





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



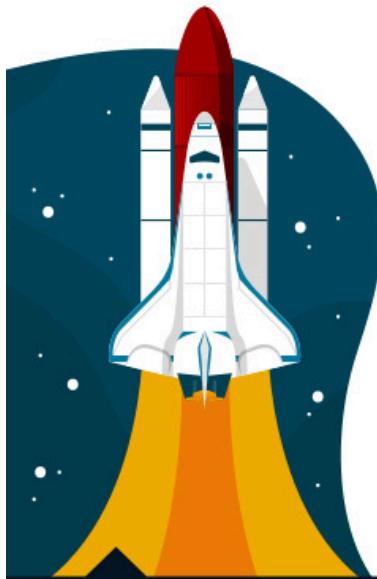
Engage in thousands of active conversations and posts



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 Security: Linux in Physical, Virtual, and Cloud



Red Hat Enterprise Linux 9.2 RH415

Red Hat Security: Linux in Physical, Virtual, and Cloud

Edition 1 20240215

Publication date 20240215

Authors: Ashish Lingayat, Austin Garrigus, Iván Chavero, Mauricio Santacruz
Course Architect: Steven Bonneville
DevOps Engineers: Artur Glogowski, Trey Feagle
Editors: Julian Cable, Rachel Lee

© 2024 Red Hat, Inc.

The contents of this course and all its modules and related materials, including handouts to audience members, are © 2024 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 [mailto: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, 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 American, Inc. and/or its affiliates.

XFS® is a registered trademark of Hewlett Packard Enterprise Development LP 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 a 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 the Square O Design, together or apart, are trademarks or registered trademarks of OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. Red Hat, Inc. is not affiliated with, endorsed by, or sponsored by the OpenStack Foundation or the OpenStack community.

All other trademarks are the property of their respective owners.

Contributors: **Jonika Shank, Roberto Velazquez, Shashi Singh, Yuvaraj Balaraju**

Document Conventions	xi
Admonitions	xi
Inclusive Language	xii
Introduction	xiii
Red Hat Security: Linux in Physical, Virtual, and Cloud	xiii
Orientation to the Classroom Environment	xiv
Performing Lab Exercises	xviii
1. Managing Security and Risk	1
Managing Security and Risk	2
Quiz: Managing Security and Risk	8
Managing RHEL Security with Red Hat Errata	10
Guided Exercise: Managing RHEL Security with Red Hat Errata	18
Reviewing Recommended Security Practices	22
Guided Exercise: Implementing Recommended Security Practices	29
Lab: Managing Security and Risk	33
Summary	39
2. Automating Configuration and Remediation with Ansible	41
Configuring Ansible for Security Automation	42
Guided Exercise: Configuring Ansible for Security Automation	55
Managing Playbooks with Automation Controller	59
Guided Exercise: Managing Playbooks with Automation Controller	71
Lab: Automating Configuration and Remediation with Ansible	74
Summary	81
3. Protecting Data with LUKS and NBDE	83
Managing Storage Device Encryption with LUKS	84
Guided Exercise: Managing Storage Device Encryption with LUKS	89
Automating Storage Device Decryption with NBDE	93
Guided Exercise: Automating Storage Device Decryption with NBDE	100
Lab: Protecting Data with LUKS and NBDE	106
Summary	114
4. Restricting USB Device Access	115
Controlling USB Access with USBGuard	116
Guided Exercise: Controlling USB access with USBGuard	124
Lab: Restricting USB Device Access	131
Summary	141
5. Controlling Authentication with PAM	143
Selecting the PAM Configuration	144
Guided Exercise: Selecting the PAM Configuration	152
Configuring Password Quality Requirements	156
Guided Exercise: Configuring Password Quality Requirements	162
Limiting Access after Failed Logins	165
Guided Exercise: Limiting Access after Failed Logins	169
Lab: Controlling Authentication with PAM	173
Summary	178
6. Recording System Events with Audit	179
Configuring Audit to Record System Events	180
Guided Exercise: Configuring Audit to Record System Events	185
Inspecting Audit Logs	191
Guided Exercise: Inspecting Audit Logs	198
Writing Custom Audit Rules	204
Guided Exercise: Writing Custom Audit Rules	209

Enabling Prepackaged Audit Rule Sets	213
Guided Exercise: Enabling Prepackaged Audit Rule Sets	216
Lab: Recording System Events with Audit	220
Summary	229
7. Monitoring File-system Changes	231
Detecting File-system Changes with AIDE	232
Guided Exercise: Detecting File-system Changes with AIDE	237
Investigating File-system Changes with AIDE	245
Guided Exercise: Investigating File-system Changes with AIDE	252
Lab: Monitoring File-system Changes	259
Summary	267
8. Mitigating Risk with SELinux	269
Enabling SELinux from the Disabled State	270
Guided Exercise: Enabling SELinux from the Disabled State	280
Controlling Access with Confined Users	286
Guided Exercise: Controlling Access with Confined Users	291
Auditing the SELinux Policy	298
Guided Exercise: Auditing the SELinux Policy	309
Lab: Mitigating Risk with SELinux	315
Summary	327
9. Managing Compliance with OpenSCAP	329
Installing OpenSCAP	330
Guided Exercise: Installing OpenSCAP	335
Scanning and Analyzing Compliance	340
Guided Exercise: Scanning and Analyzing Compliance	346
Customizing OpenSCAP Policy	349
Guided Exercise: Customizing OpenSCAP Policy	354
Remediating OpenSCAP Issues with Ansible	358
Guided Exercise: Remediating OpenSCAP Issues with Ansible	362
Lab: Managing Compliance with OpenSCAP	366
Summary	373
10. Analyzing and Remediating Issues with Red Hat Insights	375
Red Hat Insights and Security	376
Guided Exercise: Registering Systems with Red Hat Insights	380
Creating and Reviewing Red Hat Insights Reports	382
Guided Exercise: Creating and Reviewing Red Hat Insights Reports	386
Running OpenSCAP Reports from Red Hat Insights	388
Guided Exercise: Running OpenSCAP Reports from Red Hat Insights	394
Integrating Red Hat Insights and Automation Controller	401
Guided Exercise: Integrating Red Hat Insights and Automation Controller	408
Summary	413
11. Automating Compliance with Red Hat Satellite	415
Configuring Red Hat Satellite for OpenSCAP	416
Guided Exercise: Configuring Red Hat Satellite for OpenSCAP	420
Scan OpenSCAP Compliance with Red Hat Satellite	423
Guided Exercise: Scan OpenSCAP Compliance with Red Hat Satellite	428
Customize the OpenSCAP Policy in Red Hat Satellite	432
Guided Exercise: Customize OpenSCAP Policy in Red Hat Satellite	435
Lab: Automating Compliance with Red Hat Satellite	438
Summary	443
12. Comprehensive Review	445
Comprehensive Review	446

Lab: Protecting Data with LUKS and NBDE	448
Lab: Restricting USB Device Access	456
Lab: Recording Events and Monitoring File-system Changes with PAM, Audit, and AIDE ..	467
Lab: Mitigating Risk with SELinux	478
Lab: Managing Compliance with OpenSCAP and Ansible	487

Document Conventions

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

Admonitions

Red Hat Training courses use the following admonitions:



References

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



Note

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



Important

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



Warning

Do not ignore warnings. 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 that are covered in Red Hat Training courses. Red Hat appreciates your patience during this process.

Introduction

Red Hat Security: Linux in Physical, Virtual, and Cloud

Red Hat Security: Linux in Physical, Virtual, and Cloud (RH415) is designed for security administrators and system administration personnel who manage the secure operation of computer systems running Red Hat Enterprise Linux on physical hardware, as virtual machines, or as cloud instances both in private data centers and on public cloud platforms. Develop skills needed to reduce risk and to implement, manage, and remediate compliance and security issues in an efficient way at scale.

Course Objectives

- Learn how to manage the security of computer systems by using technologies, tools, and resources that help you implement and comply with security requirements.
- Prepare for the *Red Hat Certified Specialist in Security: Linux* certification exam.

Audience

- System Administrator - responsible for supporting the organization's physical and virtual computing infrastructure, systems, and servers.
- IT Security Practitioner / Compliance & Auditor - responsible for ensuring the technology environment is protected from attacks and is in compliance with security/privacy rules and regulations.
- Automation Architect - responsible for the organization's automation development and for optimizing cloud tools and infrastructure to achieve automation goals.

Prerequisites

- Red Hat Certified Engineer (EX300 / RHCE) certification or equivalent Red Hat Enterprise Linux knowledge and experience.

Orientation to the Classroom Environment

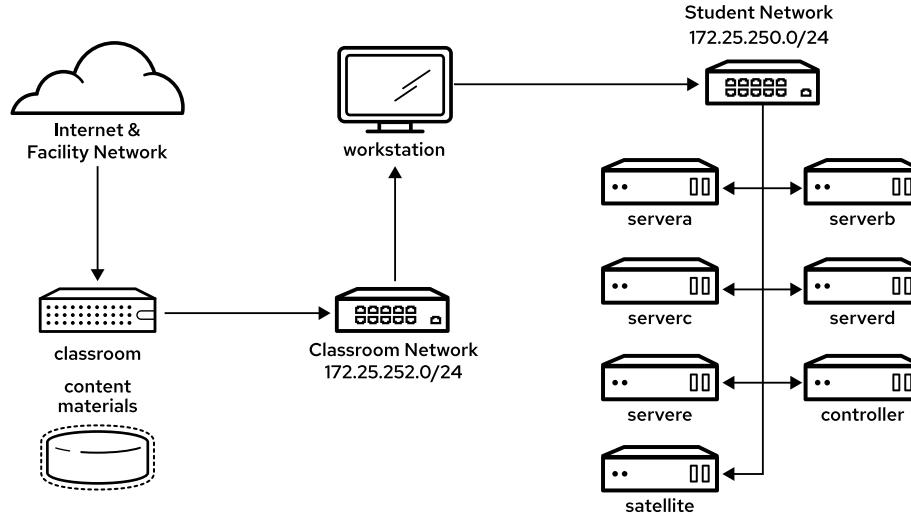


Figure 0.1: Classroom environment

In this course, the main computer system used for hands-on learning activities is **workstation**. The systems called **bastion**, **classroom**, and **utility** must always be running for proper use of the lab environment. Seven other machines are also used by students for these activities: **servera**, **serverb**, **serverc**, **serverd**, **servere**, **controller**, and **satellite**. All eleven of these systems are in the `lab.example.com` DNS domain.

All student computer systems have a standard user account, **student**, which has the **student** password. The **root** password on all student systems is **redhat**. The **admin** user password for Red Hat Satellite Server and for the automation controller is **redhat**.

Classroom Machines

Machine name	IP addresses	Role
<code>workstation.lab.example.com</code>	172.25.250.254	Graphical workstation used 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"
<code>serverc.lab.example.com</code>	172.25.250.12	Managed server "C"
<code>serverd.lab.example.com</code>	172.25.250.13	Managed server "D"
<code>servere.lab.example.com</code>	172.25.250.16	Managed server "E"
<code>controller.lab.example.com</code>	172.25.250.14	Automation controller

Introduction

Machine name	IP addresses	Role
satellite.lab.example.com	172.25.250.15	Red Hat Satellite Server
utility.lab.example.com	172.25.250.220	Utility server

One additional function of **workstation** is that it acts as a router between the network that connects the student machines and the classroom network. If **workstation** is down, then other student machines will only be able to access systems on the student network.

Controlling Your Systems

You are assigned remote computers in a Red Hat Online Learning (ROLE) classroom. Self-paced courses are accessed through a web application that is hosted at rol.redhat.com [<http://rol.redhat.com>]. 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.

The screenshot shows a web-based interface for managing a lab environment. At the top, there are tabs for 'Table of Contents', 'Course', 'Lab Environment', and icons for star and question mark. Below the tabs, a section titled '▶ Lab Controls' contains instructions: 'Click CREATE to build all of the virtual machines needed for the classroom lab environment. This may take several minutes to complete. Once created the environment can then be stopped and restarted to pause your experience.' It also states that deleting the lab will remove all virtual machines and lose progress. Below this is a button bar with 'DELETE' (red), 'STOP' (teal), and an info icon. A table lists five virtual machines: 'bastion' (active), 'classroom' (active), 'servera' (building), 'serverb' (building), and 'workstation' (active). Each row has an 'ACTION -' dropdown and an 'OPEN CONSOLE' button.

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 it just started, it still might be starting services.
stopped	The virtual machine is shut down. On starting, the virtual machine boots into the same state it was in before shutdown. The disk state is preserved.

Classroom Actions

Button or action	Description
CREATE	Create the ROLE classroom. Creates and starts all the virtual machines that are needed for this classroom.
CREATING	The ROLE classroom virtual machines are being created. Creation can take several minutes to complete.
DELETE	Delete the ROLE classroom. Destroys all virtual machines in the classroom. All saved work on those systems' 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 action is equivalent to removing the power from a physical machine.
ACTION > Reset	Forcefully shut down the virtual machine and reset associated storage 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 that 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, then click DELETE to remove the entire classroom environment. After the lab is deleted, then 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 environment 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

You might see the following lab activity types in this course:

- A *guided exercise* is a hands-on practice exercise that follows a presentation section. It walks you through a procedure to perform, step by step.
- A *quiz* is typically used when checking knowledge-based learning, or when a hands-on activity is impractical for some other reason.
- An *end-of-chapter lab* is a gradable hands-on activity to help you to check your learning. You work through a set of high-level steps, based on the guided exercises in that chapter, but the steps do not walk you through every command. A solution is provided with a step-by-step walk-through.
- A *comprehensive review lab* is used at the end of the course. It is also a gradable hands-on activity, and might cover content from the entire course. You work through a specification of what to accomplish in the activity, without receiving the specific steps to do so. Again, a solution is provided with a step-by-step walk-through that meets the specification.

To prepare your lab environment at the start of each hands-on activity, run the `lab start` command with a specified activity name from the activity's instructions. Likewise, at the end of each hands-on activity, run the `lab finish` command with that same activity name to clean up after the activity. Each hands-on activity has a unique name within a course.

The syntax for running an exercise script is as follows:

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

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

The `start` action 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. You can perform an exercise at any time, even without performing preceding exercises.

grade

For gradable activities, the `grade` action directs the `lab` command to evaluate your work, and shows 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

The `finish` action cleans up resources that were configured during the exercise. You can perform an exercise as many times as you want.

The `lab` command supports tab completion. For example, to list all exercises that you can start, enter `lab start` and then press the Tab key twice.

Chapter 1

Managing Security and Risk

Goal

Define and implement strategies to manage security on Red Hat Enterprise Linux systems.

Sections

- Managing Security and Risk (and Quiz)
- Managing RHEL Security with Red Hat Errata (and Guided Exercise)
- Reviewing Recommended Security Practices (and Guided Exercise)

Lab

- Managing Security and Risk

Managing Security and Risk

Objectives

- Describe fundamental concepts of security management for Red Hat Enterprise Linux servers and how to approach the security management process.

Risk Management

A *risk* is any event that presents the possibility of loss, such as financial loss, productivity loss, or service delays. Continuous risk management is the process of taking a proactive approach to discover potential risk, assess facts, and act based on the facts to resolve those risks. Monitoring risk indicators and the actions that are taken to resolve them are important for effective risk management. Communicating to stakeholders what those risk indicators are, the actions that are taken to resolve them, and the results, all promote confidence and stability in the process.

The following diagram illustrates a process for continuously managing security risks by maintaining constant attention to potential security vulnerabilities and taking a proactive approach to maintaining a secure computing environment.

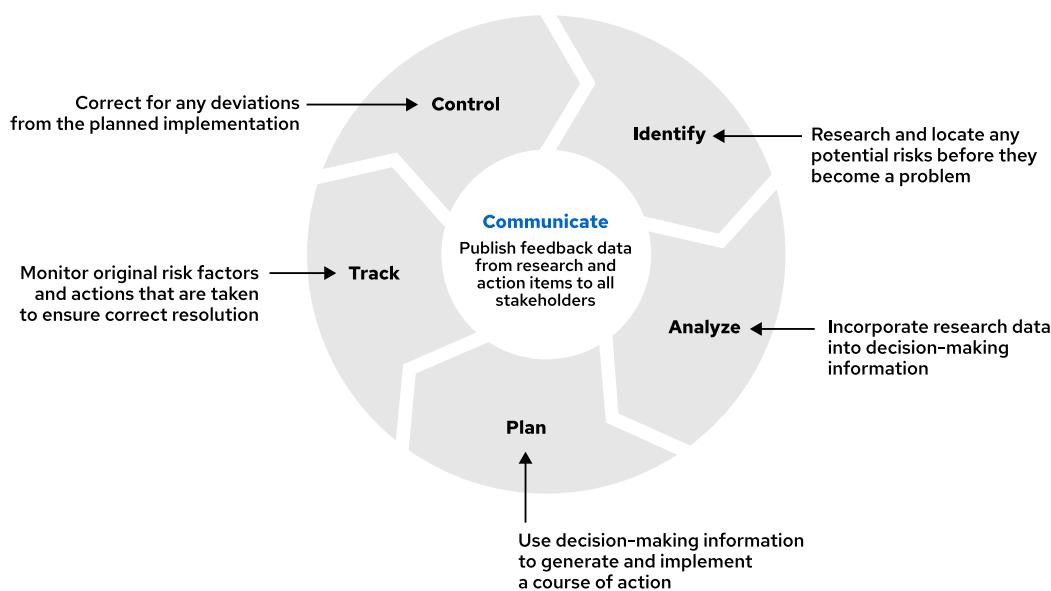


Figure 1.1: Continuous risk management lifecycle

Managing Security

Managing security in your computing environments is a collection of continuous activities. Red Hat offers solutions that enable security in each phase of the continuous security lifecycle. The following diagram illustrates a continuous security lifecycle that incorporates the risk management lifecycle.

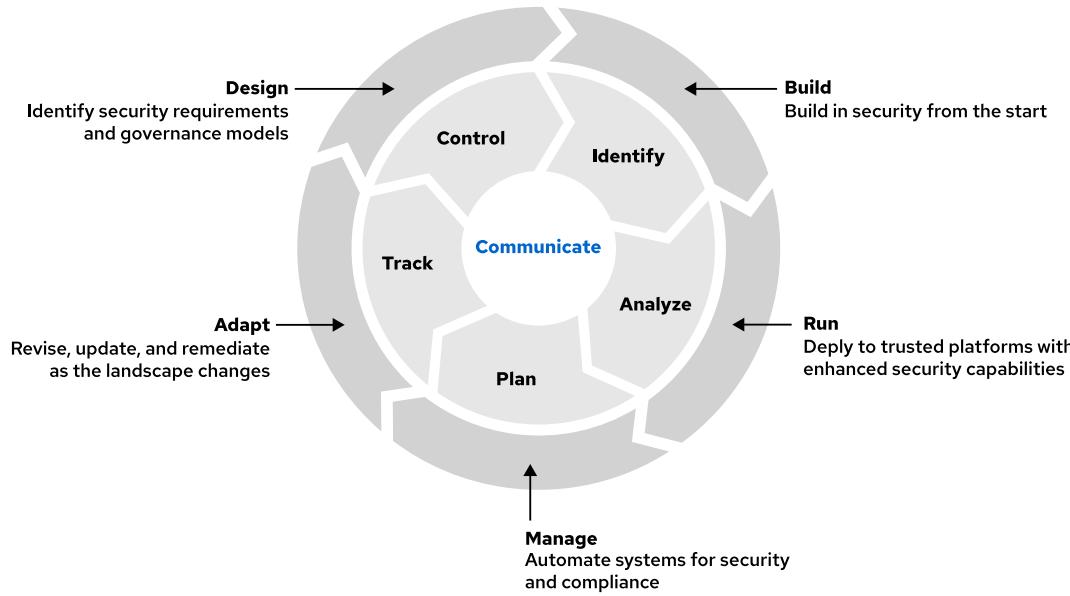


Figure 1.2: Continuous security and risk management lifecycle

Continuous Security

Security must be both proactive and reactive. It must be considered at every stage of your application and infrastructure lifecycle. You must integrate security experts into your application, deployment, and infrastructure teams.

Design

Develop security requirements, design security processes and procedures, and communicate so that all team members take these considerations into account when designing applications and infrastructure for continuous security and compliance. Collaboratively develop and communicate your security policy and procedures.

In this stage, you might consider what security policies you must comply with, and implement the controls and technologies to maintain compliance.

Red Hat consultants and partners are available to help customers design security into their applications and infrastructure as well as into their application development lifecycle and operations management process.

Build

Build security features into your applications and your infrastructure.

- Automate and integrate security testing into your continuous integration and continuous deployment process. Use test outcomes to trigger appropriate action: deploy or prevent deployment.
- Define security profiles and automate the configuration of those profiles whenever new systems are deployed.

You implement your designs and build your applications, and determine how to deploy them to production in a repeatable, reliable, and secure way. In this stage, you could also deploy and provision the infrastructure to support these servers, by using tools such as Kickstart, Red Hat Satellite, and Red Hat Ansible Automation Platform.

Chapter 1 | Managing Security and Risk

Red Hat provides stable and safe development platforms and middleware to design and create your applications.

Run

Run on trusted platforms with built-in security features, such as SELinux.

You can take advantage of the tools that Red Hat provides to help operate your systems with full support. Red Hat helps to secure your systems by providing tools such as SELinux, which help to protect your systems from hackers who exploit security-related issues.

Manage

Maintain an up-to-date catalog of assets, and monitor access and usage. Deploy a management solution that provides a single management interface across your footprints: physical, virtual, and private and public cloud. You can actively manage users and access.

Red Hat Satellite can help you to track systems that are deployed in your environment and to keep them up-to-date with errata. You can use other management tools, such as Red Hat Ansible Automation Platform, to remediate configuration drift and to ensure that your servers and applications are correctly configured and deployed.

Using these solutions, you can automatically apply, audit, and remediate security and regulatory compliance policies across workloads, whether deployed to bare-metal or virtual servers, with a container, on-premise, or in a public, private, or hybrid cloud.

Adapt

Use analytics and automation to adapt, revise, update, and remediate as the landscape changes. You must update access settings when roles and responsibilities change.

You can use tools such as Red Hat Insights to proactively adapt to emerging issues such as common misconfigurations or new security issues that Red Hat identifies.

As you adapt to the ever-changing security landscape, you can revisit the design stage, and start the cycle over again.

How Red Hat Can Help You Manage Security

Securing computing infrastructures has changed significantly in recent years. Keeping your operating systems updated with the latest security patches is no longer sufficient. Operating system providers must be more proactive in combating security problems.

The Red Hat Product Security team analyzes threats and vulnerabilities against all products, and provides relevant advice and updates through the Red Hat Customer Portal. Red Hat provides patches to supported versions of Red Hat products, often on the same day that the vulnerability is first published. These patches minimize disruption, because enterprises can continue to work safely with current versions, and upgrade to later versions when the business is ready.

The Red Hat Product Security team provides the needed guidance, stability, and security to deploy enterprise solutions.

- Help to protect customers from security concerns when using Red Hat products and services.
- Investigate, track, and explain security issues that might affect users of Red Hat supported products and services.
- Be the point of contact for customers, users, and researchers who find security issues in Red Hat products and services, and publish the procedures for resolving those issues.
- Ensure timely security fixes, and help customers to find, obtain, and publish security advisories and updates.

- Help customers keep their systems updated to minimize the risk of security issues and to provide automated analysis of security practices.
- Work with other Linux vendors and open source software to reduce the risk of security issues through information sharing and peer review.

Red Hat Security Reporting

Red Hat takes security seriously, and aims to take immediate action to address security-related problems that involve Red Hat products and services.

Red Hat Security Response

The Red Hat Product Security team is a point of contact for security issues with Red Hat products. This team investigates and verifies the issues. The team analyzes the affected products, identifies the impact, and provides the required corrective action. When a security update is released, the team also publishes a public notification in a Red Hat Security Advisory (RHSA).

Report any suspected security vulnerability for a Red Hat product or service to Red Hat Product Security at secalert@redhat.com. Only members of the Red Hat Product Security team, which is a restricted and carefully chosen group of Red Hat employees, have access to material that is sent to the secalert@redhat.com email address.

Email that is sent to secalert@redhat.com is read and acknowledged by a team member within three working days. For complex issues that require significant attention, Red Hat opens an investigation and provides you with a mechanism to check the status of the team's progress at any time. Any information that you share with Red Hat about security issues that are not public knowledge is kept confidential within Red Hat. This information is not provided to any third parties without your permission.

Communicating Risk Information

Red Hat Product Security provides objective information about security risks that affect you, regardless of possible media hype. Red Hat uses the following workflow to communicate accurate information about how these vulnerabilities affect you, so that you can make informed decisions.

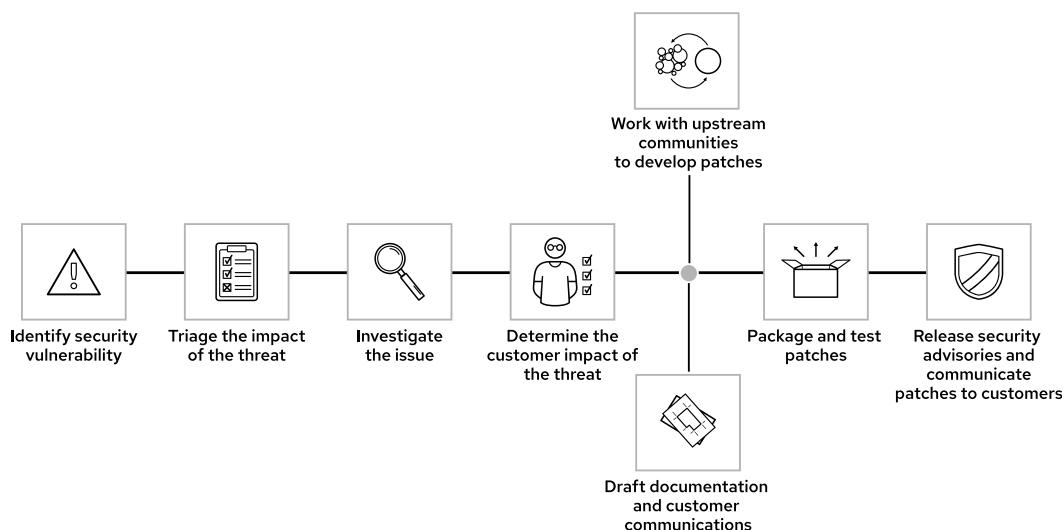


Figure 1.3: Security risk awareness workflow

Red Hat Security Severity Ratings

The Red Hat Product Security team rates the impact of security issues in Red Hat products by using a 4-point scale (Critical, Important, Moderate, and Low), as well as *Common Vulnerability Scoring System* (CVSS) scores.

These ratings provide a prioritized risk assessment to help you to understand and schedule upgrades to your systems, so that you can make informed decisions about the risk that each issue poses to your unique environment.

The ratings consider the potential risk based on a technical analysis of the exact flaw and its type, but not the current threat level. A given rating might not change if an exploit or worm is released later for a flaw, or if the flaw is available before the release of a fix.

Issue Severity Classification

Severity rating	Description
Critical impact	<p>This rating is given to flaws that a remote unauthenticated attacker could exploit, and that could compromise the system (arbitrary code execution) without requiring user interaction.</p> <p>Worms could exploit these types of vulnerabilities: Flaws that require an authenticated remote user, a local user, or an unlikely configuration are not classed as Critical impact.</p>
Important impact	<p>This rating is given to flaws that can compromise the confidentiality, integrity, or availability of resources.</p> <p>These flaws are the types of vulnerabilities that allow local users to gain privileges, by allowing unauthenticated remote users to view resources, execute arbitrary code, or cause a denial of service.</p>
Moderate impact	<p>This rating is given to flaws that might be more difficult to exploit, but could lead to some compromise of the confidentiality, integrity, or availability of resources under certain circumstances.</p> <p>These types of vulnerabilities could have a Critical or Important impact, but are less exploited, based on a technical evaluation of the flaw, or they affect unlikely configurations.</p>
Low impact	<p>This rating is given to all other issues that have a security impact.</p> <p>These types of vulnerabilities are believed to require unlikely circumstances to be exploited, or are cases where a successful exploit would have minimal consequences.</p>

A Red Hat security advisory can contain fixes for several vulnerabilities, and for packages for several products. Each issue in an advisory has an impact rating for each product. The overall severity of an advisory is the highest severity out of all the individual issues, across all the products that the advisory targets. For simplicity, advisories show only the overall severity (except for kernel advisories, which list the severity of each issue). The advisories contain links to the relevant entries in Red Hat's bug-tracking system, where you can examine individual impacts and additional commentary.

When a technology is enabled by default that blocks the exploitation of a vulnerability across all architectures, Red Hat adjusts the severity level of that vulnerability. When a technology reduces the risk of a vulnerability, Red Hat adjusts the severity level and explains the decision in the bug-tracking entry.



References

What Is Risk Management?

<https://www.redhat.com/en/topics/management/what-is-risk-management>

Product Security Center

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

Product Security Overview

<https://access.redhat.com/security/overview/>

Security Contacts and Procedures

<https://access.redhat.com/security/team/contact/#contact>

Notifications and Advisories

<https://access.redhat.com/security/updates/advisory>

Security Data

<https://access.redhat.com/security/data>

Severity Ratings

<https://access.redhat.com/security/updates/classification>

Red Hat Security

<https://www.redhat.com/en/blog/channel/security>

Red Hat Enterprise Linux Life Cycle

<https://access.redhat.com/support/policy/updates/errata>

► Quiz

Managing Security and Risk

Choose the correct answers to the following questions:

- ▶ 1. During which security management lifecycle stage do you adapt, revise, update, and remediate systems as the security landscape changes?
 - a. Build
 - b. Run
 - c. Manage
 - d. Adapt
 - e. Design

- ▶ 2. To send a report of any suspected security vulnerability in a Red Hat product or service to the Red Hat Product Security team, which email address do you use?
 - a. rhsecure@redhat.com
 - b. security@redhat.com
 - c. rhtraining@redhat.com
 - d. secalert@redhat.com

- ▶ 3. Which three statements describe a Critical impact severity classification? (Choose three.)
 - a. Requires an authenticated remote user.
 - b. Does not require user interaction to invoke a possible system compromise by arbitrary code execution.
 - c. Exposes a vulnerability that worms can exploit.
 - d. Requires a local user.
 - e. Is easily exploited by an unauthenticated attacker.

- ▶ 4. Red Hat uses which numbering and naming standard to consistently report and track security-related software issues?
 - a. Customized email notifications from the rhsecurity@redhat.com bug-tracking team
 - b. Common Vulnerabilities and Exposures (CVE)
 - c. Common Vulnerability Scoring System (CVSS)
 - d. Red Hat Security Vulnerability Response (RHSV)

► Solution

Managing Security and Risk

Choose the correct answers to the following questions:

- ▶ 1. **During which security management lifecycle stage do you adapt, revise, update, and remediate systems as the security landscape changes?**
 - a. Build
 - b. Run
 - c. Manage
 - d. Adapt
 - e. Design

- ▶ 2. **To send a report of any suspected security vulnerability in a Red Hat product or service to the Red Hat Product Security team, which email address do you use?**
 - a. rhsecure@redhat.com
 - b. security@redhat.com
 - c. rhtraining@redhat.com
 - d. secalert@redhat.com

- ▶ 3. **Which three statements describe a Critical impact severity classification? (Choose three.)**
 - a. Requires an authenticated remote user.
 - b. Does not require user interaction to invoke a possible system compromise by arbitrary code execution.
 - c. Exposes a vulnerability that worms can exploit.
 - d. Requires a local user.
 - e. Is easily exploited by an unauthenticated attacker.

- ▶ 4. **Red Hat uses which numbering and naming standard to consistently report and track security-related software issues?**
 - a. Customized email notifications from the rhsecurity@redhat.com bug-tracking team
 - b. Common Vulnerabilities and Exposures (CVE)
 - c. Common Vulnerability Scoring System (CVSS)
 - d. Red Hat Security Vulnerability Response (RHSV)

Managing RHEL Security with Red Hat Errata

Objectives

- Describe Red Hat's development process and security response practices, and install Red Hat errata.

Red Hat's Secure Development Lifecycle

The Red Hat Product Security team collaborates with the relevant teams to promote secure development practices and software features that meet customers' business needs. Red Hat Secure Software Management Lifecycle (SSML), a Software Development Lifecycle (SDL) approach, aligns with the NIST Secure Software Development Framework (NIST SSDF SP-800-218) as well as OWASP guidance and various ISO standards. The goal of these SDL practices and standards is to reduce the number of vulnerabilities in released software, to mitigate the potential impact of undetected or unaddressed vulnerabilities, and to address the root causes of vulnerabilities to prevent future recurrences.

Responding to Vulnerabilities

Every day, new and increasingly complex vulnerabilities are discovered that affect computer systems. The process of interpreting and addressing threats in a vulnerability scan can be challenging. Scan results with more findings might seem more accurate than scans with fewer findings, but this conclusion could be misleading. A truly accurate scan might remove duplicates, use vendor-disclosed data, and assist the security professional in prioritizing the elimination of risk.

However, vulnerability scanning can also generate false positives, false negatives, and data discrepancies, which makes the results difficult to compare and understand. Vulnerability scanning becomes even more challenging when the complexity of the software increases. For example, containerized environments such as Red Hat OpenShift include platform dependencies in addition to container packages.

- Red Hat monitors weaknesses and flaws before they become known vulnerabilities.
- Regardless of whether an offering includes an affected component, Red Hat's analysis includes a review of the environmental aspects of how the component is built, the libraries that are used, and any default configurations.
- All vulnerabilities receive a Common Vulnerabilities and Exposures (CVE) record.
- Red Hat provides a severity rating, which prioritizes the risk assessment for customers. This rating is based on both the base flaw metrics and the additional environmental aspects that were previously mentioned.
- Red Hat openly publishes vulnerabilities by using industry standards and formats.

Backporting Security Fixes

Red Hat uses the term *backporting* to describe when it takes a fix for a security flaw out of the most recent version of an upstream software package, and applies that fix to an earlier version of the package that Red Hat distributes.

In such cases, Red Hat can backport the updates. When backporting security fixes, Red Hat takes the following steps:

- Identify the fixes and isolate them from any other changes.
- Ensure that the fixes do not introduce unwanted side effects.
- Apply the fixes to previously released versions.

For example, Red Hat provided version 5.3 of PHP in Red Hat Enterprise Linux 6. The upstream developers of PHP stopped supporting PHP 5.3 on August 14, 2014, and were no longer releasing updates of any kind for PHP 5.3. At that point, if security issues occurred, then the upstream project fixed them only in later versions of PHP.

On October 14, 2014, a buffer overflow flaw was found in PHP that Red Hat rated as Important. This issue could allow a remote attacker to crash a PHP application, or possibly execute arbitrary code with the privileges of the user that runs the PHP application.

The upstream project developed a fix, and released it for PHP 5.4. However, because the upstream project no longer supported PHP 5.3, the project developers did not update that version of PHP.

However, Red Hat still had customers who used PHP 5.3. Red Hat did not want to force them to migrate to PHP 5.4 due to the risk of backward compatibility problems between versions 5.3 and 5.4. Therefore, Red Hat backported the fix for this issue from PHP 5.4 to PHP 5.3 without changing other features, and released updated packages for Red Hat Enterprise Linux 6 customers. Backporting enabled Red Hat customers to continue to run PHP 5.3 and not be vulnerable to that particular issue, even though the last upstream release of PHP 5.3 was still vulnerable to it.

For most products, although the default practice is to backport security fixes, Red Hat does sometimes provide version updates for some packages after testing and analysis. Red Hat might update a package instead of backporting fixes to it when those updates have minimal impact on the behavior of the system, or if users of that package also expect frequent feature enhancements.

The Relationship Between Software Versions and Vulnerabilities

Although backporting brings advantages, it can create confusion when it is not understood. Customers must know that they cannot determine the vulnerability of a package from its version number alone. This version number can cause confusion, because even after installing updated packages from a vendor, customers might not have the latest upstream version. Customers might instead have an earlier upstream version with applied backported patches.

Some security scanning and auditing tools make decisions about vulnerabilities based solely on the version number of the components. These tools generate false positives because the tools do not consider backported security fixes.

For each version of Red Hat Enterprise Linux, Red Hat explains in security advisories how it fixed an issue: either by moving to a new upstream version or by backporting patches to the existing version. Red Hat now attaches CVE records to all advisories, so that users can refer to vulnerabilities and their fixes independently from version numbers.

Red Hat Advisories and CVEs

Red Hat periodically releases *errata*, which are corrections or updates to software packages based on security issues, bugs, or the availability of new features. Each errata is associated with an *advisory*, a text description of what the issue is, which fix is being provided, the products that are affected, and other relevant information.

Red Hat releases three types of advisories:

Red Hat Security Advisory (RHSA)

RHSAs contain security fixes and might also contain bug or enhancement fixes. Many organizations consider RHSAs to be the most important type of errata. RHSAs are ranked with a rating of Critical, Important, Moderate, and Low in decreasing severity order of the vulnerability.

The RHSA provides a unique advisory ID, a severity rating, an issue date, a brief description of the issue that the updated packages fix, and the list of included packages in the advisory that require updating. Customers use the severity rating to determine the relevance of a specific security update to their environment.

Red Hat Enhancement Advisory (RHEA)

RHEAs contain enhancements or new features and do not contain bug fixes or security fixes. An RHEA is released when new features are added and an updated package is shipped.

Red Hat Bug Advisory (RHBA)

RHBAs always contain bug fixes and might contain enhancements, but do not contain security fixes. Because RHBAs are released for bug fixes, they are often considered a higher priority than an RHEA.

Red Hat customers can subscribe to email notifications for new or updated advisories on the Customer Portal. Log in to <https://access.redhat.com>, and click your account icon in the upper-right corner of the web UI. Select **Account details**, and then on the next page select **Errata notifications**. If you are logged in to the Customer Portal, you can also go directly to the URL <https://www.redhat.com/wapps/ugc/protected/notif.html> to reach that page.

All email advisories from Red Hat are digitally signed.

CVE Records and OVAL Definitions

The CVE Program is a cross-vendor organization whose mission is to identify, define, and catalog publicly disclosed cybersecurity vulnerabilities. They maintain a Common Vulnerabilities and Exposures database that contains a *CVE record*, or CVE, for each identified vulnerability. Each CVE record consists of an ID number (including the year and a sequence number to uniquely identify it), a short description to summarize the issue, and a list of links to content that is relevant to the vulnerability. More information is available at <https://www.cve.org>.

A Red Hat advisory references which CVEs it resolves, which bugs are addressed, or which features the relevant errata added.



Important

A single CVE record might result in multiple Red Hat Security Advisories when the issue is addressed in different products.

Likewise, a single Red Hat Security Advisory might address several CVE records in one software update. Red Hat might also update multiple related software packages as part of the advisory.

Security Data

Red Hat aims to provide clarity about backporting, and to help customers to keep up-to-date with security fixes. Red Hat provides various electronic documents to help vulnerability scanners and other tools determine which issues might affect a system.

CSAF/VEX Documents

The Common Security Advisory Framework (CSAF) standard enables organizations to share information about security issues by using a consistent and common format. Red Hat provides security advisories in CSAF format by using the Vulnerability Exploitability eXchange (VEX) profile.

OVAL Definitions

Open Vulnerability and Assessment Language (OVAL) definitions are available for vulnerabilities that were addressed in errata for Red Hat Enterprise Linux and certain additional products.

Red Hat publishes vulnerability data in different formats in the following locations:

- Red Hat CVE Database [<https://access.redhat.com/security/security-updates/cve>]
- Red Hat Security Data API [https://access.redhat.com/documentation/en-us/red_hat_security_data_api/1.0/html/red_hat_security_data_api/index]
- Index of CSAF v2 [<https://access.redhat.com/security/data/csaf/v2/>]
- Index of OVAL v2 [<https://access.redhat.com/security/data/oval/v2/>]



Important

Red Hat uses OVAL/DS v2 security data. OVAL/DS v1 security data is deprecated and archived.

Common Vulnerability Reporting Framework (CVRF) files are discontinued from 1 September 2023; customers must migrate to using CSAF files.

Using DNF to Manage Security Errata

The `dnf` package manager includes several security-related features to search, list, display, and install security errata.

A Red Hat subscription must be attached to the host to manage security updates.

