **redhat**.

```
serverb login: root  
Password: redhat  
[root@serverb ~]#
```

2. Repair the boot issue.

- 2.1. Query the current boot entries.

```
[root@serverb ~]# grubby --info=ALL  
index=0  
kernel="/boot/vmlinuz-4.18.0-305.el8.x86_64"  
args="ro no_timer_check net.ifnames=0 crashkernel=auto $tuned_params"  
root="/dev/vda3"  
initrd="/boot/initramfs-4.18.0-305.el8.x86_64.img $tuned_initrd"  
title="Red Hat Enterprise Linux (4.18.0-305.el8.x86_64) 8.4 (Ootpa)"  
id="ffffffffffffffffffff-4.18.0-305.el8.x86_64"  
index=1  
kernel="/boot/vmlinuz-newboot"  
args="ro no_timer_check net.ifnames=0 crashkernel=auto"  
root="/dev/vda3"  
initrd="/boot/initramfs-newboot.img"  
title="NewBootHopeThisWorks"  
id="b39b2a9094e24603888d9d5b9ddc2740-newboot"
```

- 2.2. Determine the default grub2 boot entry.

```
[root@serverb ~]# grubby --default-kernel  
/boot/vmlinuz-newboot
```

- 2.3. Change the default grub2 menu entry to use the kernel that does not have the **newboot** label. The version of the kernel will vary.

```
[root@serverb ~]# grub --set-default=/boot/vmlinuz-4.18.0-305.el8.x86_64  
The default is /boot/loader/entries/  
ffffffffffffffffff-4.18.0-305.el8.x86_64.conf with index 0 and  
kernel /boot/vmlinuz-4.18.0-305.el8.x86_64
```

2.4. Verify that the default kernel is set correctly.

```
[root@serverb ~]# grub --default-kernel  
/boot/vmlinuz-4.18.0-305.el8.x86_64
```

2.5. Reboot the system and verify that the default grub2 menu entry boots correctly and displays the login prompt.

```
[root@serverb ~]# reboot  
...output omitted...  
serverb login:
```

2.6. Return to **workstation** as the **student** user, with the appropriate technique for your classroom environment.

Evaluation

As the **student** user on the **workstation** machine, use the **lab** command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab grade compreview-review2
```

Finish

As the **student** user on the **workstation** machine, use the **lab** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish compreview-review2
```

This concludes the lab.

► Lab

Resolving Authentication Issues

In this review, you repair an authentication issue.

Outcomes

You should be able to successfully repair an authentication issue.

Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

This command configures your system as an IdM client.

```
[student@workstation ~]$ lab start compreview-review3
```

Specifications

- The servera system is an IdM client for user authentication.
 - The IdM server is `utility.lab.example.com`.
 - The IdM server authenticates the `admin` and `operator1` users.
 - The `operator1` user reports that they cannot log in.
 - The `operator1` user should be able to log in without a password prompt.
 - The password for the `operator1` and `admin` user accounts is `RedHat123^`.

Evaluation

As the student user on the workstation machine, use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab grade compreview-review3
```

Finish

As the student user on the workstation machine, use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish compreview-review3
```

This concludes the lab.

► Solution

Resolving Authentication Issues

In this review, you repair an authentication issue.

Outcomes

You should be able to successfully repair an authentication issue.

Before You Begin

As the student user on the workstation machine, use the lab command to prepare your system for this exercise.

This command configures your system as an IdM client.

```
[student@workstation ~]$ lab start compreview-review3
```

1. Re-create the issue. From the workstation machine, attempt to log in to the operator1 account on the servera machine. The login either fails to connect or presents a password prompt. Press Ctrl+C at any time to exit the authentication process.