Identifying Security Updates

The following example lists all available security updates that are not installed:

```
[root@host ~]# dnf updateinfo list updates security
...output omitted...
RHSA-2023:4412 Important/Sec. openssh-8.7p1-30.el9_2.x86_64
RHSA-2023:4412 Important/Sec. openssh-clients-8.7p1-30.el9_2.x86_64
RHSA-2023:4412 Important/Sec. openssh-server-8.7p1-30.el9_2.x86_64
RHSA-2023:3722 Moderate/Sec. openssl-1:3.0.7-16.el9_2.x86_64
RHSA-2023:3722 Moderate/Sec. openssl-libs-1:3.0.7-16.el9_2.x86_64
RHSA-2023:3595 Important/Sec. python-unversioned-command-3.9.16-1.el9_2.1.noarch
RHSA-2023:3595 Important/Sec. python3-3.9.16-1.el9_2.1.x86_64
RHSA-2023:3595 Important/Sec. python3-libs-3.9.16-1.el9_2.1.x86_64
```

Chapter 1 | Managing Security and Risk

```
RHSA-2023:4349 Moderate/Sec. python3-libxml2-2.9.13-3.el9_2.1.x86_64
RHSA-2023:3723 Important/Sec. python3-perf-5.14.0-284.18.1.el9_2.x86_64
RHSA-2023:4377 Important/Sec. python3-perf-5.14.0-284.25.1.el9_2.x86_64
RHSA-2023:4350 Moderate/Sec. python3-requests-2.25.1-7.el9_2.noarch
...output omitted...
```

The following example lists all security updates that are installed:

```
[root@host ~]# dnf updateinfo list security --installed
...output omitted...
RHSA-2023:2645 Moderate/Sec. openssh-8.7p1-29.el9_2.x86_64
RHSA-2023:2645 Moderate/Sec. openssh-clients-8.7p1-29.el9_2.x86_64
RHSA-2023:2645 Moderate/Sec. openssh-server-8.7p1-29.el9_2.x86_64
RHSA-2022:6224 Moderate/Sec. openssl-1:3.0.1-41.el9_0.x86_64
RHSA-2022:7288 Important/Sec. openssl-1:3.0.1-43.el9_0.x86_64
RHSA-2023:0946 Important/Sec. openssl-1:3.0.1-47.el9_1.x86_64
RHSA-2023:2523 Low/Sec. openssl-1:3.0.7-6.el9_2.x86_64
RHSA-2022:6224 Moderate/Sec. openssl-libs-1:3.0.1-41.el9_0.x86_64
RHSA-2022:7288 Important/Sec. openssl-libs-1:3.0.1-43.el9_0.x86_64
RHSA-2023:0946 Important/Sec. openssl-libs-1:3.0.1-47.el9_1.x86_64
RHSA-2023:2523 Low/Sec. openssl-libs-1:3.0.7-6.el9_2.x86_64
...output omitted...
```

The following example shows information for a specific advisory:

```
[root@host ~]# dnf updateinfo info RHSA-2023:4412
Last metadata expiration check: 0:06:16 ago on Wed Aug 9 00:15:00 2023.
=====
Important: openssh security update
=====
Update ID: RHSA-2023:4412
Type: security
Updated: 2023-08-01 09:35:02
Bugs: 2224173 - CVE-2023-38408 openssh: Remote code execution in ssh-agent
PKCS#11 support
CVEs: CVE-2023-38408
Description: OpenSSH is an SSH protocol implementation supported by a number of
Linux, UNIX, and similar operating systems. It includes the core files necessary
for both the OpenSSH client and server.
:
: Security Fix(es):
:
: * openssh: Remote code execution in ssh-agent PKCS#11 support
(CVE-2023-38408)
:
: For more details about the security issue(s), including the impact,
a CVSS score, acknowledgments, and other related information, refer to the CVE
page(s) listed in the References section.
Severity: Important
```

The following example identifies RHSAs with the Important severity rating:

```
[root@host ~]# dnf updateinfo list updates security | grep Important
...output omitted...
RHSA-2023:4412 Important/Sec. openssh-8.7p1-30.el9_2.x86_64
RHSA-2023:4412 Important/Sec. openssh-clients-8.7p1-30.el9_2.x86_64
RHSA-2023:4412 Important/Sec. openssh-server-8.7p1-30.el9_2.x86_64
...output omitted...
```

The following example identifies the required packages that resolve the CVE-2023-38408 CVE:

```
[root@host ~]# dnf updateinfo list updates security --cve CVE-2023-38408
...output omitted...
RHSA-2023:4354 Moderate/Sec. libcurl-7.76.1-23.el9_2.x86_64
RHSA-2023:4347 Moderate/Sec. libeconf-0.4.1-3.el9_2.x86_64
RHSA-2023:4325 Moderate/Sec. libsmclient-4.17.5-103.el9_2.x86_64
RHSA-2023:4325 Moderate/Sec. libwbclient-4.17.5-103.el9_2.x86_64
RHSA-2023:4349 Moderate/Sec. libxml2-2.9.13-3.el9_2.1.x86_64
RHSA-2023:4412 Important/Sec. openssh-8.7p1-30.el9_2.x86_64
RHSA-2023:4412 Important/Sec. openssh-clients-8.7p1-30.el9_2.x86_64
RHSA-2023:4412 Important/Sec. openssh-server-8.7p1-30.el9_2.x86_64
RHSA-2023:3722 Moderate/Sec. openssl-1:3.0.7-16.el9_2.x86_64
RHSA-2023:3722 Moderate/Sec. openssl-libs-1:3.0.7-16.el9_2.x86_64
...output omitted...
```

Installing Security Updates

The following example installs the security update for a specific advisory:

```
[root@host ~]# dnf update --advisory=RHSA-2023:3722
Last metadata expiration check: 0:21:29 ago on Wed Aug 9 00:15:00 2023.
Dependencies resolved.
=====
Package      Arch    Version           Repository          Size
=====
Upgrading:
openssl      x86_64  1:3.0.7-17.el9_2   rhel-9-for-x86_64-baseos-rpms 1.2 M
openssl-libs x86_64  1:3.0.7-17.el9_2   rhel-9-for-x86_64-baseos-rpms 2.2 M

Transaction Summary
=====
Upgrade 2 Packages
...output omitted...

Upgraded:
openssl-1:3.0.7-17.el9_2.x86_64      openssl-libs-1:3.0.7-17.el9_2.x86_64

Complete!
```

The following example installs all security updates:

```
[root@host ~]# dnf update --security
Last metadata expiration check: 0:30:20 ago on Wed Aug 9 00:15:00 2023.
Dependencies resolved.
```

```
=====
Package      Arch   Version            Repository      Size
=====
Installing:
kernel       x86_64  5.14.0-284.25.1.el9_2  rhel-9-for-x86_64-baseos-rpms 3.4 M
kernel-core   x86_64  5.14.0-284.25.1.el9_2  rhel-9-for-x86_64-baseos-rpms 17 M
kernel-modules
              x86_64  5.14.0-284.25.1.el9_2  rhel-9-for-x86_64-baseos-rpms 37 M
...output omitted...

Upgraded:
...output omitted...
python-unversioned-command-3.9.16-1.el9_2.1.noarch
python3-3.9.16-1.el9_2.1.x86_64
python3-libs-3.9.16-1.el9_2.1.x86_64
python3-libxml2-2.9.13-3.el9_2.1.x86_64
python3-perf-5.14.0-284.25.1.el9_2.x86_64
python3-requests-2.25.1-7.el9_2.noarch
samba-client-libs-4.17.5-103.el9_2.x86_64
samba-common-4.17.5-103.el9_2.noarch
samba-common-libs-4.17.5-103.el9_2.x86_64
webkit2gtk3-jsc-2.38.5-1.el9_2.3.x86_64

Installed:
grub2-tools-efi-1:2.06-61.el9_2.1.x86_64
grub2-tools-extra-1:2.06-61.el9_2.1.x86_64
kernel-5.14.0-284.25.1.el9_2.x86_64
kernel-core-5.14.0-284.25.1.el9_2.x86_64
kernel-modules-5.14.0-284.25.1.el9_2.x86_64
kernel-modules-core-5.14.0-284.25.1.el9_2.x86_64

Complete!
```



References

An Overview of Red Hat's Secure Development Lifecycle (SDL) Practices

https://access.redhat.com/articles/red_hat SDL

Tutorial on How to Process Vulnerability Scans

https://access.redhat.com/articles/red_hat_vulnerability_tutorial

What Is Backporting and How Does It Affect Red Hat Enterprise Linux (RHEL)?

<https://access.redhat.com/solutions/57665>

Backporting Security Fixes

<https://access.redhat.com/security/updates/backporting/>

Red Hat Security Advisories

<https://access.redhat.com/security/security-updates/security-advisories>

The CVE Program

<https://www.cve.org>

Red Hat CVE Database

<https://access.redhat.com/security/security-updates/cve>

Red Hat and OVAL Compatibility

<https://access.redhat.com/articles/221883>

Managing and Monitoring Security Updates (RHEL 9)

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html-single/managing_and_monitoring_security_updates/index

► Guided Exercise

Managing RHEL Security with Red Hat Errata

Query a RHEL system for security threats, and install Red Hat Security errata to resolve the Important security issues.

Outcomes

- Identify all Critical, Important, and Moderate security notices.
- Update the system to eliminate potential Important security vulnerabilities.

Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

```
[student@workstation ~]$ lab start securityrisk-errata
```

Instructions

- 1. Identify all Critical, Important, and Moderate security updates on the `servera` machine.

- 1.1. Log in to the `servera` machine as the `student` user.

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

- 1.2. Change to the `root` user. Use `student` as the password.

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

- 1.3. List the summary of the security updates. Among the 16 total security notices, eight are Important and eight are Moderate. The total security number might be different on your system.

```
[root@servera ~]# dnf updateinfo --security  
...output omitted...  
Updates Information Summary: available  
 16 Security notice(s)  
    8 Important Security notice(s)  
    8 Moderate Security notice(s)
```

- 2. List the security-related packages that are available to update.

```
[root@servera ~]# dnf updateinfo list updates security
...output omitted...
RHSA-2023:3725 Moderate/Sec. less-590-2.el9_2.x86_64
RHSA-2023:4354 Moderate/Sec. libcurl-7.76.1-23.el9_2.2.x86_64
RHSA-2023:4347 Moderate/Sec. libeconf-0.4.1-3.el9_2.x86_64
RHSA-2023:4325 Moderate/Sec. libsmbclient-4.17.5-103.el9_2.x86_64
RHSA-2023:4325 Moderate/Sec. libwbclient-4.17.5-103.el9_2.x86_64
RHSA-2023:4349 Moderate/Sec. libxml2-2.9.13-3.el9_2.1.x86_64
RHSA-2023:4412 Important/Sec. openssh-8.7p1-30.el9_2.x86_64
RHSA-2023:4412 Important/Sec. openssh-clients-8.7p1-30.el9_2.x86_64
RHSA-2023:4412 Important/Sec. openssh-server-8.7p1-30.el9_2.x86_64
RHSA-2023:3722 Moderate/Sec. openssl-1:3.0.7-16.el9_2.x86_64
RHSA-2023:3722 Moderate/Sec. openssl-libs-1:3.0.7-16.el9_2.x86_64
...output omitted...
```

► 3. List the RHSAs with an Important severity rating.

```
[root@servera ~]# dnf updateinfo list updates security | grep Important
...output omitted...
RHSA-2023:4412 Important/Sec. openssh-8.7p1-30.el9_2.x86_64
RHSA-2023:4412 Important/Sec. openssh-clients-8.7p1-30.el9_2.x86_64
RHSA-2023:4412 Important/Sec. openssh-server-8.7p1-30.el9_2.x86_64
...output omitted...
```

► 4. View the information of one Important RHSA to validate its content.

Read through the **Description** and **Security Fix(es)** sections of RHSA-2023:4412 to better understand this advisory and the subsequent CVE. Note the CVE ID so that you can use it later to update only the packages that relate to this RHSA.

```
[root@servera ~]# dnf updateinfo info RHSA-2023:4412
...output omitted...
=====
Important: openssh security update
=====
Update ID: RHSA-2023:4412
Type: security
Updated: 2023-08-01 09:35:02
Bugs: 2224173 - CVE-2023-38408 openssh: Remote code execution in ssh-agent
PKCS#11 support
CVEs: CVE-2023-38408
Description: OpenSSH is an SSH protocol implementation supported by a number of Linux, UNIX, and similar operating systems. It includes the core files necessary for both the OpenSSH client and server.
:
: Security Fix(es):
:
```

```
: * openssh: Remote code execution in ssh-agent PKCS#11 support
(CVE-2023-38408)
:
: For more details about the security issue(s), including the impact,
a CVSS score, acknowledgments, and other related information, refer to the CVE
page(s) listed in the References section.
Severity: Important
```

- 5. List the required RHSAs to resolve the CVE-2023-38408 CVE.

```
[root@servera ~]# dnf updateinfo list updates security --cve CVE-2023-38408
...output omitted...
RHSA-2023:4412 Important/Sec. openssh-8.7p1-30.el9_2.x86_64
RHSA-2023:4412 Important/Sec. openssh-clients-8.7p1-30.el9_2.x86_64
RHSA-2023:4412 Important/Sec. openssh-server-8.7p1-30.el9_2.x86_64
```

- 6. Use DNF and the CVE ID to update the system with the necessary packages that provide the security fixes.

```
[root@servera ~]# dnf update --cve CVE-2023-38408
...output omitted...
Dependencies resolved.
=====
Package Arch Version Repository Size
=====
Upgrading:
  openssh x86_64 8.7p1-30.el9_2 rhel-9.2-for-x86_64-baseos-additional-rpms 460 k
  openssh-clients
    x86_64 8.7p1-30.el9_2 rhel-9.2-for-x86_64-baseos-additional-rpms 709 k
  openssh-server
    x86_64 8.7p1-30.el9_2 rhel-9.2-for-x86_64-baseos-additional-rpms 459 k

Transaction Summary
=====
Upgrade 3 Packages
...output omitted...
Is this ok [y/N]: y
...output omitted...
Upgraded:
  openssh-8.7p1-30.el9_2.x86_64           openssh-clients-8.7p1-30.el9_2.x86_64
  openssh-server-8.7p1-30.el9_2.x86_64

Complete!
```

- 7. List the summary of the security updates to confirm that the number of Important notices is reduced.

```
[root@servera ~]# dnf updateinfo --security
...output omitted...
Updates Information Summary: available
 15 Security notice(s)
    7 Important Security notice(s)
    8 Moderate Security notice(s)
```

- 8. Return to the workstation machine as the student user.

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

Finish

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

```
[student@workstation ~]$ lab finish securityrisk-errata
```

Reviewing Recommended Security Practices

Objectives

- Implement recommended practices to improve the security of a RHEL system.

Baseline Standard Operating Environment

Every additional software component on your system increases the chance that the system is subject to some security vulnerability. If that software is not used, then you are needlessly increasing your risk by including it.

Red Hat recommends creating a *Standard Operating Environment* (SOE). The SOE is a standardized baseline installation with only the required software packages for your server installations. When you develop applications to run in your SOE, you can ensure that they add only the components that they need and which the SOE does not supply.

When building a prototype SOE and you are installing from DVD or USB media, then include only the minimum required software. Alternatively, you can use RHEL Image Builder to create a custom SOE image from an existing installation. If additional packages become necessary, then you can add them to the system later.

Manual Installations

To specify which packages to install, click **Software Selection** on the **Installation Summary** page. The package groups are organized into **Base Environments**. These environments are predefined sets of packages with a specific purpose. You can choose only one base environment at installation time.

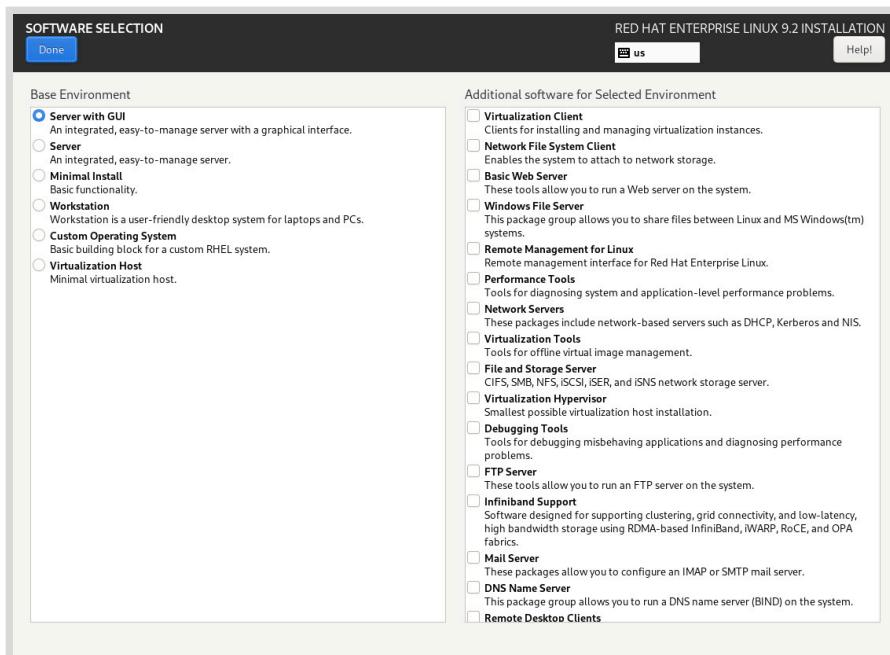


Figure 1.4: Software selection interface

You can customize your system by using the predefined environments and add-ons, but a manual installation does not provide individual packages to install. If you are not sure which packages to install, then Red Hat recommends that you choose **Minimal Install**.

A minimal installation installs a basic version of Red Hat Enterprise Linux with a minimal amount of additional software. This installation substantially reduces the chance of a vulnerability affecting the system. After the system is installed and you log in for the first time, you can use the `dnf` package manager to install any additional software.

Kickstart Installation

Kickstart installations partially or fully automate the installation process. Kickstart files contain all the information that the installation program requires, such as what time zone you want the system to use, how the drives must be partitioned, and which packages must be installed. By providing a prepared Kickstart file when the installation begins, you can install automatically without any user intervention. This process is especially useful when deploying Red Hat Enterprise Linux on many systems at once.

Automating the installation with a Kickstart file enables much higher control over installed packages. You can specify environments, groups, and individual packages in the `%packages` section of the Kickstart file.

Securing Services

A server typically runs many processes that are not associated with a terminal to provide various support functions for the system, users, and local or remote programs. These processes are called *daemons*.

Red Hat Enterprise Linux includes various daemons that act as network services. If a daemon acts as a network service, then its processes listen for connections on one or more network ports. Treat each of these services as a potential security risk.

Potential Risks to Services

Network services can pose many risks for Linux systems. You should be familiar with common attacks:

Denial of Service Attacks (DoS)

A denial of service attack overwhelms the service with requests, and can render a system unusable as it tries to log and answer each request.

Distributed Denial of Service Attacks (DDoS)

A DDoS attack is similar to a DoS attack, but a DDoS attack uses multiple compromised machines (which often number in the thousands) to direct a coordinated attack on a service, which overwhelms the service with requests and renders it unusable.

Script Vulnerability Attacks

When a server uses scripts to execute server-side actions, which web servers commonly do, an attacker can target incorrectly written scripts. These script vulnerability attacks can lead to a buffer overflow condition or allow the attacker to alter files on the system.

Buffer Overflow Attacks

Services that listen on privileged ports, such as ports 1 through 1023, must either be started with administrative privileges or the `CAP_NET_BIND_SERVICE` capability must be set for them. When a process is bound to a port and is listening on it, the privileges or the capability are often dropped. If the privileges or the capability are not dropped, and the application has an exploitable buffer overflow, then an attacker could access the system as the user that runs

the daemon. Because exploitable buffer overflows exist, exploiters use automated tools to identify systems with vulnerabilities. When exploiters gain access, they use automated rootkits to maintain their access to the system.

To promote strong security measures, most network services that are installed with Red Hat Enterprise Linux 9 are turned off by default. Notable exceptions are services such as the cups default print server and the sshd OpenSSH server. Any network service is potentially a security risk, which is why turning off unused services is so important.

Identifying which network services are available to start at boot time is not enough. You must also determine which ports are open and listening.

Use the `ss` command to list open ports in the listening state. The `-t l w` options display TCP sockets, listening sockets, and raw sockets, respectively. Raw sockets receive packets of a type that the kernel does not explicitly support.

```
[root@host ~]# ss -t l w
Netid State Recv-Q Send-Q Local Address:Port          Peer Address:Port
tcp   LISTEN  0      128                *:sunrpc           *:*
tcp   LISTEN  0      128                *:ssh              *:*
tcp   LISTEN  0      100               127.0.0.1:smtp       *:*
tcp   LISTEN  0      128                :::sunrpc          :::*
tcp   LISTEN  0      128                :::ssh             :::*
tcp   LISTEN  0      100                ::1:smtp            :::*
```

**Note**

On some systems, the deprecated `netstat` command might be installed instead of the `ss` command. The `netstat` command is part of the earlier `net-tools` package, which is not always available. The `netstat` command also takes slightly different options. The `ss` command is part of the modern `iproute` package, which is included in a minimal installation as part of the `Core` package group.

Your environment often determines whether to leave services running. Network services that you think are secure can quickly become a security risk. Exploits for services are routinely discovered and patched, which is why it is important to regularly update packages that are associated with any network service.

Protecting SSH Authentication

When you use the `ssh-keygen` command to generate a key pair, you are prompted to enter a passphrase. This passphrase encrypts the private key, and is required to decrypt the key for use. Red Hat recommends protecting your private key with a passphrase, because it helps to prevent unauthorized access to machines that the key is registered to. If the private key is stolen, then it is difficult for someone other than the issuer to use a passphrase-protected key. To avoid the inconvenience of entering a passphrase every time, use the `ssh-add` command to provide your passphrase to the `ssh-agent` utility. The `ssh-agent` utility stores the passphrase and provides it as needed while you remain logged in to your current session.

By default, the generated SSH keys are stored in your home directory. You can specify a different file name or directory interactively, or by setting the `ssh-keygen` command `-f` option.

The private key permissions must be set to 600, and the public key permissions must be set to 644. The ssh command automatically rejects keys with incorrect permissions. Additionally, SELinux in Enforcing mode rejects ssh keys without the ssh_home_t context label.

```
[user@host ~]$ ls -lz /user/home/.ssh
total 8
-rw----- 1 user user unconfined_u:object_r:ssh_home_t:s0 2602 10:57 id_rsa
-rw-r--r-- 1 user user unconfined_u:object_r:ssh_home_t:s0 571 10:57 id_rsa.pub
```

Red Hat Enterprise Linux 9 is configured to automatically reject root user login. Earlier systems might not be configured to reject root user login by default. If your environment contains earlier systems, then consider modifying the SSH service configuration on these systems to disable root login. Red Hat recommends allowing login only for non-privileged users, and then allowing specific users to escalate privileges with the sudo or sudo -i commands.

Cryptographic Policies

Over time, Red Hat deprecates cryptographic ciphers when the ciphers become less secure. From this deprecation process, some legacy systems such as Red Hat Enterprise Linux 6 might become inaccessible. Legacy systems that do not support modern cryptographic standards require more modern systems to use earlier protocols to communicate. Although some earlier protocols are removed in modern systems, most of those protocols are disabled by system-wide cryptographic policies.

Cryptographic policies are lists of service and protocol combinations that define which combinations are permitted. Red Hat Enterprise Linux 9 has the following preconfigured cryptographic policies:

LEGACY

The LEGACY policy is more permissive and therefore less secure than the DEFAULT policy. The LEGACY policy enables maximum compatibility with legacy systems.

DEFAULT

The DEFAULT policy is suitable for modern standards. This policy aligns with most systems and also remains secure. The NEXT policy is the same as the DEFAULT policy.

FUTURE

The FUTURE policy is more restrictive than the DEFAULT policy, and is designed to have a high likelihood of withstanding near-future attacks. The policy might prevent communication with common systems that use weaker protocols.

FIPS

The FIPS policy conforms to the Federal Information Processing Standard (FIPS) 140-2 requirements, which might be required by law in some countries and sectors.

Red Hat recommends using the DEFAULT policy in most circumstances and using the LEGACY policy only when necessary. The LEGACY policy might introduce security vulnerabilities by permitting outdated protocols.

Changing the Cryptographic Policy

You can verify the current cryptographic policy with the update-crypto-policies --show command. Although this command returns the active cryptographic policy, it might not indicate whether all services are using the policy. Because some long-running services might not restart when the policy is changed, Red Hat recommends rebooting a system after changing the cryptographic policy.

```
[user@host ~]$ update-crypto-policies --show  
DEFAULT
```

You can verify that the current policy matches the generated policy files by using the `update-crypto-policies --check` command. The command shows a discrepancy if the generated policy files are updated but the policy is not reapplied.

```
[user@host ~]$ update-crypto-policies --check  
The configured policy matches the generated policy
```

To change the cryptographic policy, use the `update-crypto-policies --set` command. This command applies a cryptographic policy, and restarts any known affected services. The command requires administrative privileges.

```
[user@host ~]$ sudo update-crypto-policies --set LEGACY  
sudo user@host:  
Setting system policy to LEGACY  
Note: System-wide crypto policies are applied on application start-up.  
It is recommended to restart the system for the change of policies  
to fully take place.
```

Creating Custom Cryptographic Policies

The `update-crypto-policies` command detects additional cryptographic policy files in the `/etc/crypto-policies/policies/` directory. You can copy an existing policy file in the `/usr/share/crypto-policies/policies/` directory and modify it with a text editor. A policy file contains a list of key-value pairs. Each key is a type of protocol or policy. The value is a list of algorithms that are enabled for the type.

The following example allows the AES-256-GCM and AES-256-CCM algorithms for the `cipher` type:

```
cipher = AES-256-GCM AES-256-CCM
```

You can use the `type@subtype` syntax to define subtypes. You can also use an asterisk (*) to match multiple items, or prepend a hyphen (-) to disable an item.

The following example disables all CBC algorithms for the SSH subtype of the `cipher` type:

```
cipher@SSH = -*~CBC
```

Escalating User Privileges

As a system administrator, you perform certain tasks with administrative access. Directly accessing the system as the `root` user is potentially dangerous and can compromise your entire computing infrastructure. The `sudo` command allows specific users to perform tasks that would normally be available only to the `root` user, and also maintains a higher level of control and system security.

Using Sudo to Gain Privileges

The `sudo` command offers an approach to give users administrative access. When trusted users precede an administrative command with `sudo`, they are prompted for their own password. Then,

when the user is authenticated, and assuming that the administrative command is permitted, the administrative command executes as if they were the `root` user.

The format of the `sudo` command is as follows:

```
[user@host ~]$ sudo command
```

In the preceding example, `command` would be replaced by a command that is normally reserved for the `root` user, such as `mount`.

Another advantage of the `sudo` command is that an administrator can allow different users access to specific commands based on their needs. Use the `visudo` command to edit the `/etc/sudoers` configuration file. To give administrative privileges to a user called `user`, type the `visudo` command and add the following line to the section on user privilege specification:

```
user ALL=(ALL) ALL
```

In the preceding example, a user named `user` can use `sudo` from any host and execute any command.

Use the `sudo -i` command to switch to the `root` user's login environment. The `-i` option is an abbreviation for the `--login` options. The `sudo -i` command changes to the `root` user's home directory and opens an interactive login shell based on the `root` user's environment variables. Because you have an entry in the `/etc/sudoers` file, you do not need to know the `root` password.

```
[user@host ~]$ sudo -i  
Password: user's password  
[root@host ~]# whoami; pwd; echo $PATH  
root  
/root  
/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin:/root/bin  
[root@host ~]#
```



Important

The `sudo` command resets the `PATH` variable based on the `secure_path` directive in the `/etc/sudoers` file. The `PATH` variable might then be modified based on which command `sudo` is executing. The case of `sudo -i` is discussed in more detail in the `sudoers(5)` man page.



References

`ssh-keygen(1)`, `ssh-copy-id(1)`, `ssh-agent(1)`, `ssh-add(1)`, `crypto-policies(7)`, `update-crypto-policies(8)`, `sudo(8)`, `visudo(8)`, and `su(1)` man pages

For more information, refer to the *Configuring Software Selection* chapter in the *Red Hat Enterprise Linux 9 Installation* guide at https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html/performing_a_standard_rhel_9_installation/graphical-installation_graphical-installation#configuring-software-selection_configuring-software-settings

► Guided Exercise

Implementing Recommended Security Practices

Use recommended security practices to configure and operate remote access methods and cryptographic algorithms.

Outcomes

- Apply the legacy cryptographic policy to enable communication with legacy hosts.
- Configure a system to use SSH key-based authentication.
- Disable SSH password-based authentication.

Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

```
[student@workstation ~]$ lab start securityrisk-recommend
```

Instructions

In this exercise, the `serverb` host acts as a legacy host where the current cryptographic algorithms are disabled.

- 1. Configure the `servera` host to use the legacy cryptographic policy to enable communication with the `serverb` host.
- 1.1. From `workstation`, log in to `servera` as the `student` user with the `student` password.

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

- 1.2. Verify that `serverb` is inaccessible due to legacy cryptographic protocols by attempting to log in with the `student` user.

```
[student@servera ~]$ ssh student@serverb  
Unable to negotiate with 172.25.250.11 port 22: no matching cipher found. Their  
offer: aes128-cbc
```

- 1.3. Apply the legacy cryptographic policy to `servera` by using the `update-crypto-policies` command. Use the `student` password when prompted. You do not need to reboot the `servera` host or to restart services because when you run `ssh` to remotely access `serverb`, the command uses the new cryptographic policy. Although other services might still use the previous cryptographic policy, it does not affect this test.

```
[student@servera ~]$ sudo update-crypto-policies --set legacy
[sudo] password for student:
Setting system policy to LEGACY
Note: System-wide crypto policies are applied on application start-up.
It is recommended to restart the system for the change of policies
to fully take place.
```

- 1.4. Verify that **serverb** is now accessible by logging in with the **student** user and the **student** password. If prompted, answer yes to connect. Log out of **serverb** after verifying accessibility.

```
[student@servera ~]$ ssh student@serverb
The authenticity of host 'serverb (172.25.250.11)' can't be established.
ED25519 key fingerprint is SHA256:0vIwK0Rq0bSrxkFEqzB1rLw5IvvJCBWxJmmx85TtVkg.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'serverb' (ED25519) to the list of known hosts.
student@serverb's password: student
[student@serverb ~]$ logout
Connection to serverb closed.
[student@servera ~]$
```

- ▶ 2. Configure the **serverb** host to allow SSH key-based authentication. Ensure that the **student** user on **servera** can log in to **serverb** by using SSH key-based authentication.
- 2.1. On **servera** as the **student** user, create an SSH key pair with no passphrase.

```
[student@servera ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/student/.ssh/id_rsa): Enter
Created directory '/home/student/.ssh'.
Enter passphrase (empty for no passphrase): Enter
Enter same passphrase again: Enter
Your identification has been saved in /home/student/.ssh/id_rsa.
Your public key has been saved in /home/student/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:8CttStESwlNmmGot+dXN9cR9x4Hw46gdRmVM5XFj+TQ student@servera.lab.example.com
The key's randomart image is:
+---[RSA 2048]----+
|      o+ .+o0+|
|     .o+      =o=EX|
|    ++ o. o o = o=|
|   = .o.=. + o o .|
|   . o .o S + . |
|     . + . . |
|     o + . . |
|     . + . . |
|     . . . |
+---[SHA256]-----+
```

- 2.2. On **servera**, use the **ssh-copy-id** command to copy the SSH public key to the **student** account on **serverb**.

```
[student@servera ~]$ ssh-copy-id student@serverb
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/student/.ssh/
id_rsa.pub"
The authenticity of host 'serverb (172.25.250.11)' can't be established.
ECDSA key fingerprint is SHA256:BMdnasLF5CBGg42Dx77nuXodXdI9dKoEBQGFK500HRI.
ECDSA key fingerprint is MD5:9e:a8:ec:0c:86:d2:70:34:ef:5a:94:15:6d:48:73:db.
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
student@serverb's password: student

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'student@serverb'"
and check to make sure that only the key(s) you wanted were added.
```

- 2.3. On servera, test key-based authentication by using the ssh command to connect to serverb and to remotely execute the hostname command. The ssh command should not prompt for a password. The results are returned to the terminal on servera.

```
[student@servera ~]$ ssh serverb 'hostname'
serverb
```

- 3. Configure SSH on the serverb host to prevent password authentication.
- 3.1. Confirm that the current configuration on serverb allows users to log in by using ssh password authentication. On servera, log in to serverb as the visitor user with the redhat password.

```
[student@servera ~]$ ssh visitor@serverb
visitor@serverb's password: redhat
[visitor@serverb ~]$
```

Log out of serverb.

```
[visitor@serverb ~]$ logout
Connection to serverb closed.
[student@servera ~]$
```

- 3.2. From servera, log in to serverb as the student user.

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

- 3.3. As the student user on serverb use the sudo -i command to change to the root user. If prompted, use the student password.

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

- 3.4. As the `root` user, edit the `/etc/ssh/sshd_config` configuration file so that the `PasswordAuthentication` entry is set to no. When finished, save your changes and exit the text editor.

```
[root@serverb ~]# vim /etc/ssh/sshd_config  
...output omitted...  
PasswordAuthentication no  
...output omitted...
```

- 3.5. Reload the SSH service on `serverb`. When finished, log out of `serverb`.

```
[root@serverb ~]# systemctl reload sshd  
[root@serverb ~]# logout  
[student@serverb ~]$ logout  
Connection to serverb closed.  
[student@servera ~]$
```

- 3.6. On `servera`, verify that the `visitor` user cannot log in to `serverb`. Then, verify that the `student` user can log in by using the previously created SSH keys.

```
[student@servera ~]$ ssh visitor@serverb  
Permission denied (publickey,gssapi-keyex,gssapi-with-mic).  
[student@servera ~]$ ssh student@serverb  
...output omitted...  
[student@serverb ~]$
```

- 3.7. Log out of `serverb` and `servera`.

```
[student@serverb ~]$ logout  
Connection to serverb closed.  
[student@servera ~]$ logout  
Connection to servera closed.  
[student@workstation ~]$
```

Finish

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

```
[student@workstation ~]$ lab finish securityrisk-recommend
```

▶ Lab

Managing Security and Risk

Identify all security notices that relate to a RHEL system, assess the severity of the notices, and update the system to eliminate any Moderate security issues.

Outcomes

- Identify all Critical, Important, and Moderate security notices.
- Update the system to eliminate potential Moderate security vulnerabilities.

Before You Begin

As the **student** user on the **workstation** machine, use the **lab** command to prepare your environment for this exercise, and to ensure that all required resources are available.

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

Instructions

1. Identify all Critical, Important, and Moderate security updates on the **serverb** machine.
2. List the security-related packages that are available to update.
3. List the RHSAs with a Moderate severity rating.
4. View the information of the RHSA-2023:4354 RHSA to validate its content.
5. Use DNF and the RHSA ID to update the system with the necessary packages that provide the security fixes.
6. List the summary of the security updates again to confirm that the number of notices is reduced for Moderate severity.
7. Return to the **workstation** machine as the **student** user.

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 securityrisk-review
```

Finish

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

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


► Solution

Managing Security and Risk

Identify all security notices that relate to a RHEL system, assess the severity of the notices, and update the system to eliminate any Moderate security issues.

Outcomes

- Identify all Critical, Important, and Moderate security notices.
- Update the system to eliminate potential Moderate security vulnerabilities.

Before You Begin

As the **student** user on the **workstation** machine, use the **lab** command to prepare your environment for this exercise, and to ensure that all required resources are available.

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

Instructions

1. Identify all Critical, Important, and Moderate security updates on the **serverb** machine.

- 1.1. Log in to the **serverb** machine as the **student** user.

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

- 1.2. Change to the **root** user. Use **student** as the password.

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

- 1.3. List the summary of the security updates. Among the 16 total security notices, eight are Important and eight are Moderate. The total security number might be different on your system.

```
[root@serverb ~]# dnf updateinfo --security  
...output omitted...  
Updates Information Summary: available  
 16 Security notice(s)  
    8 Important Security notice(s)  
    8 Moderate Security notice(s)
```

2. List the security-related packages that are available to update.

```
[root@serverb ~]# dnf updateinfo list updates security
...output omitted...
RHSA-2023:4354 Moderate/Sec. curl-7.76.1-23.el9_2.2.x86_64
...output omitted...
RHSA-2023:4354 Moderate/Sec. libcurl-7.76.1-23.el9_2.2.x86_64
...output omitted...
```

3. List the RHSA with a Moderate severity rating.

```
[root@serverb ~]# dnf updateinfo list updates security | grep Moderate
RHSA-2023:4354 Moderate/Sec. curl-7.76.1-23.el9_2.2.x86_64
...output omitted...
RHSA-2023:4354 Moderate/Sec. libcurl-7.76.1-23.el9_2.2.x86_64
...output omitted...
```

4. View the information of the RHSA-2023:4354 RHSA to validate its content.

```
[root@serverb ~]# dnf updateinfo info RHSA-2023:4354
...output omitted...
=====
Moderate: curl security update
=====
Update ID: RHSA-2023:4354
Type: security
Updated: 2023-08-01 03:58:15
Bugs: 2196786 - CVE-2023-28321 curl: IDN wildcard match may lead to
Improper Certificate Validation
: 2196793 - CVE-2023-28322 curl: more POST-after-PUT confusion
CVEs: CVE-2023-28321
: CVE-2023-28322
Description: The curl packages provide the libcurl library and the curl utility
for downloading files from servers using various protocols, including HTTP, FTP,
and LDAP.
:
: Security Fix(es):
:
: * curl: IDN wildcard match may lead to Improper Certificate Validation
(CVE-2023-28321)
:
: * curl: more POST-after-PUT confusion (CVE-2023-28322)
:
: For more details about the security issue(s), including the impact,
a CVSS score, acknowledgments, and other related information, refer to the CVE
page(s) listed in the References section.
Severity: Moderate
```

Chapter1 | Managing Security and Risk

5. Use DNF and the RHSA ID to update the system with the necessary packages that provide the security fixes.

```
[root@serverb ~]# dnf update --advisory RHSA-2023:4354
...output omitted...
Dependencies resolved.
=====
Package           Arch    Version      Repository      Size
=====
Upgrading:
curl x86_64 7.76.1-23.el9_2.2 rhel-9.2-for-x86_64-baseos-additional-rpms 298 k
libcurl
x86_64 7.76.1-23.el9_2.2 rhel-9.2-for-x86_64-baseos-additional-rpms 286 k

Transaction Summary
=====
Upgrade 2 Packages

...output omitted...
Is this ok [y/N]: y
...output omitted...
Upgraded:
curl-7.76.1-23.el9_2.2.x86_64          libcurl-7.76.1-23.el9_2.2.x86_64

Complete!
```

6. List the summary of the security updates again to confirm that the number of notices is reduced for Moderate severity.

```
[root@serverb ~]# dnf updateinfo --security
...output omitted...
Updates Information Summary: available
 15 Security notice(s)
    8 Important Security notice(s)
    7 Moderate Security notice(s)
```

7. Return to the workstation machine as the student user.

```
[root@serverb ~]# logout
[student@serverb ~]$ logout
Connection to serverb closed.
[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 securityrisk-review
```

Finish

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

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

Summary

- Risk management is a continuous process of proactively discovering potential risk, assessing facts, and taking action based on the facts to resolve those risks.
- Red Hat analyzes threats and vulnerabilities against all Red Hat products every day, and provides relevant advice and updates through the Red Hat Customer Portal.
- Common Vulnerabilities and Exposures (CVE) records provide a standardized format for reporting and tracking security-related software issues.
- Base your servers on a standard operating environment (SOE) that provides the minimum packages that all your systems need, and add only the packages that the server applications require.
- Every daemon that provides a network service increases the risk of a successful remote attack. To reduce this risk, minimize the number of unnecessary services.
- By default, Red Hat Enterprise Linux 9 disables SSH login for the `root` user.
- When possible, turn off password-based SSH access and require key-based authentication for remote logins.
- Red Hat recommends the `DEFAULT` cryptographic policy in most scenarios.

Chapter 2

Automating Configuration and Remediation with Ansible

Goal

Remediate configuration and security issues automatically with Ansible Playbooks.

Sections

- Configuring Ansible for Security Automation (and Guided Exercise)
- Managing Playbooks with Automation Controller (and Guided Exercise)

Lab

- Automating Configuration and Remediation with Ansible

Configuring Ansible for Security Automation

Objectives

- Install and configure an Ansible control node, prepare managed hosts for Ansible access, and configure hosts by using a playbook with an Ansible inventory.

Ansible Concepts and Architecture

Ansible is an open source automation platform that uses an automation language to describe the intended configuration of servers and applications.

The Ansible architecture consists of two types of machines: *control nodes* and *managed hosts*. The control node runs Ansible, and it also has copies of your Ansible project files. Managed hosts are listed in an *inventory*, which also organizes those systems into groups for easier management. You can define the inventory statically in a text file, or dynamically by using scripts that obtain group and host information from external sources.

Instead of writing complex scripts, Ansible users write high-level *plays* that specify the intended state of particular managed hosts. A play performs a series of *tasks* on the hosts, in the order that the play specifies. These plays are expressed in YAML format in a text file. A *playbook* is a file that contains one or more plays.

Each task runs a *module*, which is a small piece of code (written in Python, PowerShell, or some other language) with specific arguments. Each module is essentially a tool in your toolkit. Ansible ships with hundreds of useful modules that can perform wide-ranging automation tasks. Modules can act on system files, install software, or make API calls.

Most modules are *idempotent*, which means that they can be run safely multiple times, and if the system is already in the correct state, they do nothing. Tasks, plays, and playbooks are also designed to be idempotent.

Ansible also uses *plug-ins*. Plug-ins extend Ansible to new uses and platforms.

The Ansible architecture is agentless. Typically, when an administrator runs an Ansible Playbook, the control node connects to the managed host by using SSH (by default) or WinRM. Therefore, an Ansible-specific agent does not need to be installed on managed hosts, and you do not need to allow any additional communication between the control node and managed hosts.

Security Automation

Managing security for many systems can be challenging. When new threats or issues are discovered, you might need to ensure that all the systems are configured or provisioned correctly.

Many administrators use automation to ensure that work is applied consistently and correctly to all machines under their care, and to speed up the correct deployment of security remediation.

A good automation solution must be able to deploy and provision systems from scratch. Administrators must be able to run automated tasks on a system that is already configured, and either confirm that it is already correctly configured and do nothing, or change that system to ensure that it is correctly configured.

Chapter 2 | Automating Configuration and Remediation with Ansible

You can use many tools for this purpose. In this course, you use Red Hat Ansible Automation Platform to address security issues. Using Ansible in a security context offers several advantages:

- You can install Ansible with minimal requirements for managed systems, and it does not need an agent.
- Working with Ansible does not require special coding skills.
- Ansible provides human-readable automation instructions (*Ansible Playbooks*).
- Ansible provides various tools that can identify and detect security-related issues that can help remediate the issues. This course reviews two tools in detail: the SCAP Security Guide for OpenSCAP and Red Hat Insights.

When your security policy is defined in Ansible, scanning and remediation of site-wide security policies can be integrated into other automated processes. Instead of being an afterthought, security is an integral part of everything that is deployed.

Installing Ansible

To use Ansible, you must install the Ansible software on a host (the control node) from which you run Ansible Playbooks. Ensure that your managed hosts are set up so that you can use Ansible to connect to them, become the `root` user, and run Ansible plays.

Installing Ansible on the Control Node

To run Ansible Playbooks, install the automation content navigator (`ansible-navigator`) on your control node and download an execution environment. Hosts that Ansible manages do not require the `ansible-navigator` tool to be installed; you need to install this tool only on the control node that you run Ansible Playbooks from.

Python 3.8 or later must be installed on the control node before you install the `ansible-core` package.

You must have a valid Red Hat Ansible Automation Platform subscription to install the automation content navigator on your control node.

If you activated Simple Content Access for your organization in the Red Hat Customer Portal, then you do not need to attach the subscription to your system.



Note

You do not need to run these exact steps in your classroom environment, because it is preconfigured to download the `ee-supported-rhel9` execution environment.

- Attach a valid Red Hat Ansible Automation Platform subscription:

```
[root@controlnode ~]# subscription-manager repos \
--enable ansible-automation-platform-2.4-for-rhel-9-x86_64-rpms
```

- Install the automation content navigator on your control nodes:

```
[root@controlnode ~]# dnf install ansible-navigator
```

- Verify that the automation content navigator is installed on the system:

```
[user@controlnode ~]$ ansible-navigator --version
ansible-navigator 3.4.1
```

- Log in to the container registry that contains your automation execution environments (in this case, `registry.redhat.io`):

```
[user@controlnode ~]$ podman login registry.redhat.io
Username: your-registry-username
Password: your-registry-password
Login Succeeded!
```

- Display the list of locally available container images. The automation content navigator automatically downloads the default execution environment when you run the `ansible-navigator` command:

```
[user@controlnode ~]$ ansible-navigator images
  Image          Tag    Execution environment   Created      Size
0/ee-supported-rhel9  latest  True                10 days ago  1.63 GB

^b/PgUp page up  ^f/PgDn page down  ↑ scroll  esc back  [0-9] goto  :help
```

Preparing Managed Hosts for Ansible Automation

Ansible is a cross-platform solution that can manage Linux hosts, Microsoft Windows systems, and even managed networking devices. This course focuses on configuring Linux managed hosts.

One benefit of Ansible is that a special agent does not need to be installed on managed hosts. The Ansible control node connects to managed hosts by using a standard network protocol to ensure that the systems are in the specified state.

Depending on how the control node connects to the managed hosts, and which modules are run on them, managed hosts might have the following requirements:

- On Linux and UNIX managed hosts, Python 3.8 or later must be installed for most modules to work.
- If SELinux is enabled on the managed hosts, then ensure that the `python3-libselinux` package is installed before using modules that are related to any copy, file, or template functions.
- Ansible must connect to the managed machine by using SSH. When Ansible connects as a regular user, it must be able to use `sudo` to gain superuser access.

When configuring authentication and privilege escalation in Ansible, you have several options based on your policies and needs. A common method involves using SSH public key authentication to connect as a non-privileged user and using `sudo` to escalate privileges to the `root` user.

You can choose one of the following options to configure the Ansible user with `sudo` access. For the first option, configure `sudo` to require the Ansible user to enter their password to escalate privileges. The default configuration of the `/etc/sudoers` file permits `sudo` access for any member of the `wheel` group, so you might add the Ansible user to that group.

For the second option, add a rule to explicitly allow `sudo` access for a user. For example, by creating a file named `/etc/sudoers.d/ansible-testuser` that includes the following line,

you allow the `ansible-testuser` user to use `sudo` to run any command as any user after they enter their password to authenticate.

```
ansible-testuser  ALL=(ALL)  ALL
```

You can allow the Ansible user to run `sudo` without entering a password. This approach is less secure, but it might be more convenient than requiring a password. For example, you could set up `/etc/sudoers.d/user` to allow the `ansible-testuser` user to use `sudo` to run any command as any user without further authentication, in this way:

```
ansible-testuser  ALL=(ALL)  NOPASSWD:ALL
```



Note

Various security trade-offs apply here. If you require the Ansible user to provide their password to escalate privileges, then the administrators who run Ansible might need to know that user's password and it must be the same everywhere. Automation controller protects the `sudo` password from Ansible administrators and allows privilege escalation.

An Ansible user with no password can be created, to help to ensure that the only way to access that account is through SSH public key authentication or the local `root` account.

Managing a Host Inventory

An *inventory* defines a collection of hosts that Ansible will manage. These hosts can also be assigned to *groups*, which can be managed collectively. Groups can contain child groups, and hosts can be members of multiple groups. The inventory can also set variables that apply to the hosts and groups that it defines.

An inventory can be of two types, *static* or *dynamic*. A *static* host inventory can be defined by an INI-like text file. A *dynamic* host inventory can be generated by a script or other program as needed, by using external information providers.

In its simplest form, a static inventory is a list of hostnames or IP addresses of the managed hosts, each on a single line:

```
server1.example.com
server2.example.com
server3.example.com
```

You can organize managed hosts into host groups. Host groups enable you to more effectively run Ansible against a collection of systems. In this example, each section starts with a host group name in square brackets ([]). This section is followed by the hostname or an IP address for each managed host in the group, each on a single line. Hosts can be and often are members of multiple host groups, which makes them easier to manage.

Host groups can also be nested. The `:children` suffix on a name in a square-bracketed section sets up a nested group, and each line is the name of another group. All hosts in those groups are also members of the parent group.

This example inventory file creates a `webservers` group that contains the `server1.example.com` and `server2.example.com` hosts. This inventory also creates another group named `fileservers`, which contains the `server3.example.com` and `server4.example.com` servers. Finally, the inventory creates a group named `myservers` that includes the `webservers` and `fileservers` groups as nested groups inside it. (The inventory could also have included a `[myservers]` section to add individual hosts to that group.)

```
[webservers]
server1.example.com
server2.example.com

[fileservers]
server3.example.com
server4.example.com

[myservers:children]
webservers
fileservers
```

**Note**

Two host groups always exist:

- The `all` host group, which includes every host listed in the inventory.
- The `ungrouped` host group, which contains every listed host in the inventory that is not a member of another group.

Configuring Ansible Operation

To configure the behavior of Ansible and the `ansible-navigator` command, you can create and edit two files in each of your Ansible project directories:

- The `ansible.cfg` file, which configures the behavior of several Ansible tools.
- The `ansible-navigator.yml` file, which changes the default options for the `ansible-navigator` command.

You can customize the behavior of an Ansible installation by modifying settings in the Ansible configuration file. Ansible chooses its configuration file from one of several possible locations on the control node.

The `ansible` package provides a default configuration file in the `/etc/ansible/ansible.cfg` file. However, the recommended practice is to create an `ansible.cfg` file in a directory from which you run Ansible commands. This directory would also contain any files that your Ansible project uses, such as an inventory and a playbook.

**Note**

The `ansible --version` command shows the location of the configuration file in use.

The Ansible configuration file consists of several sections, where each section contains settings that are defined as key-value pairs. Section titles are in square brackets. Two sections are essential for basic operations:

- [defaults] sets defaults for Ansible operation.
- [privilege_escalation] configures how Ansible escalates privileges on managed hosts.

The following content is a typical example of the `ansible.cfg` file:

```
[defaults]
inventory = ./inventory
remote_user = user
ask_pass = false

[privilegeEscalation]
become = true
becomeMethod = sudo
becomeUser = root
becomeAskPass = false
```

The directives in this file are explained in the following table:

Directive	Description
<code>inventory</code>	Specifies the path to the inventory file.
<code>remote_user</code>	The name of the user to log in as on the managed hosts. If not specified, the current user's name is used. (In a container-based automation execution environment that the <code>ansible-navigator</code> runs, the user is always <code>root</code> .)
<code>ask_pass</code>	Whether to prompt for an SSH password. Can be <code>false</code> if using SSH public key authentication.
<code>become</code>	Whether to automatically switch user on the managed host (typically to <code>root</code>) after connecting. A play can also specify this directive.
<code>becomeMethod</code>	How to switch user (typically <code>sudo</code> , which is the default, although <code>su</code> is an option).
<code>becomeUser</code>	The user to switch to on the managed host (typically <code>root</code> , which is the default).
<code>becomeAskPass</code>	Whether to prompt for a password for your <code>becomeMethod</code> . Defaults to <code>false</code> .

The `ansible-navigator config` command displays the current Ansible configuration. The command displays the value that Ansible uses for each parameter and from which source it retrieves that value, configuration file, or environment variable.

The following output is from the `ansible-navigator config` command that is run from a `/home/student/project/` directory that contains an `ansible.cfg` file. Each line describes an Ansible configuration parameter:

Name	Default	Source	Current
...output omitted...			
44 Default ask pass	True	default	False
45 Default ask vault pass	True	default	False
46 Default become	False	/home/student/project/ansible.cfg	True
47 Default become ask pass	False	/home/student/project/ansible.cfg	True
...output omitted...			
50 Default become method	False	/home/student/project/ansible.cfg	sudo
51 Default become user	False	/home/student/project/ansible.cfg	root
...output omitted...			

- The `Default ask pass` and `Default ask vault pass` parameters use their default values, which the `True` value in the `Default` column indicates.
- The `Default become` and `Default become ask pass` parameters are manually configured to `True` in the `/home/student/project/ansible.cfg` configuration file. The `Default` column is `False` for these two parameters. The `Source` column provides the path to the configuration file that defines these parameters, and the `Current` column shows that the value for these two parameters is `True`.
- The `Default become method` parameter has the current value of `sudo`, and the `Default become user` parameter has the current value of `root`.

Press Esc or type :q to exit the interactive mode of the `ansible-navigator config` command.



Note

If the project does not include an `ansible.cfg` file, then Ansible tries to use a `~/.ansible.cfg` file in the home directory of the user that runs Ansible. If that file does not exist, then Ansible tries to use a `/etc/ansible/ansible.cfg` file.

However, the `ansible-navigator` command, if used, looks for these files *inside the automation execution environment*. On the current execution environments that Red Hat supports, these files do not exist or are empty.

If you are using the earlier `ansible-playbook` command or other Ansible commands that do not use automation execution environments, then Ansible looks for these files on your workstation.

Ansible Playbooks

The power of Ansible is that you can use playbooks to run multiple complex tasks against a set of targeted hosts in a repeatable manner.

You can use plays to change a lengthy, complex set of manual administrative tasks into a repeatable routine with predictable and successful outcomes. In a playbook, you can save the sequence of tasks in a play in a human-readable and immediately runnable form. The tasks themselves, because of the way in which they are written, document the steps to deploy your application or infrastructure.

Formatting an Ansible Playbook

The following example contains one play with a single task:

```
---
```

```
- name: Configure important user consistently
hosts: server1.example.com
tasks:
  - name: Newbie exists with UID 4000
    ansible.builtin.user:
      name: newbie
      uid: 4000
      state: present
```

A playbook is a text file in YAML format, and is normally saved with the `.yml` extension. The playbook uses indentation with space characters to indicate the structure of its data. Although YAML does not place strict requirements on how many spaces are used for the indentation, two main rules apply:

- Data elements at the same level in the hierarchy (such as items in the same list) must have the same indentation.
- Items that are children of another item must be indented more than their parents.

You can also add blank lines for readability.



Important

Use only space characters for indentation; do not use tab characters.

A playbook usually begins with a line that consists of three dashes (---) to indicate the start of the document. A playbook might end with three dots (...) to indicate the end of the document, although in practice this boundary marker is often omitted.

Between those markers, the playbook is defined as a list of plays. An item in a YAML list starts with a single dash followed by a space. For example, a YAML list might appear as follows:

```
- apple
- orange
- grape
```

The play itself is a collection of key-value pairs. Keys in the same play must have the same indentation. The following example shows a YAML snippet with three keys. The first two keys have single values. The third key has a list of three items as a value.

```
---
```

```
name: just an example
hosts: webservers
tasks:
  - first
  - second
  - third
```

The initial example play has three keys: `name`, `hosts`, and `tasks`. These keys all have the same indentation.

Chapter 2 | Automating Configuration and Remediation with Ansible

The first line of the example play starts with a dash and a space (to indicate that the play is the first item of a list), and then the first key, `name`. The `name` key associates an arbitrary string with the play as a label that identifies the purpose of the play. The `name` key, though optional, is recommended because it helps to document your playbook. This `name` key is especially useful when a playbook contains multiple plays.

```
- name: Configure important user consistently
```

The second key in the play is a `hosts` key, which specifies the hosts that the play's tasks are run against. The `hosts` key takes a host pattern as a value, such as the names of managed hosts or groups in the inventory.

```
hosts: server1.example.com
```

Finally, the last key in the play is `tasks`, whose value specifies a list of tasks to run for this play. This example has a single task, which runs the `ansible.builtin.user` module with specific arguments (to ensure that the `newbie` user exists and has UID 4000).

```
tasks:
- name: newbie exists with UID 4000
  ansible.builtin.user:
    name: newbie
    uid: 4000
    state: present
```

The `tasks` key is the part of the play that lists, in order, the tasks to be run on the managed hosts. Each task in the list is itself a collection of key-value pairs.

In this example, the only task in the play has two keys:

- The `name` label optionally documents the purpose of the task. Red Hat recommends naming all your tasks to help to document the purpose of each step of the automation process.
- The `ansible.builtin.user` module is run for this task. This module's arguments are passed as a collection of key-value pairs, which are children of the module (`name`, `uid`, and `state`).

The following output is another example of a `tasks` key with multiple tasks, where each task uses the `ansible.builtin.service` module to ensure that a service starts at boot:

```
tasks:
- name: Web server is enabled
  ansible.builtin.service:
    name: httpd
    enabled: true

- name: NTP server is enabled
  ansible.builtin.service:
    name: chronyd
    enabled: true

- name: Postfix is enabled
```

```
ansible.builtin.service:  
  name: postfix  
  enabled: true
```



Important

The order in which the plays and tasks are listed in a playbook is important, because Ansible runs them in the same order.

Finding Modules for Tasks

Plays use modules to run tasks. Hundreds of modules are written for different purposes. You can usually find a tested, special-purpose module that does what you need, often as part of the default automation execution environment.

Ansible Content Collections distribute multiple types of Ansible content, such as playbooks, modules, documentation, and more.

The `ansible-core` package provides a single Ansible Content Collection named `ansible.builtin`. These modules are always available to you. Visit <https://docs.ansible.com/ansible/latest/collections/ansible/builtin/> for a list of modules in the `ansible.builtin` collection.

In addition, the default automation execution environment that `ansible-navigator` uses in Red Hat Ansible Automation Platform 2.4, `ee-rhel9-supported`, includes several other Ansible Content Collections.

You can browse these collections by running the `ansible-navigator collections` command. In the interactive UI, you can type a colon (:) followed by the line number of a collection to get more information about it, including the list of modules and other Ansible content that it provides. You can do the same thing with the line number of a module to get documentation about that module. Press `Esc` to go back to the preceding list.

Name	Version	Shadowed	Type	Path
0 amazon.aws	6.0.1	False	contained	/usr/share/ansible/collections
1 ansible.builtin	2.15.4	False	contained	/usr/lib/python3.9/site-packages
2 ansible.controller	4.4.2	False	contained	/usr/share/ansible/collections
3 ansible.netcommon	5.1.1	False	contained	/usr/share/ansible/collections
4 ansible.network	2.0.0	False	contained	/usr/share/ansible/collections
5 ansible.posix	1.5.4	False	contained	/usr/share/ansible/collections

Modules are named by using fully qualified collection names (FQCNs). The same name can then be used for different modules in two Ansible Content Collections without causing conflicts. For example, the `copy` module from the `ansible.builtin` Ansible Content Collection has `ansible.builtin.copy` as its FQCN.



Note

Some modules might have their own additional requirements. For example, the `ansible.builtin.dnf` module, which can install packages on current Fedora systems, requires the `python3-dnf` package.

Running Playbooks

The `ansible-navigator run` command runs playbooks. The control node executes the command, and passes the name of the playbook to be run as an argument.

Running the `ansible-navigator run` command with the `-m stdout` option prints the output of the playbook to standard output. If the `-m stdout` option is not provided, then the `ansible-navigator` command runs in interactive mode. (Interactive mode is covered in the *DO374: Developing Advanced Automation with Red Hat Ansible Automation Platform* course.)

```
[user@controlnode ~]$ ansible-navigator run \
-m stdout site.yml
```

When you run the playbook, the output shows the play and the tasks that are executed. The output also reports the result of each task that is executed.

The following example shows the contents of a playbook:

```
[user@controlnode playdemo]$ cat webserver.yml
---
- name: Play to set up web server
  hosts: server1.example.com
  tasks:
    - name: Latest httpd version installed
      ansible.builtin.dnf:
        name: httpd
        state: latest
...output omitted...
```

When you run the playbook, the output shows that the latest version of the `httpd` package is installed:

```
[user@controlnode playdemo]$ ansible-navigator run \
-m stdout webserver.yml

PLAY [Play to set up web server] ****
TASK [Gathering Facts] ****
ok: [server1.example.com]

TASK [Latest httpd version installed] ****
changed: [server1.example.com]

PLAY RECAP ****
server1.example.com    : ok=2    changed=1    unreachable=0    failed=0
                      skipped=0   rescued=0   ignored=0
```

The values of the `name` key for each play and task are displayed when the playbook is run. The `ansible.builtin.setup` module usually runs the `Gathering Facts` special task automatically at the start of a play. For playbooks with multiple plays and tasks, setting `name` attributes makes it easier to monitor the progress of a playbook's execution.

In the sample output, the `Latest httpd version installed` task is changed for `server1.example.com`. The task changed something on that host to ensure that its

specification was met. In this case, it means that the `httpd` package was not previously installed or was not the latest version.

Increasing Output Verbosity

The default output from the `ansible-navigator run` command does not provide detailed task execution information. The `-v` option provides additional information, with up to four levels.

Configuring the Output Verbosity of Playbook Execution

Option	Description
<code>-v</code>	Displays task results.
<code>-vv</code>	Displays task results and task configuration.
<code>-vvv</code>	Displays extra information about connections to managed hosts.
<code>-vvvv</code>	Adds extra verbosity options to the connection plug-ins, including the users on managed hosts who executed scripts, and which scripts were executed.

Syntax Verification

Before executing a playbook, it is good practice to validate its syntax. You can use the `ansible-navigator run --syntax-check` command to validate the syntax of a playbook. The following example shows the successful syntax validation of a playbook:

```
[user@controlnode playdemo]$ ansible-navigator run \
-m stdout webserver.yml --syntax-check
playbook: /home/user/playdemo/webserver.yml
```

When syntax validation fails, a syntax error is reported. The output also includes the approximate location of the syntax issue in the playbook.

Executing Playbooks in Check Mode

You can use the `--check` option to run a playbook in *check mode*, which executes a playbook without altering your systems. In check mode, Ansible reports which changes would have occurred if the playbook was executed, but it does not make any changes to managed hosts.

The following example shows the check mode of a playbook that contains a single task for ensuring that the latest version of the `httpd` package is installed on a managed host. In this case, the dry run reports that the task would make a change on the managed host.

```
[user@controlnode playdemo]$ ansible-navigator run \
-m stdout webserver.yml --check

PLAY [Play to set up web server] ****
TASK [Gathering Facts] ****
ok: [server1.example.com]

TASK [Latest httpd version installed] ****
```

```
changed: [server1.example.com]

PLAY RECAP ****
server1.example.com    : ok=2      changed=1      unreachable=0      failed=0
skipped=0      rescued=0      ignored=0
```



References

Ansible

<https://www.ansible.com>

How Ansible Works

<https://www.ansible.com/how-ansible-works>

Introducing Ansible Automation Platform 2

<https://www.ansible.com/blog/introducing-ansible-automation-platform-2>

ansible-navigator Settings: Ansible Navigator Documentation

<https://ansible.readthedocs.io/projects/navigator/settings/>

Ansible Playbooks

https://docs.ansible.com/ansible/latest/user_guide/playbooks_intro.html

Ansible Community Documentation

<https://docs.ansible.com>

► Guided Exercise

Configuring Ansible for Security Automation

Install Ansible on a control node, set up an inventory file, and run a single task to ensure that multiple systems are in the same state.

Outcomes

- Install `ansible-navigator`.
- Create a basic configuration file.
- Create an inventory file.
- Confirm managed host configuration on multiple hosts.

Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

```
[student@workstation ~]$ lab start ansible-configure
```

Instructions

- 1. On the workstation machine, install the `ansible-navigator` RPM package that provides automation content navigator, to use that machine as your control node.

```
[student@workstation ~]$ sudo dnf install ansible-navigator
[sudo] password for student: student
...output omitted...
Is this ok [y/N]: y
...output omitted...
Complete!
```

- 2. Ensure that for the `ansible-navigator` tool, the execution environment container image is downloaded.



Note

You do not need to download the `ee-supported-rhel9` execution environment, because it is preloaded in your classroom.

```
[student@workstation ~]$ ansible-navigator images
```

```
Image           Tag     Execution environment  Created      Size
0|ee-supported-rhel9    latest   True                  10 days ago  1.63 GB
```

```
^b/PgUp page up  ^f/PgDn page down  ↑ scroll  esc back  [0-9] goto  :help
```

Press Esc to exit the image list.

- 3. Create the /home/student/security-ansible directory.

```
[student@workstation ~]$ mkdir ~student/security-ansible
```

- 4. Navigate to the /home/student/security-ansible directory.

```
[student@workstation ~]$ cd ~student/security-ansible
[student@workstation security-ansible]$
```

- 5. In the /home/student/security-ansible directory, create an Ansible configuration file named ansible.cfg. Use the following content and values.

```
[student@workstation security-ansible]$ cat ansible.cfg
[defaults]
inventory      = ./inventory
remote_user    = ansible-testuser

[privilegeEscalation]
become        = true
become_method = sudo
become_user   = root
become_ask_pass = false
```

- 6. In the /home/student/security-ansible directory, create the inventory file named inventory. Use the following content.

```
[student@workstation security-ansible]$ cat inventory
[Boston]
servera
serverb

[Raleigh]
workstation

[cities:children]
Boston
Raleigh
```

- 7. List all the managed hosts that are present in the inventory.

```
[student@workstation security-ansible]$ ansible-navigator inventory \
-m stdout --graph
@all:
|--@ungrouped:
|--@cities:
| |--@Boston:
| | |--servera
| | |--serverb
| |--@Raleigh:
| | |--workstation
```

- 8. In the /home/student/security-ansible directory, create an Ansible Playbook file named sshd_config.yml. Use the following content and values.

```
[student@workstation security-ansible]$ cat sshd_config.yml
---
- name: Play to disable SSH password-based authentication
  hosts: Boston
  tasks:
    - name: Task to disable SSH password-based authentication
      ansible.builtin.lineinfile:
        path: /etc/ssh/sshd_config
        regexp: 'PasswordAuthentication yes'
        backrefs: yes
        line: 'PasswordAuthentication no'
    - name: Restart sshd
      ansible.builtin.service:
        name: sshd
        state: restarted
...
...
```

- 9. Before running your playbook, validate the sshd_config.yml playbook syntax. Correct any reported errors before continuing.

```
[student@workstation security-ansible]$ ansible-navigator run \
-m stdout sshd_config.yml --syntax-check
playbook: /home/student/security-ansible/sshd_config.yml
```

- 10. Run the sshd_config.yml playbook. Read through the generated output to ensure that all tasks completed successfully.

```
[student@workstation security-ansible]$ ansible-navigator run \
-m stdout sshd_config.yml
PLAY [Play to disable SSH password-based authentication] ****
TASK [Gathering Facts] ****
ok: [serverb]
ok: [servera]

TASK [Task to disable SSH password-based authentication] ****
```

```
changed: [serverb]
changed: [servera]

TASK [Restart sshd] ****
changed: [serverb]
changed: [servera]

PLAY RECAP ****
servera    : ok=2      changed=2      unreachable=0      failed=0      skipped=0
             rescued=0     ignored=0
serverb    : ok=2      changed=2      unreachable=0      failed=0      skipped=0
             rescued=0     ignored=0
```

- ▶ **11.** Verify that you cannot log in to the `servera` or `serverb` machines from the `serverc` machine by using password authentication.

11.1. Log in to the `serverc` machine as the `student` user. No password is required.

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

11.2. Verify that you cannot log in to the `servera` or `serverb` machines by using password authentication.

```
[student@serverc ~]$ ssh student@servera
student@servera: Permission denied (publickey,gssapi-keyex,gssapi-with-mic).
[student@serverc ~]$ ssh student@serverb
student@serverb: Permission denied (publickey,gssapi-keyex,gssapi-with-mic).
[student@serverc ~]$
```

11.3. Return to the `workstation` machine when done.

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

Finish

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

```
[student@workstation ~]$ lab finish ansible-configure
```

Managing Playbooks with Automation Controller

Objectives

- Run playbooks and manage access to authentication credentials by using automation controller from Red Hat Ansible Automation Platform.

Introduction to Automation Controller

Red Hat Ansible Automation Platform 2 includes a component called *automation controller*, which was called *Red Hat Ansible Tower* in earlier versions of Ansible Automation Platform. Automation controller provides a centralized hub to run your Ansible automation code.

Enterprise IT organizations need a way to define and embed automation workflows for other tools and processes. The organizations also need reliable and scalable automation execution, and a centralized system that supports auditing.

With automation controller, companies can standardize how automation is deployed by using a centralized location.

Automation controller provides a framework for running and managing Ansible efficiently on an enterprise scale. Automation controller maintains organization security by introducing features such as a centralized web UI for playbook management, role-based access control (RBAC), and centralized logging and auditing. You can integrate an enterprise's existing workflows and tool sets, such as enabling continuous integration and deployment, by using the automation controller REST API. Automation controller provides mechanisms to enable the centralized use and control of machine credentials and other secrets without exposing the credentials or secrets to the users of automation controller.

From a security perspective, automation controller helps you control, secure, and manage your Ansible automation at scale. You can use automation controller to control who has access to run particular playbooks and which inventories they can use or manage. You can allow users of the automation controller to use machine credentials and control who can manage them, without allowing users to transfer them or see their contents. Automation controller logs who ran what Ansible jobs on what hosts when, and what the results of those jobs were. It provides a way for you to centrally manage your Ansible inventories, to ensure that no hosts are missed.

Automation Controller Architecture

The environment in Red Hat Ansible Automation Platform 2 decouples the automation controller control plane from its execution environment.

Automation controller uses automation execution environments instead of using the system executables, or uses the Python virtual environments. These environments are container images that you can pull from a central container registry, install on automation controller, and manage through a web UI. If needed, then you can create custom automation execution environments. After confirming that a custom automation execution environment works with your automation code, you can publish the container image to a container registry and then provide the new or updated automation execution environment to automation controller. Using the same automation execution environment helps to ensure that the automation code runs consistently on both your system and in automation controller.

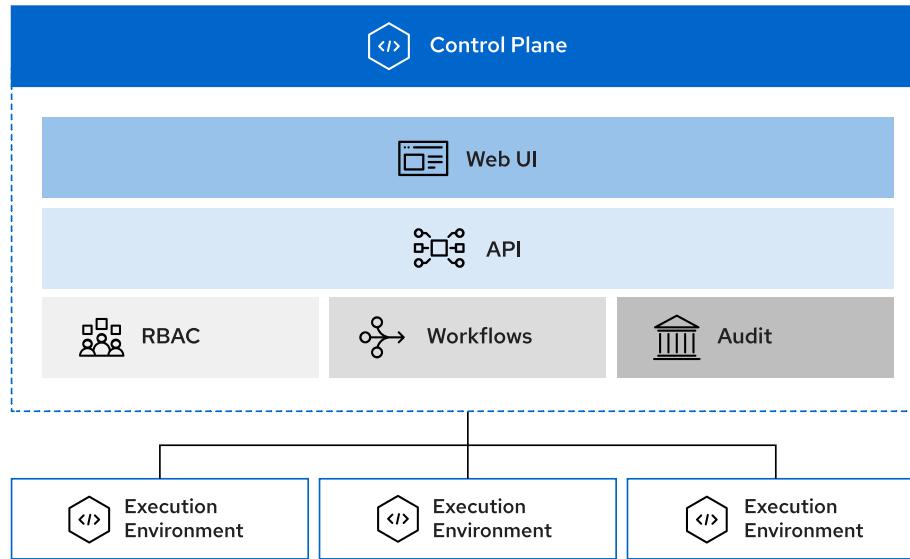


Figure 2.1: Architecture of Ansible Automation Platform 2

Automation controller design provides the following features:

- Decentralized and modular application
- Decoupled control plane and execution plane
- Containerized virtual environments
- On-demand scaling by using container orchestrators

Automation controller runs the control plane (with the web UI and API) on the automation controller system, and runs automation execution environments on other machines that are closer to the managed systems. This design increases efficiency and scaling.

Resources in Automation Controller

Automation controller provides a centralized location for organizations to run their Ansible Playbooks.

Several resources must exist before you can create a job template to test a playbook in automation controller.

Use the following resources to create a job template:

- A machine credential for connecting to the managed hosts.
- A source control credential for downloading and synchronizing remote content, such as from a Git repository.
- A project that specifies the location of content, such as playbooks.
- An inventory with at least one host.

The Automation Controller Web Interface

The automation controller dashboard provides a summary of information about hosts, inventories, and projects.

Chapter 2 | Automating Configuration and Remediation with Ansible

The left navigation bar provides quick access to resources, such as **Projects**, **Inventories**, **Job Templates**, and **Jobs**.

At the upper right of the interface, you can access your user profile, find the About page, view related documentation, and log out.

The dashboard contains the following sections:

- A **Job Status** graph that shows the number of successful and failed jobs over a specified time.
- A list of **Recent Jobs** that shows which jobs were recently run, their status, and the time that they were run.
- A list of **Recent Templates** that shows a summary of the most recently used templates.

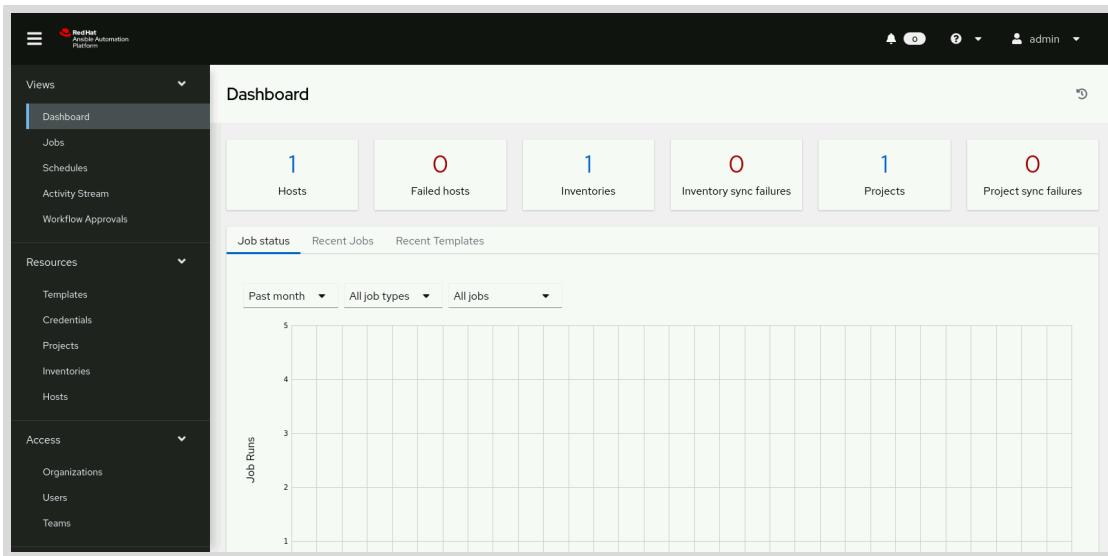


Figure 2.2: Automation controller web interface

Controlling User Access in Automation Controller

Different people who use an automation controller need different levels of access. Some users need to run existing job templates against a preconfigured inventory of machines. Others need to be able to modify particular inventories, job templates, and playbooks. Still others might need access to change anything in the automation controller installation.

You can create the following types of users in automation controller:

- *Normal users* have read and write access that is limited to the resources (such as inventory, projects, and job templates) for which that user is granted the appropriate roles and privileges.
- *System auditors* implicitly inherit the read-only capability for all objects within the environment.
- A *system administrator* (also known as a *superuser*) has full system administration privileges with full read and write privileges over the entire automation controller. A system administrator is typically responsible for managing all aspects of automation controller and for managing access to various users.

**Note**

The initial user (usually `admin`) that the installation process creates is a superuser. One superuser must always exist. To delete the `admin` user account, you must first create another superuser account.

For large deployments, it can be helpful to categorize users, teams, projects, and inventories into different departments. An *organization* is a logical collection of users, teams, projects, and inventories. All users must belong to an organization, and as part of installation, a *Default* organization is created. A *user* is someone who has access to automation controller with associated permissions and credentials. A *team* is a subdivision of an organization with associated users, projects, credentials, and permissions. A *project* is a logical collection of Ansible Playbooks. An *inventory* is a collection of hosts against which jobs must be launched.

Managing Credentials

Credentials are used for authentication when launching jobs against machines, synchronizing with inventory sources, and importing project content from a version control system.

You can grant users and teams the ability to use these credentials, without exposing the credential to the user. If a user moves to a different team or leaves the organization, then you do not have to rekey all the systems just because that credential was available in automation controller.

The automation controller uses SSH to connect to remote hosts (or the Windows equivalent). To pass the key from the automation controller to SSH, the key must be decrypted before it can be written to a named pipe. The automation controller then uses that pipe to send the key to SSH (so that it is never written to disk).

If passwords are used, then the automation controller handles the request by responding directly to the password prompt and decrypting the password before writing it to the prompt.

**Important**

After sensitive authentication data is entered into a credential and encrypted, it can no longer be retrieved in decrypted form through the automation controller web interface.

Some credential types that automation controller uses for authentication are as follows:

- *Machine credentials* are used by automation controller to authenticate to managed hosts on which it is running Ansible jobs.
- *SCM (source control manager)* credentials are used by projects to clone and update the contents of source code repositories from a remote revision control system, such as Git.

Listing Credentials

Navigate to **Resources > Credentials** to list credentials. Automation controller displays the name and credential type for each credential.

Click the name of any credential to display credential details.

Figure 2.3: List of existing credentials

Creating a Machine Credential

To create a machine credential, navigate to **Resources > Credentials** and click **Add**. Specify a name for the credential and choose the machine credential type. Specify additional settings and then click **Save**.

Setting	Description
Username	Automation controller uses this username to connect to the managed hosts (similar to the <code>remote_user</code> setting in an <code>ansible.cfg</code> file).
Password	Enter the password, or prompt for the password when the credential is used. A password or SSH private key is required. Using a password does not work if a managed host prevents password-based authentication over SSH.
SSH Private Key	This key is the associated private key for a public SSH key that is already copied to the managed host. A password or SSH private key is required.
Private Key Passphrase	If the private SSH key is password-protected, then enter the passphrase, or prompt for the passphrase when the credential is used.
Privilege Escalation Method	Enter the method that is used for privilege escalation (the equivalent of the <code>become_method</code> setting in an <code>ansible.cfg</code> file).
Privilege Escalation Username	The automation controller uses this user for tasks that require privilege escalation (the equivalent of the <code>become_user</code> setting in an <code>ansible.cfg</code> file).
Privilege Escalation Password	If the user requires a password for privilege escalation, then enter the password, or prompt for the password when the credential is used.

Creating a Source Control Credential

To create a source control credential, navigate to **Resources > Credentials** and click **Add**. Specify a name for the credential and choose the source control credential type.

Setting	Description
Username	Automation controller connects to the source control repository as this user.
Password	The password that is associated with the user. The source control repository might disable password-based authentication. In that case, use the SCM private key.
SCM Private Key	This key is the associated private key for a public SSH key to the managed host.
Private Key Passphrase	If the private SCM key is password-protected, then enter the passphrase.

Controlling Access to Credentials

Private credentials (credentials that are not assigned to an organization) are accessible only to their creators or to users that have the **System Administrator** or **System Auditor** user type. Other users cannot be assigned roles on private credentials.

To assign roles to credentials, the credential must have an organization. Users and teams in that organization can then share that credential through role assignments.

The following table lists the available credential roles:

Credential role	Description
Admin	Grants users full permissions on a credential. These permissions include deleting and modifying the credential, as well as the ability to use the credential in a job template.
Use	Grants users the ability to use a credential in a job template. How to use a credential in a job template is discussed later in this course.
Read	Grants users the ability to view the details of a credential. These users cannot decrypt the secrets that belong to that credential through the web UI.

When you create an organization credential, it is accessible only by the owner and users with either the **Admin** or **Auditor** role in the organization in which you created it. You must configure any additional access, if needed.

You must first save a credential before you can edit it to assign additional roles.

Use the following procedure to grant permissions to an existing organization credential:

1. Log in as a user with the **Admin** role for the organization in which the credential was created.
2. Click **Resources > Credentials** and then click the name of the credential to edit.
3. On the credential editor page, on the **Access** tab, click **Add** to add permissions.

4. Click either **Users** or **Teams**, and then click **Next** to select the users or teams to be assigned roles.
5. Click **Next** to display the list of available roles to apply.
6. Click **Save**.

**Important**

You can also add permissions for credentials through either the user or team management pages.

Creating Project Resources

Automation controller uses project resources to provide access to Ansible Playbooks. If a project uses source control, such as Git or Subversion, then automation controller synchronizes content from the remote source control repository.

Navigate to **Resources > Projects** to list existing projects. Click the **>** icon to the left of a project name to expand the project information. You can synchronize projects that use source control from the **Projects** page.

Use the **Revision** column to assess whether your project is current. For a Git repository, the revision string matches a commit hash.

The screenshot shows the Red Hat Ansible Automation Platform web interface. The left sidebar has a 'Views' section with links to Dashboard, Jobs, Schedules, Activity Stream, Workflow Approvals, and a 'Resources' section with links to Templates, Credentials, Projects (which is selected), Inventories, and Hosts. The main content area is titled 'Projects' and shows a table with two rows. The columns are Name, Status, Type, Revision, and Actions. The first row is for 'Controller Playbooks Project' with status 'Successful', type 'Git', revision 'd339950', and actions edit, sync, and delete. The second row is for 'Demo Project' with status 'Sync for revision', type 'Git', and actions edit, sync, and delete. At the top of the main area, there are buttons for Add, Delete, and search, along with a pagination indicator '1-2 of 2'.

Figure 2.4: List of existing projects

To create a project, navigate to **Resources > Projects** and click **Add**.

Projects can specify a default execution environment for job templates. If the environment is not specified, then the project defaults to the execution environment that is defined for automation controller. You can choose job templates for different execution environments.

Choose a source control credential type, such as Git, specify additional settings, and then click **Save**.

Setting	Description
Source Control URL	The URL to clone the remote repository.

Setting	Description
Source Control Branch/Tag/Commit	(Optional) A specific branch, tag, or commit for the repository. Use the Allow Branch Override option so that job templates can use a different branch, tag, or commit.
Source Control Credential	An <i>existing</i> credential to synchronize the remote repository.

Creating Inventory Resources

Similar to a local inventory file or directory, you can create inventory resources within automation controller. Inventories can contain groups of hosts as well as ungrouped hosts. You can configure variables that apply to the entire inventory (similar to the `all` group), to a specific group (similar to using the `groups_vars` directory), and to a specific host (similar to using the `host_vars` directory). When creating a job template, you must specify an inventory resource to use. This inventory is similar to specifying the `--inventory (-i)` option of the `ansible-navigator` command.

Automation controller can contain many inventories, both static and dynamic. Inventories that contain dynamic content display the status of the most recent inventory synchronization attempt. Static inventories display **Disabled** in the **Status** column.

Name	Status	Type	Organization	Actions
Controller Playbooks Inventory (Dev)	Success	Inventory	Default	
Demo Inventory	Disabled	Inventory	Default	

Figure 2.5: List of existing inventories

To create an inventory, navigate to **Resources > Inventories** and click **Add > Add inventory**. Specify a name for the inventory and click **Save**.

After creating the inventory, populate the inventory with groups and hosts. You can manually add the groups and hosts, or you can automatically populate them by using an existing inventory file in the project repository.

Manually Adding Groups and Hosts

To manually add a group, navigate to the **Groups** tab and then click **Add**. Enter a name for the group and then click **Save**.

To manually add an ungrouped host, which you can do at any time, navigate to the **Hosts** tab for the inventory and click **Add**. Enter a name for the host (often the fully qualified domain name for the host) and then click **Save**. To manually add a host to a group, navigate to the **Groups** tab for the inventory, click the group name, and then click the **Hosts** tab for the group. You can either associate an existing host within the inventory or add a new host to the group. When adding a new host, enter a name for the host (often the fully qualified domain name for the host) and then click **Save**.

The breadcrumb navigation displays the inventory name and the group name. In the following example, the `serverb.lab.example.com` host is added to the `apac` group in the `Web Servers (Prod)` inventory.

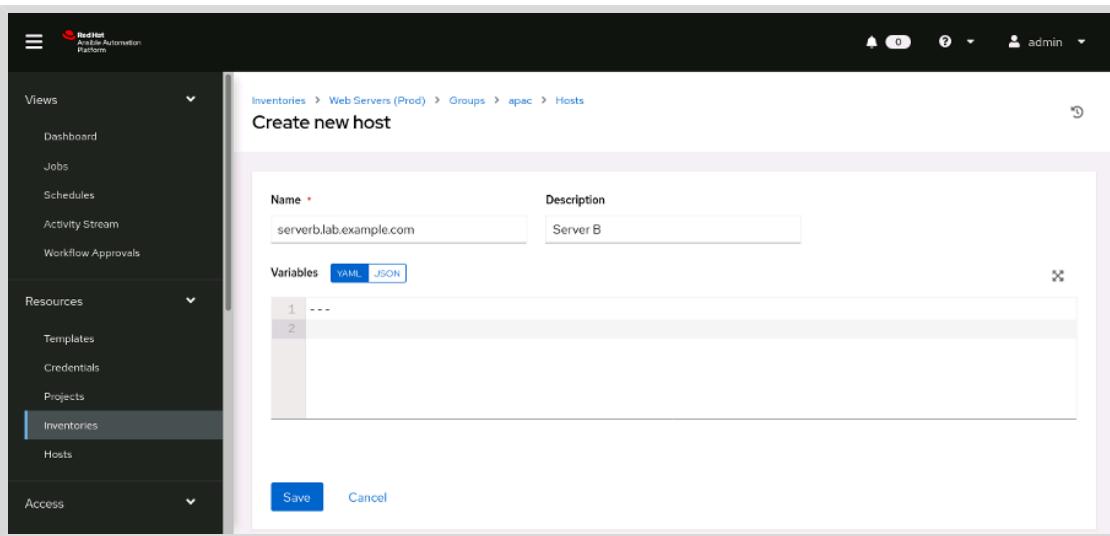


Figure 2.6: Adding a new host to a group

Populating Groups and Hosts by Using a Project Inventory File

If a project already contains an inventory file, then you can use the file to automatically populate groups and hosts within automation controller.

Within an existing group, navigate to the **Sources** tab and then click **Add**. Enter a name for the source and then select **Sourced from a Project** from the **Source** menu. Choose an existing project, select an inventory file that exists within the project, and then click **Save**.

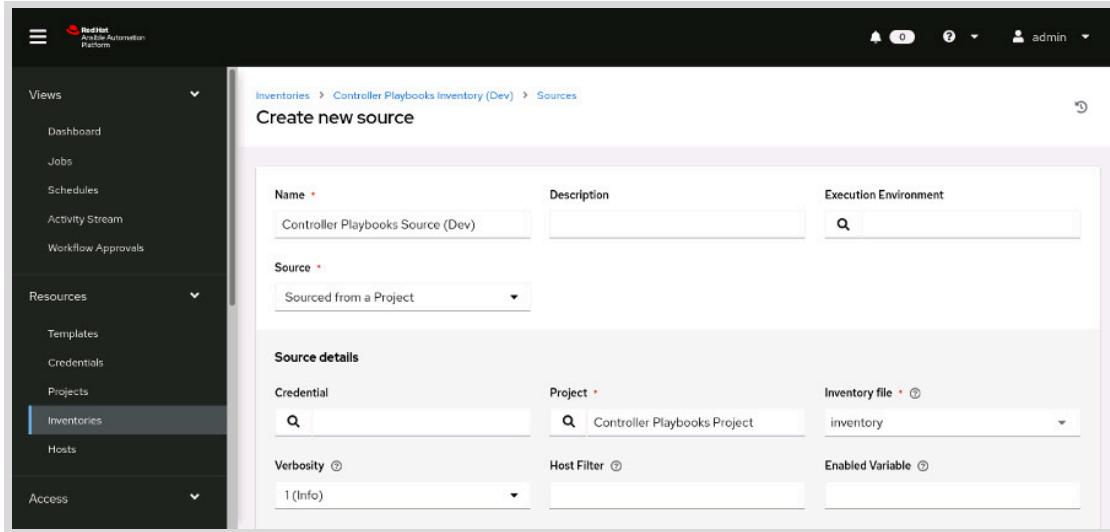


Figure 2.7: Adding a new inventory source from a project

Within an existing group, navigate to the **Sources** tab and then click **Start sync process** for the source. A successful inventory synchronization creates the hosts and groups that are defined in the selected inventory file.

Creating Job Template Resources

A job template is equivalent to running a playbook. Among other things, a job template includes an inventory, and might specify variables to override, or tags to either use or skip.

The **Templates** page lists existing templates. Expand template information by clicking the **>** icon to the left of any template name. Initiate a new job run by clicking **Launch Template**.

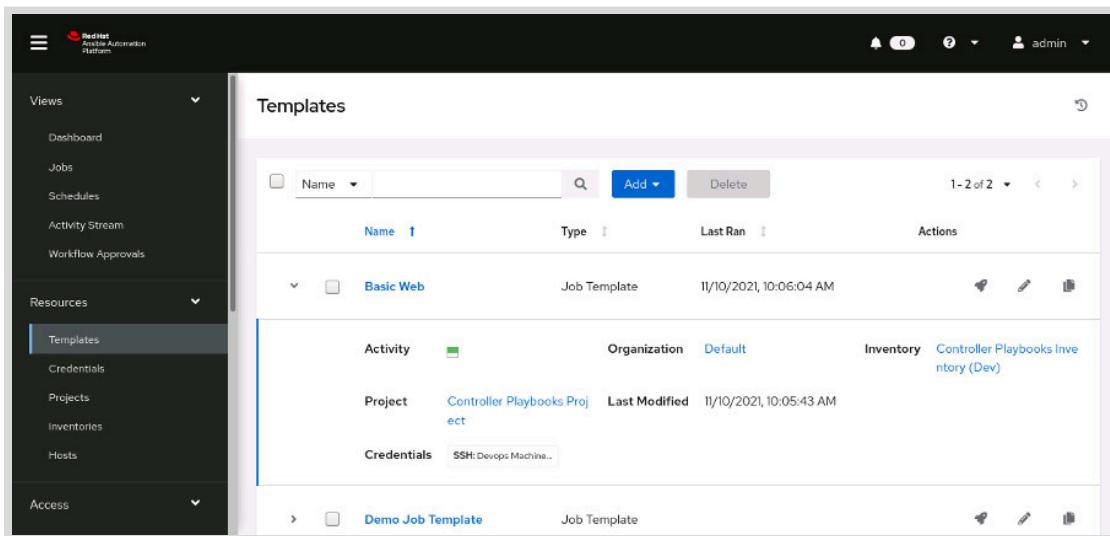


Figure 2.8: List of existing job templates

To create a job template, navigate to **Resources > Templates** and then click **Add > Add job template**. At a minimum, enter a name, select a project, and select a playbook. You can either choose an existing inventory or prompt for an inventory when the job template is launched. Job templates must specify necessary credentials, such as the machine credential that automation controller needs to connect to the managed hosts.

**Important**

Organizations that forbid storing credentials, such as passwords or passphrases, must configure job templates to prompt for credentials. Any job template that prompts for passwords (or other settings) must be run interactively.

Job templates include additional optional fields. When you run a playbook with the `ansible-navigator` command, many of these options have corresponding command-line options. You can specify these options as part of the job template, or be prompted for customizations when the job template is launched.

Job template setting or option	Equivalent command-line option
Inventory	<code>--inventory (-i)</code>
Execution environment	<code>--execution-environment-image (--eei)</code>
Variables	<code>--extra-vars (-e)</code>
Forks	<code>--forks (-f)</code>
Limit	<code>--limit (-l)</code>
Verbosity	<code>--verbose (-v)</code>
Job tags	<code>--tags (-t)</code>
Skip tags	<code>--skip-tags</code>
Privilege escalation	<code>--become (-b)</code>

Launching and Reviewing Jobs

When you launch a job template, automation controller uses an execution environment to run the playbook by using all the supplied information. If you configure the job template to prompt for information, such as an inventory, credentials, or variables, then automation controller prompts you for this information. Similarly, if the credentials for your job template prompt for passwords or passphrases, then automation controller prompts you for this information. You must interactively run the job templates that prompt for any information.

Navigate to `Views > Jobs` to review the job output. In addition to the output of playbook runs, the `Jobs` page displays inventory synchronization jobs and source control update jobs. Each job contains a number to show the order in which automation controller ran the job.

Expand job information by clicking the `>` icon to the left of any job name. Expanded details include who launched the job, which inventory was used, and which execution environment was used. Click a job name to see the full job output.

The screenshot shows the Red Hat Ansible Automation Platform web interface. The left sidebar has a 'Views' dropdown with 'Dashboard', 'Jobs' (which is selected), 'Schedules', 'Activity Stream', and 'Workflow Approvals'. Under 'Resources', it lists 'Templates', 'Credentials', 'Projects', 'Inventories', 'Hosts', and 'Access'. The main area is titled 'Jobs' and displays a table with three rows. The columns are 'Name', 'Status', 'Type', 'Start Time', 'Finish Time', and 'Actions'. The first row is '6 – Basic Web' (Successful, Playbook Run, 11/10/2021, 10:05:48 AM). The second row is '4 – Controller Playbooks Inventory (Dev) - Controller Playbooks Source (Dev)' (Successful, Inventory Sync, 11/10/2021, 10:04:00 AM). The third row is '1 – Controller Playbooks Project' (Successful, Source Control Update, 11/10/2021, 10:02:23 AM). There are search and filter buttons at the top of the table.

Figure 2.9: List of jobs



References

For more information, refer to the *Automation Controller User Guide v4.2.1* at <https://docs.ansible.com/automation-controller/4.2.1/html/userguide/index.html>

For more information, refer to *Automation Controller* at <https://www.redhat.com/en/technologies/management/ansible/automation-controller>

For more information, refer to *What's New in Ansible Automation Controller 4.0* at <https://access.redhat.com/articles/6184841>

► Guided Exercise

Managing Playbooks with Automation Controller

Run a playbook by using a preconfigured automation controller from Red Hat Ansible Automation Platform.

Outcomes

- Log in to an existing automation controller as a non-administrative user.
- Run an Ansible Playbook by using Red Hat Ansible Automation Platform.
- Confirm that the non-administrative user can use the credential object but not modify or view it.

Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