- 1.1. The following output is an example of incorrect behavior:

```
[student@workstation ~]$ ssh operator1@servera
Connection closed by 172.25.250.10 port 22
```

- 1.2. The following output is another possible example of incorrect behavior:

```
[student@workstation ~]$ ssh operator1@servera
Password: RedHat123^
Password: RedHat123^
Password: RedHat123^
operator1@servera's password: RedHat123^
Permission denied, please try again.
operator1@servera's password: RedHat123^
Received disconnect from 172.25.250.10 port 22:2: Too many authentication failures
Disconnected from 172.25.250.10 port 22
[student@workstation ~]$
```

2. Log in to servera.

- 2.1. As the student user on the workstation machine, log in to the servera machine and switch to the root user.

```
[student@workstation ~]$ ssh student@servera  
...output omitted...  
[student@servera ~]$ sudo -i  
[sudo] password for student: student  
[root@servera ~]#
```

3. Ensure that the Identity Management service is providing Kerberos tickets.
 - 3.1. Use the `kinit admin` command to authenticate and obtain Kerberos credentials. When prompted, use `RedHat123^` as the password.

```
[root@servera ~]# kinit admin  
Password for admin@LAB.EXAMPLE.NET : RedHat123^
```

- 3.2. Use the `klist` command to verify that the Kerberos ticket is created.

```
[root@servera ~]# klist  
Ticket cache: KCM:0  
Default principal: admin@LAB.EXAMPLE.NET  
  
Valid starting     Expires            Service principal  
11/23/2021 20:50:11  11/24/2021 20:50:06  krbtgt/LAB.EXAMPLE.NET@LAB.EXAMPLE.NET
```

4. The previous step indicates that both LDAP and Kerberos are functioning normally, so the problem might be the SSSD configuration. Clear the SSSD log files on the `servera` machine and attempt to log in as the `operator1` user to generate fresh logs. Search the log files for possible errors.

- 4.1. Enable detailed SSSD debug logging.

```
[root@servera ~]# sssctl debug-level 6
```

- 4.2. Invalidate all objects in the SSSD cache.

```
[root@servera ~]# sss_cache -E
```

- 4.3. Clear previous SSSD logs to minimize the troubleshooting data set.

```
[root@servera ~]# sssctl logs-remove  
Truncating log files...
```

- 4.4. From a new terminal on `workstation`, attempt to log in to the `operator1` user account to regenerate the error.

```
[student@workstation ~]$ ssh operator1@servera  
Connection closed by 172.25.250.10 port 22
```

- 4.5. On `servera`, review the SSSD logs for information about a failed request. The log file indicates that no rules were found for the host.

```
[root@servera ~]# tail /var/log/sssd/sssd_lab.example.net.log
(2021-11-23 21:51:12): [be[lab.example.net]] [sdap_account_expired] (0x0400): IPA
access control succeeded, checking AD access control
(2021-11-23 21:51:12): [be[lab.example.net]] [sdap_account_expired_ad] (0x0400):
Performing AD access check for user [ operator1@lab.example.net ]
(2021-11-23 21:51:12): [be[lab.example.net]] [sdap_get_generic_ext_step]
(0x0400): calling ldap_search_ext with [(&(objectClass=ipaHost)
(fqdn=serverc.lab.example.com))][cn=accounts,dc=lab,dc=example,dc=net].
(2021-11-23 21:51:12): [be[lab.example.net]] [sdap_get_generic_op_finished]
(0x0400): Search result: Success(0), no errmsg set
(2021-11-23 21:51:12): [be[lab.example.net]] [ipa_pam_access_handler_done]
(0x0020): No HBAC rules found, denying access
(2021-11-23 21:51:12): [be[lab.example.net]] [dp_req_done] (0x0400): DP Request
[PAM Account #9]: Request handler finished [0]: Success
(2021-11-23 21:51:12): [be[lab.example.net]] [_dp_req_recv] (0x0400): DP Request
[PAM Account #9]: Receiving request data.
(2021-11-23 21:51:12): [be[lab.example.net]] [dp_req_destructor] (0x0400): DP
Request [PAM Account #9]: Request removed.
(2021-11-23 21:51:12): [be[lab.example.net]] [dp_req_destructor] (0x0400): Number
of active DP request: 0
(2021-11-23 21:51:12): [be[lab.example.net]] [sbus_issue_request_done] (0x0400):
sssd.dataprovider.pamHandler: Success
```

5. Check and repair any issues in the /etc/sssd/sssd.conf file.

- 5.1. Inspect the file. The ipa_hostname parameter is incorrect. Fix the parameter so that it matches the following output.

```
[root@servera ~]# cat /etc/sssd/sssd.conf | grep ipa_hostname
ipa_hostname = servera.lab.example.com
```

- 5.2. Restart the sssd service.

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

6. Verify that the operator1 user is not prompted for a password when logging in to the servera machine.

- 6.1. From the additional terminal on workstation, log in to the operator1 account on the servera machine.

The login is successful. Exit the terminal when connected.

```
[student@workstation ~]$ ssh operator1@servera
...output omitted...
[operator1@servera ~]$ exit
```

- 6.2. Close the terminal that is connected to servera and return to workstation as the student user.

```
[root@servera ~]# exit  
[student@servera ~]$ exit  
[student@workstation ~]$
```

Evaluation

As the **student** user on the **workstation** machine, use the **lab** command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab grade compreview-review3
```

Finish

As the **student** user on the **workstation** machine, use the **lab** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish compreview-review3
```

This concludes the lab.

► Lab

Repairing a Web Server Issue

In this review, you repair a failing service and a corrupted file system.

Outcomes

You should be able to successfully repair a failing service and a corrupted file system.

Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command installs a web server and prepares files.

```
[student@workstation ~]$ lab start compreview-review4
```

Specifications

- A web server that is configured on the `servera` system is not providing the correct pages. While troubleshooting, your colleague broke the web service and can no longer start it. Repair the web service and ensure that the server provides the correct pages.
 - The `/dev/vdb1` device contains the required HTML files.
 - The server expects files in the `/var/www/html` directory.
 - If the web server is working, then it returns `<h1>Home</h1>`.

Evaluation

As the student user on the `workstation` machine, use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab grade compreview-review4
```

Finish

As the student user on the `workstation` machine, use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish compreview-review4
```

This concludes the lab.

► Solution

Repairing a Web Server Issue

In this review, you repair a failing service and a corrupted file system.

Outcomes

You should be able to successfully repair a failing service and a corrupted file system.

Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

This command installs a web server and prepares files.

```
[student@workstation ~]$ lab start compreview-review4
```

1. Gather information about the failing service.

- 1.1. Log in to `servera` and switch to the `root` user.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

- 1.2. Determine the status of the web service.

The service is in a failed state. The `/usr/sbin/httpd` binary file could not start.

```
[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 Tue 2021-11-16 17:58:38 EST; 40s ago
    Docs: man:httpd.service(8)
 Process: 1953 ExecStart=/usr/sbin/httpd $OPTIONS -DFOREGROUND (code=exited, status=203/EXEC)
 Main PID: 1953 (code=exited, status=203/EXEC)

Nov 16 17:58:38 servera.lab.example.com systemd[1]: Starting The Apache HTTP
Server...
```

Chapter 11 | Comprehensive Review

```
Nov 16 17:58:38 servera.lab.example.com systemd[1]: httpd.service: Main process exited, code=exited, status=203/EXEC
Nov 16 17:58:38 servera.lab.example.com systemd[1]: httpd.service: Failed with result 'exit-code'.
Nov 16 17:58:38 servera.lab.example.com systemd[1]: Failed to start The Apache HTTP Server.
```

1.3. Inspect the `httpd` binary file.

The binary file is missing executable privileges.

```
[root@servera ~]# ls -la /usr/sbin/httpd
-rw-r--r--. 1 root root 579968 Jan 27 2021 /usr/sbin/httpd
```

1.4. Check which package provided the `httpd` binary.

```
[root@servera ~]# rpm -qf /usr/sbin/httpd
httpd-2.4.37-39.module+el8.4.0+9658+b87b2deb.x86_64
```

1.5. Check the status of the package.

```
[root@servera ~]# rpm -qV httpd
.M..... /usr/sbin/apachectl
.M..... /usr/sbin/fcgistarterm
.M..... /usr/sbin/htcacheclean
.M..... /usr/sbin/httpd
.M..... /usr/sbin/rotatelogs
.M..... /usr/sbin/suexec
```