```
[student@workstation ~]$ lab start ansible-controller-intro
```

Instructions

- ▶ 1. On **workstation**, open Firefox to access <https://controller.lab.example.com>. If you encounter a warning that the dashboard's TLS certificate cannot be authenticated in this classroom, ignore the warning and add the certificate to your browser as an exception.
- ▶ 2. Log in to the automation controller dashboard as the `admin` user with the `redhat` password.
- ▶ 3. Verify that the `ansible-testuser` user exists in the automation controller.
 - 3.1. Navigate to the **Access > Users** dashboard. You should see the `ansible-testuser` user in the list that appears.
- ▶ 4. Add the `ansible-testuser` user to the **Default** organization with the **Member** role.
 - 4.1. Navigate to the **Access > Organizations** dashboard.
 - 4.2. Click the **Default** organization and navigate to the **Access** tab for the organization.
 - 4.3. Click **Add** and follow the prompts to add the `ansible-testuser` user to the organization. Apply the **Member** role to the user from the roles list.
- ▶ 5. Verify that the `ansible-testuser-credential` machine credential exists. Also, grant the **Use** role to the `ansible-testuser` user for the machine credential so that the user can authenticate to managed hosts while executing the Ansible Playbook.

- 5.1. Navigate to the **Resources > Credentials** menu. Verify that the **ansible-testuser-credential** item appears in the displayed list of available credentials.
 - 5.2. Click the **ansible-testuser-credential** link and navigate to the **Access** tab.
 - 5.3. Click **Add** and follow the prompts to add the **ansible-testuser** user. Apply the **Use** role from the roles list.
- **6.** Verify that the **RH415_project** project exists in the automation controller. Also, grant the **Use** role to the **ansible-testuser** user on this project.
- 6.1. Navigate to the **Resources > Projects** menu. Verify that the **RH415_project** project appears in the list that is displayed.
 - 6.2. Click the **RH415_project** link and navigate to the **Access** tab for the project.
 - 6.3. Click **Add** and follow the prompts to add the **ansible-testuser** user. Assign the **Use** role from the roles list.
- **7.** Create the **RH415-inventory** static inventory.
- 7.1. Navigate to the **Resources > Inventories** menu.
 - 7.2. Click **Add** to add an inventory. Use **RH415-inventory** for the inventory name and click **Save**.
- **8.** Grant the **Use** and **Ad Hoc** permissions to the **ansible-testuser** user for the **RH415-inventory** inventory.
- 8.1. Navigate to the **Access** tab for the **RH415-inventory** inventory.
 - 8.2. Follow the prompts to add the **ansible-testuser** user. Assign the **Use** and **Ad Hoc** roles from the role list.
- **9.** Create the **webservers** group within the **RH415-inventory** inventory to include the **servera** and **serverb** managed hosts.
- 9.1. Navigate to the **Groups** tab for the **RH415-inventory** inventory.
 - 9.2. Click **Add**.
 - 9.3. Specify **webservers** for the group name and click **Save**.
 - 9.4. Navigate to the **Hosts** tab and click **Add** to add a new host. Specify **servera** for the hostname and click **Save**.
 - 9.5. Repeat the same steps to add the **serverb** host as well.
- **10.** As the **ansible-testuser** user, verify that the **ansible-testuser-credential** machine credential exists.
- 10.1. Log out of the **admin** user by clicking the username in the upper right corner of the page and selecting **Logout**.
 - 10.2. Log in as the **ansible-testuser** user with the **redhat** password in the automation controller dashboard.

- 10.3. Navigate to the **Resources > Credentials** dashboard. You cannot modify the properties of the credential, because the `ansible-testuser` user is limited to the **Use** role. The **Use** role allows you to use this credential object to authenticate to the managed hosts while executing the Ansible Playbook, but it does not grant you the privileges to modify the properties of the credential object.
- **11.** Create a job template called `RH415-job` in the `RH415_project` project.
- 11.1. Navigate to the **Resources > Templates** dashboard.
 - 11.2. Click **Add** to add a new job template. Specify `RH415-job` for the job template name and use `RH415-inventory` for the inventory. You can click the **Search** button that is associated with the inventory field to pick the inventory from a list.
 - 11.3. Ensure that `RH415_project` is selected for the project.
 - 11.4. Select `hello_world.yml` from the drop-down menu in the **Playbook** field.
 - 11.5. Click the **Search** button of the **Credentials** field and select the radio button beside the `ansible-testuser-credential` item. Click **Select** to confirm the selection.
 - 11.6. Click **Save**.
- **12.** Launch the `RH415-job` job template.
- 12.1. Navigate to the **Resources > Templates** dashboard and click the **Launch Template** button with a rocket icon to the right of the `RH415-job` template. Verify that the job completes with a **Successful** state.

Finish

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

```
[student@workstation ~]$ lab finish ansible-controller-intro
```

► Lab

Automating Configuration and Remediation with Ansible

Ensure that your workstation is prepared to use Ansible and is configured with an appropriate configuration file and inventory, and use a provided playbook to ensure that several servers are in the correct configuration.

Outcomes

- Install the `ansible-navigator` package.
- Configure and use Ansible `inventories` and `ansible.cfg` files.
- Confirm that Ansible is working correctly and can connect to managed hosts.
- Run Ansible Playbooks to configure managed hosts.

Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

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

Instructions

1. On the workstation machine, install the `ansible-navigator` package that provides automation content navigator, to use that machine as your control node.
2. Ensure that the execution environment container image is downloaded for the `ansible-navigator` tool.



Note

You do not need to download the `ee-supported-rhel9` execution environment, because it is preloaded in your classroom.

3. Create the `/home/student/ansible-review` directory.
4. Navigate to the `/home/student/ansible-review` directory.
5. In the `/home/student/ansible-review` directory, create an Ansible configuration file named `ansible.cfg`. Use the following content and values.
6. In the `/home/student/ansible-review` directory, create the `inventory` file. Use the following content.
7. List all the managed hosts that are present in the inventory.
8. In the `/home/student/ansible-review` directory, create an Ansible Playbook file named `webserver.yml`. Use the following content and values.

9. Before running your playbook, validate the `webserver.yml` playbook syntax. Correct any reported errors before continuing.
10. Run the `webserver.yml` playbook. Read through the generated output to ensure that all tasks completed successfully.
11. Use the `curl` command to verify that both `servera.lab.example.com` and `serverb.lab.example.com` are configured as HTTPD servers.

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 ansible-review
```

Finish

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

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

► Solution

Automating Configuration and Remediation with Ansible

Ensure that your workstation is prepared to use Ansible and is configured with an appropriate configuration file and inventory, and use a provided playbook to ensure that several servers are in the correct configuration.

Outcomes

- Install the `ansible-navigator` package.
- Configure and use Ansible inventories and `ansible.cfg` files.
- Confirm that Ansible is working correctly and can connect to managed hosts.
- Run Ansible Playbooks to configure managed hosts.

Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

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

Instructions

1. On the workstation machine, install the `ansible-navigator` package that provides automation content navigator, to use that machine as your control node.
 - 1.1. Install the `ansible-navigator` package on the workstation machine.

```
[student@workstation ~]$ sudo dnf install ansible-navigator
[sudo] password for student: student
...output omitted...
Is this ok [y/N]: y
...output omitted...
Complete!
```

2. Ensure that the execution environment container image is downloaded for the `ansible-navigator` tool.



Note

You do not need to download the `ee-supported-rhel9` execution environment, because it is preloaded in your classroom.

- 2.1. Use the `ansible-navigator` tool to list the container images.

```
[student@workstation ~]$ ansible-navigator images
```

- 2.2. Press Esc to exit the image list.

Image	Tag	Execution environment	Created	Size
0 ee-supported-rhel9	latest	True	4 months ago	1.63 GB

```
^b/PgUp page up ^f/PgDn page down ↑ scroll esc back [0-9] goto :help help
```

3. Create the /home/student/ansible-review directory.

```
[student@workstation ~]$ mkdir ~/ansible-review
```

4. Navigate to the /home/student/ansible-review directory.

```
[student@workstation ~]$ cd ~/ansible-review  
[student@workstation ansible-review]$
```

5. In the /home/student/ansible-review directory, create an Ansible configuration file named `ansible.cfg`. Use the following content and values.

```
[student@workstation ansible-review]$ cat ansible.cfg  
[defaults]  
inventory      = ./inventory  
remote_user    = ansible-labuser  
  
[privilegeEscalation]  
become        = true  
becomeMethod   = sudo  
becomeUser     = root  
becomeAskPass  = false
```

6. In the /home/student/ansible-review directory, create the `inventory` file. Use the following content.

```
[student@workstation ansible-review]$ cat inventory  
[prod]  
servera  
serverb  
  
[test]  
workstation  
  
[webserver:children]  
prod  
test
```

7. List all the managed hosts that are present in the inventory.

```
[student@workstation ansible-review]$ ansible-navigator inventory \
-m stdout --graph
@all:
|--@ungrouped:
|--@webserver:
| |--@prod:
| | |--servera
| | |--serverb
| |--@test:
| | |--workstation
```

8. In the /home/student/ansible-review directory, create an Ansible Playbook file named `webserver.yml`. Use the following content and values.

```
[student@workstation ansible-review]$ cat webserver.yml
---
- name: Start a webserver
  hosts: prod
  tasks:
    - name: Install httpd package
      yum:
        name: httpd
        state: present

    - name: Start firewalld service
      service:
        name: firewalld
        state: started
        enabled: true

    - name: Copy content
      copy:
        content: "Welcome to RH415 webserver\n"
        dest: /var/www/html/index.html

    - name: Add http service to firewalld
      firewalld:
        service: http
        state: enabled
        immediate: true
        permanent: true

    - name: Start httpd service
      service:
        name: httpd
        state: started
        enabled: true
...
```

9. Before running your playbook, validate the `webserver.yml` playbook syntax. Correct any reported errors before continuing.

```
[student@workstation ansible-review]$ ansible-navigator run -m stdout \
  webserver.yml --syntax-check
playbook: /home/student/ansible-review/webserver.yml
```

10. Run the `webserver.yml` playbook. Read through the generated output to ensure that all tasks completed successfully.

```
[student@workstation ansible-review]$ ansible-navigator run -m stdout \
  webserver.yml
PLAY [Start a webserver] ****
TASK [Gathering Facts] ****
ok: [serverb]
ok: [servera]

TASK [Install httpd package] ****
changed: [serverb]
changed: [servera]

TASK [Start firewalld service] ****
ok: [serverb]
ok: [servera]

TASK [Copy content] ****
changed: [serverb]
changed: [servera]

TASK [Add http service to firewalld] ****
changed: [serverb]
changed: [servera]

TASK [Start httpd service] ****
changed: [serverb]
changed: [servera]

PLAY RECAP ****
servera    : ok=6    changed=4      unreachable=0      failed=0      skipped=0
             rescued=0   ignored=0
serverb    : ok=6    changed=4      unreachable=0      failed=0      skipped=0
             rescued=0   ignored=0
```

11. Use the `curl` command to verify that both `servera.lab.example.com` and `serverb.lab.example.com` are configured as HTTPD servers.

```
[student@workstation ansible-review]$ curl servera.lab.example.com
Welcome to RH415 webserver
[student@workstation ansible-review]$ curl serverb.lab.example.com
Welcome to RH415 webserver
```

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 ansible-review
```

Finish

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

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

Summary

- Effective automation tools help you to manage security by ensuring that all machines are correctly and consistently configured and patched.
- Automation helps you to mitigate human error and helps to ensure that your IT infrastructure is in a consistent and correct state.
- Ansible is an open source automation platform that can adapt to many workflows and environments.
- Red Hat Ansible Automation Platform is a fully supported version of Ansible that includes components and tools to help you develop, deploy, and manage your automation code.
- An Ansible Playbook consists of one or more plays. Each play targets a set of hosts with a list of tasks that are executed in order. Playbooks also confirm whether the system is in a certain state. If the system is not in the intended state, then the Ansible Playbook sets the system in that state.
- Automation content navigator (`ansible-navigator`) helps you develop and run your Ansible automation code.
- An inventory file lists the hosts and groups that you can use in your playbook.
- Automation controller is a service that helps you control, secure, and centrally manage your Ansible automation at scale.
- Automation controller provides central logging and management so that you can track who ran playbooks from the control node, at what time, affecting which hosts, and the results of those runs.

Chapter 3

Protecting Data with LUKS and NBDE

Goal

Encrypt data on storage devices with Linux Unified Key Setup (LUKS), and use Network-bound Disk Encryption (NBDE) to manage automatic decryption when servers are booted.

Sections

- Managing Storage Device Encryption with LUKS (and Guided Exercise)
- Automating Storage Device Decryption with NBDE (and Guided Exercise)

Lab

- Protecting Data with LUKS and NBDE

Managing Storage Device Encryption with LUKS

Objectives

- Create encrypted storage devices with LUKS, and manually open and mount storage on LUKS-encrypted devices.

Encrypting Storage with Linux Unified Key Setup (LUKS)

The threat of a system's physical compromise puts sensitive data in its storage devices at risk. Although this risk applies particularly to mobile systems such as laptops or removable media, servers might also have security requirements to protect the data that they store. Encrypting this data helps mitigate the risk of its exposure if the system is lost.

Red Hat Enterprise Linux supports block device encryption with LUKS technology. Encrypting a block device (such as a disk partition or LVM physical volume) can be completed during installation, but LUKS can also be configured after installation. However, to set up LUKS encryption after installation, you must reformat the file systems on the device.

In Red Hat Enterprise Linux, the default format for LUKS encryption is LUKS2. The earlier LUKS1 format remains fully supported and it is provided as a compatible format with earlier Red Hat Enterprise Linux releases. Compared to LUKS1 re-encryption, LUKS2 re-encryption is more robust and safer to use.

The LUKS2 format enables future updates of various parts without needing to modify binary structures. Internally, the LUKS2 format uses the JSON text format for metadata, provides redundancy of metadata, detects metadata corruption, and automatically repairs from a metadata copy.

LUKS is not recommended in the following scenarios:

- Solutions that require disk encryption to protect the data only when your system is off. After the system is on and LUKS has decrypted the disk, the files on that disk are available to anyone who has access to them.
- Scenarios that require multiple users to have distinct access keys to the same device. The LUKS1 format provides eight key slots and LUKS2 provides up to 32 key slots.
- Applications that require file-level encryption.

Online Re-encryption

The LUKS2 format supports re-encrypting encrypted devices when the devices are in use. For example, you do not have to unmount the file system on the device to perform the following tasks:

- Change the volume key
- Change the encryption algorithm

When encrypting a non-encrypted device, you must still unmount the file system. You can remount the file system after a short initialization of the encryption.

The LUKS1 format does not support online re-encryption.

LUKS Operations

LUKS performs the following operations for encrypting block devices:

- LUKS encrypts entire block devices and is therefore well suited for protecting content on mobile devices such as removable storage media or laptop disk drives.
- LUKS encrypts the swap devices by encrypting underlying contents of the block device. This encryption can also be useful with certain databases that use specially formatted block devices for data storage.
- LUKS uses the existing device mapper in the kernel subsystem.
- LUKS provides passphrase strengthening, which protects against dictionary attacks.
- LUKS devices contain multiple key slots, so users can add backup keys or passphrases.

Creating Encrypted Devices at Installation

When installing interactively, select **Encrypt** during partition creation. The system prompts for a passphrase to decrypt the partition when this option is selected. You must manually enter the passphrase every time that the system boots. If you are creating a custom partition table, then you can select which partitions to encrypt.

With automated installations, Kickstart can create encrypted block devices. If you prefer automated partitioning, then you can specify the use of encryption with the following directive:

```
autopart --type=lvm --encrypted --passphrase=PASSPHRASE
```

If you are configuring specific disk partitions, then you must specify the `--encrypted` and `--passphrase` options for each partition to be encrypted. For example, the following line in a Kickstart profile encrypts the existing `/dev/vda2` partition by using the specified passphrase for decryption, formats it with an ext4 file system, and mounts it on `/home`:

```
part /home --fstype=ext4 --size=10000 --onpart=vda2  
--encrypted --passphrase=PASSPHRASE
```

You can use similar syntax to encrypt an LVM physical volume:

```
part pv.01 --size=10000 --encrypted --passphrase=PASSPHRASE
```

The passphrase, `PASSPHRASE`, is stored in the Kickstart profile in plain text, so the Kickstart profile must be secured. If you omit the `--passphrase` option, then the installer prompts for the passphrase during installation.

Encrypting Devices with LUKS after Installation

You can use the `cryptsetup` command to encrypt existing devices after installation. The `cryptsetup` package that contains this command is available in the default Red Hat Enterprise Linux installation. You can use the `cryptsetup luksFormat` command to encrypt a partition.

The following example encrypts the `/dev/vdb1` device:

```
[root@host ~]# cryptsetup luksFormat /dev/vdb1

WARNING!
=====
This will overwrite data on /dev/vdb1 irreversibly.

Are you sure? (Type 'yes' in capital letters): YES
Enter passphrase for /dev/vdb1: PASSPHRASE
Verify passphrase: PASSPHRASE
```

**Warning**

The `cryptsetup luksFormat` command reformats the targeted block device, and deletes any data that is currently stored on that device.

You can use the `cryptsetup luksDump` command to verify the encryption information for an encrypted device. This command displays information such as the LUKS header information, and the key slots (each of which might contain a valid passphrase) that the LUKS-encrypted device uses. The output also reports the cipher that encrypts the device (by default, `aes-xts-plain64`).

```
[root@host ~]# cryptsetup luksDump /dev/vdb1

LUKS header information
Version:          2
Epoch:            3
Metadata area:   16384 [bytes]
Keyslots area:   16744448 [bytes]
UUID:             3f386240-06a3-4336-881f-7a1f96a2cd01
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:    argon2id
    Time cost: 4
    Memory:   748531
    Threads:  2
    Salt:     cb 4e 45 73 e4 dd 35 ac 12 41 0f 4d d6 52 0d 28
              d9 1c 41 33 44 d8 61 9a bf 31 d7 e1 79 9b ec 00
    AF stripes: 4000
```

```
AF hash: sha256
Area offset:32768 [bytes]
Area length:258048 [bytes]
Digest ID: 0
Tokens:
Digests:
 0: pbkdf2
Hash: sha256
Iterations: 94296
Salt: bb c5 e2 aa 9d 9c e1 f2 9b 8a 17 f4 3c ac f4 af
      3b b3 06 50 45 cf 50 0f 7c 23 b3 3c ee 77 49 64
Digest: df f4 af 04 03 a5 1b e5 24 69 6b 0d dc d5 28 40
      90 c6 d8 86 5f ff db 73 53 77 cf db 32 b9 9b 9c
```



Note

If you are creating a LUKS-encrypted file system after installation, be aware that some mount points have specific SELinux context assignments (the `/home` and `/tmp` directories, for example). You might need to use the `restorecon` command on the file system after it is mounted to its permanent location.

Opening and Mounting Encrypted Devices

You can use the `cryptsetup open` command to manually open a LUKS-encrypted partition and access its data. This command maps the partition to a decrypted, logical device-mapper block device, in the `/dev/mapper` directory. Then, if the decrypted device contains a file system, you can provide the name of this logical device as an input to the `mount` command to access it.

The following example decrypts the `/dev/vdb1` device and maps it to the `vdb1_encrypted` logical device-mapper device. To decrypt the partition, the `cryptsetup open` command prompts for the passphrase that was used to encrypt it.

```
[root@host ~]# cryptsetup open /dev/vdb1 vdb1_encrypted
Enter passphrase for /dev/vdb1: PASSPHRASE
```

The following example shows the status of the encrypted block device:

```
[root@host ~]# cryptsetup status vdb1_encrypted
/dev/mapper/vdb1_encrypted is active.
  type:   LUKS2
  cipher:  aes-xts-plain64
  keysize: 512 bits
  key location: keyring
  device:  /dev/vdb1
  sector size: 512
  offset:  32768 sectors
  size:    2015232 sectors
  mode:    read/write
```

**Important**

You can also configure LUKS-encrypted devices so that they are decrypted and mounted automatically at boot time. This configuration might require entering a password on the system's console at boot time, or you can use Network-Bound Disk Encryption (NBDE) to automatically decrypt the device if certain conditions are met. This topic is covered in a later section.

Unmounting and Closing Encrypted Devices

Before manually closing a LUKS-encrypted device, ensure that you unmounted its file systems and are not using that device for active logical volumes.

In the following example, the `cryptsetup close` command unmaps the LUKS-encrypted `/dev/vdb1` partition device from the `vdb1_encrypted` logical device-mapper device.

```
[root@host ~]# cryptsetup close vdb1_encrypted
```

**References**

`cryptsetup(8)` man page

For more information, refer to the *Encrypting Block Devices Using LUKS* chapter in the *Security Hardening* guide at

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html/security_hardening/encrypting-block-devices-using-luks_security-hardening

► Guided Exercise

Managing Storage Device Encryption with LUKS

Create an encrypted partition with LUKS. You then open it, format it with an XFS file system, and demonstrate that you can mount it. Finally, you unmount the file system and close the partition.

Outcomes

- Encrypt a partition by using LUKS.

Before You Begin

As the **student** user on the **workstation** machine, use the **lab** command to prepare your environment for this exercise, and to ensure that all required resources are available.

```
[student@workstation ~]$ lab start luks-manage
```

Instructions

- 1. Verify that the **/dev/vdb** additional disk is available on the **servera** machine. In this exercise, you use this disk to create an encrypted partition.

- 1.1. Log in to the **servera** machine as the **student** user.

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

- 1.2. Change to the **root** user. Use **student** as the password.

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

- 1.3. Verify that the **/dev/vdb** disk is available, and has no partition.

```
[root@servera ~]# parted -l  
Error: /dev/vdb: unrecognised disk label  
Model: Virtio Block Device (virtblk)  
Disk /dev/vdb: 1074MB  
Sector size (logical/physical): 512B/512B  
Partition Table: unknown  
Disk Flags:  
  
...output omitted...
```

Chapter 3 | Protecting Data with LUKS and NBDE

- 2. Create a partition on the /dev/vdb disk on the servera machine.

- 2.1. Use the **parted** command to create a partition on the /dev/vdb disk on the servera machine.

```
[root@servera ~]# parted /dev/vdb mklabel msdos mkpart primary xfs 1M 1G
Information: You may need to update /etc/fstab.
```

- 2.2. Verify that the partition is available.

```
[root@servera ~]# parted /dev/vdb print
Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 1074MB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Partition Flags:

Number  Start   End     Size    File system  Name  Flags
 1      1049kB  1073MB  1072MB        xfs
```

- 3. Encrypt the /dev/vdb1 partition with LUKS. Use the **redhatRHT** passphrase.

```
[root@servera ~]# cryptsetup luksFormat /dev/vdb1
WARNING!
=====
This will overwrite data on /dev/vdb1 irreversibly.

Are you sure? (Type 'yes' in capital letters): YES
Enter passphrase for /dev/vdb1: redhatRHT
Verify passphrase: redhatRHT
```

- 4. Map the encrypted partition to the **encryptedvdb1** logical device.

- 4.1. Use the **cryptsetup open** command to map the encrypted partition to the **encryptedvdb1** logical device.

```
[root@servera ~]# cryptsetup open /dev/vdb1 encryptedvdb1
Enter passphrase for /dev/vdb1: redhatRHT
```

- 4.2. Verify that the partition is now mapped to the /dev/mapper/encryptedvdb1 logical device.

```
[root@servera ~]# ls /dev/mapper/encryptedvdb1
/dev/mapper/encryptedvdb1
```

- 5. Create an XFS file system on the encrypted partition, and mount this file system on the /encrypted directory. Then, create a file in the /encrypted directory.

- 5.1. Create an XFS file system on the /dev/mapper/encryptedvdb1 device.

Chapter 3 | Protecting Data with LUKS and NBDE

```
[root@servera ~]# mkfs.xfs /dev/mapper/encryptedvdb1
meta-data=/dev/mapper/encryptedvdb1 isize=512    agcount=4, agsize=64384 blks
          =                      sectsz=512  attr=2, projid32bit=1
          =                      crc=1    finobt=1, sparse=1, rmapbt=0
          =                      reflink=1 bigtime=1 inobtcount=1
data     =                      bsize=4096   blocks=257536, imaxpct=25
          =                      sunit=0    swidth=0 blks
naming   =version 2           bsize=4096   ascii-ci=0, ftype=1
log      =internal log        bsize=4096   blocks=1566, version=2
          =                      sectsz=512  sunit=0 blks, lazy-count=1
realtime =none                extsz=4096   blocks=0, rtextents=0
```

5.2. Create the `/encrypted` directory.

```
[root@servera ~]# mkdir /encrypted
```

5.3. Mount the `/dev/mapper/encryptedvdb1` device on the `/encrypted` directory.

```
[root@servera ~]# mount -t xfs /dev/mapper/encryptedvdb1 /encrypted
```

5.4. Verify that the `/dev/vdb1` partition is correctly mounted.

```
[root@servera ~]# mount | grep encryptedvdb1
/dev/mapper/encryptedvdb1 on /encrypted type xfs
(rw,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=32k,noquota)
```

5.5. Create a file called `testfile` in the `/encrypted` directory by using the `touch` command.

```
[root@servera ~]# touch /encrypted/testfile
```

► 6. Unmount the file system and unmap the encrypted partition.

6.1. Unmount the file system from the `/encrypted` directory.

```
[root@servera ~]# umount /encrypted
```

6.2. Unmap the encrypted partition.

```
[root@servera ~]# cryptsetup close encryptedvdb1
```

► 7. Return to the workstation machine as the student user.

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



Important

The encrypted `/dev/vdb1` device that is created in this exercise is required to complete the *Guided Exercise: Automating Storage Device Decryption with NBDE*. If you choose to skip the latter guided exercise, then run the `lab finish luks-nbde` command after running the `lab finish luke-manage` command at the end of this guided exercise.

Finish

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

```
[student@workstation ~]$ lab finish luks-manage
```

Automating Storage Device Decryption with NBDE

Objectives

- Manage decryption policy and automatically decrypt storage when specified conditions are met, by using NBDE.

Introducing Network-bound Disk Encryption (NBDE)

In many environments, the default workflow to decrypt devices that are encrypted by using Linux Unified Key Setup (LUKS) is to manually type the password for disk decryption. When a system starts, the password must be entered manually to allow the system to mount the encrypted disk. Typing a passphrase is either not convenient or not possible for large data centers or cloud VM instances. Network-bound Disk Encryption (NBDE) securely automates the decryption of those encrypted disks without manually entering any passphrase at boot time, by ensuring that certain criteria are met. You can use NBDE to automatically decrypt LUKS storage devices that contain root and non-root file systems.

Overview of NBDE Components

The architecture of NBDE uses two key components: the Clevis framework and the Tang server. The Clevis framework is a pluggable framework that supports NBDE on the client side and automates unlocking LUKS-encrypted block devices. The Tang server supports NBDE on the server side, and makes data available on a system when that system can reach the Tang server over a specific secure network. The Clevis framework supports plug-ins, also called *pins*, to interact with Tang servers. Both the Clevis framework and the Tang server use the JOSE framework as their back end for encryption and decryption.

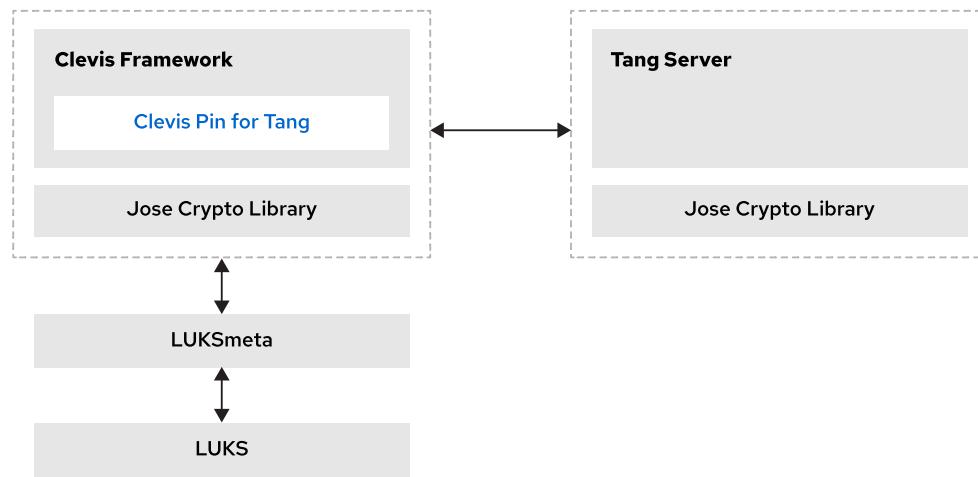


Figure 3.1: NBDE architecture with Clevis and Tang

Clevis gets a list of the keys for each Tang server, and uses one of those keys to generate an encryption key. The Clevis pin then uses that encryption key to encrypt the data. The encryption

generates some state information, which is stored in the LUKS header. That state information is accessible with the `luksmeta show` command.

Configuring NBDE with System Roles

The RHEL System Roles collection provides Ansible roles and modules to create consistent configurations for RHEL systems.

The `rhel-system-roles` package contains the `nbde_client` and `nbde_server` roles to automate the deployment of NBDE by using Clevis and Tang in one or more servers.

To use the RHEL System Roles, you must install the `ansible-core` and `rhel-system-roles` packages on the control node, and you must grant permissions and access to the managed nodes.

Setting up Tang Servers

Use the `nbde_server` role to deploy and configure Tang servers:

```
---
- hosts: all

vars:
  nbde_server_rotate_keys: yes
  nbde_server_manage_firewall: true
  nbde_server_manage_selinux: true

roles:
  - rhel-system-roles.nbde_server
```

Setting up the Clevis Client

The `nbde_client` role configures the Clevis client. This role requires the volumes to be already encrypted with LUKS, and that the role can bind a volume to one or more NBDE servers. You can preserve the existing passphrase or delete it. If you remove the original passphrase, then the volume can be unlocked by using only NBDE. If you provide a passphrase and a key file in the playbook, then the role uses the first entry and ignores all other entries.

You can use a Tang binding to map a device to one or more servers. You can have multiple bindings for the same device. To create or update a binding, set the `state` variable to the `present` value. To remove a binding, set the `state` variable to the `absent` value.

The following example configures the Clevis client for two devices with two Tang servers:

```
---
- hosts: all

vars:
  nbde_client_bindings:
    - device: /dev/rhel/root
      encryption_key_src: /etc/luks/keyfile
      servers:
        - http://server1.example.com
        - http://server2.example.com
    - device: /dev/rhel/swap
      encryption_key_src: /etc/luks/keyfile
```

```
servers:  
  - http://server1.example.com  
  - http://server2.example.com  
  
roles:  
  - rhel-system-roles.nbde_client
```

**Note**

The Clevis client requires networking to be available when mounting Tang bindings. You can use the `grubby` tool to ensure that networking is available during the early boot process.

```
[root@host ~]# grubby --update-kernel=ALL --args="rd.neednet=1"
```

Manual NBDE Configuration

If the RHEL System Role configuration method is not possible, then you can instead manually install and configure NBDE.

Configuring a Tang Server

To begin configuring a Tang server, install the `tang` package. This package installs the `tangd` service, which uses `systemd` socket activation, so it starts with the first connection.

Use the following command to enable socket activation for the `tangd` service:

```
[root@host ~]# systemctl enable tangd.socket --now  
Created symlink from /etc/systemd/system/multi-user.target.wants/tangd.socket to /  
usr/lib/systemd/system/tangd.socket.
```

**Note**

By default, the `tangd` service binds to the 80/TCP port. Red Hat Enterprise Linux blocks this port by default.

To stop the `tangd` service, disable the socket activation for that service as follows:

```
[root@host ~]# systemctl disable tangd.socket --now  
Removed symlink /etc/systemd/system/multi-user.target.wants/tangd.socket.
```

**Note**

You can test whether a Tang server is accessible:

```
[root@host ~]# curl -f http://demotang.lab.example.com/adv
```

Managing Keys for Tang Servers

Red Hat recommends that you periodically rotate the signature and exchange keys that a Tang server uses. The rotation interval for those keys depends on your application, key sizes, and organizational policies. The Tang server stores the signature and exchange keys in the /var/db/tang directory. To rotate the Tang server keys, rename the existing keys to include a dot as a prefix, and then generate new keys with the jose command for the JOSE framework, as follows:

```
[root@host ~]# cd /var/db/tang
[root@host tang]# jose jwk gen -i '{"alg":"ES512"}' \
    -o signature.jwk
[root@host tang]# jose jwk gen -i '{"alg":"ECMR"}' \
    o exchange.jwk
```

Existing client connections use the earlier keys and the Tang server uses the new keys automatically for new client bindings. When all existing client connections finish, you can safely remove the earlier keys.



Note

You must update the clients to use the new keys. To refresh the client keys usage, bind the encrypted device with the tang server again by using the following command:

```
[root@host ~]# clevis luks bind -d /dev/DEVICE tang '{"url":"http://
demotang.lab.example.com"}'
```

Configuring the Clevis Framework

To configure the Clevis framework with LUKS support, install the clevis, clevis-luks, and clevis-dracut packages. The clevis luks bind command binds a block device to a Tang server. The command also associates a binding policy with the device, which decrypts the device based on the Tang server availability. Available binding policies include the REST HTTP escrow server policy, Shamir's Secret Sharing (SSS) policy, and the Tang binding server policy. The following example binds the /dev/vda1 device to the demotang.lab.example.com Tang server:

```
[root@host ~]# clevis luks bind -d /dev/vda1 tang '{"url":"http://
demotang.lab.example.com"}'
```

The previous example uses the -d option to specify the device that binds to the Tang server.

The example also associates a binding policy, the Tang binding policy, to bind the device to the Tang server. A Tang binding server policy definition requires the base URL for the Tang server. Other policies are also available, such as the SSS policy.

If the policy requirements are not met, such as if the Tang server is unavailable, then the Clevis framework prompts for the LUKS passphrase during the boot process.



Important

If you are configuring Clevis to decrypt a block device that contains the root file system, then you must re-create the `initramfs` image with the `dracut -f` command. For other block devices, you must configure path-based execution by enabling the `clevis-luks-askpass.path` systemd unit as follows:

```
[root@host ~]# systemctl enable clevis-luks-askpass.path
Created symlink from /etc/systemd/system/remote-fs.target.wants/clevis-luks-
askpass.path to /usr/lib/systemd/system/clevis-luks-askpass.path.
```

Shamir's Secret Sharing (SSS)

The SSS policy supports the definition of complex policies with multiple Tang servers. You can define a policy with SSS that requires a minimum number of Tang servers to be available before a LUKS-encrypted block device can be decrypted.

The following example binds the `/dev/vdb1` device to an SSS policy that defines three Tang servers and requires at least two of them to be available for automatic decryption to occur.

```
[root@host ~]# cfg='{"t":2,"pins":{"tang":[\n    {"url":"http://demotang1.lab.example.com"},\n    {"url":"http://demotang2.lab.example.com"},\n    {"url":"http://demotang3.lab.example.com"}]}'
[root@host ~]# clevis luks bind -d /dev/vdb1 sss "$cfg"
The advertisement contains the following signing keys:
```

```
4R1tkfaTw-67bw0uxTAmTprUPoo
```

Do you wish to trust these keys? [ynYN] Y

The advertisement contains the following signing keys:

```
gks_IaVo1yog0KuQei95rg_yGns
```

Do you wish to trust these keys? [ynYN] Y

The advertisement contains the following signing keys:

```
vA5xAeUiKPqvkg4UyR4TemzXoAw
```

Do you wish to trust these keys? [ynYN] Y

You are about to initialize a LUKS device for metadata storage.

Attempting to initialize it may result in data loss if data was already written into the LUKS header gap in a different format. A backup is advised before initialization is performed.

Do you wish to initialize /dev/vdb1? [yn] y

Enter existing LUKS password: **demopass**

For each Tang server, you must trust the key for that server.

Persistently Mounting LUKS File Systems

Edit the `/etc/crypttab` file to specify the encrypted block devices to open, decrypt, and map at boot time. This file contains an entry for each encrypted block device. Two entries in `/etc/crypttab` might appear as follows:

```
decrypted1 /dev/vdb1 none _netdev
decrypted2 UUID=43d8995e-b876-4385-b124-7e402446d6c7 none _netdev
```

The first field contains the name to use for the decrypted block device. This name maps to the corresponding `/dev/mapper/name` device file. In the example, `/dev/mapper/decrypted1` is the name of the first mapped and decrypted block device.

The second field includes either the device name or the LUKS UUID (from the `cryptsetup luksUUID <device>` command) for the encrypted LUKS device. In this example, the encrypted `/dev/vdb1` block device maps to the decrypted `/dev/mapper/decrypted1` device mapping. The encrypted block device that is labeled with the `43d8995e-b876-4385-b124-7e402446d6c7` LUKS UUID maps to the decrypted `/dev/mapper/decrypted2` device mapping.

The third field specifies the absolute path to a file that contains the encryption password. The field is set to `none` in the example, which has special meaning. In this case, either the boot process pauses for the manual entry of the encryption password on the machine's console, or the boot process triggers block device decryption with NBDE.

Finally, the last field includes a comma-separated list of options, and in this case specifies only the `_netdev` option. At boot time, NBDE attempts to unlock all block devices with the `_netdev` option.



Important

If you are using NBDE Tang Servers to decrypt devices, include the `_netdev` option in the `/dev/fstab` file. NBDE uses the network to contact those servers, and so decryption cannot occur until networking is available.

If you are not using NBDE and the device does not need the network to be available to be decrypted, then you can omit the fourth field or use other options such as the `discard` option.

When the LUKS-encrypted block device maps to the `/dev/mapper/name` device mapping, you can use the device mapping to mount this device to a specific mount point with the `/etc/fstab` file.

The following example mounts the file system of the decrypted `/dev/mapper/decrypted1` device mapping to the `/encrypted` directory. This example assumes that the underlying file system on the device is XFS.