2. Repair the failing service.

2.1. Restore the permissions of the files in the package.

```
[root@servera ~]# rpm --setperms httpd
```

2.2. Check the status of the package again.

```
[root@servera ~]# rpm -qV httpd
```

2.3. Restart the web service.

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

2.4. Verify that the web service 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: active (running) since Tue 2021-11-16 18:01:19 EST; 6s ago
    Docs: man:httpd.service(8)
```

```
Main PID: 3066 (httpd)
  Status: "Started, listening on: port 80"
    Tasks: 213 (limit: 11049)
  Memory: 27.2M
  CGroup: /system.slice/httpd.service
    ├─3066 /usr/sbin/httpd -DFOREGROUND
    ├─3067 /usr/sbin/httpd -DFOREGROUND
    ├─3068 /usr/sbin/httpd -DFOREGROUND
    ├─3069 /usr/sbin/httpd -DFOREGROUND
    └─3070 /usr/sbin/httpd -DFOREGROUND

Nov 16 18:01:19 servera.lab.example.com systemd[1]: Starting The Apache HTTP
Server...
Nov 16 18:01:19 servera.lab.example.com systemd[1]: Started The Apache HTTP
Server.
Nov 16 18:01:19 servera.lab.example.com httpd[3066]: Server configured, listening
on: port 80
```

- 2.5. View the default page that the web server returned.

The server provides the default Apache2 page, not the required <h1>Home</h1> page. The required pages are on the /dev/vdb1 device.

```
[root@servera ~]# curl localhost
...output omitted...
```

3. Gather information about the required web files and the /dev/vdb1 device.

- 3.1. List the contents of the /var/www/html directory.

```
[root@servera ~]# ls -l /var/www/html
total 0
```

- 3.2. List the file system of the /var/www/html directory.

```
[root@servera ~]# df -T /var/www/html
Filesystem      Type 1K-blocks   Used Available Use% Mounted on
/dev/vda3        xfs    10371052 2308320   8062732  23% /
```

- 3.3. Mount the correct file system.

An error is expected.

```
[root@servera ~]# mount /dev/vdb1 /var/www/html
mount: /var/www/html: mount(2) system call failed: Structure needs cleaning.
```

- 3.4. Determine the device file system type.

```
[root@servera ~]# lsblk -f /dev/vdb1
NAME FSTYPE LABEL UUID                                     MOUNTPOINT
vdb1 xfs          73f56278-a55b-4e25-beba-436a880cb0c5
```

4. Repair the corrupted device and mount the required files.

4.1. Perform a read-only repair of the file system.

```
[root@servera ~]# xfs_repair -n /dev/vdb1
...output omitted...
Invalid inode number 0x75bcd15
xfs_dir_ino_validate: XFS_ERROR_REPORT
Metadata corruption detected at 0x561f28d32dd0, inode 0x80 data fork
couldn't map inode 128, err = 117, can't compare link counts
No modify flag set, skipping filesystem flush and exiting.
```

4.2. Repair the file system.

```
[root@servera ~]# xfs_repair /dev/vdb1
...output omitted...
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 137, moving to lost+found
Phase 7 - verify and correct link counts...
done
```

4.3. Attempt to mount the device again.

```
[root@servera ~]# mount /dev/vdb1 /var/www/html
```

4.4. Inspect the contents of the /var/www/html directory.

New files are present; however, the `index.html` file is still missing.

```
[root@servera ~]# ls -l /var/www/html
total 12
-rw-r--r--. 1 root root 15 Feb 24 2016 about.html
drwxr-xr-x. 2 root root 16 Feb 24 2016 lost+found
-rw-r--r--. 1 root root 18 Feb 24 2016 products.html
-rw-r--r--. 1 root root 17 Feb 24 2016 support.html
```

4.5. Inspect the `lost+found` directory.

```
[root@servera ~]# ls -la /var/www/html/lost+found
total 4
drwxr-xr-x. 2 root root 16 Feb 24 2016 .
drwxr-xr-x. 3 root root 79 Feb 24 2016 ..
-rw-r--r--. 1 root root 14 Feb 24 2016 137
```

4.6. Move the orphaned file to `index.html` in the directory.