```
...output omitted...
/dev/mapper/decrypted1  /encrypted      xfs      _netdev      1 2
```

Device Decryption at Boot Time

You can use the `/etc/crypttab` file alone to configure device decryption at boot time with LUKS. LUKS decrypts the device by prompting you to manually enter passwords into the console

Chapter 3 | Protecting Data with LUKS and NBDE

at boot time or by storing passwords in plain text on a decrypted device. This way of decryption is inadequate for use cases where servers must boot unattended and still keep the decryption password secure.

NBDE provides a mechanism to automatically decrypt LUKS devices at boot time securely. This mechanism is based on two key components:

- The Clevis client, which runs on the system, decrypts devices, and defines the policy whose conditions must be met for the data to be decrypted.
- The Tang server, which Clevis clients contact to determine whether they are booting with access to a secure network and therefore decrypt their LUKS devices.

By using NBDE, you can prevent automatic decryption of a device that is removed from the data center.



References

`clevis-encrypt-tang(1)`, `clevis-encrypt-sss(1)`, and the `jose-jwk-gen(1)` man pages

For more information, refer to the *Network-bound Disk Encryption* chapter in the *Red Hat Enterprise Linux 9 Security Hardening Guide* at

[https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html/security_hardening/configuring-automated-unlocking-of-encrypted-volumes-using-policy-based-decryption](https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html/security_hardening/configuring-automated-unlocking-of-encrypted-volumes-using-policy-based-decryption_security-hardening#network-bound-disk-encryption_configuring-automated-unlocking-of-encrypted-volumes-using-policy-based-decryption)

► Guided Exercise

Automating Storage Device Decryption with NBDE

Use NBDE to automatically decrypt a LUKS device at boot time, by using three Tang servers if they are available, and falling back to manual decryption by password if they are not.

Outcomes

- Install and configure a Tang server.
- Decrypt a LUKS partition with multiple Tang servers.

Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.



Note

You must complete the *Guided Exercise: Managing Storage Device Encryption with LUKS* before starting this section.

```
[student@workstation ~]$ lab start luks-nbde
```

Instructions

- 1. Use Ansible to configure the `serverb`, `serverc`, and `serverd` machines as Tang servers. Those hosts are in the `servers` group in the `/home/student/nbde_servers.yml` inventory file. Write and run an Ansible Playbook that uses the `nbde_server` system role to complete this task.
- 1.1. Use a text editor to write the `/home/student/nbde_servers.yml` Ansible Playbook that configures hosts in the `servers` group by using the `nbde_server` system role. Add the following content:

```
---
- hosts: servers
  become: true
  become_method: sudo

  vars:
    nbde_server_rotate_keys: yes
    nbde_server_manage_firewall: true
    nbde_server_manage_selinux: true

  roles:
    - rhel-system-roles.nbde_server
```

Chapter 3 | Protecting Data with LUKS and NBDE

- 1.2. Run the `/home/student/nbde_servers.yml` Ansible Playbook by using the `ansible-playbook` command. Include the `--ask-become-pass` option to provide the student sudo password interactively. Use the `/home/student/RH415/labs/luks-nbde/inventory` inventory file.

```
[student@workstation ~]$ ansible-playbook \
  -i /home/student/RH415/labs/luks-nbde/inventory \
  --ask-become-pass ~/nbde_servers.yml
...output omitted...
PLAY RECAP
*****
serverb                  : ok=26    changed=5      unreachable=0    failed=0
skipped=29    rescued=0   ignored=0
serverc                  : ok=26    changed=5      unreachable=0    failed=0
skipped=29    rescued=0   ignored=0
serverd                  : ok=25    changed=5      unreachable=0    failed=0
skipped=30    rescued=0   ignored=0
```

- 2. Configure the encrypted partition to automatically decrypt and mount its file system on the `/encrypted` directory at boot time.

- 2.1. Log in to the `servera` machine as the `student` user.

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

- 2.2. Change to the `root` user. Use `student` as the password.

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

- 2.3. Edit the `/etc/crypttab` file to open the encrypted partition at boot time. Add the following content:

```
encryptedvdb1      /dev/vdb1  none  _netdev
```

- 2.4. Update the `/etc/fstab` file to mount the file system in the encrypted partition on the `/encrypted` directory.

```
...output omitted...
/dev/mapper/encryptedvdb1  /encrypted      xfs      _netdev      1 2
```

**Note**

A problem in `/etc/fstab` might bring the machine to an unusable state. If this problem happens, then rebuild the classroom environment.

- 2.5. Return to the `workstation` machine as the `student` user.

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

- 3. Configure the hosts in the `clients` section in the `/home/student/RH415/labs/luks-nbde/inventory` Ansible inventory file as NBDE (Clevis) clients. Write and run an Ansible Playbook that uses the `nbde_client` system role to complete this task.

- 3.1. Use a text editor to write the `/home/student/nbde_clients.yml` Ansible Playbook. Configure hosts in the `clients` group as NBDE clients of your Tang servers by using the `nbde_client` system role. Set the threshold for the minimum number of available Tang servers to three. Add the following content:

```
---
- hosts: clients
  become: true
  become_method: sudo

vars:
  nbde_client_bindings:
    - device: /dev/vdb1
      encryption_password: redhatRHT
      servers:
        - http://serverb.lab.example.com
        - http://serverc.lab.example.com
        - http://serverd.lab.example.com
  threshold: 3

roles:
  - rhel-system-roles.nbde_client
```

- 3.2. Run the `/home/student/nbde_clients.yml` Ansible Playbook by using the `ansible-playbook` command. Include the `--ask-become-pass` option to provide the student sudo password interactively. Use the `/home/student/RH415/labs/luks-nbde/inventory` inventory file.

```
[student@workstation ~]$ ansible-playbook \
  -i /home/student/RH415/labs/luks-nbde/inventory \
  --ask-become-pass ~/nbde_clients.yml
PLAY RECAP
*****
servera : ok=20  changed=9  unreachable=0  failed=0  skipped=1  rescued=0
           ignored=0
```

- 4. Turn off one of the Tang servers to test manual authentication at boot. Disable socket activation for the Tang server on `serverb`. Reboot `servera`, one of the NBDE clients. Verify that the `/dev/vdb1` LUKS-encrypted partition on the `servera` machine is decrypted, and that its file system is mounted automatically on the `/encrypted` directory, by providing the encrypted partition passphrase at boot time. The system asks for the passphrase at boot time because only two Tang servers are available.

Chapter 3 | Protecting Data with LUKS and NBDE

- 4.1. Log in to the **serverb** machine as the **student** user. You do not need to enter any password.

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

- 4.2. Change to the **root** user. Use **student** as the password.

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

- 4.3. Disable socket activation for the Tang server. When done, log out of the **serverb** machine.

```
[root@serverb ~]# systemctl disable tangd.socket --now  
Removed symlink /etc/systemd/system/multi-user.target.wants/tangd.socket.  
[root@serverb ~]# logout  
[student@serverb ~]$ logout  
Connection to serverb closed.  
[student@workstation ~]$
```

- 4.4. Log in to the **servera** machine as the **student** user. You do not need to enter any password.

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

- 4.5. Change to the **root** user and reboot the machine. Use **student** as the password.

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

- 4.6. Open the **servera** console. The console prompts you to manually enter the passphrase for the encrypted partition. Enter **redhatRHT** as the passphrase.

- 4.7. On **workstation**, log in to **servera** as the **student** user. You do not need to enter any password.

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

- 4.8. Use the **sudo -i** command to change to the **root** user. Use **student** as the password.

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

- 4.9. Verify that the file system in the encrypted partition is mounted on the /encrypted directory. When done, log out of the servera machine.

```
[root@servera ~]# mount | grep /encrypted
/dev/mapper/encryptedvdb1 on /encrypted type xfs
(rw,relatime,seclabel,attr2,inode64,noquota,_netdev)
[root@servera ~]# logout
[student@servera ~]$ logout
[student@workstation ~]$
```

- 5. Make sure that all the Tang servers are running, to test automatic authentication and decryption at boot. Enable socket activation for the Tang server on the serverb machine. Reboot the servera machine to verify that the /dev/vdb1 LUKS-encrypted partition is decrypted and that its file system is mounted automatically on the /encrypted directory.
- 5.1. Log in to the serverb machine as the student user. You do not need to enter any password.

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

- 5.2. Change to the root user. Use student as the password.

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

- 5.3. Enable socket activation for the Tang server. When done, log out of the serverb machine.

```
[root@serverb ~]# systemctl enable tangd.socket --now
Created symlink /etc/systemd/system/multi-user.target.wants/tangd.socket → /usr/
lib/systemd/system/tangd.socket.
[root@serverb ~]# logout
[student@serverb ~]$ logout
Connection to serverb closed.
[student@workstation ~]$
```

- 5.4. After the servera machine completes the reboot, log in as the student user. You do not need to enter any password.

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

- 5.5. Verify that the file system in the encrypted partition is mounted on the /encrypted directory.

```
[student@servera ~]$ mount | grep /encrypted
/dev/mapper/encryptedvdb1 on /encrypted type xfs
(rw,relatime,seclabel,attr2,inode64,noquota,_netdev)
```

- 5.6. Verify that the previously created file, `testfile`, is still available in the `/encrypted` directory. When done, log out of the `servera` machine.

```
[student@servera ~]$ ls /encrypted
testfile
[student@servera ~]$ logout
Connection to servera closed.
[student@workstation ~]$
```

- ▶ 6. Rotate the keys for the Tang servers by running the `/home/student/nbde_servers.yml` playbook again.
- 6.1. Run the `/home/student/nbde_servers.yml` Ansible Playbook by using the `ansible-playbook` command. Include the `--ask-become-pass` option to provide the student sudo password interactively. Use the `/home/student/RH415/labs/luks-nbde/inventory` inventory file.

```
[student@workstation ~]$ ansible-playbook \
  -i /home/student/RH415/labs/luks-nbde/inventory \
  --ask-become-pass ~/nbde_servers.yml
...output omitted...
PLAY RECAP
*****
serverb                  : ok=25    changed=1      unreachable=0      failed=0
  skipped=30   rescued=0    ignored=0
serverc                  : ok=26    changed=1      unreachable=0      failed=0
  skipped=29   rescued=0    ignored=0
serverd                  : ok=26    changed=1      unreachable=0      failed=0
  skipped=29   rescued=0    ignored=0
```

Finish

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

```
[student@workstation ~]$ lab finish luks-nbde
```

► Lab

Protecting Data with LUKS and NBDE

Create an encrypted storage device with LUKS and configure it to automatically decrypt at boot time securely by using NBDE.

Outcomes

- Encrypt a partition with LUKS.
- Decrypt a LUKS partition with multiple Tang servers.

Before You Begin

As the **student** user on the **workstation** machine, use the **lab** command to prepare your environment for this exercise, and to ensure that all required resources are available.

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

Instructions

- Verify that an additional disk is available on the **serverb** machine.
- Create a partition on the additional disk on the **serverb** machine. Use the following configuration for the partition:

Field	Value
Disk label	msdos
Partition	primary
FS type	xfs
Starting block	1M
Ending block	1G

- Encrypt the **vdb1** partition with LUKS. Use **redhatRHT** as the encryption password.
- Open the device with an **encryptedvdb1** map name.
- Create an XFS file system on the encrypted partition, and mount this file system on the **/encrypted** directory. Create a file called **testfile** in the **/encrypted** directory by using the **touch** command.
- Unmount the file system from the **/encrypted** mount point and lock the encrypted partition.
- Edit and apply the **/home/student/RH415/labs/luks-review/nbde_setup.yml** Ansible Playbook to associate the LUKS-encrypted partition on the **/dev/vdb1** device with Tang servers on the **serverc** and **serverd** machines. Use the **/home/student/RH415/**

`labs/luks-review/inventory` inventory file. Configure SSS encryption so that at least two Tang servers must be available to decrypt the partition.

8. On the `serverb` machine configure the encrypted partition to automatically decrypt and mount on the `/encrypted` directory at boot time. Reboot the `serverb` machine.
9. After the `serverb` machine reboots, verify that the LUKS-encrypted partition on the `/dev/vdb1` device is decrypted and is mounted automatically on the `/encrypted` directory.
10. Rotate the keys for the Tang servers by running the `/home/student/RH415/labs/luks-review/nbde_setup.yml` Ansible Playbook again.

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 luks-review
```

Finish

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

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

► Solution

Protecting Data with LUKS and NBDE

Create an encrypted storage device with LUKS and configure it to automatically decrypt at boot time securely by using NBDE.

Outcomes

- Encrypt a partition with LUKS.
- Decrypt a LUKS partition with multiple Tang servers.

Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

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

Instructions

- Verify that an additional disk is available on the `serverb` machine.
 - Log in to the `serverb` machine as the `student` user. You do not need to enter any password.

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

- Use the `sudo -i` command to change to the `root` user. Use `student` as the password.

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

- Verify that the `vdb` disk is available and has no partition.

```
[root@serverb ~]# parted -l
...output omitted...
Error: /dev/vdb: unrecognised disk label
Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 1074MB
Sector size (logical/physical): 512B/512B
Partition Table: unknown
Disk Flags:
```

- Create a partition on the additional disk on the `serverb` machine. Use the following configuration for the partition:

Field	Value
Disk label	msdos
Partition	primary
FS type	xfs
Starting block	1M
Ending block	1G

- 2.1. Use the `parted` command to create a partition on the additional disk on the `serverb` machine. Use the parameters from the previous table.

```
[root@serverb ~]# parted /dev/vdb \
  mklabel msdos
...output omitted...
[root@serverb ~]# parted /dev/vdb \
  mkpart primary xfs 1M 1G
...output omitted...
```

- 2.2. Verify that the partition is available.

```
[root@serverb ~]# parted /dev/vdb print
Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 1074MB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start   End     Size    Type      File system  Flags
1        1049kB  1074MB  1073MB  primary
```

3. Encrypt the `vdb1` partition with LUKS. Use `redhatRHT` as the encryption password.

- 3.1. Use the `cryptsetup luksFormat` command to encrypt the `vdb1` partition with LUKS.

```
[root@serverb ~]# cryptsetup luksFormat /dev/vdb1
WARNING!
=====
This will overwrite data on /dev/vdb1 irreversibly.

Are you sure? (Type uppercase yes): YES
Enter passphrase: redhatRHT
Verify passphrase: redhatRHT
```

4. Open the device with an `encryptedvdb1` map name.

- 4.1. Use the `cryptsetup luksOpen` command to name the encrypted partition `encryptedvdb1`.

Chapter 3 | Protecting Data with LUKS and NBDE

```
[root@serverb ~]# cryptsetup luksOpen /dev/vdb1 encryptedvdb1
Enter passphrase for /dev/vdb1: redhatRHT
```

- 4.2. Verify that the partition is now available at the /dev/mapper/encryptedvdb1 device mapping.

```
[root@serverb ~]# ls /dev/mapper/encryptedvdb1
/dev/mapper/encryptedvdb1
```

5. Create an XFS file system on the encrypted partition, and mount this file system on the /encrypted directory. Create a file called testfile in the /encrypted directory by using the touch command.

- 5.1. Create an XFS file system on the /dev/mapper/encryptedvdb1 device mapping.

```
[root@serverb ~]# mkfs.xfs /dev/mapper/encryptedvdb1
meta-data=/dev/mapper/encryptedvdb1 isize=512    agcount=4, agsize=65344 blks
          =                      sectsz=512  attr=2, projid32bit=1
          =                      crc=1    finobt=0, sparse=0
data     =                      bsize=4096   blocks=261376, imaxpct=25
          =                      sunit=0    swidth=0 blks
naming   =version 2           bsize=4096   ascii-ci=0 ftype=1
log      =internal log       bsize=4096   blocks=855, version=2
          =                      sectsz=512  sunit=0 blks, lazy-count=1
realtime =none                extsz=4096   blocks=0, rtextents=0
```

- 5.2. Create the /encrypted directory.

```
[root@serverb ~]# mkdir /encrypted
```

- 5.3. Mount the /dev/mapper/encryptedvdb1 device mapping on the /encrypted directory.

```
[root@serverb ~]# mount -t xfs /dev/mapper/encryptedvdb1 /encrypted
```

- 5.4. Verify that the /dev/vdb1 partition is correctly mounted.

```
[root@serverb ~]# mount | grep /encrypted
/dev/mapper/encryptedvdb1 on /encrypted type xfs
(rw,relatime,seclabel,attr2,inode64,noquota)
```

- 5.5. Create a file called testfile in the /encrypted directory by using the touch command.

```
[root@serverb ~]# touch /encrypted/testfile
```

6. Unmount the file system from the /encrypted mount point and lock the encrypted partition.

- 6.1. Unmount the file system from the /encrypted mount point.

```
[root@serverb ~]# umount /encrypted
```

- 6.2. Lock the encrypted partition by using the cryptsetup luksClose command.

```
[root@serverb ~]# cryptsetup luksClose encryptedvdb1
```

- 6.3. Return to the workstation machine when done.

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

7. Edit and apply the /home/student/RH415/labs/luks-review/nbde_setup.yml Ansible Playbook to associate the LUKS-encrypted partition on the /dev/vdb1 device with Tang servers on the serverc and serverd machines. Use the /home/student/RH415/labs/luks-review/inventory inventory file. Configure SSS encryption so that at least two Tang servers must be available to decrypt the partition.

- 7.1. Use a text editor to edit the /home/student/RH415/labs/luks-review/nbde_setup.yml Ansible Playbook. Add the following content:

```
---  
- hosts: servers  
  become: yes  
  become_method: sudo  
  
  vars:  
    nbde_server_rotate_keys: yes  
    nbde_server_manage_firewall: true  
    nbde_server_manage_selinux: true  
  
  roles:  
    - rhel-system-roles.nbde_server  
- hosts: clients  
  become: yes  
  become_method: sudo  
  
  vars:  
    nbde_client_bindings:  
      - device: /dev/vdb1  
        encryption_password: redhatRHT  
        servers:  
          - http://serverc.lab.example.com  
          - http://serverd.lab.example.com  
        threshold: 2  
  
  roles:  
    - rhel-system-roles.nbde_client
```

- 7.2. Apply the /home/student/RH415/labs/luks-review/nbde_setup.yml Ansible Playbook.

Chapter 3 | Protecting Data with LUKS and NBDE

```
[student@workstation ~]$ ansible-playbook \
-i ~/RH415/labs/luks-review/inventory \
--ask-become-pass ~/RH415/labs/luks-review/nbde_setup.yml
...output omitted...
```

8. On the **serverb** machine configure the encrypted partition to automatically decrypt and mount on the **/encrypted** directory at boot time. Reboot the **serverb** machine.

- 8.1. Log in to the **serverb** machine as the **student** user.

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

- 8.2. Change to the **root** user. Use the **student** sudo password.

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

- 8.3. Edit the **/etc/crypttab** file to open the encrypted partition at boot time. Add the following content:

```
encryptedvdb1      /dev/vdb1  none  _netdev
```

- 8.4. Update the **/etc/fstab** file to mount the encrypted partition on the **/encrypted** directory.

```
...output omitted...
/dev/mapper/encryptedvdb1  /encrypted        xfs    _netdev      1 2
```

- 8.5. Reboot the **serverb** machine by using the **reboot** command.

```
[root@serverb ~]# reboot
Connection to serverb closed.
[student@workstation ~]$
```

9. After the **serverb** machine reboots, verify that the LUKS-encrypted partition on the **/dev/vdb1** device is decrypted and is mounted automatically on the **/encrypted** directory.

- 9.1. Log in to the **serverb** machine as the **student** user. You do not need to enter a password. The **serverb** machine might take a few minutes to boot. You can check the boot progress by clicking **Open Console** on the lab control page. If the **serverb** machine fails to boot, then you might need to rebuild your lab environment.

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

- 9.2. Change to the **root** user. Use the **student** sudo password.

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

9.3. Verify that the encrypted partition is mounted on the /encrypted directory.

```
[root@serverb ~]# mount | grep /encrypted  
/dev/mapper/encryptedvdb1 on /encrypted type xfs  
(rw,relatime,seclabel,attr2,inode64,noquota,_netdev)
```

9.4. Verify that the previously created **testfile** file is still available in the /encrypted directory. When done, return to the **workstation** machine as the **student** user.

```
[root@serverb ~]# ls /encrypted  
testfile  
[root@serverb ~]# logout  
[student@serverb ~]$ logout  
[student@workstation ~]$
```

10. Rotate the keys for the Tang servers by running the /home/student/RH415/labs/luks-review/nbde_setup.yml Ansible Playbook again.

10.1. Apply the /home/student/RH415/labs/luks-review/nbde_setup.yml Ansible Playbook.

```
[student@workstation ~]$ ansible-playbook \  
-i ~/RH415/labs/luks-review/inventory \  
--ask-become-pass ~/RH415/labs/luks-review/nbde_setup.yml  
...output omitted...
```

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 luks-review
```

Finish

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

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

Summary

- Red Hat Enterprise Linux supports block device encryption with Linux Unified Key Setup (LUKS).
- When installing the operating system automatically, Kickstart can create encrypted block devices.
- You can use the `cryptsetup` command to encrypt existing devices after installation.
- A passphrase is required at boot time to decrypt a LUKS-encrypted block device.
- NBDE automates the decryption of LUKS-encrypted disks without manually entering a passphrase at boot time.
- You can automate NBDE configuration with Ansible by using the `rhel-system-roles.nbde_client` and `rhel-system-roles.nbde_server` Ansible Roles.
- NBDE uses the Clevis framework on the client side (decryption), and queries Tang servers to determine whether the client is running on a secure network.
- The Clevis framework provides binding policies that enable the use of multiple Tang servers.
- Red Hat recommends that you periodically rotate the signature and exchange keys for a Tang server.

Chapter 4

Restricting USB Device Access

Goal

Protect systems from rogue USB device access with USBGuard.

Sections

- Controlling USB Access with USBGuard (and Guided Exercise)

Lab

- Restricting USB Device Access

Controlling USB Access with USGuard

Objectives

- Configure and use USGuard to selectively control USB device access.

Introduction to USGuard

USGuard is a software framework that protects your systems against rogue USB devices by implementing basic allowlist and blocklist capabilities based on device attributes. The Linux kernel authorizes USB devices by enforcing the settings that USGuard configures. For example, you can define what types of USB devices are allowed and how a USB device can interact with your system.

The USGuard framework provides the following components:

- A system service component with an inter-process communication (IPC) interface for dynamic interaction and policy enforcement.
- A command-line interface to interact with a running USGuard instance.
- A rule language for writing USB device authorization policies.
- A C++ API for interacting with the system service component that is implemented in a shared library.

Installing USGuard

The `usbguard` package provides the `usbguard` service, which works with the kernel to enforce the USGuard policies, and provides the `usbguard` command to manage the USGuard policies. USGuard provides its own SELinux policy, which is confined under the `usbguard_t` domain.

Run the following command to install the `usbguard` package:

```
[root@host ~]# dnf install usbguard
```

Use the following optional packages for managing USB devices:

usbutils

The `usbutils` package provides the `lsusb` command to display information about the USB buses in the system and connected devices.

udisks2

The `udisks2` package provides interfaces to count and perform operations on disks and storage devices. The `udisks2` package provides the `udisksctl` tool, which interacts with the `udisksd` daemon. The `udisksctl` command `status` option provides high-level information about disk drives and block devices.

Use the following command to install the optional supporting packages:

```
[root@host ~]$ dnf install usbutils udisks2
```

Start the `usbguard` daemon:

```
[root@host ~]# systemctl enable --now usbguard.service
```

Using USGuard

The USGuard daemon determines whether to allow a USB device based on a policy that a set of rules defines. When a USB device is inserted into the system, the daemon scans the existing rules sequentially. When a matching rule is found, the daemon allows, blocks, or rejects the device based on the rule target. If no matching rule is found, then the decision is based on an implicit default target. This implicit default is to block the device until you decide to allow it.

Rule Targets

The rule target determines whether the system allows you to use a USB device that matches that rule. The following rule targets help to manage the authorization of a USB device:

allow

Allow the device. The device and its interfaces are allowed to communicate with the system.

block

Disallow the device. The device is visible to the system but remains in a blocked state until it is allowed.

reject

Disallow and remove the device from the system. The device must be reinserted to become visible to the system again.

Using the USGuard Command-line Interface (CLI)

The `usbguard` command provides a CLI to the USGuard daemon.

List all USB devices that the USGuard daemon recognizes:

```
[root@host ~]# usbguard list-devices
```

Allow a device that the device number identifies to interact with the system:

```
[root@host ~]# usbguard allow-device 1
```

Disallow a device that the device number identifies:

```
[root@host ~]# usbguard block-device 2
```

Disallow and remove a device that the device number identifies:

```
[root@host ~]# usbguard reject-device 2
```

List the rule set (policy) that the `usbguard` daemon uses:

```
[root@host ~]# usbguard list-rules
```

Chapter 4 | Restricting USB Device Access

Generate a rule set (policy) that allows the currently connected USB devices:

```
[root@host ~]# usbguard generate-policy > /etc/usbguard/rules.conf
```

Remove a rule that is identified by the rule number:

```
[root@host ~]# usbguard remove-rule 3
```

Creating an Initial Rule Set

Persistent USBGuard rules are stored in the `/etc/usbguard/rules.conf` file. The initial file that is installed on the system is empty, and the `root` user owns it. Use the `usbguard generate-policy` command to create a persistent initial rule set that allows the currently connected USB devices.

```
[root@host ~]# usbguard generate-policy > /etc/usbguard/rules.conf
```

After generating the policy, you must start or restart the `usbguard` service to apply the changes from the `/etc/usbguard/rules.conf` file:

```
[root@host ~]# systemctl enable --now usbguard
```

Use the following command to inspect the contents of the `rules.conf` file:

```
[root@host ~]# usbguard list-rules
4: allow id 1d6b:0001 serial "0000:00:01.2" name "UHCI Host Controller" hash
"FRDEjz70hdJbNjmJ8zityiNX/Lu0+ovKC07I0bOfjao=" parent-hash "9+Zsfvo9IR/AEQ/
Fn4mzd0PGk0rqpqjku6uErfS09K4c=" with-interface 09:00:00 with-connect-type ""
```

Rule Composition

Using the fourth rule in the list (rule 4) as an example, the components are as follows:

4

This field represents the rule number.

allow

This field designates the rule target (policy) that is applied to the device.

id

The USB device ID is a colon-delimited pair in the `vendor_id:product_id` form. For all USB devices, the manufacturer assigns this ID, and it must uniquely identify a USB product. Both `vendor_id` and `product_id` values are 16-bit numbers that are represented in hexadecimal base. You can use an asterisk character to match either any device ID `* : *` or any product ID from a specific vendor, for example, `1234 : *`.

serial

This field shows the USB serial device attribute.

name

This field shows the USB device name attribute.

Chapter 4 | Restricting USB Device Access**hash**

The hash field contains a hash that is computed from the device attribute values and the USB descriptor data. USBDGuard computes the hash for every device. The hash attribute is the most specific value to identify a device.

parent -hash

The parent -hash field contains a hash value that is computed similarly to the one in the hash field, but for the parent USB device of the USB device. For example, the parent USB device might be a USB hub that the device is plugged into.

with-interface

This field shows the interface that the USB device provides.

Dynamically Allow a Device to Interact with the System

Dynamically allowing a USB device is useful for granting temporary access to a device. If a USB device is attached to the system and the default policy does not allow access to the system, then the device is assigned the **block** rule target. Use the **usbdguard** command to list and modify the authorization of the devices.

```
[root@host ~]# usbdguard list-devices
1: allow id 1d6b:0001 serial "0000:00:01.2" name "UHCI Host Controller" hash
"FRDEjz70hdJbNjmJ8zityiNX/Lu0+ovKC07I0b0Fjao=" parent-hash "9+Zsfvo9IR/AEQ/
Fn4mzdoPGk0rqpjku6uErfS09K4c=" via-port "usb1" with-interface 09:00:00 with-
connect-type ""
2: block id 0627:0001 serial "42" name "QEMU USB Tablet" hash
"V2g2ylTpBl8oDeqJ8qb2rZNeKY0z38+qm2Z4bkSK+20=" parent-hash
"FRDEjz70hdJbNjmJ8zityiNX/Lu0+ovKC07I0b0Fjao=" via-port "1-1" with-interface
03:00:00 with-connect-type "unknown"
```

Use the **usbdguard** command to dynamically allow the USB device with the 2 device number:

```
[root@host ~]# usbdguard allow-device 2
```

List the devices to verify that the device is now allowed to interact with the system:

```
[root@host ~]# usbdguard list-devices
1: allow id 1d6b:0001 serial "0000:00:01.2" name "UHCI Host Controller" hash
"FRDEjz70hdJbNjmJ8zityiNX/Lu0+ovKC07I0b0Fjao=" parent-hash "9+Zsfvo9IR/AEQ/
Fn4mzdoPGk0rqpjku6uErfS09K4c=" via-port "usb1" with-interface 09:00:00 with-
connect-type ""
2: allow id 0627:0001 serial "42" name "QEMU USB Tablet" hash
"V2g2ylTpBl8oDeqJ8qb2rZNeKY0z38+qm2Z4bkSK+20=" parent-hash
"FRDEjz70hdJbNjmJ8zityiNX/Lu0+ovKC07I0b0Fjao=" via-port "1-1" with-interface
03:00:00 with-connect-type "unknown"
```

The **usbdguard** command does not add the rule target in the **/etc/usbdguard/rules.conf** file, and the changes do not persist after reboot.

```
[root@host ~]# usbdguard list-rules
1: allow id 1d6b:0001 serial "0000:00:01.2" name "UHCI Host Controller" hash
"FRDEjz70hdJbNjmJ8zityiNX/Lu0+ovKC07I0b0Fjao=" parent-hash "9+Zsfvo9IR/AEQ/
Fn4mzdoPGk0rqpjku6uErfS09K4c=" with-interface 09:00:00 with-connect-type ""
```

Persistently Allow a Device to Interact with the System

If you dynamically allow a USB device to interact with the system by setting a temporary rule, then you might want that rule to persist after reboots to permanently allow the device. The `usbguard allow-device` command `-p` option allows a USB device that is identified by the device number to persistently interact with the system across reboots. The `-p` option gives persistent authorization by adding the allow rule target to the policy in the `/etc/usbguard/rules.conf` file.

```
[root@host ~]# usbguard list-devices
1: allow id 1d6b:0001 serial "0000:00:01.2" name "UHCI Host Controller" hash
  "FRDEjz70hdJbNjmJ8zityiNX/Lu0+ovKC07I0b0Fjao=" parent-hash "9+Zsfvo9IR/AEQ/
Fn4mzd0PGk0rqpjku6uErfS09K4c=" via-port "usb1" with-interface 09:00:00 with-
connect-type ""
2: allow id 0627:0001 serial "42" name "QEMU USB Tablet" hash
  "V2g2ylTpBl8oDeqJ8qb2rZNeKY0z38+qm2Z4bkSK+20=" parent-hash
  "FRDEjz70hdJbNjmJ8zityiNX/Lu0+ovKC07I0b0Fjao=" via-port "1-1" with-interface
  03:00:00 with-connect-type "unknown"
```

Use the following command to add a persistent allow rule target for the 2 device number:

```
[root@host ~]# usbguard allow-device -p 2
```

After adding a rule to the `/etc/usbguard/rules.conf` file, you must restart the `usbguard` service to apply the changes:

```
[root@host ~]# systemctl restart usbguard.service
```

The `usbguard list-rules` command shows the USB with the `0627:0001` device ID in the output. The output confirms an entry in the `/etc/usbguard/rules.conf` file, which means that the changes will persist across reboots.

```
[root@host ~]# usbguard list-rules
1: allow id 1d6b:0001 serial "0000:00:01.2" name "UHCI Host Controller" hash
  "FRDEjz70hdJbNjmJ8zityiNX/Lu0+ovKC07I0b0Fjao=" parent-hash "9+Zsfvo9IR/AEQ/
Fn4mzd0PGk0rqpjku6uErfS09K4c=" with-interface 09:00:00 with-connect-type ""
2: allow id 0627:0001 serial "42" name "QEMU USB Tablet" hash
  "V2g2ylTpBl8oDeqJ8qb2rZNeKY0z38+qm2Z4bkSK+20=" parent-hash
  "FRDEjz70hdJbNjmJ8zityiNX/Lu0+ovKC07I0b0Fjao=" with-interface 03:00:00 with-
connect-type "unknown"
```

Preventing a Device from Interacting with the System

To prevent devices from interacting with the system, you can block or reject the device. When you block a device, the system can see it but not use it. When you reject a device, the system cannot see or use the device.

Use the `usbguard block-device` command with the device number to set the block rule target:

```
[root@host ~]# usbguard block-device 2
```

Chapter 4 | Restricting USB Device Access

Use the `-p` option to block a device persistently:

```
[root@host ~]# usbguard block-device -p 2
```

Use the `usbguard reject-device` command with the device number to set the reject rule target:

```
[root@host ~]# usbguard reject-device 2
```

Use the following command to persistently reject a device:

```
[root@host ~]# usbguard reject-device -p 2
```

USBGuard Daemon Configuration

The `/etc/usbguard/usbguard.conf` file is the default system configuration file. The USBGuard daemon loads this file after it parses the command-line options to configure the runtime parameters of the USBGuard daemon. You can override the default file with another file by using the `usbguard-daemon` command `-c` option.

Selected USBGuard Configuration Options:

RuleFile=absolute path to file

The `usbguard` daemon loads the policy rule set from this file, and also writes new rules that are received through the IPC interface to this file.

ImplicitPolicyTarget=allow, block, or reject

The target value determines how the USBGuard daemon treats the devices that do not match any rule in the policy. The default value is `block`. The value can be the `allow`, `block`, or `reject` rule target.

PresentDevicePolicy=policy

The value determines the policy for treating devices that are already connected when the daemon starts:

- `allow`: This policy allows every connected device.
- `block`: This policy disallows every connected device.
- `reject`: This policy removes every connected device.
- `keep`: This policy synchronizes the internal state and keeps the device's current state.
- `apply-policy`: This policy evaluates the rule set for every connected device.

PresentControllerPolicy=policy

The value determines the policy for treating USB controllers that are already connected when the daemon starts. This policy also accepts the values that are available in the `PresentDevicePolicy` policy.

IPCAuthenticatedUsers=usernames

You can provide a space-delimited list of user names that the daemon accepts IPC connections for.

IPCAuthenticatedGroups=groupnames

You can provide a space-delimited list of group names that the daemon accepts IPC connections for.

IPCAccessControlFiles=absolute path to directory

You can provide a path to a directory with the IPC access control files.

Allowing Users and Groups to Use the USBDaemon IPC Interface

The usbdam daemon uses the `/etc/usbdam/usbdam-daemon.conf` configuration file to load rules from the `/etc/usbdam/rules.conf` file, and allows only users from the `usbdam` group to use the IPC interface.

The following example allows the `wheel` group to use the IPC interface:

```
IPCAllowedGroups=wheel
```

The `usbdam add-user` and `usbdam remove-user` commands specify the IPC Access Control List (ACL).

In the following example, the `usbdam add-user` command allows users from the `usbdam` group to modify the USB device authorization state, list USB devices, listen to exception events, and list the USB authorization policy:

```
[root@host ~]# usbdam add-user -g usbdam \
> --devices=modify,list,listen --policy=list --exceptions=listen
```

In Red Hat Enterprise Linux, the access to this interface is by default limited to the `root` user. You must set either the `IPCAccessControlFiles` option (recommended) or set the `IPCAllowedUsers` and `IPCAllowedGroups` options to limit access to the IPC interface. Leaving the ACL unconfigured exposes the IPC interface to all local users, which means that a user could manipulate the authorization state of USB devices and modify the USBDaemon policy.

Applying Rules to Specific Devices and Classes of Devices

If the devices can be examined and identified accurately, then you can implement policies to allow or block devices. If accurate identification of the device is not possible, then you can add more flexible rules to the `/etc/usbdam/rules.conf` file based on device characteristics. Blocking a device based on its USB class or characteristics might not be strict enough for some organizations. In this case, you might reject devices based on their USB classes or sets of characteristics.

Creating Policies that Match a Specific Device

The following example allows a specific YubiKey device to be connected via a specific port. The policy is written to reject everything else on that port.

```
allow 1050:0011 name "Yubico Yubikey II" serial "0001234567" via-port "1-2" hash
  "044b5e168d40ee0245478416caf3d998"
reject via-port "1-2"
```

In the previous example, you could use just the hash to match the device. However, by using the name and serial attributes, you can quickly assign rules to specific devices without computing the hash. That said, the hash is the most specific value for identifying a device in USBDaemon, so it is the best attribute to use for a rule to match only one device.

Creating Policies that Match Multiple Devices

You can add a policy in the `/etc/usbdam/rules.conf` file to allow USB mass storage devices (USB flash disks) and to block everything else.

Chapter 4 | Restricting USB Device Access

This policy blocks any device that has capabilities in addition to mass storage, such as a hidden keyboard. Devices with a single mass storage interface are allowed to interact with the operating system.

In the following output, the interface type represents a USB interface, and must be formatted as three 8-bit colon-delimited hexadecimal numbers:

```
{ interface class:subclass:protocol }
```

The policy consists of a single rule:

```
allow with-interface equals { 08:*:* }
```

In this policy, the blocking is implicit because a block rule was not included. Implicit blocking is useful to desktop users because a desktop applet that is listening to USBDaemon events can ask the user for a decision if an implicit target was selected for a device.

Reject Devices with a Suspicious Combination of Interfaces

A USB flash disk that implements a keyboard or a network interface is suspicious. The following set of rules forms a policy that allows USB flash disks and explicitly rejects devices with additional suspicious interfaces.

```
allow with-interface equals { 08:*:* }
reject with-interface all-of { 08:*:* 03:00:*
reject with-interface all-of { 08:*:* 03:01:*
reject with-interface all-of { 08:*:* e0:*:*
reject with-interface all-of { 08:*:* 02:*
```

The policy rejects all USB flash disk devices with an interface from the Human Interface Devices (HID), communications, and wireless classes. Red Hat recommends blocking devices by default and explicitly allowing devices that are needed. This policy assumes that blocking is the default approach. Red Hat also recommends rejecting sets of devices that are considered to be bad, such as flash devices that include a suspicious keyboard interface.



References

`usbdemux(1)`, `usbdemux-daemon(8)`, `usbdemux-daemon.conf(5)`, and `usbdemux-rules.conf(5)` man pages.

For more information, refer to the *Protecting Systems Against Intrusive USB Devices* chapter in the *Red Hat Enterprise Linux 9 Security Hardening* guide at https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html/security_hardening/protecting-systems-against-intrusive-usb-devices_security-hardening

► Guided Exercise

Controlling USB access with USBDGuard

Install USBDGuard, configure dynamic policy, and use USBDGuard to block, reject, or permit access by certain USB devices or types of USB devices.

Outcomes

- Generate policies to control USB device authorization.
- Create USBDGuard **block** and **reject** policies.
- Create dynamic and persistent USBDGuard rule sets.
- Use command-line tools to list the access statuses of USB devices.

Before You Begin

As the **student** user on the **workstation** machine, use the **lab** command to prepare your environment for this exercise, and to ensure that all required resources are available.

```
[student@workstation ~]$ lab start usb-usb
```

Instructions

- 1. As the **root** user on the **workstation** machine, verify the run state of the **usbdguard** virtual machine (VM). Start the **usbdguard** VM if it is not running.
- 1.1. On the **workstation** machine, log in as the **student** user and switch to the **root** user:

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

- 1.2. Verify the state of the **usbdguard** VM:

```
[root@workstation ~]# virsh domstate usbdguard  
shut off
```

- 1.3. Start the **usbdguard** VM if necessary. Allow the **usbdguard** VM about two minutes to complete the startup process.

```
[root@workstation ~]# virsh start usbdguard  
Domain 'usbdguard' started
```

- 2. Log in to the console as the **student** user by using **student** as the password, and switch to the **root** user. If the console delays in displaying the login prompt, then press **Enter** to proceed to the prompt.

Chapter 4 | Restricting USB Device Access

- 2.1. Open the console of the `usbguard` VM, and log in to the console as the `student` user by using `student` as the password:

```
[root@workstation ~]# virsh console usbguard
...output omitted...
Connected to domain 'usbguard'
Escape character is ^] (Ctrl + ])
<Enter>

Red Hat Enterprise Linux 9.2 (Plow)
Kernel 5.14.0-284.11.1.el9_2.x86_64 on an x86_64

Activate the web console with: systemctl enable --now cockpit.socket

localhost login: student
Password: student
```

- 2.2. Switch to the `root` user:

```
[student@localhost ~]$ sudo -i
[root@localhost ~]#
```

- 3. As the `root` user on the `usbguard` VM, install the RPM packages to configure, control, and manage USB devices.

- 3.1. Install the `usbguard`, `usbutils`, and `udisks2` packages:

```
[root@localhost ~]# dnf install usbguard usbutils udisks2
...output omitted...
Is this ok [y/N]: y
...output omitted...
Complete!
```

- 4. Generate a rule set (policy) that authorizes the currently connected USB devices. Copy the output into the `/etc/usbguard/rules.conf` file. Use the `-X` option to suppress the generation of hash attributes for each device.

```
[root@localhost ~]# usbguard generate-policy -X | \
tee /etc/usbguard/rules.conf
allow id 1d6b:0002 serial "0000:00:04.7" name "EHCI Host Controller" with-
interface 09:00:00 with-connect-type ""
allow id 1d6b:0001 serial "0000:00:04.0" name "UHCI Host Controller" with-
interface 09:00:00 with-connect-type ""
allow id 1d6b:0001 serial "0000:00:04.1" name "UHCI Host Controller" with-
interface 09:00:00 with-connect-type ""
allow id 1d6b:0001 serial "0000:00:04.2" name "UHCI Host Controller" with-
interface 09:00:00 with-connect-type ""
```

- 5. Start the `usbguard` service to persist across reboots. Use the `usbguard` command to verify the USBGuard rules.

- 5.1. Start the `usbguard` service to persist across reboots:

```
[root@localhost ~]# systemctl enable --now usbguard
Created symlink /etc/systemd/system/basic.target.wants/usbguard.service → /usr/
lib/systemd/system/usbguard.service.
```

- 5.2. List the rule set (policy) that the `usbguard` daemon uses:

```
[root@localhost ~]# usbguard list-rules
1: allow id 1d6b:0002 serial "0000:00:04.7" name "EHCI Host Controller" with-
interface 09:00:00 with-connect-type ""
2: allow id 1d6b:0001 serial "0000:00:04.0" name "UHCI Host Controller" with-
interface 09:00:00 with-connect-type ""
3: allow id 1d6b:0001 serial "0000:00:04.1" name "UHCI Host Controller" with-
interface 09:00:00 with-connect-type ""
4: allow id 1d6b:0001 serial "0000:00:04.2" name "UHCI Host Controller" with-
interface 09:00:00 with-connect-type ""
```

- 6. List all of the USB devices that the `USBGuard` daemon recognizes. This option lists each device's hash attribute, which is the most specific value to identify a device.

```
[root@localhost ~]# usbguard list-devices
5: allow id 1d6b:0002 serial "0000:00:04.7" name "EHCI Host Controller"
hash "CsKOZ6IY8v3eojsc1fqKDW84V+MMhD6HsjojcZBjSg=" parent-hash
"MhPzffrQEhx5CwP3GXco7JXDbaMzFbd5FPUFFE7nfu0=" via-port "usb1" with-interface
09:00:00 with-connect-type ""
6: allow id 1d6b:0001 serial "0000:00:04.0" name "UHCI Host Controller" hash
"sKXn6PthDDlGgdxZHdnluQ9DR0kH/YSojkBlfpncnsaU=" parent-hash "9Ii0Zm8Mvu2nYz9z/
EgAXJ/ed6bLW8Ctv1iUD5rh6qY=" via-port "usb2" with-interface 09:00:00 with-connect-
type ""
7: allow id 1d6b:0001 serial "0000:00:04.1" name "UHCI Host Controller" hash
"6t6CPSS/v2EqQsw6CMq8DVf0hgUG02f+bEBX7R2yz0=" parent-hash "t7Z0XTvKnMmdqAm1vU
+noU318kZsRQQV+JorpRTThQ7c=" via-port "usb3" with-interface 09:00:00 with-connect-
type ""
8: allow id 1d6b:0001 serial "0000:00:04.2" name "UHCI Host Controller"
hash "BSaNQWAoDaBI31jUqbck0N56uRuh3uVT1VkrdoD0ghs=" parent-hash "UyZQuRI
+gcw41fsM6Kgyty6pgYN0zfyYjqSpJv7na1E=" via-port "usb4" with-interface 09:00:00
with-connect-type ""
```

- 7. Open another terminal window on the `workstation` machine, and attach the `usb-disk-red.img` disk image to the `usbguard` VM:

```
[student@workstation ~]$ sudo virsh attach-device usbguard \
/home/student/RH415/labs/usb-usb/usb-disk-red.xml
[sudo] password for student: student
Device attached successfully
```

- 8. Return to the terminal window of the `usbguard` VM, and see the kernel messages. Notice that the RED USB device is not authorized for usage. Press `Enter` to return to the command prompt.

```
[ 1554.522039] usb 1-1: new high-speed USB device number 2 using ehci-pci
[ 1554.662285] usb 1-1: New USB device found, idVendor=46f4, idProduct=0001,
bcdDevice= 0.00
[ 1554.669411] usb 1-1: New USB device strings: Mfr=1, Product=2, SerialNumber=3
[ 1554.674855] usb 1-1: Product: QEMU USB HARDDRIVE
[ 1554.677274] usb 1-1: Manufacturer: QEMU
[ 1554.679123] usb 1-1: SerialNumber: RED
[ 1554.689326] usb 1-1: Device is not authorized for usage
<Enter>
```

- ▶ 9. Debug the newly attached device and display information about the devices.

- 9.1. Verify that the new device is attached to the system.

```
[root@localhost ~]# lsusb
Bus 001 Device 002: ID 46f4:0001 QEMU QEMU USB HARDDRIVE
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 004 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 003 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 002 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
```

- 9.2. List the available block devices to confirm that the newly added device is absent.

```
[root@localhost ~]# lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
vda    252:0    0   10G  0 disk
└─vda1 252:1    0    1M  0 part
└─vda2 252:2    0  200M  0 part /boot/efi
└─vda3 252:3    0  500M  0 part /boot
└─vda4 252:4    0  9.3G  0 part /
```

- 9.3. List the high-level information for the disk drives and block devices. This output confirms that the newly attached device is not available for mounting.

```
[root@localhost ~]# udisksctl status
MODEL           REVISION  SERIAL          DEVICE
-----
VirtIO Disk                  vda
```

- 9.4. List the blocked USB devices that the USBDaemon recognizes. The device numbers might differ in your classroom.

The output displays the device number 9, the 46f4:0001 device ID, and the RED serial name with the block target policy.

```
[root@localhost ~]# usbdemux list-devices --blocked
9: block id 46f4:0001 serial "RED" name "QEMU USB HARDDRIVE"
hash "AKmuakTNktSFF54t2IHFRMaukoUw47v3lu/9ZebOsNo=" parent-hash
"CsKOZ6IY8v3eojsc1fqKDW84V+MMhd6HsijojcZBjSg=" via-port "1-1" with-interface
08:06:50 with-connect-type ""
```

Chapter 4 | Restricting USB Device Access

- 10. Create a persistent rule that allows the new device to access the system and to be available for mounting. The device numbers might differ in your classroom.

10.1. Allow the device with a persistent rule to access the system:

```
[root@localhost ~]# usbguard allow-device -p 9
[54509.376970] usb 1-1: authorized to connect
[54509.406321] usb-storage 1-1:1.0: USB Mass Storage device detected
[54509.413177] scsi host6: usb-storage 1-1:1.0
[54509.415700] usbcore: registered new interface driver usb-storage
[54509.424303] usbcore: registered new interface driver uas
[54510.476342] scsi 6:0:0:0: Direct-Access      QEMU      QEMU HARDDISK  2.5+ PQ:
0 ANSI: 5
[54510.498202] scsi 6:0:0:0: Attached scsi generic sg0 type 0
[54510.517945] sd 6:0:0:0: Power-on or device reset occurred
[54510.524460] sd 6:0:0:0: [sda] 65536 512-byte logical blocks: (33.6 MB/32.0 MiB)
[54510.536556] sd 6:0:0:0: [sda] Write Protect is off
[54510.546850] sd 6:0:0:0: [sda] Write cache: enabled, read cache: enabled,
doesn't support DPO or FUA
[54510.586024] sd 6:0:0:0: [sda] Attached SCSI disk
<Enter>
```

10.2. Reboot the usbguard VM to confirm that the allow rule persists across reboots. Log in to the console as the student user by using student as the password.

```
[root@localhost ~]# reboot
...output omitted...
Red Hat Enterprise Linux 9.2 (Plow)
Kernel 5.14.0-284.11.1.el9_2.x86_64 on an x86_64

Activate the web console with: systemctl enable --now cockpit.socket

localhost login: student
Password: student
```

10.3. Switch to the root user:

```
[student@localhost ~]$ sudo -i
[root@localhost ~]#
```

10.4. List the device to confirm that the entry for the device with the RED serial name has the allow rule target. The device numbers might differ in your classroom.

```
[root@localhost ~]# usbguard list-devices | grep RED
9: allow id 46f4:0001 serial "RED" name "QEMU USB HARDDRIVE"
hash "AKmuakTNktSFF54t2IHFRMaukoUw47v3lu/9ZebOsNo=" parent-hash
"CsK0Z6IY8v3eojsc1fqKDw84V+MMhD6HsjojcZBjSg=" via-port "1-1" with-interface
08:06:50 with-connect-type "unknown"
```

10.5. List the rule set (policy) that the usbguard daemon uses. Notice the persistent rule for device ID 46f4:0001. The device numbers might differ in your classroom.

```
[root@localhost ~]# usbguard list-rules | grep RED
10: allow id 46f4:0001 serial "RED" name "QEMU USB HARDDRIVE"
hash "AKmuakTNktSFF54t2IHFRMaukoUw47v3lu/9ZebOsNo=" parent-hash
"CsKOZ6IY8v3eojsc1fqKDw84V+MMhD6HsjojcZBjSg=" with-interface 08:06:50 with-
connect-type "unknown"
```

► 11. Debug the attached device and display the information for the device.

- 11.1. List the available block devices to confirm that the new device is present:

```
[root@localhost ~]# lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
sda      8:0    0   32M  0 disk
vda     252:0    0   10G  0 disk
└─vda1  252:1    0    1M  0 part
└─vda2  252:2    0  200M  0 part /boot/efi
└─vda3  252:3    0  500M  0 part /boot
└─vda4  252:4    0  9.3G  0 part /
```

- 11.2. List the high-level information about disk drives and block devices. Confirm that the attached device is available to be mounted.

```
[root@localhost ~]# udisksctl status
MODEL           REVISION  SERIAL          DEVICE
-----
VirtIO Disk
QEMU QEMU HARDDISK       2.5+    RED        vda
                                         sda
```

► 12. Return to the workstation machine as the student user.

- 12.1. Log out of the usbguard VM terminal session:

```
[root@localhost ~]# logout
[student@localhost ~]$ logout
```

- 12.2. Exit the usbguard VM console:

```
Red Hat Enterprise Linux 9.2 (Plow)
Kernel 5.14.0-284.11.1.el9_2.x86_64 on an x86_64

Activate the web console with: systemctl enable --now cockpit.socket

localhost login:
Ctrl+]
```

- 12.3. Return to the workstation machine as the student user:

```
[root@workstation ~]# logout
[student@workstation ~]$
```

Finish

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

```
[student@workstation ~]$ lab finish usb-usb
```

▶ Lab

Restricting USB Device Access

Selectively control which USB devices may access or be accessed by the system by using USBGuard.

Outcomes

- Create a permanent USBGuard rule that allows a specific USB device to interact with the system.
- Generate a base policy that maintains the current rules that are defined, and that rejects any additional USB devices that attempt to connect to the system.
- Use `usbguard`, `lsblk`, and other command-line tools to confirm USB device access policies.

Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

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

Instructions

1. As the `root` user on the `workstation` machine, ensure that the `usbguard` virtual machine (VM) is running.
2. On the `usbguard` VM, install the RPM packages to configure, control, and manage USB device access.
3. Start the USBGuard service and configure it to persist across reboots. List the default devices.
4. Set a permanent USBGuard rule to allow the RED USB device access to the system.
5. Return to the `workstation` machine, and attach the BLUE USB device to the `usbguard` VM. Confirm that the BLUE USB device is blocked from interacting with the `usbguard` VM, and then detach the USB device from the `usbguard` VM.
6. Generate a new base policy with a `reject` rule target that ignores any additional USB devices that try to interact with the system. Verify that the `reject` rule is added.
7. Attach the BLUE USB device to the `usbguard` VM. Debug the attached device and display the information for all devices.
8. Return to the `workstation` machine as the `student` user.

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 usb-review
```

Finish

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

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

► Solution

Restricting USB Device Access

Selectively control which USB devices may access or be accessed by the system by using USBDGuard.

Outcomes

- Create a permanent USBDGuard rule that allows a specific USB device to interact with the system.
- Generate a base policy that maintains the current rules that are defined, and that rejects any additional USB devices that attempt to connect to the system.
- Use `usbdguard`, `lsblk`, and other command-line tools to confirm USB device access policies.

Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

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

Instructions

1. As the `root` user on the `workstation` machine, ensure that the `usbdguard` virtual machine (VM) is running.
 - 1.1. On the `workstation` machine, log in as the `student` user and switch to the `root` user.

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

- 1.2. Verify the state of the `usbdguard` VM.

```
[root@workstation ~]# virsh list --all  
 Id   Name     State  
-----  
 -    usbdguard  shut off
```

- 1.3. Start the `usbdguard` VM if it is not running. Allow the `usbdguard` VM about two minutes to complete the startup process.

```
[root@workstation ~]# virsh start usbdguard  
Domain 'usbdguard' started
```

Chapter 4 | Restricting USB Device Access

2. On the **usbguard** VM, install the RPM packages to configure, control, and manage USB device access.
 - 2.1. Open the console of the **usbguard** VM, log in to the console as the **student** user by using **student** as the password, and switch to the **root** user. If the console delays in displaying the login prompt, then press **Enter** to proceed to the prompt.

```
[root@workstation ~]# virsh console usbguard
Connected to domain 'usbguard'
Escape character is ^] (Ctrl + ])
Enter
localhost login: student
Password: student
[student@localhost ~]$
```

- 2.2. Switch to the **root** user.

```
[student@localhost ~]$ sudo -i
[root@localhost ~]#
```

- 2.3. Install the **usbguard**, **usbutils**, and **udisks2** packages.

```
[root@localhost ~]# dnf install usbguard usbutils udisks2
...output omitted...
Is this ok [y/N]: y
...output omitted...
Complete!
```

3. Start the USBGuard service and configure it to persist across reboots. List the default devices.

- 3.1. Configure the **usbguard** service to persist across reboots.

```
[root@localhost ~]# systemctl enable --now usbguard
Created symlink /etc/systemd/system/basic.target.wants/usbguard.service → /usr/
lib/systemd/system/usbguard.service.
```

- 3.2. List all of the USB devices that USBGuard recognizes.

The device numbers might differ in your classroom.

```
[root@localhost ~]# usbguard list-devices
1: allow id 1d6b:0002 serial "0000:00:04.7" name "EHCI Host Controller"
hash "CsKOZ6IY8v3eojsc1fqKDw84V+MMhD6HsjjjojcZBjSg=" parent-hash
"MhPzffrQEhx5CWP3GXco7JXDbaMzFbD5FPUFFE7nfu0=" via-port "usb1" with-interface
09:00:00 with-connect-type ""
2: allow id 1d6b:0001 serial "0000:00:04.0" name "UHCI Host Controller" hash
"sKXn6PthDDlGgdxZHdnLUQ9DR0kH/YSojkBlfpncsaU=" parent-hash "9Ii0Zm8Mvu2nYz9z/
EgAXJ/ed6bLW8Ctv1iUD5rh6qY=" via-port "usb2" with-interface 09:00:00 with-connect-
type ""
3: allow id 1d6b:0001 serial "0000:00:04.1" name "UHCI Host Controller" hash
"6t6CPSS/v2EqQwsW6CMq8DVf0hgUG02f+bEBX7R2yz0=" parent-hash "tZ0XTvKnMmdqAm1vU
+nOU318kZsRQQV+JorpRThQ7c=" via-port "usb3" with-interface 09:00:00 with-connect-
type ""
4: allow id 1d6b:0001 serial "0000:00:04.2" name "UHCI Host Controller"
hash "BSaNQWADaBI31juqbck0N56uRuh3uVT1Vk4rd0D0ghs=" parent-hash "UyZQuRI
+gcw41fsM6Kgyty6pgYN0zfyYjqSpJv7na1E=" via-port "usb4" with-interface 09:00:00
with-connect-type ""
```

4. Set a permanent USBGuard rule to allow the RED USB device access to the system.

- 4.1. Open another console on the workstation machine and attach the `usb-disk-red.img` disk image to the `usbguard` VM.

```
[student@workstation ~]$ sudo virsh attach-device usbguard \
~/RH415/labs/usb-usb/usb-disk-red.xml
[sudo] password for student: student
Device attached successfully
```

Return to the console of the `usbguard` VM, and notice the kernel messages that indicate that the RED USB device is not authorized for usage. Press `Enter` to return to the command prompt.

```
[ 511.777583] usb 1-1: new high-speed USB device number 2 using ehci-pci
[ 511.911206] usb 1-1: New USB device found, idVendor=46f4, idProduct=0001,
bcdDevice= 0.00
[ 511.915994] usb 1-1: New USB device strings: Mfr=1, Product=2, SerialNumber=3
[ 511.920071] usb 1-1: Product: QEMU USB HARDDRIVE
[ 511.922966] usb 1-1: Manufacturer: QEMU
[ 511.925236] usb 1-1: SerialNumber: RED
[ 511.934510] usb 1-1: Device is not authorized for usage
Enter
```

- 4.2. On the `usbguard` VM, list the blocked USB devices and record the device number for the RED USB device.

The device numbers might differ in your classroom.

```
[root@localhost ~]# usbguard list-devices --blocked
5: block id 46f4:0001 serial "RED" name "QEMU USB HARDDRIVE"
hash "AKmuakTNktSFF54t2IHFRMaukoUw47v3lu/9Zeb0sNo=" parent-hash
"CsKOZ6IY8v3eojsc1fqKDw84V+MMhD6HsjjjojcZBjSg=" via-port "1-1" with-interface
08:06:50 with-connect-type ""
```

Chapter 4 | Restricting USB Device Access

- 4.3. Allow the device to access the system persistently.

```
[root@localhost ~]# usbguard allow-device -p 5  
[ 5014.063516] usb 1-1: authorized to connect  
...output omitted...  
Enter
```

- 4.4. List the persistent rules and ensure that the RED USB device is added.

```
[root@localhost ~]# usbguard list-rules  
6: allow id 46f4:0001 serial "RED" name "QEMU USB HARDDRIVE"  
hash "AKmuakTNktSff54t2IHFRMaukoUw47v3lu/9ZebOsNo=" parent-hash  
"CsK0Z6IY8v3eojsc1fqKDw84V+MMhD6HsjojcZBjSg=" with-interface 08:06:50 with-  
connect-type ""
```

**Note**

If the device is not listed, then restart the `usbguard` services to ensure that the USBGuard daemon loads the `/etc/usbguard/rules.conf` file.

```
[root@localhost ~]# systemctl restart usbguard
```

- 4.5. List the devices to ensure that the RED USB device has a rule target of `allow`.

```
[root@localhost ~]# usbguard list-devices | grep RED  
5: allow id 46f4:0001 serial "RED" name "QEMU USB HARDDRIVE"  
hash "AKmuakTNktSff54t2IHFRMaukoUw47v3lu/9ZebOsNo=" parent-hash  
"CsK0Z6IY8v3eojsc1fqKDw84V+MMhD6HsjojcZBjSg=" via-port "1-1" with-interface  
08:06:50 with-connect-type ""
```

5. Return to the `workstation` machine, and attach the BLUE USB device to the `usbguard` VM. Confirm that the BLUE USB device is blocked from interacting with the `usbguard` VM, and then detach the USB device from the `usbguard` VM.

- 5.1. Attach the `usb-disk-blue.img` disk image to the `usbguard` VM.

```
[student@workstation ~]$ sudo virsh attach-device usbguard \  
~/RH415/labs/usb-usb/usb-disk-blue.xml  
[sudo] password for student: student  
Device attached successfully
```

Return to the console of the `usbguard` VM, and notice the `kernel` messages indicate that the BLUE USB device is not authorized for usage. Press `Enter` to return to the command prompt.

Chapter 4 | Restricting USB Device Access

```
[ 7200.497544] usb 1-2: new high-speed USB device number 3 using ehci-pci
[ 7200.633369] usb 1-2: New USB device found, idVendor=46f4, idProduct=0001,
bcdDevice= 0.00
[ 7200.638668] usb 1-2: New USB device strings: Mfr=1, Product=2, SerialNumber=3
[ 7200.640789] usb 1-2: Product: QEMU USB HARDDRIVE
[ 7200.642356] usb 1-2: Manufacturer: QEMU
[ 7200.643703] usb 1-2: SerialNumber: BLUE
[ 7200.648247] usb 1-2: Device is not authorized for usage
Enter
```

- 5.2. On the **usbguard** VM, list the USB devices to confirm that the system allows the **RED** USB device and blocks the **BLUE** USB device because of the current rules and their rule targets.

The device numbers might differ in your classroom.

```
[root@localhost ~]# usbguard list-devices | grep "RED\|BLUE"
5: allow id 46f4:0001 serial "RED" name "QEMU USB HARDDRIVE"
hash "AKmuakTNktSFF54t2IHFRMaukoUw47v3lu/9ZebOsNo=" parent-hash
"CsKOZ6IY8v3eojsc1fqKDW84V+MMhD6HsjojcZBjSg=" via-port "1-1" with-interface
08:06:50 with-connect-type ""
7: block id 46f4:0001 serial "BLUE" name "QEMU USB HARDDRIVE"
hash "GT0vx1ANTDVd0aekgV1a9GmXHc2Mwrx4o3w6gXae5Lo=" parent-hash
"CsKOZ6IY8v3eojsc1fqKDW84V+MMhD6HsjojcZBjSg=" via-port "1-2" with-interface
08:06:50 with-connect-type ""
```

- 5.3. Return to the **workstation** machine and detach the **BLUE** USB device from the **usbguard** VM.

```
[student@workstation ~]$ sudo virsh detach-device usbguard \
~/RH415/labs/usb-usb/usb-disk-blue.xml
[sudo] password for student: student
Device detached successfully
```

6. Generate a new base policy with a **reject** rule target that ignores any additional USB devices that try to interact with the system. Verify that the **reject** rule is added.

- 6.1. Generate a new base policy with a **reject** rule target.

```
[root@localhost ~]# usbguard generate-policy -x \
-t reject > /etc/usbguard/rules.conf
```

- 6.2. Restart the **usbguard** service.

```
[root@localhost ~]# systemctl restart usbguard.service
```

- 6.3. Confirm the **allow** rule target for the **RED** USB device followed by a catch-all **reject** rule target.

```
[root@localhost ~]# usbguard list-rules
1: allow id 1d6b:0002 serial "0000:00:04.7" name "EHCI Host Controller" with-
interface 09:00:00 with-connect-type ""
2: allow id 1d6b:0001 serial "0000:00:04.0" name "UHCI Host Controller" with-
interface 09:00:00 with-connect-type ""
3: allow id 1d6b:0001 serial "0000:00:04.1" name "UHCI Host Controller" with-
interface 09:00:00 with-connect-type ""
4: allow id 1d6b:0001 serial "0000:00:04.2" name "UHCI Host Controller" with-
interface 09:00:00 with-connect-type ""
5: allow id 46f4:0001 serial "RED" name "QEMU USB HARDDRIVE" with-interface
08:06:50 with-connect-type "unknown"
6: reject
```

7. Attach the BLUE USB device to the usbguard VM. Debug the attached device and display the information for all devices.
 - 7.1. Return to the workstation machine to attach the BLUE USB device to the usbguard VM.

```
[student@workstation ~]$ sudo virsh attach-device usbguard \
~/RH415/labs/usb-usb/usb-disk-blue.xml
[sudo] password for student: student
Device attached successfully
```

The command output on the usbguard VM indicates that the BLUE USB device is successfully attached. The output further confirms that the attempt to attach a USB device is not authorized. The system shows a blocked USB device, which is not allowed to mount. The system ignores a rejected USB device, and therefore, the device is not displayed in command-line tool listings.

- 7.2. Query the `systemd` journal records, and notice the kernel action as well as the USBGuard action. Press `q` to return to the command prompt.

```
[root@localhost ~]# journalctl -b -e
...output omitted...
Jul 14 15:38:00 localhost.localdomain kernel: usb 1-2: new high-speed USB devic>
Jul 14 15:38:00 localhost.localdomain kernel: usb 1-2: New USB device found, id>
Jul 14 15:38:00 localhost.localdomain kernel: usb 1-2: New USB device strings: >
Jul 14 15:38:00 localhost.localdomain kernel: usb 1-2: Product: QEMU USB HARDDR>
Jul 14 15:38:00 localhost.localdomain kernel: usb 1-2: Manufacturer: QEMU
Jul 14 15:38:00 localhost.localdomain kernel: usb 1-2: SerialNumber: BLUE
Jul 14 15:38:00 localhost.localdomain usbguard-daemon[11644]: uid=0 pid=11642 r>
Jul 14 15:38:00 localhost.localdomain usbguard-daemon[11644]: uid=0 pid=11642 r>
Jul 14 15:38:00 localhost.localdomain kernel: usb 1-2: Device is not authorized>
Jul 14 15:38:00 localhost.localdomain kernel: usb 1-2: USB disconnect, device n>
Jul 14 15:38:00 localhost.localdomain usbguard-daemon[11644]: uid=0 pid=11642 r>
...output omitted...
q
```

- 7.3. List all USB devices that the USBGuard daemon recognizes, and confirm that the RED USB device is listed, but that the BLUE USB device is ignored and is not listed.
- The device numbers might differ in your classroom.

Chapter 4 | Restricting USB Device Access

```
[root@localhost ~]# usbguard list-devices
7: allow id 1d6b:0002 serial "0000:00:04.7" name "EHCI Host Controller"
hash "CsKOZ6IY8v3eojsc1fqKDw84V+MMhD6HsjjojcZBjSg=" parent-hash
"MhPzffrQEhx5CWP3Gxco7JXDbaMzFbD5FPuFFE7nfu0=" via-port "usb1" with-interface
09:00:00 with-connect-type ""
8: allow id 1d6b:0001 serial "0000:00:04.0" name "UHCI Host Controller" hash
"sKXn6PthDDlGgdxZHdnluQ9DR0kH/YSojkBlfpncsaU=" parent-hash "9Ii0Zm8Mvu2nYz9z/
EgAXJ/ed6bLW8Ctv1iUD5rh6qY=" via-port "usb2" with-interface 09:00:00 with-connect-
type ""
9: allow id 1d6b:0001 serial "0000:00:04.1" name "UHCI Host Controller" hash
"6t6CPSS/v2EqQsw6CMq8DVf0hgUG02f+bEBX7R2yz0=" parent-hash "t7Z0XTvKnMmdqAm1vU
+nOU318kZsRQQV+JorpRThQ7c=" via-port "usb3" with-interface 09:00:00 with-connect-
type ""
10: allow id 1d6b:0001 serial "0000:00:04.2" name "UHCI Host Controller"
hash "BSaNQWADaBI31juqbck0N56uRuh3uVT1Vk4rd0D0ghs=" parent-hash "UyZQuRI
+gcw41fsM6Kgyty6pgYN0zfyYjqSpJv7na1E=" via-port "usb4" with-interface 09:00:00
with-connect-type ""
11: allow id 46f4:0001 serial "RED" name "QEMU USB HARDDRIVE"
hash "AKmuakTNktSFF54t2IHFRMaukoUw47v3lu/9Zeb0sNo=" parent-hash
"CsKOZ6IY8v3eojsc1fqKDw84V+MMhD6HsjjojcZBjSg=" via-port "1-1" with-interface
08:06:50 with-connect-type "unknown"
```

8. Return to the workstation machine as the student user.

8.1. Log out of the usbguard VM console session.

```
[root@localhost ~]# logout
[student@localhost ~]$ logout
```

8.2. Exit the usbguard VM console.

```
Red Hat Enterprise Linux 9.2 (Plow)
Kernel 5.14.0-284.11.1.el9_2.x86_64 on an x86_64

Activate the web console with: systemctl enable --now cockpit.socket

localhost login:
Ctrl+]
```

8.3. Return to the workstation machine as the student user.

```
[root@workstation ~]# logout
[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 usb-review
```

Finish

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

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

Summary

- The USBDaemon daemon protects your systems against suspicious USB devices by implementing allowlist and blocklist capabilities based on device attributes.
- The USBDaemon daemon allows a USB device based on the policy that a set of rules defines.
- The `usbguard` command manages the USB device authorization rules.
- The USBDaemon daemon uses the `/etc/usbguard/usbguard-daemon.conf` configuration file to load rules that allow users and groups to use the IPC interface.
- The USBDaemon daemon authorizes specific devices, or classes of devices.
- You can create policies that match multiple devices and that reject devices with a suspicious combination of interfaces.

Chapter 5

Controlling Authentication with PAM

Goal

Manage authentication, authorization, session settings, and password controls by configuring Pluggable Authentication Modules (PAM).

Sections

- Selecting the PAM Configuration (and Guided Exercise)
- Configuring Password Quality Requirements (and Guided Exercise)
- Limiting Access after Failed Logins (and Guided Exercise)
- Controlling Authentication with PAM

Lab

Selecting the PAM Configuration

Objectives

- Interpret PAM configuration files and select appropriate configuration profiles with the authselect utility.

Overview of PAM

The *Pluggable Authentication Modules* (PAM) system provides a generic way for applications to implement support for authentication and authorization.

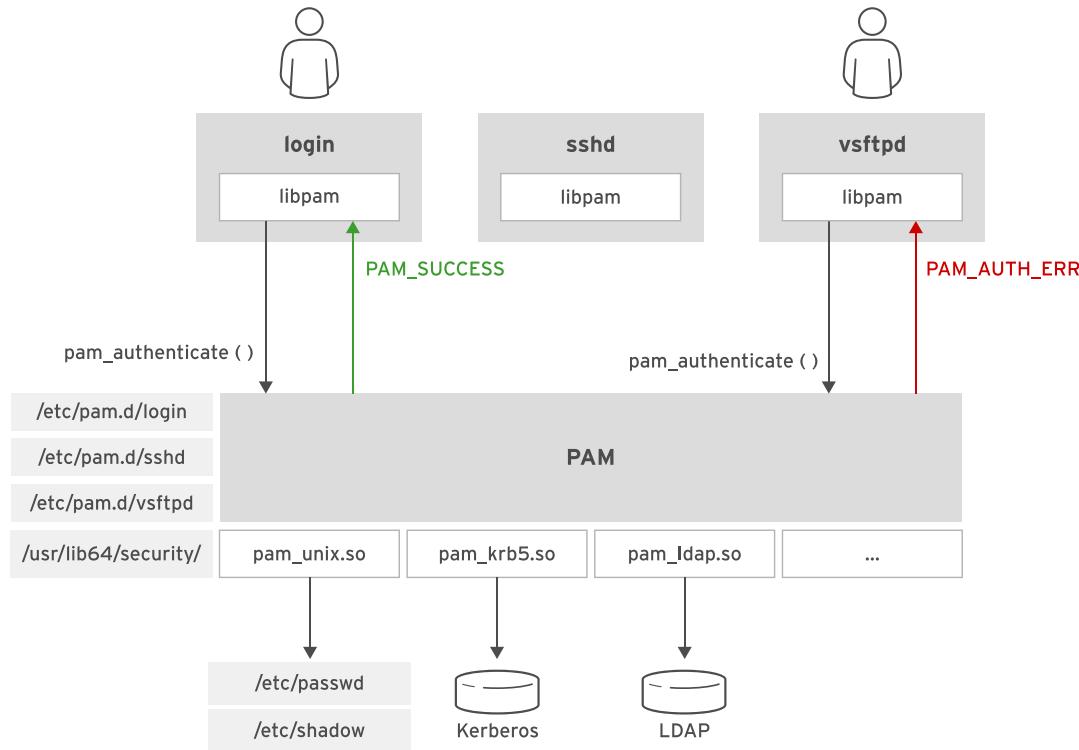
Initially, an application that needed to authenticate users used the local `/etc/passwd` and `/etc/shadow` files. Eventually, other authentication mechanisms were invented. For example, your company might decide to implement Kerberos, in which case the application must verify the correctness of the passwords against the Kerberos Key Distribution Center (KDC). Alternatively, you might choose to deploy a Lightweight Directory Access Protocol (LDAP) infrastructure, and so the application must contact the LDAP server for authentication.

As a result, every time that a new authentication method emerged, applications had to be rewritten to support it. Each application was handled separately, which led to a duplication of effort, and led to many implementations that had to be audited for correctness.

A PAM-enabled application calls the PAM library, `libpam`, to perform all authentication tasks on its behalf and to return a pass or fail response to the application.

The PAM modules implement these different authentication methods. For example, the `pam_krb5` module performs authentication against Kerberos. The `pam_ldap` module authenticates against an LDAP server. The `pam_unix` module uses the standard local files, `/etc/passwd` and `/etc/shadow`. The PAM-enabled applications can directly use these authentication methods without having to be modified or recompiled.

The following figure provides an overview of how PAM-enabled applications authenticate users.

**Figure 5.1: PAM authentication**

In this figure, the `login` program contacts PAM for authentication. PAM reads the `/etc/pam.d/login` configuration file to retrieve the list of modules to use for authentication. PAM calls the modules, in the `/usr/lib64/security/` directory, and these modules perform the authentication. PAM returns a successful authentication status to the `login` program. The `vsftpd` FTP server also uses PAM to authenticate FTP users, but in the figure, the user provides incorrect credentials, and PAM returns a fail code to the `vsftpd` FTP server.

Authentication and Authorization

Determining whether a client is permitted to access a resource is a two-step process. First, the application must authenticate the client. That is, the application must prove that the client is who they claim to be. Then, the application must determine whether that client is authorized to access the resource.

For example, a client that types the correct username and password at a login prompt has authenticated that they are that user. However, access control restrictions might not authorize that user to log in at the current time, so the client might be returned to the login prompt.

PAM manages both authentication and authorization.

Configuring PAM

PAM configuration files determine the modules that are used for each application. Each PAM-enabled application has its own PAM configuration file in the `/etc/pam.d/` directory, and generally this file has the same name as the application. For example, the `login` program uses the `/etc/pam.d/login` file, and the `sshd` daemon uses the `/etc/pam.d/sshd` file. If the service file for an application is missing, then PAM uses the `/etc/pam.d/other` file by default.

These configuration files contain rules that specify the modules to call for authentication and authorization. In this way, each application can use a different authentication method. However, applications often use the same sets of authentication methods. Most PAM configuration files use the `include` or `substack` directives to include a common configuration from the `password-auth` or `system-auth` files, which are also in the `/etc/pam.d/` directory.

In Red Hat Enterprise Linux 8 and later versions, administrators are no longer expected to edit PAM configuration files directly. Instead, you can select security profiles by using the `authselect` tool. When a profile is selected, the `authselect` tool modifies the `/etc/nsswitch.conf` file and the configuration files in the `/etc/pam.d/` and `/etc/dconf/db/distro.d/` directories. When using the `authselect` tool to manage PAM configurations, do not edit the managed configuration files manually. The `authselect` tool overwrites any modifications to managed files when the tool runs, such as during system updates.



Warning

You do not need to use the `authselect` tool if any of the following conditions apply:

- Your host is part of Red Hat Enterprise Linux Identity Management (IdM). Joining your host to an IdM domain with the `ipa-client-install` command automatically configures the System Security Services Daemon (SSSD) authentication on your host.
- Your host is part of Active Directory via SSSD. Calling the `realm join` command to join your host to an Active Directory domain automatically configures SSSD authentication on your host.

Red Hat recommends against changing the `authselect` profiles that are configured by the `ipa-client-install` or `realm join` commands. If you need to modify the `authselect` profiles, then display the current settings before making any modifications, so that you can revert to the current settings if necessary:

```
[root@host ~]# authselect current
Profile ID: sssd
Enabled features:
- with-sudo
- with-mkhomedir
- with-smartcard
```

Selecting Security Profiles

Red Hat Enterprise Linux 8 and later versions provide several security profiles. You can list the available security profiles by using the `authselect list` command.

minimal

This profile is for minimal installations with only local users.

nis

This profile enables Network Information Service (NIS) for system authentication.

sssd

This profile enables the SSSD for system authentication. This profile is the typical profile to use with LDAP or other identity providers.

winbind

This profile enables `winbind` for system authentication.

The security profile that you choose depends on your organization's environment and security requirements.

Each security profile can implement additional features, which enable specific additional PAM modules or which adjust existing PAM configurations. To see the available additional features for a security profile, use the `authselect list-features` command. The following example lists the additional feature options for the `minimal` profile:

```
[root@host ~]# authselect list-features minimal
with-altfiles
with-custom-aliases
with-custom-automount
with-custom-ethers
with-custom-group
with-custom-hosts
with-custom-initgroups
with-custom-netgroup
with-custom-networks
with-custom-passwd
with-custom-protocols
with-custom-publickey
with-custom-rpc
with-custom-services
with-custom-shadow
with-ecryptfs
with-faillock
with-mkhomedir
with-pamaccess
with-pwhistory
with-silent-lastlog
without-nullok
```

To select a security profile, use the `authselect select` command. The following example selects the `minimal` security profile:

```
[root@host ~]# authselect select minimal
Profile "minimal" was selected.
...output omitted...
```

You can select additional features by appending them to the `authselect select` command. The following example selects the `minimal` security profile with the `with-pwhistory` and `with-mkhomedir` additional features:

```
[root@host ~]# authselect select minimal with-pwhistory with-mkhomedir
Profile "minimal" was selected.
...output omitted...
```

You can view the current `authselect` security profile and additional features by using the `authselect current` command.

```
[root@host ~]# authselect current
Profile ID: sssd
Enabled features:
- with-mkhomedir
- with-silent-lastlog
- with-mdns4
```

Security features with additional options have configuration files in the `/etc/security/` directory. You can edit these configuration files to further customize the security options on a system. Because each configuration file contains different options that are specific to the application or PAM module, Red Hat recommends reviewing the man page for a configuration file before editing it.

Reading PAM Configuration Files

You can view the PAM configuration files that the `authselect` tool generates in the `/etc/pam.d/` directory to see the exact configuration effects. Application configuration files in the `/etc/pam.d/` directory follow a standard format for their rules:

```
type control module [module arguments]
```

The following output shows the `/etc/pam.d/system-auth` file as an example.

```
[root@host ~]# cat /etc/pam.d/system-auth
# Generated by authselect
# Do not modify this file manually, use authselect instead. Any user changes will
# be overwritten.
# See authselect(8) for more details.

auth      required          pam_env.so
auth      required          pam_faildelay.so ...
auth      [default=1 ignore=ignore success=ok]  pam_usertype.so isregular
auth      [default=1 ignore=ignore success=ok]  pam_localuser.so
auth      sufficient        pam_unix.so nullok
auth      [default=1 ignore=ignore success=ok]  pam_usertype.so isregular
auth      sufficient        pam_sss.so forward_pass
auth      required          pam_deny.so

account   required          pam_unix.so
account   sufficient        pam_localuser.so
account   sufficient        pam_usertype.so issystem
account   [default=bad success=ok user_unknown=ignore] pam_sss.so
account   required          pam_permit.so

password  requisite         pam_pwquality.so ...
password  sufficient        pam_unix.so ...
password  [success=1 default=ignore]  pam_localuser.so
password  sufficient        pam_sss.so use_authok
password  required          pam_deny.so

session   optional          pam_keyinit.so revoke
session   required          pam_limits.so
-session  optional          pam_systemd.so
```

session	optional	pam_oddjob_mkhomedir.so
session	[success=1 default=ignore]	pam_succeed_if.so ...
session	required	pam_unix.so
session	optional	pam_sss.so

The first field is the type, and organizes the tests into four *management groups*: **auth**, **account**, **password**, and **session**. The application invokes each group at a different time during a user authentication and authorization process.

PAM Rule Types

Type	Description
auth	The application verifies the rules in that management group when a user is authenticating. Users must pass these rules to validate their identity. In the previous output, one of the rules in this group calls the pam_unix module. This module verifies the provided username and password against the /etc/shadow file.
account	The application uses the account management group to verify that an account is valid at this time, and that passwords are not expired. In other words, the application determines whether access by that account is <i>authorized</i> . In the preceding example /etc/pam.d/system-auth file, a rule also calls the pam_unix module in the account management group. In this group, the pam_unix module determines from the expiration information in the /etc/shadow file whether the password is still valid.
password	Modules in the password management group control password changes. The rules in this management group do not involve authentication or authorization. If the application provides a feature for users to change their password, then PAM calls these rules when a user attempts to change their password through the application. In the previous /etc/pam.d/system-auth file, the pam_pwquality module is the first in this group. This module verifies the quality of the new password from the user. The pam_pwquality module is described in more detail in the following section. The pam_unix module is called next. In that password management group, the pam_unix module stores the new password in the /etc/shadow file.
session	The application calls the rules in this management group at the start and at the end of a user session. These rules manage tasks such as logging, and device or console ownership.

The order of the rules in a management group is essential, because PAM rules are parsed and executed from top to bottom.

For a dash (-) character in front of a type, such as **-session** near the end of the **/etc/pam.d/system-auth** file, PAM silently skips the rule if the module file is missing.

Some modules, such as the **pam_unix** module, can be called in multiple management groups. When PAM calls these modules, they know in which group they are running and adapt their behavior accordingly.

The next field, the control field, determines how the associated test affects the group's overall result. Of the many possible values for this field, the most common ones are included in the following table:

Common PAM Controls

Control	Description
<code>required</code>	The associated module must succeed. If the associated module fails, then PAM sets the management group overall result to <code>fail</code> . The other rules are still tested, to disguise the reason for the failure from a potential attacker.
<code>requisite</code>	This control is similar to <code>required</code> , but <code>requisite</code> stops testing on error. PAM directly gives back the control to the application or to the calling file. The <code>substack</code> control describes how to perform such a call.
<code>sufficient</code>	This control returns success immediately to the application or to the calling file if the associated module succeeds and no previous module failed. If the module fails, then PAM ignores the test and continues checking.
<code>optional</code>	PAM ignores the result of the test, even if it fails.
<code>include</code>	This control includes the rules from the provided PAM configuration file as if you directly entered them at that point.
<code>substack</code>	This control works like <code>include</code> , except that a failed test in the called file gives back the control to the current file instead of giving the control to the application.

The next field is the PAM module itself. Although you can give the full path to the module, usually you give only the relative path. PAM looks for the modules in the `/usr/lib64/security/` directory.

Modules return a code that PAM interprets as either success or error. PAM modules can also take options or arguments to alter their behavior. Not every module takes arguments, and options or arguments are usually unique to a specific module.

The SSSD Security Profile

The `sssd` security profile allows user authentication against remote identity management services, such as LDAP and Kerberos. When a remote user authenticates on the local system, SSSD stores their credentials and authentication parameters to a local cache. This way, even if the remote authentication server is not reachable, the remote user can still log in to the local system. This feature is particularly useful for laptops that are disconnected from the company network, and it also helps to reduce the load on the remote authentication services.

Accessing Security Profile and PAM Documentation

You can view documentation for `authselect` security profiles by using the `authselect show` command.

This documentation includes a description of the security profile, information about additional configuration, and information about each optional feature of the profile.

Documentation for configuration files for security features in the `/etc/security/` directory can be found in the man pages. Each configuration file has an associated man page with descriptions, options, defaults, and syntax information.

For example, the `/etc/security/faillock.conf` configuration file is documented by the `faillock.conf(5)` man page.

Chapter 5 | Controlling Authentication with PAM

Additionally, PAM modules have documentation through the man pages. Most of the modules have a dedicated man page with the following information:

- A description of the module.
- The module groups in which you can use it: auth, account, password, or session. Some modules can be used in several groups. At runtime, the behavior of such modules depends on the module group that they are running in.
- A list and description of the module parameters.
- The return codes.
- Some usage examples.

For example, the `pam_unix` module is documented by the `pam_unix(8)` man page. The man page names do not include the `.so` module extension.

The `pam.conf(5)` man page describes the full syntax of the PAM configuration files.

The PAM man pages typically start with the `pam_` prefix. The `man -k pam_` command returns a long list of related man pages.



References

The `authselect(8)`, `pam.conf(5)`, and `pam_*`(8) man pages.

For more information, refer to the *Configuring User Authentication Using authselect* chapter in the *Red Hat Enterprise Linux 9 Configuring Authentication and Authorization in Red Hat Enterprise Linux* guide at

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html-single/configuring_authentication_and_authorization_in_rhel/index#configuring-user-authentication-using-authselect_configuring-authentication-and-authorization-in-rhel

► Guided Exercise

Selecting the PAM Configuration

Select a PAM profile, and interpret the PAM configuration results.

Outcomes

- Use the `authselect` tool to list and select security profiles.
- List and select additional options for the security profiles.

Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

```
[student@workstation ~]$ lab start pam-selecting
```

Instructions

- 1. Log in as the `root` user on the `serverc` machine.
- 1.1. Use the `ssh student@serverc` command to log in to the `student` user on the `serverc` host.

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

- 1.2. Use the `sudo -i` command to change to the `root` user. Use the `student` sudo password.