```
[root@servera ~]# mv /var/www/html/lost+found/137 /var/www/html/index.html
```

4.7. Run the `curl` command against the web server.

A correct message is displayed.

```
[root@servera ~]# curl localhost  
<h1>Home</h1>
```

4.8. Return to workstation as the student user.

```
[root@servera ~]# exit  
[student@servera ~]$ exit  
[student@workstation ~]$
```

Evaluation

As the **student** user on the **workstation** machine, use the **lab** command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab grade compreview-review4
```

Finish

As the **student** user on the **workstation** machine, use the **lab** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish compreview-review4
```

This concludes the lab.

► Lab

Resolving Network Delay Issues

In this review, you repair a network delay issue.

Outcomes

You should be able to successfully repair a network delay issue.

Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command installs a custom application on your system.

```
[student@workstation ~]$ lab start compreview-review5
```

Specifications

- Users on the `servera` system report delays when connecting to the `serverb` system. Specifically, users notice a small delay before connecting via SSH. Investigate and repair the issue that is causing the delay.
- The systems use the following IP address scheme:
 - `servera`: `172.25.250.10/24, fd37:5265:6448:6174::a/64`
 - `serverb`: `172.25.250.11/24, fd37:5265:6448:6174::b/64`

Evaluation

As the student user on the `workstation` machine, use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab grade compreview-review5
```

Finish

As the student user on the `workstation` machine, use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish compreview-review5
```

This concludes the lab.

► Solution

Resolving Network Delay Issues

In this review, you repair a network delay issue.

Outcomes

You should be able to successfully repair a network delay issue.

Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

This command installs a custom application on your system.

```
[student@workstation ~]$ lab start compreview-review5
```

1. Re-create the issue.

- 1.1. Log in to `servera` and switch to the `root` user.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

- 1.2. Use the `ssh` command to perform a remote command on the `serverb` system as the `student` user. Note the time that it takes to connect. When prompted, provide `student` as the password.

A small delay occurs before the connection is established.

```
[root@servera ~]# ssh student@serverb echo hello
...delay...
The authenticity of host 'serverb (172.25.250.11)' can't be established.
ECDSA key fingerprint is SHA256:NJAyJMx8B2AeIYHRnVLAuJ1XZwblomyOKowyfTwGrTY.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'serverb,172.25.250.11' (ECDSA) to the list of known
hosts.
student@serverb's password: student
hello
```

2. Gather information.

- 2.1. Use `strace` to run the `ssh` command to discover whether the command stalls at a specific point.

The output pauses on a specific `connect` system call.

```
[root@servera ~]# strace ssh student@serverb echo hello
...output omitted...
```

This system call, and the subsequent `getsockname` call, show a failed attempt to connect to an IPv6 socket on the `serverb` system. When these attempts fail, an IPv4 socket is successfully opened.

```
connect(5, {sa_family=AF_INET6, sin6_port=htons(22), sin6_flowinfo=htonl(0),
inet_nton(AF_INET6, "fd37:5265:6448:6174::b", &sin6_addr), sin6_scope_id=0}, 28
```

2.2. Inspect the IP addresses of the `serverb` system.

The IPv6 address on the `eth1` interface is correctly configured. Because the IPv4 protocol successfully connects, it is likely that a rule is preventing IPv6 traffic from reaching `serverb`.

```
[root@servera ~]# ssh student@serverb ip addr
student@serverb's password: student
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    qlen 1000
        link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
        inet 127.0.0.1/8 scope host lo
            valid_lft forever preferred_lft forever
        inet6 ::1/128 scope host
            valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group
    default 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 noprefixroute eth0
            valid_lft forever preferred_lft forever
        inet6 fe80::6dab:7b1:80c9:4f7f/64 scope link noprefixroute
            valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 8942 qdisc fq_codel state UP group
    default qlen 1000
        link/ether 52:54:00:01:fa:0b brd ff:ff:ff:ff:ff:ff
        inet6 fd37:5265:6448:6174::b/64 scope global noprefixroute
            valid_lft forever preferred_lft forever
        inet6 fe80::a969:1f7e:a542:51b2/64 scope link noprefixroute
            valid_lft forever preferred_lft forever
```

3. Fix and then verify that the issue is resolved.

3.1. From the `workstation` system, open a new terminal. Log in to `serverb` and switch to the `root` user.