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

- 2. List the available security profiles by using the `authselect list` command.

```
[root@serverc ~]# authselect list  
- minimal Local users only for minimal installations  
- sssd Enable SSSD for system authentication (also for local users only)  
- winbind Enable winbind for system authentication
```

- 3. List the available options and view the documentation for the `minimal` security profile.

- 3.1. Use the `authselect list-features minimal` command to list the available options for the `minimal` security profile.

```
[root@serverc ~]# authselect list-features minimal
with-altfiles
with-custom-aliases
with-custom-automount
with-custom-ethers
with-custom-group
with-custom-hosts
with-custom-initgroups
with-custom-netgroup
with-custom-networks
with-custom-passwd
with-custom-protocols
with-custom-publickey
with-custom-rpc
with-custom-services
with-custom-shadow
with-ecryptfs
with-faillock
with-mkhomedir
with-pamaccess
with-pwhistory
with-silent-lastlog
without-nullok
```

- 3.2. Use the `authselect show minimal` command to view the documentation for the `minimal` security profile.

```
[root@serverc ~]# authselect show minimal
Local users only for minimal installations
=====
```

Selecting this profile will enable local files as the source of identity and authentication providers.

This profile can be used on systems that require minimal installation to save disk and memory space. It serves only local users and groups directly from system files instead of going through other authentication providers. Therefore SSSD, winbind and fprintd packages can be safely removed.

Unless this system has strict memory and disk constraints, it is recommended to keep SSSD running and use 'sssd' profile to avoid functional limitations.

```
AVAILABLE OPTIONAL FEATURES
+-----+
with-faillock::
    Enable account locking in case of too many consecutive
    authentication failures.

with-mkhomedir::
    ...output omitted...
```

- 4. Select the minimal profile with the with-pwhistory option and verify the changes.

- 4.1. View the existing PAM configuration in the /etc/pam.d/system-auth file.

```
[root@serverc ~]# cat /etc/pam.d/system-auth
...output omitted...

auth      required          pam_env.so
auth      required          pam_faildelay.so
auth      sufficient        pam_unix.so nullok
auth      required          pam_deny.so

account   required          pam_unix.so

password  requisite         pam_pwquality.so
password  sufficient        pam_unix.so sha512 shadow
password  nullok use_authtok
password  required          pam_deny.so

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

**Note**

The /etc/pam.d/password-auth file is identical to the /etc/pam.d/system-auth file.

- 4.2. Use the authselect select minimal with-pwhistory command to apply the minimal security profile with the with-pwhistory additional option.

```
[root@serverc ~]# authselect select minimal with-pwhistory
Profile "minimal" was selected.
The following nsswitch maps are overwritten by the profile:
- aliases
- automount
- ethers
- group
- hosts
- initgroups
- netgroup
- networks
- passwd
- protocols
- publickey
- rpc
- services
- shadow
```

Chapter 5 | Controlling Authentication with PAM

- 4.3. Use the `authselect current` command to verify that the security profile is applied.

```
[root@serverc ~]# authselect current
Profile ID: minimal
Enabled features:
- with-pwhistory
```

- 4.4. View the changes to the `/etc/pam.d/system-auth` file.

```
[root@serverc ~]# cat /etc/pam.d/system-auth
...output omitted...

auth      required          pam_env.so
auth      required          pam_faildelay.so
delay=2000000
auth      sufficient        pam_unix.so nullok
auth      required          pam_deny.so

account   required          pam_unix.so

password  requisite         pam_pwquality.so
password  [default=1 ignore=ignore success=ok]  pam_localuser.so
password  requisite         pam_pwhistory.so
use_authok
password  sufficient        pam_unix.so sha512 shadow
nullok use_authok
password  required          pam_deny.so

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

- 5. Return to the workstation machine.

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

Finish

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

```
[student@workstation ~]$ lab finish pam-selecting
```

Configuring Password Quality Requirements

Objectives

- Implement password quality and complexity requirements by using PAM modules.

Overview of the PAM Password Quality Module

Many organizations require user passwords to comply with a particular set of rules, which often define the password length or character types. On Red Hat Enterprise Linux, you can use PAM to enforce these security policies and recommended practices.

When a user attempts to change their password through an open application session, PAM uses the rules in the `password` management group. The following examples show the `password` management group rules from files in the `/etc/pam.d/` directory:

```
[root@host ~]# grep ^password /etc/pam.d/sshd
password    include      password-auth

[root@host ~]# grep ^password /etc/pam.d/login
password    include      system-auth
```

From the example, the `sshd` and `login` login sessions use `include` calls to either the `system-auth` or `password-auth` files. The password rules in the `password-auth` and `system-auth` files are the same.



Note

The `authselect` tool automatically generates the files in the `/etc/pam.d` directory based on the selected `authselect` policy. If a configuration file notes that the `authselect` tool generated it, then do not manually edit the file, because edits might be overwritten when the `authselect` tool next runs.

Inspect the `/etc/pam.d/system-auth` file.

```
[root@host ~]# grep ^password /etc/pam.d/system-auth
password    requisite      pam_pwquality.so try_first_pass local_users_only retry=3
            authok_type=
password    sufficient    pam_unix.so sha512 shadow nullok try_first_pass
            use_authok
password    required      pam_deny.so
```

PAM rules are parsed and executed from top to bottom. When a user attempts to change their password, PAM calls the `requisite pam_pwquality.so` entry first.

The `pam_pwquality` module takes the new password that the user provides, compares it with a dictionary in the `/usr/share/cracklib/` directory, and checks it for patterns and easily guessed combinations. By default, the `pam_pwquality` module expects passwords to be longer

than eight characters. Because the module is called with a `requisite` controller, if the password is deemed not compliant, then the password change immediately stops and control returns to the application that initiated the PAM call. No further PAM rules are inspected.

If the user passes the `pam_pwquality` check, then PAM executes the `sufficient pam_unix.so` rule. In the `password` management group, the `pam_unix` rule updates the user's password in various back-end storage mechanisms, such as the `/etc/passwd` file or the `/etc/shadow` file.

Because the `pam_unix` rule uses the `sufficient` controller, if the rule successfully updates the user's password, then PAM closes and passes control back to the application that called it.

PAM reviews the `required pam_deny.so` rule only if the `sufficient pam_unix.so` rule fails to update the user's password. The `pam_deny` rule always returns a failure result. Because the `pam_deny` rule is required, PAM returns an overall failure to the application that attempts to change the user's password.

Configuring the PAM Password Quality Module

To configure the requirements for password complexity, edit the `/etc/security/pwquality.conf` file. You can add configuration files in the `/etc/security/pwquality.conf.d/` directory. Entries in additional files overwrite the entries in previously loaded files.



Important

In earlier RHEL versions than Red Hat Enterprise Linux 8, manually editing the `/etc/pam.d/system-auth` and `/etc/pam.d/password-auth` files was the preferred configuration method. In Red Hat Enterprise Linux 8 and later versions, you can apply changes that would be configured in those files by selecting the `authselect` profile that meets your needs.

The `/etc/security/pwquality.conf` file contains a list of options that can be set to positive or negative values to achieve specific effects on password requirements:

`minlen`

This value is the minimum length for a password. By default, all characters contribute one to the overall length score. However, with the `credit` system, an administrator can give some character types more value, and thus fewer overall characters are required to meet the same length.

`lcredit`

This value is the amount of credit that lowercase characters contribute to password length. If you set this value to a negative number, then the value specifies the minimum number of lowercase characters.

`ucredit`

This value is the amount of credit that uppercase characters contribute to password length. If you set this value to a negative number, then the value specifies the minimum number of uppercase characters.

`dcredit`

This value is the amount of credit that digits contribute to password length. If you set this value to a negative number, then this value specifies the minimum number of digit characters.

ocredit

This value is the amount of credit that other characters contribute to password length. Other characters are symbols and all the characters that are not included in lowercase, uppercase, and digit. If you set this value to a negative number, then the value specifies the minimum number of other characters.

You can combine the different length and complexity requirements with the `pam_pwquality` module. The chosen combination depends on the security policy or password policy of your organization.

Configuring a Password Policy with Specific Character Class Requirements

You can create minimum requirements for different character classes by using negative values in the `/etc/security/pwquality.conf` configuration file.

Consider the following example policy:

- Passwords must be a minimum of eight characters.
- Passwords must contain at least one uppercase character.
- Passwords must contain at least two digits.
- Passwords must contain at least one special character.

To implement this policy, edit the `/etc/security/pwquality.conf` file to set the following parameters:

```
[root@host ~]# vim /etc/security/pwquality.conf
minlen = 8
lcredit = 0
ucredit = -1
dcredit = -2
ocredit = -1
```

Negative values indicate the minimum number of characters for each class. The default value for each entry is listed in the `/etc/security/pwquality.conf` file.

The Password Credit Mechanism

Although password policies such as the previous one are widespread, they can be a source of frustration for users, who must create and remember passwords with complex combinations of different classes of characters. One solution is to use a variable password requirement.

With variable password requirements, a user can create a long password with few or no special characters, or a shorter password with many special characters.



Note

A 2011 study by Carnegie Mellon University and the United States National Institute of Standards and Technology (NIST) found that requiring users to create longer passwords, with no additional requirements for complexity or variation of types of characters in the passwords, produced passwords with more entropy that were easier for users to remember and create. Saranga Komanduri et al. "Of Passwords and People: Measuring the Effect of Password-Composition Policies." In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. Association for Computing Machinery. May, 2011. <https://doi.org/10.1145/1978942.1979321>

Later studies corroborate the 2011 study, such as a 2016 study by Carnegie Mellon researchers. Richard Shay et al. "Designing Password Policies for Strength and Usability." Association for Computing Machinery. May, 2016. <https://doi.org/10.1145/2891411>

The `pam_pwquality` credit mechanism is an implementation of variable password requirements, and is triggered when the credit parameters have positive values.

When the credit mechanism is enabled, the `minlen` parameter functions as a quality level that the password must reach for the module to accept it:

- The password earns one point for each character.
- An additional credit is given for each lowercase character, up to the value of the `lccredit` parameter.
- An additional credit is given for each uppercase character, up to the value of the `ucredit` parameter.
- An additional credit is given for each digit, up to the value of the `dcredit` parameter.
- An additional credit is given for each other character, up to the value of the `ocredit` parameter.

If the resulting score is equal to or above the `minlen` parameter, then the password is accepted.

The following `/etc/security/pwquality.conf` configuration file uses the password credit mechanism:

```
[root@host ~]# cat /etc/security/pwquality.conf
minlen = 18
lccredit = 1
ucredit = 1
dccredit = 2
ocredit = 5
```

With the configuration in the previous example, the following table shows how different passwords relate to the configuration:

Password	Length	Credit	Status
passwordlowercase	17	<ul style="list-style-type: none"> length = 17 lowercase = +1 uppercase = 0 digit = 0 other = 0 <p>Credit = 18</p>	Pass
WithFourUpperChr	16	<ul style="list-style-type: none"> length = 16 lowercase = +1 uppercase = +1 digit = 0 other = 0 <p>Credit = 18</p>	Pass
withaspecialch!	15	<ul style="list-style-type: none"> length = 15 lowercase = +1 uppercase = 0 digit = 0 other = +1 <p>Credit = 17</p>	Fail
!with-o?char.	13	<ul style="list-style-type: none"> length = 13 lowercase = +1 uppercase = 0 digit = 0 other = +4 <p>Credit = 18</p>	Pass
Sco:10+1=11	11	<ul style="list-style-type: none"> length = 11 lowercase = +1 uppercase = +1 digit = +2 other = +3 <p>Credit = 18</p>	Pass



References

The authselect(8), pam_pwquality(8), and pwquality.conf(5) man pages

For more information, refer to *Set Password Policy & Complexity for RHEL 8 & 9 via pam_pwhistory, pam_pwquality & pam_faillock* at
<https://access.redhat.com/solutions/5027331>

For more information about the authselect tool, refer to the *Configuring User Authentication Using authselect* documentation at

https://access.redhat.com/documentation/es-es/red_hat_enterprise_linux/9/html-single/configuring_authentication_and_authorization_in_rhel/index#configuring-user-authentication-using-authselect_configuring-authentication-and-authorization-in-rhel

► Guided Exercise

Configuring Password Quality Requirements

Configure password complexity requirements that PAM enforces when passwords are changed.

Outcomes

- Review and adjust password quality requirements by configuring the /etc/security/pwquality.conf file.

Before You Begin

As the student user on the workstation machine, use the lab command to prepare your environment for this exercise, and to ensure that all required resources are available.

```
[student@workstation ~]$ lab start pam-configuring
```

Instructions

- 1. Set the password policy on the serverc machine to require a password of at least 12 characters that include at least one symbol and one digit.
- 1.1. Log in to the serverc machine as the student user. No password is required.

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

- 1.2. Use the sudo -i command to change to the root user. Use the student sudo password.

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

- 1.3. Use a text editor to modify the /etc/security/pwquality.conf file to require a minimum password length of 12 characters, a minimum of one digit, and a minimum of one special character. Read through the options in the file to know what each line modifies. Uncomment and modify the lines to match the following output:

```
minlen = 12  
dcredit = -1  
ocredit = -1
```

- 2. Verify that you meet the new requirements by setting the student user password to each of the following values:

Chapter 5 | Controlling Authentication with PAM

- `alpha42numeric` should not work (2 digits, 14 characters, but no symbol)
 - `symbol+1digit` should work (1 symbol, 1 digit, and 13 characters)
- 2.1. Log out of the `root` account.

```
[root@serverc ~]# logout  
[student@serverc ~]$
```

- 2.2. Use the `passwd` command to set the password for the `student` user.

```
[student@serverc ~]$ passwd  
Changing password for user student.  
Current password: student  
New password: alpha42numeric  
BAD PASSWORD: The password contains less than 1 non-alphanumeric characters  
passwd: Authentication token manipulation error  
[student@serverc ~]$ passwd  
Changing password for user student.  
Current password: student  
New password: symbol+1digit  
Retype new password: symbol+1digit  
passwd: all authentication tokens updated successfully.  
[student@serverc ~]$
```

- 2.3. Use the `sudo -i` command to become the `root` user again, and change back the password for the `student` user to `student`. If the `sudo` command prompts for a password, then use the new one.

```
[student@serverc ~]$ sudo -i  
[sudo] password for student: symbol+1digit  
[root@serverc ~]# passwd student  
Changing password for user student.  
New password: student  
BAD PASSWORD: The password contains less than 1 digits  
Retype new password: student  
passwd: all authentication tokens updated successfully.  
[root@serverc ~]#
```

- 2.4. Return to the `workstation` machine.

```
[root@serverc ~]# logout  
[student@serverc ~]$ logout  
[student@workstation ~]$
```

Finish

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

```
[student@workstation ~]$ lab finish pam-configuring
```


Limiting Access after Failed Logins

Objectives

- Implement account locking after a specified number of failed logins.

Locking Accounts with Multiple Failed Logins

Many security policies have requirements for locking user accounts after multiple failed login attempts. You can use the PAM `pam_faillock` module to implement this requirement. The `pam_faillock` module can lock accounts after a specific number of failed login attempts, and it can automatically unlock these accounts after a predefined period. You can use the `faillock` command, from the `pam` package, to view a report of failed login attempts or to reset a user's failed attempts.



Note

The `pam_faillock` module was added in Red Hat Enterprise Linux 6.1, alongside the existing `pam_tally2` module. The `pam_faillock` module provides extra functions. For example, the `pam_faillock` module also controls failed login attempts on local screen savers. The `authselect` tool can configure the `pam_faillock` module, but not the `pam_tally2` module.

Consider the risk of denial-of-service attacks when enabling the `pam_faillock` module. A malicious user can lock an account by intentionally entering incorrect passwords. The legitimate user is then denied access to their account during the lock period.

PAM also provides the `pam_access` module to restrict users and groups. You can enable the `pam_access` module by including the `with-pamaccess` feature when selecting an `authselect` profile. To configure the `pam_access` module, edit the `/etc/security/access.conf` configuration file. For more information about the `pam_access` module, see the `pam_access(8)` and `access.conf(5)` man pages.

Enabling the `pam_faillock` Module

To enable the `pam_faillock` module, use the `authselect` tool to select your security profile and add the `with-faillock` option. The following example enables the `pam_faillock` module with the `mimimal` security profile:

```
[root@host ~]# authselect select minimal with-faillock --force
Backup stored at /var/lib/authselect/backups/2023-10-09-17-32-43.Jr656T
Profile "minimal" was selected.
The following nsswitch maps are overwritten by the profile:
- aliases
- automount
- ethers
- group
- hosts
- initgroups
```

- netgroup
- networks
- passwd
- protocols
- publickey
- rpc
- services
- shadow

Configuring the pam_faillock Module

You can configure the `pam_faillock` module by editing the `/etc/security/faillock.conf` configuration file. The file contains descriptions of each option, along with the default value.

`dir`

This string configures the directory with the user files with the failure records.

`audit`

This Boolean enables logging of user names if the user is not found.

`silent`

This Boolean disables the printing of informative messages.

`no_log_info`

This Boolean disables the logging of informative messages in the Syslog file.

`local_users_only`

This Boolean configures the `pam_faillock` module to ignore users that are not in the `/etc/passwd` file. This option is useful if your environment already locks users in LDAP or other identity management servers.

`deny`

This integer configures the number of allowed consecutive authentication failures within the interval before a user is locked.

`fail_interval`

This integer configures the number of seconds that authentication failures are measured within.

`unlock_time`

This integer configures the number of seconds that a user is locked for.

`even_deny_root`

This Boolean enables locking for the `root` user.

`root_unlock_time`

This integer configures the number of seconds that the `root` user is locked for. If this option is not specified, then the value defaults to match the `unlock_time` option.

`admin_group`

This string configures a group that follows the configuration for the `root` user.



Warning

If you configure the `pam_faillock` module to apply to the `root` user by using the `even_deny_root` option, then an attacker might be able to purposefully lock your `root` user in a denial-of-service attack.

Ensure that you at least have an alternative way to unlock the `root` user if you enable the `even_deny_root` option.

Managing Locked Accounts

You can list failed login attempts with the `faillock` command.

```
[root@host ~]# faillock
user1:
When          Type  Source           Valid
2023-07-14 11:46:35 RHOST 10.1.2.12      V
2023-07-14 11:46:31 RHOST 10.1.2.12      V
2023-07-14 11:46:44 RHOST 10.1.2.12      V
user2:
When          Type  Source           Valid
2023-07-14 11:48:01 TTY   tty2            V
2023-07-14 11:48:31 TTY   tty2            V
2023-07-14 11:48:37 TTY   tty2            V
root:
When          Type  Source           Valid
```

The `--user` option restricts the output to a specific account.

```
[root@host ~]# faillock --user user1
user1:
When          Type  Source           Valid
2023-07-14 11:46:31 RHOST 10.1.2.12      V
2023-07-14 11:46:44 RHOST 10.1.2.12      V
2023-07-14 11:46:35 RHOST 10.1.2.12      V
```

The `Type` column indicates the source of the connection: `RHOST` for remote connections, such as `SSH`, or `TTY` for console connections. The `Source` column gives the origin of the connection: hostname, IP address, or `TTY`. The `Valid` column indicates whether the record is still valid (V) or not (I).

The `--reset` option removes the failure records for a user. As a side effect, this option also unlocks the account if it was locked.

```
[root@host ~]# faillock --user user1 --reset
[root@host ~]# faillock --user user1
user1:
When          Type  Source           Valid
[root@host ~]#
```



References

The `faillock.conf(5)`, `pam_faillock(8)`, `faillock(8)`, `pam_access(8)`, and `access.conf(5)` man pages

For more information, refer to the *What Is pam_faillock and How to Use It in Red Hat Enterprise Linux 8 & 9?* article at

<https://access.redhat.com/solutions/62949>

► Guided Exercise

Limiting Access after Failed Logins

Implement restrictions that cause accounts to lock automatically after a specified number of failed logins, test the restriction, and use administrative tools to manually unlock locked accounts.

Outcomes

- Configure the `pam_faillock` module to lock accounts after a set number of failed logins.
- Test the locking operation.
- Manually unlock a locked account.

Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

```
[student@workstation ~]$ lab start pam-limiting
```

Instructions

- 1. Edit the `/etc/security/faillock.conf` configuration file on the `serverc` machine to set a limit of five failed attempts in a 15-minute (900-second) interval. Configure the locked accounts to automatically unlock after 15 minutes (900 seconds). Do not apply account locking to the `root` account.

- 1.1. Log in to the `serverc` machine as the `student` user. No password is required.

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

- 1.2. Use the `sudo -i` command to switch your identity to the `root` user. Use `student` as the password.

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

- 1.3. Use a text editor to modify the `/etc/security/faillock.conf` configuration file to lock accounts after five failed login attempts, and to unlock after 15 minutes (900 seconds). Read through the options in the file to know what each line modifies. Uncomment and modify the lines to match the following parameters:

```
deny = 5
unlock_time = 900
```

14. Use the `authselect select minimal with-faillock` command to apply the `minimal` authselect profile with the `with-faillock` option.

```
[root@serverc ~]# authselect select minimal with-faillock
...output omitted...
Profile "minimal" was selected.
...output omitted...
```

- 2. Verify that the `operator2` account locks after five failed login attempts by using the `ssh operator2@localhost` command. Enter an incorrect password at least five times before attempting to use the correct `redhat` password.
- 2.1. Use the `ssh operator2@localhost` command to fail to log in, by using an incorrect password. Because the `ssh` command prompts for a password only three times, run the command again after the first three incorrect attempts.

```
[root@serverc ~]# ssh operator2@localhost
...output omitted...
operator2@localhost's password: wrongpass1
Permission denied, please try again.
operator2@localhost's password: wrongpass2
Permission denied, please try again.
operator2@localhost's password: wrongpass3
Permission denied (publickey,gssapi-keyex,gssapi-with-mic,password).
[root@serverc ~]# ssh operator2@localhost
operator2@localhost's password: wrongpass4
Permission denied, please try again.
operator2@localhost's password: wrongpass5
Permission denied, please try again.
operator2@localhost's password: wrongpass6
Permission denied (publickey,gssapi-keyex,gssapi-with-mic,password).
```

- 2.2. Confirm that the `operator2` account is locked by attempting to use the `ssh operator2@localhost` command with the correct `redhat` password.

```
[root@serverc ~]# ssh operator2@localhost
operator2@localhost's password: redhat
Permission denied, please try again.
operator2@localhost's password: redhat
Permission denied, please try again.
operator2@localhost's password: redhat
Permission denied (publickey,gssapi-keyex,gssapi-with-mic,password).
```

- 2.3. Use the `faillock --user operator2` command to list the invalid login attempts.

```
[root@serverc ~]# faillock --user operator2
operator2:
When          Type   Source           Valid
2018-07-12 18:42:22 RHOST  localhost      V
2018-07-12 18:42:25 RHOST  localhost      V
2018-07-12 18:42:29 RHOST  localhost      V
2018-07-12 18:42:35 RHOST  localhost      V
2018-07-12 18:42:40 RHOST  localhost      V
```

Because you configured pam_faillock to lock accounts after five failed attempts, the operator2 account is now locked.

► 3. Unlock the operator2 account and confirm that the user can log in again.

- 3.1. Use the `faillock --user operator2 --reset` command to unlock the account.

```
[root@serverc ~]# faillock --user operator2 --reset
[root@serverc ~]#
```

- 3.2. Use the `faillock --user operator2` command to confirm that the account is unlocked.

```
[root@serverc ~]# faillock --user operator2
operator2:
When          Type   Source           Valid

```

- 3.3. Log in to the operator2 user by using the `ssh operator2@localhost` command with the correct redhat password. Log out of the operator2 user when done.

```
[root@serverc ~]# ssh operator2@localhost
operator2@localhost's password: redhat
...output omitted...
Last failed login: Fri Oct  6 18:52:52 EDT 2023 from ::1 on ssh:notty
There were 8 failed login attempts since the last successful login.
...output omitted...
[operator2@serverc ~]$ logout
[root@serverc ~]#
```

- 3.4. Return to the workstation machine.

```
[root@serverc ~]# logout
[student@serverc ~]$ logout
[student@workstation ~]$
```

Finish

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

```
[student@workstation ~]$ lab finish pam-limiting
```


▶ Lab

Controlling Authentication with PAM

Configure system authentication with PAM to use SSSD for authentication, enforce a specified password complexity policy on password changes, and lock user accounts after a specified number of failed logins.

Outcomes

- Apply an authselect security profile.
- Configure accounts to lock after failed logins.
- Configure password quality requirements.

Before You Begin

As the **student** user on the **workstation** machine, use the **lab** command to prepare your environment for this exercise, and to ensure that all required resources are available.

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

Instructions

1. Log in to the **serverd** machine as the **root** user.
2. Apply the **minimal** security profile with the **with-faillock** additional option.
3. Configure the **pam_faillock** module to lock an account after two failed attempts within a five-minute period and to unlock after five minutes.
4. Configure the **pam_pwquality** module to use the password credit mechanism with the following requirements:

Field	Value
Minimum password length or complexity	15
Minimum required lowercase characters	1
Credits for uppercase characters	2
Credits for digits	2
Credits for special characters	3

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 pam-review
```

Finish

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

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

► Solution

Controlling Authentication with PAM

Configure system authentication with PAM to use SSSD for authentication, enforce a specified password complexity policy on password changes, and lock user accounts after a specified number of failed logins.

Outcomes

- Apply an authselect security profile.
- Configure accounts to lock after failed logins.
- Configure password quality requirements.

Before You Begin

As the **student** user on the **workstation** machine, use the **lab** command to prepare your environment for this exercise, and to ensure that all required resources are available.

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

Instructions

1. Log in to the **serverd** machine as the **root** user.
 - 1.1. Use the **ssh student@serverd** command to log in to the **serverd** machine.

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

 - 1.2. Use the **sudo -i** command to change to the root user with the **student** sudo password.

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

2. Apply the **minimal** security profile with the **with-faillock** additional option.
 - 2.1. Use the **authselect select minimal with-faillock** command to select the **minimal** security profile with the **with-faillock** option.

```
[root@serverd ~]# authselect select minimal with-faillock
Profile "minimal" was selected.
The following nsswitch maps are overwritten by the profile:
- aliases
- automount
```

```
- ethers  
- group  
- hosts  
- initgroups  
- netgroup  
- networks  
- passwd  
- protocols  
- publickey  
- rpc  
- services  
- shadow
```

3. Configure the `pam_faillock` module to lock an account after two failed attempts within a five-minute period and to unlock after five minutes.
 - 3.1. Use a text editor to modify the `/etc/security/faillock.conf` configuration file to contain the following settings:

```
deny = 2  
fail_interval = 300  
unlock_time = 300
```

4. Configure the `pam_pwquality` module to use the password credit mechanism with the following requirements:

Field	Value
Minimum password length or complexity	15
Minimum required lowercase characters	1
Credits for uppercase characters	2
Credits for digits	2
Credits for special characters	3

- 4.1. Use a text editor to modify the `/etc/security/pwquality.conf` configuration file to contain the following settings:

```
minlen = 15  
lcredit = -1  
ucredit = 2  
dcredit = 2  
ocredit = 3
```

- 4.2. Return to the `workstation` machine.

```
[root@serverd ~]# logout  
[student@serverd ~]$ logout  
Connection to closed serverd closed.  
[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 pam-review
```

Finish

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

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

Summary

- The Pluggable Authentication Modules (PAM) system provides a generic way for applications to implement common support for authentication and authorization.
- A PAM-enabled application invokes the rules in each management group (`auth`, `account`, `password`, and `session`) at different times during the user authentication and authorization process.
- The `authselect` command is the recommended method for updating PAM configuration.
- You can select a security profile from the `authselect` security profiles.
- Each `authselect` security profile has a number of optional features, which offer additional configuration options.
- The `pam_pwquality` module uses the `/etc/security/pwquality.conf` configuration file to enforce your organization's password complexity requirements.
- The `pam_faillock` module uses the `/etc/security/faillock.conf` configuration file to lock accounts after a number of failed authentication attempts within a period of time.

Chapter 6

Recording System Events with Audit

Goal

Record and inspect system events relevant to security by using the Linux kernel's Audit system and supporting tools.

Sections

- Configuring Audit to Record System Events (and Guided Exercise)
- Inspecting Audit Logs (and Guided Exercise)
- Writing Custom Audit Rules (and Guided Exercise)
- Enabling Prepackaged Audit Rule Sets (and Guided Exercise)

Lab

- Recording System Events with Audit

Configuring Audit to Record System Events

Objectives

- Ensure that Audit is installed and configured to record system events, and that it forwards Audit messages to a central Audit server.

The Linux Audit System

The Linux Audit system is a mechanism in the kernel that provides a way to track security-related events and information on your systems. You define a set of rules that you load into the kernel to specify which events the kernel must record in the Audit log. The `auditd` system daemon writes the logged events to the local disk or forwards them to remote log servers. You can then use this information to discover and investigate the cause of violations or attempted violations of the system's security policy.

You can configure Audit to include or exclude events based on user identity, security contexts, or other labels. The Audit system can collect the following information about events:

- The date, time, type, and outcome of an event
- Sensitivity labels of subjects and objects
- The identity of the user who triggered the event
- Modifications to the Audit configuration and attempts to access Audit log files
- The use of authentication mechanisms, such as SSH, Kerberos, and others
- Changes to any trusted database, such as the `/etc/passwd` file
- Attempts to import or export information into or out of the system

You might need to use the Audit system to meet the compliance requirements of particular security certifications. The National Information Assurance Partnership (NIAP) and Best Security Industries (BSI) have evaluated the Audit system for security compliance.

The Audit System is designed to meet or exceed the requirements of the following certifications or compliance guides:

- The Controlled Access Protection Profile (CAPP)
- The Labeled Security Protection Profile (LSPP)
- Rule Set Base Access Control (RSBAC)
- The National Industrial Security Program Operating Manual (NISPOM)
- The Federal Information Security Management Act (FISMA)
- The Payment Card Industry Data Security Standard (PCI DSS)
- Security Technical Implementation Guides (STIGs)

Auditing Your System

The Audit system consists of two main parts: the kernel-side system call processing, and the user-space applications and utilities.

The kernel component receives system calls from user-space applications, and then filters them through the *exclude* filter. If a system call passes the *exclude* filter, then it is filtered through one of the following filters: *user*, *task*, *fstype*, or *exit*. Then, based on the Audit rule configuration, the kernel component sends the system call to the `audited` daemon for further processing.

The user-space Audit daemon collects the information from the kernel and creates entries in a log file. Other Audit user-space utilities interact with the `audited` daemon, the kernel Audit component, or the Audit log files. The `auditctl` command interacts with the kernel Audit component to manage rules and to control many settings and parameters of the event generation process. The remaining Audit utilities take the contents of the Audit log files as input and generate output based on the user's requirements.

Without any additional configuration, only a limited number of messages pass through the Audit system (mainly authentication and authorization messages, and SELinux messages). Administrators can add Audit rules to control the Audit system, watch files, or record information about any system call with the `auditctl` command. This customization is discussed later in this chapter. The remainder of this section focuses on how to configure the `audited` daemon to process any logged events.

Configuring the Audit System Daemon

The `audit` package provides the `audited` daemon for configuring the Audit system and searching the Audit logs.

You can configure Audit rules and the `audited` daemon with the following files:

/etc/audit/auditd.conf

This is the primary file for configuring the `audited` daemon.

/etc/audit/audit.rules

The `audited` daemon uses this file to load Audit rules.

Do not edit this file. This file is generated from the files in the `/etc/audit/rules.d` directory when the `audited` daemon starts.

/etc/audit/rules.d/

This directory contains manually configured Audit rules.

All files that have the `.rules` extension are combined in the `/etc/audit/audit.rules` file and are loaded into the kernel when the `audited` daemon starts.

This section focuses on the `/etc/audit/auditd.conf` file, which configures the behavior of the `audited` daemon. Rule files are discussed in a later section.



Important

When the `audited` daemon starts, its `systemd` unit file automatically rebuilds the Audit rule set from rule files in the `/etc/audit/rules.d/` directory and attempts to load them into the kernel. When the service is reloaded, it attempts to rebuild the rules file and load the new rules by running the `augenrules` command.

This service might fail if a control rule is loaded that makes the current rules immutable. In this case, you must reboot the system to change the loaded rule set.

Adjusting the Audit System Daemon to Manage Storage

Depending on the Audit rules that are loaded, the Audit logs can expand quickly. To prevent other processes from competing with the Audit logs for storage space and to improve performance tuning, mount a dedicated file system to the `/var/log/audit/` directory.

In addition, many security policies require that when the Audit system cannot log an event to disk due to the lack of storage space, the system must immediately halt or drop to single-user mode. This requirement ensures that no event can occur without being logged. The `auditd` daemon has mechanisms to warn you if this is about to happen, and to take various actions when storage gets low.

To configure the `auditd` daemon behavior to comply with your security policy, you can adjust a number of variables in the `/etc/audit/auditd.conf` file:

`log_file`

This option specifies the location of the file that is used for storing logs; the default setting is: `/var/log/audit/audit.log`.

`max_log_file`

This option specifies the maximum log file size in MB to use the available space on the partition by triggering the `max_log_file_action` parameter.

`max_log_file_action`

This option decides what action is taken when the limit set in the `max_log_file` parameter is reached. Set this parameter to the `keep_logs` value to prevent Audit log files from being overwritten.

`space_left`

This option specifies the amount of free space left on the disk to trigger the action set in the `space_left_action` parameter. This parameter must be set to a number that gives the administrator enough time to respond and free up disk space.

`space_left_action`

Red Hat recommends that you set the `space_left_action` parameter to the `email` or `exec` value with an appropriate notification method.

`admin_space_left`

This option specifies the absolute minimum amount of free space to trigger the action set in the `admin_space_left_action` parameter. The parameter must be set to a value that leaves enough space to log actions performed by the administrator. This parameter usually suspends the `auditd` daemon or halts the entire system, leaving a small amount of free space so that you can fix the issue.

`admin_space_left_action`

Set this parameter to the `single` value to put the system into single-user mode, which allows the administrator to free up some disk space.

`disk_full_action`

This parameter specifies an action that is triggered when no free space is available on the partition that holds the Audit log files. Set this parameter to the `halt` or `single` values. This parameter ensures that the system is either shut down or operating in single-user mode when Audit can no longer log events.

`disk_error_action`

This parameter specifies an action that is triggered in case an error is detected on the partition that holds the Audit log files. Set this parameter to the `syslog`, `single`, or

halt value, depending on your local security policies regarding the handling of hardware malfunctions.

Adjusting Audit System Settings to Improve Performance

You can improve the performance of the auditd service by adjusting the following settings in the /etc/audit/auditd.conf configuration file:

- Set the `flush` parameter to the `incremental_async` value. The `incremental_async` parameter flushes the log file asynchronously for higher performance. The `flush` parameter works in combination with the `freq` parameter, which determines how many records can be sent to the disk before forcing a hard synchronization with the hard drive.
- Set the `freq` parameter to the `100` value. These parameters ensure that Audit event data is synchronized with the log files on the disk by maintaining good performance during bursts of activity.

Remote Logging

You can send Audit messages to a remote system in two ways. You can either send messages to the `rsyslog` service and configure `rsyslog` to forward them to a remote server, or you can configure your system to send messages to the remote `auditd` service.

When configuring your system to use either method to send messages to a remote server, you should make two changes on the server in the /etc/audit/auditd.conf file:

- Set the `log_format = ENRICHED` parameter to resolve the UID, GID, system call number, architecture, and socket address information before transmitting each event. This parameter ensures that the log information makes sense on a remote system that might have different values for these mappings.
- Set the `name_format = HOSTNAME` parameter to include the machine's hostname in each message. This parameter makes it clear to which machine an Audit event belongs.

Rsyslog Remote Logging

If you are sending messages to the `rsyslog` service, then edit the /etc/audisp/plugins.d/syslog.conf file to include the `active = yes` line.

Configure the `rsyslog` service to collect remote Audit messages by editing the /etc/rsyslog.conf file. Open the 514/UDP firewall port for standard syslog logging. Other ports for other protocols are also supported by the `rsyslog` service.

Audit Remote Logging

If you are sending messages to a remote `auditd` daemon, then you must install the `audispd-plugins` package on your client. On the Audit client, edit the /etc/audit/plugins.d/audisp-remote.conf file to include the `active = yes` line. Edit the /etc/audit/audisp-remote.conf file to include a `remote_server` directive set to the IP address or hostname of the remote `auditd` server. Then, update the `port` directive if your remote server is not listening on the default 60/TCP port.

Ensure that the server's firewall allows connections to that port. Restart the `auditd` daemon to put this change into effect. You can perform a reboot to ensure that all services are restarted.



Warning

Both methods of sending Audit messages to a remote server use clear text protocols without encryption by default. This lack of encryption makes the content of the messages subject to interception and tampering.

The `rsyslog` service must be configured to use TLS authentication and encryption to protect your Audit log traffic.

You must configure the `auditd` daemon to use Kerberos authentication and encryption in the `/etc/audisp/audisp-remote.conf` file on the client and the `/etc/audit/auditd.conf` file on the server. See the `audisp-remote.conf(5)` and `auditd.conf(5)` man pages for details.



References

`auditd(8)`, `auditd.conf(5)`, `audispd.conf(5)`, `audisp-remote.conf(5)`, and `audit.rules(7)` man pages

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/9/html/security_hardening/auditing-the-system_security-hardening

The upstream project documentation for Linux Audit is available at

<https://github.com/linux-audit/audit-documentation>

► Guided Exercise

Configuring Audit to Record System Events

Verify the availability and configuration of the `auditd` service, and configure Audit to forward messages to a central server.

Outcomes

- Install and configure the Audit server to record system events.
- Forward Audit messages to a central Audit server.

Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

```
[student@workstation ~]$ lab start audit-configuring
```

Instructions

- 1. Verify that Audit is installed and running on the `servera` machine.
- 1.1. Log in to the `servera` machine as the `student` user.

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

- 1.2. Change to the `root` user. Use `student` as the password.

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

- 1.3. Verify that the `auditd` service is running and enabled.

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

- 1.4. Verify that the `/var/log/audit/audit.log` log file records the start of the previous `root` session with the `sudo` command.

This log file records the successful start of a user session for the `root` user that uses `sudo` on the `pts/1` terminal. Some of these details might be different for your Audit event.

```
[root@servera ~]# tail /var/log/audit/audit.log
type=CRED_REFR msg=audit(1697168486.971:110): pid=1184 uid=1000 auid=1000 ses=1
subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 msg='op=PAM:setcred
grantors=pam_unix acct="root" exe="/usr/bin/sudo" hostname=? addr=? terminal=/
dev/pts/0 res=success'UID="student" AUID="student"
type=USER_START msg=audit(1697168486.976:111): pid=1184 uid=1000 auid=1000
ses=1 subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
msg='op=PAM:session_open
grantors=pam_keyinit,pam_keyinit,pam_limits,pam_systemd,pam_unix acct="root" exe="/
usr/bin/sudo" hostname=? addr=? terminal=/dev/pts/0 res=success'UID="student"
AUID="student"
type=BPF msg=audit(1697168486.991:112): prog-id=32 op=LOAD
type=BPF msg=audit(1697168486.992:113): prog-id=33 op=LOAD
...output omitted...
```

- ▶ 2. Verify the values for the `flush` and `freq` variables in the Audit configuration file to adjust the performance for Audit. Modify the value of the `name_format` variable to include additional information in the Audit log file.
 - 2.1. In the `/etc/audit/auditd.conf` configuration file, verify that the value for the `flush` option is `INCREMENTAL_ASYNC` to flush Audit events asynchronously. The value for the `freq` variable is `50` to flush the Audit log after every 50 records.
Set the `name_format` variable to `HOSTNAME` to include the hostname in the Audit log file.

```
[root@servera ~]# cat /etc/audit/auditd.conf
...output omitted...
log_format = ENRICHED
flush = INCREMENTAL_ASYNC
freq = 50
max_log_file = 8
num_logs = 5
priority_boost = 4
name_format = HOSTNAME
...output omitted...
```

- 2.2. Restart the `auditd` service to update its configuration.

```
[root@servera ~]# service auditd restart
Stopping logging:
Redirecting start to /bin/systemctl start auditd.service
```



Note

The `service` command is deprecated; do not use it in production environments. The `systemctl restart` command cannot be used with the `auditd` service due to the interaction between the daemon and the Linux kernel. In production environments, reboot the machine to ensure that the new configuration is loaded.

- ▶ 3. Configure the Audit service on the `servera` machine to send Audit messages to the Audit service on the `serverb` machine.

Chapter 6 | Recording System Events with Audit

- 3.1. Install the `audispd-plugins` package.

```
[root@servera ~]# dnf install audispd-plugins
...output omitted...
Dependencies resolved.
=====
 Package           Arch   Version        Repository      Size
=====
 Installing:
 audispd-plugins  x86_64  3.0.7-103.el9  rhel-9.2-for-x86_64-baseos-rpms  52 k

Transaction Summary
=====
...output omitted...
Is this ok [y/N]: y
...output omitted...

Complete!
```

- 3.2. In the `/etc/audit/plugins.d/au-remote.conf` file, set the `active` option to the `yes` value to enable remote logging.

```
[root@servera ~]# cat /etc/audit/plugins.d/au-remote.conf
...output omitted...
active = yes
...output omitted...
```

- 3.3. In the `/etc/audit/audisp-remote.conf` file, set the `remote_server` option to the `serverb.lab.example.com` hostname. Use `60` as the port for the remote logging server.

```
[root@servera ~]# cat /etc/audit/audisp-remote.conf
#
# This file controls the configuration of the audit remote
# logging subsystem, audisp-remote.
#
remote_server = serverb.lab.example.com
port = 60
...output omitted...
```

- 3.4. Restart the `auditd` service to update its configuration.

```
[root@servera ~]# service auditd restart
Stopping logging:
Redirecting start to /bin/systemctl start auditd.service
```

- ▶ 4. Return to the workstation machine as the student user.

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

- 5. Configure the Audit service on the **serverb** machine to accept Audit messages.

- 5.1. Log in to the **serverb** machine as the **student** user.

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

- 5.2. Change to the **root** user. Use **student** as the password.

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

- 5.3. In the **/etc/audit/auditd.conf** file, uncomment the **tcp_listen_port** variable and set its value to **60** so that the Audit service listens to the TCP port.

```
[root@serverb ~]# cat /etc/audit/auditd.conf  
...output omitted...  
tcp_listen_port = 60  
...output omitted...
```

- 5.4. Open TCP port **60** to enable access to the Audit server.

```
[root@serverb ~]# firewall-cmd --zone=public --add-port=60/tcp --permanent  
success  
[root@serverb ~]# firewall-cmd --reload  
success
```

- 5.5. Restart the **auditd** service to update its configuration.

```
[root@serverb ~]# service auditd restart  
Stopping logging:  
Redirecting start to /bin/systemctl start auditd.service
```

- 5.6. Return to the **workstation** machine as the **student** user.

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

- 6. Verify that remote logging for Audit is working.

- 6.1. Log in to the **servera** machine as the **student** user.

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

- 6.2. Change to the **root** user. Use **student** as the password.

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

- 6.3. Return to the **workstation** machine as the **student** user.

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

- 6.4. Log in to the **serverb** machine as the **student** user.

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

- 6.5. Use the **sudo -i** command to change to the **root** user. Use **student** as the password.

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

- 6.6. Verify that new entries in the Audit log file exist for the message created on the **servera** machine. When done, log out of the **serverb** machine.