```
[student@workstation ~]$ ssh student@serverb
...output omitted...
[student@serverb ~]$ sudo -i
[sudo] password for student: student
[root@serverb ~]#
```

3.2. Inspect the firewall rules.

A rich rule is denying all incoming IPv6 traffic.

```
[root@serverb ~]# firewall-cmd --list-all
public (active)
target: default
icmp-block-inversion: no
interfaces: eth0 eth1
sources:
services: cockpit dhcpcv6-client http ssh
ports:
protocols:
masquerade: no
forward-ports:
source-ports:
icmp-blocks:
rich rules:
rule family="ipv6" source address="0::0/0" reject
```

- 3.3. Remove the unwanted rule and reload the firewall.

```
[root@serverb ~]# firewall-cmd --permanent --remove-rich-rule='rule family="ipv6"
source address="0::0/0" reject'
[root@serverb ~]# firewall-cmd --reload
```

- 3.4. From the terminal that is connected to `servera`, attempt the `ssh` command again. Verify that the delay no longer occurs.

```
[root@servera ~]# ssh student@serverb echo hello
...output omitted...
```

- 3.5. Close the additional terminal to `serverb` and return to `workstation` as the `student` user.

```
[root@serverb ~]# exit
[student@serverb ~]$ exit
[student@workstation ~]$ exit
```

```
[root@servera ~]# exit
[student@servera ~]$ exit
[student@workstation ~]$
```

Evaluation

As the `student` user on the `workstation` machine, use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab grade compreview-review5
```

Finish

As the student user on the workstation machine, use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish compreview-review5
```

This concludes the lab.

► Lab

Resolving Container Issues

In this review, you repair a containerized web application.

Outcomes

You should be able to successfully repair a containerized web application.

Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command starts a containerized application on your system.

```
[student@workstation ~]$ lab start compreview-review6
```

Specifications

- The `container-web.service` controls the web container.
- The container is rootless.
- The container and host listen on port `8080`.
- When repaired, the web application returns "hello from a container".

Evaluation

As the student user on the `workstation` machine, use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab grade compreview-review6
```

Finish

As the student user on the `workstation` machine, use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish compreview-review6
```

This concludes the lab.

► Solution

Resolving Container Issues

In this review, you repair a containerized web application.

Outcomes

You should be able to successfully repair a containerized web application.

Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

This command starts a containerized application on your system.

```
[student@workstation ~]$ lab start compreview-review6
```

1. Gather information about the failing application.

- 1.1. Log in to servera.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
```

- 1.2. Attempt to reach the application.

The `curl` command returns an error.

```
[student@servera ~]$ curl localhost:8080
curl: (56) Recv failure: Connection reset by peer
```

- 1.3. Inspect the container service.

```
[student@servera ~]$ systemctl --user status container-web.service
● container-web.service - Podman container-web.service
   Loaded: loaded (/home/student/.config/systemd/user/container-web.service;
             disabled; vendor preset: enabled)
     Active: active (running) since Thu 2021-11-18 14:20:01 EST; 24s ago
       Docs: man:podman-generate-systemd(1)
   Process: 34277 ExecStart=/usr/bin/podman run --common-pidfile /run/user/1000/
             container-web.pid --cidfile /run/user/1000/container-web.ctr-id --cgroups=no-
             common --replace --name web -dt -p 8080:8181 -v /var/we>
   Process: 34275 ExecStartPre=/bin/rm -f /run/user/1000/container-web.pid /run/
             user/1000/container-web.ctr-id (code=exited, status=0/SUCCESS)
             ...output omitted...
Nov 18 14:20:00 serverb.lab.example.com systemd[1433]: Starting Podman container-
             web.service...
```

```
Nov 18 14:20:00 serverb.lab.example.com podman[34277]:  
13b5f6ccedb8f8c36dfceaa0fd274833dc940f7d6e2632fb8c08eb3289d4c1d4  
Nov 18 14:20:01 serverb.lab.example.com podman[34277]:  
24e68a1628933bf615a52d71f160a27fd929c286ac52ec07b175922b38939714  
Nov 18 14:20:01 serverb.lab.example.com systemd[1433]: Started Podman container-  
web.service.
```

- 1.4. Run the `podman` command to view more information about the container.

The container is running; however, the port configuration is incorrect. The container is listening on port 8181 rather than port 8080.

```
[student@servera ~]$ podman ps  


| CONTAINER ID | IMAGE                                    | COMMAND                |
|--------------|------------------------------------------|------------------------|
| CREATED      | STATUS                                   | PORTS                  |
| NAMES        |                                          |                        |
| 24e68a162893 | registry.access.redhat.com/ubi8/httpd-24 | /usr/bin/run-http... 2 |
| minutes ago  | Up 2 minutes ago                         | 0.0.0.0:8080->8181/tcp |
|              |                                          | web                    |


```

2. Update the `systemd` service file to use the correct ports.

- 2.1. Modify the `systemd` service file so that the container listens on port 8080.

```
[student@servera ~]$ vi .config/systemd/user/container-web.service
```

- 2.2. Verify the configuration is correct.

```
[student@servera ~]$ grep -o 8080:8080 .config/systemd/user/container-web.service  
8080:8080
```

- 2.3. Restart the daemon and the service.

```
[student@servera ~]$ systemctl --user daemon-reload  
[student@servera ~]$ systemctl --user restart container-web.service
```

3. Inspect the default page that the container returns.

- 3.1. Run the `curl` command.

The output is the default Apache2 page, not the wanted output.

```
[student@servera ~]$ curl localhost:8080  
...output omitted...
```

4. Determine why the incorrect page is appearing.

- 4.1. Check the container logs.

The required volume mount has a permission error.

```
[student@servera ~]$ podman logs web
...output omitted...
[Thu Nov 18 19:24:36.929545 2021] [core:error] [pid 42:tid 140513269212928]
(13)Permission denied: [client 10.0.2.100:37912] AH00035: access to /index.html
denied (filesystem path '/var/www/html/index.html') because search permissions
are missing on a component of the path
...output omitted...
```

4.2. Inspect the container mounts.

The container mounts the `/var/webfiles` directory from the host system.

```
[student@servera ~]$ podman inspect web | grep -A 4 Mounts
"Mounts": [
{
    "Type": "bind",
    "Source": "/var/webfiles",
    "Destination": "/var/www/html",
```

4.3. Check the permissions and SELinux contexts on the `/var/webfiles` directory.

The directory's files do not have the required `container_file_t` context.

```
[student@servera ~]$ ls -lZ /var/webfiles
total 4
-rw-r--r--. 1 root root unconfined_u:object_r:var_t:s0 23 Nov 18 14:19 index.html
```

5. Update the SELinux context and verify that the container returns the desired output.

5.1. Update the SELinux context.

```
[student@servera ~]$ sudo semanage fcontext -a -t container_file_t
'/var/webfiles(/.*)?'
[sudo] password for student: student
[student@servera ~]$ sudo restorecon -Rv /var/webfiles/
Relabeled /var/webfiles from unconfined_u:object_r:var_t:s0 to
unconfined_u:object_r:container_file_t:s0
Relabeled /var/webfiles/index.html from unconfined_u:object_r:var_t:s0 to
unconfined_u:object_r:container_file_t:s0
```

5.2. Confirm that the wanted output appears.

```
[student@servera ~]$ curl localhost:8080
hello from a container
```

5.3. Return to workstation as the student user.

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

Evaluation

As the **student** user on the **workstation** machine, use the **lab** command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab grade compreview-review6
```

Finish

As the **student** user on the **workstation** machine, use the **lab** command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish compreview-review6
```

This concludes the lab.