```
[root@serverb ~]# grep servera /var/log/audit/audit.log  
...output omitted...  
node=servera type=SERVICE_STOP msg=audit(1697169930.355:216): pid=1  
uid=0 auid=4294967295 ses=4294967295 subj=system_u:system_r:init_t:s0  
msg='unit=user@1000 comm="systemd" exe="/usr/lib/systemd/systemd" hostname=?  
addr=? terminal=? res=success'UID="root" AUID="unset"  
node=servera type=SERVICE_STOP msg=audit(1697169930.362:217): pid=1  
uid=0 auid=4294967295 ses=4294967295 subj=system_u:system_r:init_t:s0  
msg='unit=systemd-hostnamed comm="systemd" exe="/usr/lib/systemd/systemd"  
hostname=? addr=? terminal=? res=success'UID="root" AUID="unset"  
node=servera type=BPF msg=audit(1697169930.366:218): prog-id=0 op=UNLOAD  
node=servera type=BPF msg=audit(1697169930.366:219): prog-id=0 op=UNLOAD  
node=servera type=SERVICE_STOP msg=audit(1697169930.379:220): pid=1 uid=0  
auid=4294967295 ses=4294967295 subj=system_u:system_r:init_t:s0 msg='unit=user-  
runtime-dir@1000 comm="systemd" exe="/usr/lib/systemd/systemd" hostname=? addr=?  
terminal=? res=success'UID="root" AUID="unset"
```

- 7. Return to the **workstation** machine as the **student** user.

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

Finish

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

```
[student@workstation ~]$ lab finish audit-configuring
```

Inspecting Audit Logs

Objectives

- Search for events and generate reports from the Audit log, and interpret the results.

Interpreting Audit Messages

Audit events that are recorded in the `/var/log/audit/audit.log` file include a lot of information in a condensed format. A single event might log multiple *Audit records* of different types to the log as separate messages. Each of these records might include several *fields* of information about the logged event.

The following example shows several Audit records that are associated with a single Audit event as it might appear in the `/var/log/audit/audit.log` file, recorded in an unprocessed raw format by the `auditd` daemon.

```
type=SYSCALL msg=audit(1371716130.596:28708): arch=c000003e syscall=2 success=yes  
exit=4 a0=261b130 a1=90800 a2=e a3=19 items=1 ppid=2548 pid=26131 auid=1000 uid=0  
gid=0 euid=0 suid=0 egid=0 sgid=0 fsgid=0 tty pts0 sess=1 comm="aureport"  
exe="/sbin/aureport" subj=unconfined_u:unconfined_r:unconfined_t:s0-  
s0:c0.c1023 key="audit-access"  
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
```

The previous output displays three Audit records, which are all part of Audit event 28708.

- The `type=SYSCALL` record indicates the `type` field. Each Audit record has a *record type*, sometimes called a *message type*, which is reflected by the `type` field that starts each record. This record is a `SYSCALL` record.
- The `msg=audit(1371716130.596:28708)` record indicates the `msg` field. The `msg` field gives the time stamp for this record and the event ID. The number before the colon (in this case, `1371716130.596`) is the time stamp in the number of seconds since the epoch, 00:00 UTC on January 1, 1970. You can convert epoch time to local time by using the `date --date=@<epoch-time>` command. The number after the colon (`28708`) is the Audit event number, which the record shares with the other records for this event.
- The `syscall=2` record indicates the `syscall` field. The `type` of the first record is `SYSCALL`, which indicates information about a system call that was made to the kernel. This `syscall` field indicates the number of the system call that was made (not its name). The mapping of system call numbers to names can vary between processor architectures, which is one reason why it can be challenging to correctly interpret a log that is in raw format without help. You can use the `ausearch` command to provide this help.
- The `auid=1000` record indicates the `auid` field. The `auid` field records the Audit UID of the user that triggered this event. This is the UID of the initial account that was used to log in to this machine by the user that triggered this event, even if they used the `sudo` or `su` commands to become another user.

- The key="audit-access" record indicates the key field. The key field is an identifier that you can use when searching for events, similar to a tag. You can set keys in your custom Auditing rules to make it easier to filter for certain types of events.

Searching for Events

The Audit system includes the ausearch command, a powerful tool for searching Audit logs. You can use the ausearch command to search for and filter various types of events.

Interpret Numeric Values in Audit Logs

The ausearch command can translate numeric values into more readable values, such as user names or system call names, so that you can interpret those events more easily. You can use the -i option to interpret the log records and translate numeric values into names. This option is useful when you have raw log files.

The -if option in the following example takes a saved raw log file as an argument, so that ausearch can analyze the raw log file rather than the current Audit log on the system.

```
[root@host ~]# ausearch -i -if ./raw_audit.log
...output omitted...
-----
type=PROCTITLE msg=audit(10/11/23 14:14:52.183:421) : proctitle=autrace /bin/ls /
tmp
type=SYSCALL msg=audit(10/11/23 14:14:52.183:421) : arch=x86_64 syscall=exit_group
a0=EXIT_SUCCESS a1=0xfffffffffffffe98 a2=0xe7 a3=0x7ffcdd36a200 items=0
ppid=1438 pid=1440 auid=student uid=root gid=root euid=root suid=root
fsuid=root egid=root sgid=root fsgid=root tty=pts0 ses=8 comm=ls exe=/usr/bin/ls
subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 key=(null)
```

Display Raw Audit Log Entries

You can use the --raw option to print raw log entries without separators between events. This option is useful if you have other tools that can parse the raw log format. The -r short option is equivalent.

In the following example, the -p option looks for Audit log entries that are recorded by a process with a PID of 1440.

```
[root@host ~]# ausearch -p 1440 --raw
...output omitted...
type=SYSCALL msg=audit(1697048092.183:420): arch=c000003e syscall=3 success=yes
exit=0 a0=2 a1=fbad2006 a2=7f366a7f69e0 a3=b items=0 ppid=1438 pid=1440 auid=1000
uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0 sgid=0 fsgid=0 tty=pts0 ses=8 comm="ls"
exe="/usr/bin/ls" subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
key=(null)ARCH=x86_64 SYSCALL=close AUID="student" UID="root" GID="root"
EUID="root" SUID="root" FSUID="root" EGID="root" SGID="root" FSGID="root"
type=PROCTITLE msg=audit(1697048092.183:420):
proctitle=61757472616365002F62696E2F6C73002F746D70
type=SYSCALL msg=audit(1697048092.183:421): arch=c000003e syscall=231 a0=0
a1=fffffffffffffe98 a2=e7 a3=7ffcd36a200 items=0 ppid=1438 pid=1440 auid=1000
uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0 sgid=0 fsgid=0 tty=pts0 ses=8 comm="ls"
exe="/usr/bin/ls" subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
key=(null)ARCH=x86_64 SYSCALL=exit_group AUID="student" UID="root" GID="root"
EUID="root" SUID="root" FSUID="root" EGID="root" SGID="root" FSGID="root"
type=PROCTITLE msg=audit(1697048092.183:421):
proctitle=61757472616365002F62696E2F6C73002F746D70
```

Display All Records Generated by an Event

A single event might generate multiple, related Audit log records. You can use the `-a <EVENT-ID>` option to show all records for a particular event based on the event ID.

```
[root@host ~]# ausearch -a 233
...output omitted...
---
time->Wed Oct 11 14:08:48 2023
type=USER_LOGIN msg=audit(1697047728.122:233): pid=1363 uid=0 auid=1000 ses=8
subj=system_u:system_r:sshd_t:s0-s0:c0.c1023 msg='op=login id=1000 exe="/usr/
sbin/sshd" hostname=? addr=172.25.250.9 terminal=/dev/pts/0 res=success'
```

Search for Records by Message Type

The message type describes the category of an Audit record. There are many message types, such as `ADD_USER`, `LOGIN`, `USER_CMD`, `VIRT_CONTROL`, and many more. By filtering for a specific message type, you can find information about a specific event, even if you do not know the exact details of the event.



Note

You can find a list of all Audit message types in the *RHEL Audit System Reference* article at <https://access.redhat.com/articles/4409591>

You can use the `-m <MESSAGE-TYPE>` option to show all events that include a record with a specific message type. The `-message` long option is equivalent.

For example, if you suspect that an unauthorized login event occurred, you might filter on the `LOGIN` or `USER_LOGIN` message types to see all login events.

```
[root@host ~]# ausearch -m LOGIN
...output omitted...
-----
time->Wed Oct 11 14:08:47 2023
type=LOGIN msg=audit(1697047727.989:227): pid=1367 uid=0
subj=system_u:system_r:init_t:s0 old-auid=4294967295 auid=1000 tty=(none) old-
ses=4294967295 ses=9 res=1
```

Search for Records by File Name

You can use the `-f <FILENAME>` option to search for all events that are related to a specific file name. The `--file` long option is equivalent.

```
[root@host ~]# ausearch -f /bin/ls
...output omitted...
-----
time->Wed Oct 11 14:24:07 2023
type=PROCTITLE msg=audit(1697048647.148:432):
proctitle=61757472616365002F62696E2F6C73002D6C002F746D70
type=PATH msg=audit(1697048647.148:432): item=1 name="/lib64/ld-linux-
x86-64.so.2" inode=8389810 dev=fc:04 mode=0100755 uid=0 ogid=0 rdev=00:00
obj=system_u:object_r:ld_so_t:s0 nametype=NORMAL cap_fp=0 cap_fi=0 cap_fe=0
cap_fver=0 cap_frootid=0
type=PATH msg=audit(1697048647.148:432): item=0 name="/bin/ls" inode=1192
dev=fc:04 mode=0100755 uid=0 ogid=0 rdev=00:00 obj=system_u:object_r:bin_t:s0
nametype=NORMAL cap_fp=0 cap_fi=0 cap_fe=0 cap_fver=0 cap_frootid=0
type=CWD msg=audit(1697048647.148:432): cwd="/root"
type=EXECVE msg=audit(1697048647.148:432): argc=3 a0="/bin/ls" a1="-l" a2="/tmp"
type=SYSCALL msg=audit(1697048647.148:432): arch=c000003e syscall=59
success=yes exit=0 a0=7fffa43d17b1 a1=7fffa43d0b30 a2=7fffa43d0b50
a3=7f1b0c7b13e0 items=2 ppid=1450 pid=1452 auid=1000 uid=0 gid=0 euid=0 suid=0
fsuid=0 egid=0 sgid=0 fsgid=0 tty=pts0 ses=8 comm="ls" exe="/usr/bin/ls"
subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 key=(null)
```

Search for Events by Key

Audit can label an event with a key that you can use in searches. You can use the `-k <KEY>` option to search for all events that are labeled with a specific key. For example, if you have several Audit rules that track changes to identity files, then you can configure Audit to label the created entries with a key such as `identity`. You can find each of these entries by using the `-k identity` option. This way, you can find only the entries that relate to the labeled rules.

Search for Events by Date

You can use the `--start [start-date] [start-time]` option to search for events that occurred after a specific start date and start time. If you do not specify a starting time, then the search assumes midnight. If you omit the starting date, then the search assumes the current day. The time format depends on your current locale setting. Other values that you can use include `recent` (the past ten minutes), `this-week`, `this-month`, and `this-year`.

You can use the `--end` option to search for events that occurred before a specific date and time with the same time format syntax as the `--start` option.

Other Audit Search Options

Other options exist to help you search for events based on user, terminal, virtual machine, or other identifiers.

For a complete list of options refer to the `ausearch(8)` manual page.

Interpreting Audit Log Entries

The following `ausearch` command returns an interpreted version of event 28708 from the previous example:

```
[root@host ]# ausearch -i -a 28708
-----
type=PATH msg=audit(07/31/2023 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(07/31/2023 10:15:30.596:28708) : cwd=/root
type=SYSCALL msg=audit(07/31/2023 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
```

Each Audit event is separated in the interpreted output by four dashes.

This output displays the PATH, CWD, and SYSCALL event types.

The PATH record is a file involved in this event. The file is named `/var/log/audit` (`name=/var/log/audit`), and is inode 17998 (`inode=17998`). The file is on the file system on a device with major/minor numbers 253,1 (`dev=fd:01`, the device numbers are in hexadecimal format). By looking in the `/dev` directory with the `ls -l` command, you can see that the `/dev/dm-1` device has those numbers and is associated with a logical volume. The file is a directory with octal permissions 750 (`mode=dir,750`), owned by the `root` user and the `root` group (`uid=root ogid=root`), with the SELinux type `auditd_log_t` (`obj=system_u:object_r:auditd_log_t:s0`).

The CWD record is the current working directory that is associated with the process that triggered this event, in this case the `/root` directory.

The SYSCALL record is the system call that triggered this event. The `open()` system call (`syscall=open`) was used to successfully (`success=yes`) open the file that is specified by the PATH record (the `/var/log/audit` directory). This call was done by a process with the 26131 PID (`pid=26131`). The call was started by the `/sbin/aureport` executable (`exe=/sbin/aureport`) and was run with the `root` effective UID (`euid=root`) and a `unconfined_t` SELinux domain (`subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023`) by the `root` user (`uid=root`). The command was run on the `pts/0` pseudoterminal (`tty=pts0`), probably a graphical terminal window or remote login session. The user originally logged in as the `student` user (`auid=student`) and has somehow become the `root` user since then. This record has the `audit-access` key set on it, to make its event easier to find with the `ausearch` command (`key=audit-access`).

Reporting on Audit Messages

Instead of reading individual Audit messages, you can use the `aureport` command to get a quick overview of Audit messages or more detailed reports on specific types of events.

When you run the `aureport` command without any options, the command returns an overview of how many different types of events are present in the logs. When you specify search options (mostly the same options as those for the `ausearch` command), the command displays a list of all events that match the search criteria. Common options include the `-i` option to interpret results, and the `--summary` option to condense the list into a summary. You can also create reports for specific types of events, such as a login report with the `--login` option, or an executable name report with the `--executable` option.

When you use the `ausearch --raw` command to search for specific events recorded in the Audit log files, you can provide the unformatted search results as input to the `aureport` command to generate formatted reports.

**Note**

Two specialized tools also exist: `aulast` and `aulastlog`. These tools replace the `last` and `lastlog` tools respectively, but they parse the Audit logs instead of the `/var/log/wtmp` and `/var/log/btmp` files.

Tracing a Program

To investigate the system calls that are performed by a process, you specify the process as an argument of the `autrace` command. The `autrace` command requires that you remove any custom Auditing rules. When execution finishes, the `autrace` command clears those rules, and then provides an example `ausearch` command to investigate those events.

This technique is useful for troubleshooting or investigating programs of interest.

The following example traces the `date` command:

```
[root@host ~]# autrace /bin/date
Waiting to execute: /bin/date
Thu Jul 31 11:38:46 CEST 2023
Cleaning up...
Trace complete. You can locate the records with 'ausearch -i -p 26472'
[root@host ~]# ausearch --raw -p 26472 | aureport --file -i

File Report
=====
# date time file syscall success exe auid event
=====
1. 07/31/2023 11:38:46 /bin/date execve yes /bin/date student 29158
2. 07/31/2023 11:38:46 /etc/ld.so.preload access no /bin/date student 29161
3. 07/31/2023 11:38:46 /etc/ld.so.cache open yes /bin/date student 29162
4. 07/31/2023 11:38:46 /lib64/librt.so.1 open yes /bin/date student 29166
5. 07/31/2023 11:38:46 /lib64/libc.so.6 open yes /bin/date student 29173
6. 07/31/2023 11:38:46 /lib64/libpthread.so.0 open yes /bin/date student 29181
7. 07/31/2023 11:38:46 /usr/lib/locale-archive open yes /bin/date student
29208
8. 07/31/2023 11:38:46 /etc/localtime open yes /bin/date student 29213
```

**Warning**

The `autrace` command requires you to remove any active rules before you run the command. This might cause you to miss events from other processes that those existing rules would record.

If the Audit rules are locked in place, then the `autrace` command cannot unload the existing rules and does not work.

**References**

`ausearch(8)`, `aureport(8)`, and `autrace(8)` man pages

For more information about Audit options, refer to the *RHEL Audit System Reference* article at

<https://access.redhat.com/articles/4409591>

For more information, refer to the *Auditing the System* chapter in the *Red Hat Enterprise Linux 9 Security Hardening* guide at

https://access.redhat.com/documentation/es-es/red_hat_enterprise_linux/9/html-single/security_hardening/index#auditing-the-system_security-hardening

► Guided Exercise

Inspecting Audit Logs

Search for events in your system, create reports for those events, and trace the execution of a command.

Outcomes

- Search for events and generate reports from the Audit log and interpret the results.

Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

```
[student@workstation ~]$ lab start audit-inspecting
```

Instructions

► 1. Generate a report of all login events on the `servera` machine.

- 1.1. Log in to `servera` as the student user. No password is required.

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

- 1.2. Use the `sudo -i` command to change to the `root` user. Use `student` as the password.

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

- 1.3. Generate a report of all login events by using the `aureport` command. Use the time stamps to determine the most recent login event from `workstation.lab.example.com` and take note of the event ID. In the following example, the event ID is 233, but your output might be different.

```
[root@servera ~]# aureport --login  
  
Login Report  
=====  
# date time auid host term exe success event  
=====  
...output omitted...  
615. 10/11/23 14:08:48 1000 172.25.250.9 /dev/pts/0 /usr/sbin/sshd yes 233
```

Chapter 6 | Recording System Events with Audit

- ▶ 2. Retrieve more information about the previous event by using the ausearch command. Use the `-i` option to render the results in a more human-readable format.

```
[root@servera ~]# ausearch -i -a 233
...output omitted...
-----
type=USER_LOGIN msg=audit(10/11/23 14:08:48.308:233) : pid=1202 uid=root
    auid=student ses=3 subj=system_u:system_r:sshd_t:s0-s0:c0.c1023 msg='op=login
    id=student exe=/usr/sbin/sshd hostname=? addr=172.25.250.9 terminal=/dev/pts/0
    res=success'
-----
...output omitted...
```

- ▶ 3. Use the aureport --summary command to generate an executable summary report of command executions. Remember that by default, Audit does not log all commands, but instead only the commands that trigger Audit events, such as the su and sudo commands. The report might look different on your system.

```
[root@servera ~]# aureport --executable --summary

Executable Summary Report
=====
total file
=====
498 /usr/sbin/crond
311 /usr/lib/systemd/systemd
243 /usr/sbin/sshd
112 /usr/sbin/iptables-multi
20 /usr/bin/kmod
20 /usr/bin/sudo
18 /usr/sbin/etables-restore
14 /usr/sbin/groupadd
8 /usr/sbin/useradd
6 /usr/lib/systemd/systemd-update-utmp
2 /usr/bin/passwd
```

- ▶ 4. Search for all Audit events of the LOGIN type, and export them in CSV format. Store the results in a `results.csv` file for future use.

```
[root@servera ~]# ausearch -m LOGIN --format csv > results.csv
[root@servera ~]# cat results.csv
NODE,EVENT,...,EVENT_KIND,SUBJ_SEC,SUBJ_KIND,ACTION,RESULT, ...
,LOGIN,...,user-login,...,root,privileged-acct,changed-login-id-to,success, ...
,LOGIN,...,user-login,...,root,privileged-acct,changed-login-id-to,success, ...
,LOGIN,...,user-login,...,root,privileged-acct,changed-login-id-to,success, ...
...output omitted...
```

- ▶ 5. Use the Audit system to trace and review the execution of the `/bin/ls /tmp` command. Create a file report for all files opened by the command.

- 5.1. Use the autrace `/bin/ls /tmp` command to trace the execution of the `/bin/ls /tmp` command.

```
[root@servera ~]# autrace /bin/ls /tmp
Waiting to execute: /bin/ls
NIC1      rht-bastion
NIC1.old   systemd-private-3c9eca035b4d449bb44a6ed0576f1305-
chronyrd.service-7LwrrU
NIC2      systemd-private-3c9eca035b4d449bb44a6ed0576f1305-dbus-broker.service-
PRd9UH
NIC2.old   systemd-private-3c9eca035b4d449bb44a6ed0576f1305-systemd-
logind.service-e1Ft7N
rclocal.log
Cleaning up...
Trace complete. You can locate the records with 'ausearch -i -p 1440'
```

- 5.2. Use the `ausearch -i -p 1440` command to review the records. Use the value that is provided in the output of the previous command.

```
[root@servera ~]# ausearch -i -p 1440
...output omitted...
-----
type=PROCTITLE msg=audit(10/11/23 14:14:52.183:418) : proctitle=autrace /bin/ls /
tmp
type=SYSCALL msg=audit(10/11/23 14:14:52.183:418) : arch=x86_64 syscall=write
success=yes exit=12 a0=0x1 a1=0x56532666b380 a2=0xc a3=0xb items=0 ppid=1438
pid=1440 auid=student uid=root gid=root euid=root suid=root fsuid=root
egid=root sgid=root fsgid=root tty=pts0 ses=8 comm=ls exe=/usr/bin/ls
subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 key=(null)
-----
type=PROCTITLE msg=audit(10/11/23 14:14:52.183:419) : proctitle=autrace /bin/ls /
tmp
type=SYSCALL msg=audit(10/11/23 14:14:52.183:419) : arch=x86_64 syscall=close
success=yes exit=0 a0=0x1 a1=0x56532666b380 a2=0x7f366a7f69e0 a3=0xb items=0
ppid=1438 pid=1440 auid=student uid=root gid=root euid=root suid=root
fsuid=root egid=root sgid=root fsgid=root tty=pts0 ses=8 comm=ls exe=/usr/bin/ls
subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 key=(null)
-----
type=PROCTITLE msg=audit(10/11/23 14:14:52.183:420) : proctitle=autrace /bin/ls /
tmp
type=SYSCALL msg=audit(10/11/23 14:14:52.183:420) : arch=x86_64 syscall=close
success=yes exit=0 a0=0x2 a1=0xfb00000000000006 a2=0x7f366a7f69e0 a3=0xb items=0
ppid=1438 pid=1440 auid=student uid=root gid=root euid=root suid=root
fsuid=root egid=root sgid=root fsgid=root tty=pts0 ses=8 comm=ls exe=/usr/bin/ls
subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 key=(null)
-----
type=PROCTITLE msg=audit(10/11/23 14:14:52.183:421) : proctitle=autrace /bin/ls /
tmp
type=SYSCALL msg=audit(10/11/23 14:14:52.183:421) : arch=x86_64 syscall=exit_group
a0=EXIT_SUCCESS a1=0xfffffffffffffe98 a2=0xe7 a3=0x7ff000000000 items=0
ppid=1438 pid=1440 auid=student uid=root gid=root euid=root suid=root
fsuid=root egid=root sgid=root fsgid=root tty=pts0 ses=8 comm=ls exe=/usr/bin/ls
subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 key=(null)
```

- 5.3. Use the `ausearch -p 1440 --raw` command to review the records in raw format. Use the same value from the previous command.

```
[root@servera ~]# ausearch -p 1440 --raw
...output omitted...
type=SYSCALL msg=audit(1697048092.183:418): arch=c000003e syscall=1 success=yes
exit=12 a0=1 a1=56532666b380 a2=c a3=b items=0 ppid=1438 pid=1440 auid=1000
uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0 sgid=0 fsgid=0 tty=pts0 ses=8 comm="ls"
exe="/usr/bin/ls" subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
key=(null)ARCH=x86_64 SYSCALL=write AUID="student" UID="root" GID="root"
EUID="root" SUID="root" FSUID="root" EGID="root" SGID="root" FSGID="root"
type=PROCTITLE msg=audit(1697048092.183:418):
proctitle=61757472616365002F62696E2F6C73002F746D70
type=SYSCALL msg=audit(1697048092.183:419): arch=c000003e syscall=3 success=yes
exit=0 a0=1 a1=56532666b380 a2=7f366a7f69e0 a3=b items=0 ppid=1438 pid=1440
auid=1000 uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0 sgid=0 fsgid=0 tty=pts0 ses=8
comm="ls" exe="/usr/bin/ls" subj=unconfined_u:unconfined_r:unconfined_t:s0-
s0:c0.c1023 key=(null)ARCH=x86_64 SYSCALL=close AUID="student" UID="root" GID="root"
EUID="root" SUID="root" FSUID="root" EGID="root" SGID="root" FSGID="root"
type=PROCTITLE msg=audit(1697048092.183:419):
proctitle=61757472616365002F62696E2F6C73002F746D70
type=SYSCALL msg=audit(1697048092.183:420): arch=c000003e syscall=3 success=yes
exit=0 a0=2 a1=fbad2006 a2=7f366a7f69e0 a3=b items=0 ppid=1438 pid=1440 auid=1000
uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0 sgid=0 fsgid=0 tty=pts0 ses=8 comm="ls"
exe="/usr/bin/ls" subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
key=(null)ARCH=x86_64 SYSCALL=close AUID="student" UID="root" GID="root"
EUID="root" SUID="root" FSUID="root" EGID="root" SGID="root" FSGID="root"
type=PROCTITLE msg=audit(1697048092.183:420):
proctitle=61757472616365002F62696E2F6C73002F746D70
type=SYSCALL msg=audit(1697048092.183:421): arch=c000003e syscall=231 a0=0
a1=fffffffffffffe98 a2=e7 a3=7ffcdd36a200 items=0 ppid=1438 pid=1440 auid=1000
uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0 sgid=0 fsgid=0 tty=pts0 ses=8 comm="ls"
exe="/usr/bin/ls" subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
key=(null)ARCH=x86_64 SYSCALL=exit_group AUID="student" UID="root" GID="root"
EUID="root" SUID="root" FSUID="root" EGID="root" SGID="root" FSGID="root"
type=PROCTITLE msg=audit(1697048092.183:421):
proctitle=61757472616365002F62696E2F6C73002F746D70
```

- 5.4. Use the `ausearch -p 1440 --raw | aureport -i --file` command to create a report file. Use the same value from the previous command.

```
[root@servera ~]# ausearch -p 1440 --raw | aureport -i --file

File Report
=====
# date time file syscall success exe auid event
=====
1. 10/11/23 14:14:51 newfstatat yes /usr/sbin/autrace student 251
2. 10/11/23 14:14:52 /bin/ls execve yes /usr/bin/ls student 255
3. 10/11/23 14:14:52 /etc/ld.so.preload access no /usr/bin/ls student 259
4. 10/11/23 14:14:52 /etc/ld.so.cache openat yes /usr/bin/ls student 260
5. 10/11/23 14:14:52 newfstatat yes /usr/bin/ls student 261
...output omitted...
```

Chapter 6 | Recording System Events with Audit

- 6. Repeat the previous trace, but now with the `/bin/ls -l /tmp` command. Note that the `-l` option triggers a call to the `statx` system call, which provides additional detailed information, such as permissions or size.

6.1. Use the `autrace /bin/ls -l /tmp` command to perform the trace.

```
[root@servera ~]# autrace /bin/ls -l /tmp
Waiting to execute: /bin/ls
total 24
-rw-r--r--. 1 root root 5 Oct 11 06:16 NIC1
-rw-r--r--. 1 root root 5 Sep 29 03:25 NIC1.old
-rw-r--r--. 1 root root 5 Oct 11 06:16 NIC2
-rw-r--r--. 1 root root 5 Sep 29 03:25 NIC2.old
-rw-r--r--. 1 root root 73 Oct 11 06:22 rclocal.log
-rw-r--r--. 1 root root 181 Oct 11 06:22 rht-bastion
drwx-----. 3 root root 17 Oct 11 10:17 systemd-
private-3c9eca035b4d449bb44a6ed0576f1305-chronyd.service-7LwrrU
drwx-----. 3 root root 17 Oct 11 10:17 systemd-
private-3c9eca035b4d449bb44a6ed0576f1305-dbus-broker.service-PRd9UH
drwx-----. 3 root root 17 Oct 11 10:17 systemd-
private-3c9eca035b4d449bb44a6ed0576f1305-systemd-logind.service-e1Ft7N
Cleaning up...
Trace complete. You can locate the records with 'ausearch -i -p 1452'
```

6.2. Use the `ausearch -i -p 1452` command to view the records from the previous `autrace` command. The output includes the `statx` syscall.

```
[root@servera ~]# ausearch -i -p 1452
...output omitted...
type=SYSCALL msg=audit(11/07/23 14:47:13.885:406) : arch=x86_64 syscall=getxattr
success=no exit=ENODATA(No data available) a0=0x7ffd5cb410c0 a1=0x5630e48c2b70
a2=0x0 a3=0x0 items=1 ppid=1275 pid=1277 auid=student uid=root gid=root euid=root
suid=root fsuid=root egid=root sgid=root fsgid=root tty=pts0 ses=3 comm=ls exe=/
usr/bin/ls subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 key=(null)
-----
type=PROCTITLE msg=audit(11/07/23 14:47:13.885:407) : proctitle=autrace /bin/ls -
l /tmp
type=PATH msg=audit(11/07/23 14:47:13.885:407) : item=0 name=/tmp/systemd-
private-16442968a22c44ef99b299991d70388e-systemd-logind.service-NcCBdG
inode=16797908 dev=fc:04 mode=dir,700 ouid=root ogid=root rdev=00:00
obj=system_u:object_r:tmp_t:s0 nametype=NORMAL cap_fp=none cap_fi=none cap_fe=0
cap_fver=0 cap_frootid=0
type=CWD msg=audit(11/07/23 14:47:13.885:407) : cwd=/root
type=SYSCALL msg=audit(11/07/23 14:47:13.885:407) : arch=x86_64 syscall=statx
success=yes exit=0 a0=0xffffffff9c a1=0x7ffd5cb410c0 a2=0x900 a3=0x25e items=1
ppid=1275 pid=1277 auid=student uid=root gid=root euid=root suid=root
fsuid=root egid=root sgid=root fsgid=root tty=pts0 ses=3 comm=ls exe=/usr/bin/ls
subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 key=(null)
-----
type=PROCTITLE msg=audit(11/07/23 14:47:13.885:408) : proctitle=autrace /bin/ls -
l /tmp
```

Chapter 6 | Recording System Events with Audit

```
type=PATH msg=audit(11/07/23 14:47:13.885:408) : item=0 name=/tmp/systemd-private-16442968a22c44ef99b299991d70388e-systemd-logind.service-NcCBdG  
inode=16797908 dev=fc:04 mode=dir,700 uid=root ogid=root rdev=00:00  
obj=system_u:object_r:tmp_t:s0 nametype=NORMAL cap_fp=none cap_fi=none cap_fe=0  
cap_fver=0 cap_frootid=0  
type=CWD msg=audit(11/07/23 14:47:13.885:408) : cwd=/root  
type=SYSCALL msg=audit(11/07/23 14:47:13.885:408) : arch=x86_64 syscall=lgetxattr  
success=yes exit=27 a0=0x7ffd5cb410c0 a1=0x7f34d288d23c a2=0x5630e54aa550 a3=0xff  
items=1 ppid=1275 pid=1277 auid=student uid=root gid=root euid=root suid=root  
fsuid=root egid=root sgid=root fsgid=root tty pts0 ses=3 comm=ls exe=/usr/bin/ls  
subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 key=(null)  
...output omitted...
```

6.3. When done, return to the workstation machine.

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

Finish

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

```
[student@workstation ~]$ lab finish audit-inspecting
```

Writing Custom Audit Rules

Objectives

- Write your own Audit rules to configure the system to collect information about particular events.

Customizing Audit Rules

One of the powerful features of the Linux Audit system is the ability to add your own Auditing rules. You can create rules to monitor system call usage and file access, or to configure Auditing rules for compliance with industry standard security certifications. You can also gain a better insight into how users and processes are using your system.

Adding Rules

You can use the `auditctl` command to add Auditing rules from the command line. By default, this command adds Auditing rules to the bottom of the current list, but you can insert rules to the start of the list as well.



Important

The order of rules is important. The first Audit rule that matches is the one that Audit applies. Audit does not apply any subsequent matching rules.

For example, you can configure two rules with separate keys, one to log all access to the `/etc/` directory, and another to log all access to the `/etc/sysconfig/` directory. Then, any access to the `/etc/sysconfig/` directory does not trigger the second rule. The second rule is not triggered because the first rule matches as an event for every operation in the `/etc/` directory and its subdirectories; the second rule is never reached.

If you reverse the order of these two rules and then access the `/etc/sysconfig/` directory, the rule that logs access to the `/etc/sysconfig/` directory matches the event. Operations on the `/etc/` directory that are needed to access the `/etc/sysconfig/` directory might match the second rule as separate events (for example, to look up the `/etc/sysconfig` inode in the `/etc/` directory). In this case, the operations on the `/etc/` directory itself are outside the scope of the `/etc/sysconfig/` rule and do not match it.

The Audit system uses three types of rules:

- *File system* (or *watch*) rules monitor access to files and directories.
- *System call* rules monitor the execution of system calls that are made by processes that communicate with the kernel to access system resources.
- *Control* rules configure the Audit system itself.

Setting File System Watch Rules

One type of rule that you can add is a *watch* rule. You can set watch rules on files and directories, and you can configure watch rules to trigger on specific types of access (read, write, attribute change, and execute). The *open* system call on a file that is monitored by a watch rule triggers the system to generate an Audit event based on the rule and the type of access that is requested. However, individual *read* and *write* calls on a watched file that is already open do not trigger the watch rule.

The basic syntax of a file-system rule is as follows:

```
[root@host ~]# auditctl -w file -p permissions -k key
```

The *-w* option takes the name of a file or directory to watch. If the path is a directory, then the rule matches recursively all contents and subdirectories in that directory *excluding subdirectories that are mount points*. The rule does not cross file systems.

The *-p* option takes a list of accesses to monitor by permission type.

- *r* for read access
- *w* for write access
- *x* for execute access
- *a* for changes to attributes

The *-k* option takes a key to set on the Audit record to make it easier to find with a specific *ausearch* query.

The following examples demonstrate some custom watch rules:

- Watch the */etc/passwd* file for write and attribute change access. Label log messages with the *user-edit* key.

```
[root@host ~]# auditctl -w /etc/passwd -p wa -k user-edit
```

- Add a recursive watch rule for the */etc/sysconfig/* directory and all its files and subdirectories. Watch for read, write, and attribute change access. Label log messages with the *sysconfig-access* key.

```
[root@host ~]# auditctl -w /etc/sysconfig/ -p rwa -k sysconfig-access
```

- Audit all executions of binaries in the */bin/* directory.

```
[root@host ~]# auditctl -w /bin/ -p x
```



Important

Remember that a watch rule does not cross file-system boundaries. If you set a watch on the */* directory, and the */tmp* directory is a mount point that has a separate file system mounted on it, then the rule does not apply to the contents of the */tmp* directory.

Setting System Call Rules

System call rules are more complex. The syntax of a system call rule is as follows:

```
[root@host ~]# auditctl -a <list>,<action> \
    [-F <filter-expression>]... \
    [-C <compare-expression>]... \
    [-S <system-call>]...
```

Rules are set on one of four lists:

- The *exit* list evaluates all system calls when they exit, and is the most commonly used.
- The *user* list evaluates events that originate in user space.
- The *exclude* list is sometimes used to filter events entirely from being logged (although there are other ways to do this).
- The *task* list is rarely used and is only checked during the `fork(2)` and `clone(2)` system calls.

If you replace the `-a` option with the `-A` option, then Audit inserts the rule at the start of the specified list, instead of at the end.

The action is either *always* (to always record an event), or *never* (to never record that event).



Note

Usually, use the `<list>,<action>` syntax (for example, `exit,always`), to always record the event when the system call exits.

You can filter events with the `-F`, `-C`, and `-S` options. The `-F` option enables you to specify a filter based on Audit record fields, such as the `auid`, `arch`, and `exit` fields. By using the `-C` option, you can compare fields, for example the Audit UID of the calling process and the `obj_uid` (owner) field of the file that is accessed. Finally, you can filter based on a system call with the `-S` option.



Note

On 64-bit systems, write rules to specify a system call by name two times; first, with the `-F arch=b32` option, and second, with the `-F arch=b64` option. This approach is necessary because every system call does not translate into the same number for the same name on 32- and 64-bit architectures, and 64-bit systems do provide the 32-bit system calls that applications might use.

The following examples show custom rules for specific circumstances:

- Audit the 32-bit version of both the `rename` and `renameat` system call for all users whose original Audit user ID is equal to or greater than the `1000` ID. Do not trigger the Audit rule if the process is under the `mysqld_t` SELinux domain, and add the `rename` key to the logs.

```
[root@host ~]# auditctl -a exit,always -F arch=b32 -F auid>=1000 -S rename \
    -S renameat -F subj_type!=mysqld_t -k rename
```

- Recursively audit every file-system access by the `root` user under the `/home/` directory to files or directories that are not owned by the original user, and who is now working as the `root` user.

```
[root@host ~]# auditctl -a exit,always -F dir=/home/ -F uid=0 -C auid!=obj_uid
```

Chapter 6 | Recording System Events with Audit

- Disable auditing of failed USER_LOGIN events for the example user.

```
[root@host ~]# auditctl -a exclude,never -F auid=example \
-F msgtype=USER_LOGIN -F success=0
```

When Audit starts, it assigns the 4294967295 Audit UID to any existing process. If you want to exclude those processes from your rules, then you can add the -F auid!=4294967295 option to your rules.



Important

System call rules are checked for every system call that is issued on the system. This can reduce system performance. Therefore, try to keep the list of system call rules short and general.

Setting Control Rules

The final class of Audit rule is the control rule for modifying the kernel configuration of Linux Audit. These rules tend to be short and are usually set at the start of the rules list, except for the -e 2 rule to prevent further changes to the Audit rule set.

Common control rules include the following examples:

- Remove all rules.

```
[root@host ~]# auditctl -D
```

- Remove a file-system rule that monitors the /bin/ directory. The path (/bin/) must match the rule exactly.

```
[root@host ~]# auditctl -W /bin/
```

- Set the currently loaded rules to be *immutable*. This rule means that the rules cannot be changed again until the system is rebooted.

```
[root@host ~]# auditctl -e 2
```

Removing Rules

You can remove file system watch rules with the -W option. You must use the -d <list>,<action> option to remove rules added with the -a or the -A options. The -d <list>,<action> syntax requires that the syscall name and every field and value match the rule that is being removed.

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

Inspecting Rules

You can inspect your current rule set with the auditctl -l command. To review the current status of Audit, you can use the auditctl -s command. This command also displays information on the internal buffer sizes to store Audit messages, as well as any rate limiting that is active.

Making Rules Immutable

With the `auditctl -e [0|1|2]` command, you can influence the current auditing mode. Use the `auditctl -e -0` command to disable auditing. Use the `auditctl -e 1` command to enable auditing again. Use the `auditctl -e 2` to configure your rules as immutable, so that you can no longer add, remove, or change rules, or stop the Audit subsystem. To be able to make changes again, you must reboot the entire system.



Important

If you set the `-e 2` rule in a rules file, then it must be the last rule loaded because subsequent attempts to set rules are not allowed after the `-e 2` rule has been set.

Persistent Rules

To make your Audit rules persistent, you can add them to the `/etc/audit/rules.d/audit.rules` file, or to any other file in the `/etc/audit/rules.d/` directory as a file with the suffix `.rules`. Rule files contain `auditctl` commands as you type them on the command line, but without the `auditctl` command itself. You can add empty lines and comments to this file. To include a comment, start the line with a number sign (#).

When the `auditd` daemon starts, it activates all rules from the `/etc/audit/rules.d/audit.rules` file, and the other `.rules` files in the `/etc/audit/rules.d` directory. You can also use the `auditctl -R filename` command to read rules from a file.



References

`auditctl(8)`, `audit.rules(7)`, and `intro(2)` man pages

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/9/html-single/security_hardening/index#auditing-the-system_security-hardening

► Guided Exercise

Writing Custom Audit Rules

Write your own Audit rules to configure the system to collect information about particular events.

Outcomes

- Write custom Audit rules.

Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

```
[student@workstation ~]$ lab start audit-custom
```

Instructions

- 1. On the `servera` machine, add a temporary Audit rule that logs every write or attribute change to any file in the `/etc/` directory. Add the `config-change` key to all log messages.

1.1. Log in to the `servera` machine as the `student` user. No password is required.

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

1.2. Use the `sudo -i` command to change to the `root` user. Use `student` as the password.

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

1.3. Use the `auditctl` command to add a temporary Audit rule that logs every write or attribute change to any file in the `/etc/` directory. Add the `config-change` key to all log messages.

```
[root@servera ~]# auditctl -w /etc/ -p wa -k config-change
```

1.4. Create an empty `/etc/sysconfig/testfile` file. When done, check your Audit logs for all of today's entries with the `config-change` key.

```
[root@servera ~]# touch /etc/sysconfig/testfile  
[root@servera ~]# ausearch --start today -k config-change  
...output omitted...  
time->Thu Oct 12 13:03:50 2023
```

```

type=PROCTITLE msg=audit(1697130230.185:112):
  proctitle=746F756368002F6574632F737973636F6E6669672F7465737466756C65
type=PATH msg=audit(1697130230.185:112): item=1 name="/etc/sysconfig/
testfile" inode=16802086 dev=fc:04 mode=0100644 uid=0 ogid=0 rdev=00:00
  obj=unconfined_u:object_r:etc_t:s0 nametype=CREATE cap_fp=0 cap_hi=0 cap_fe=0
  cap_fver=0 cap_frootid=0
type=PATH msg=audit(1697130230.185:112): item=0 name="/etc/sysconfig/"
  inode=16797929 dev=fc:04 mode=040755 uid=0 ogid=0 rdev=00:00
  obj=system_u:object_r:etc_t:s0 nametype=PARENT cap_fp=0 cap_hi=0 cap_fe=0
  cap_fver=0 cap_frootid=0
type=CWD msg=audit(1697130230.185:112): cwd="/root"
type=SYSCALL msg=audit(1697130230.185:112): arch=c000003e syscall=257
  success=yes exit=3 a0=fffffff9c a1=7ffed10196a9 a2=941 a3=1b6 items=2
  ppid=1175 pid=1204 auid=0 uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0
  sgid=0 fsgid=0 tty=pts0 ses=1 comm="touch" exe="/usr/bin/touch"
  subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 key="config-change"
...output omitted...

```

- ▶ 2. Add a temporary Audit rule that logs any file execution in the /bin/ directory by users with an Audit UID parameter of the 1000 value or higher, or an effective UID parameter of the 0 value. Use the `privileged-execution` key.
- 2.1. Use the `auditctl` command to create the temporary Audit rule. Log these Audit messages with the `privileged-execution` key.

```
[root@servera ~]# auditctl -a exit,always -F dir=/bin/ -F "auid>=1000" \
  -F "auid!=1" -F "euid=0" -p x -k privileged-execution
```

- 2.2. Execute the /bin/true command. When done, run an Audit search on all of this week's entries with the `privileged-execution` key.

```

[root@servera ~]# /bin/true
[root@servera ~]# ausearch --start this-week -i \
  -k privileged-execution
...output omitted...
-----
type=PROCTITLE msg=audit(10/12/23 14:00:12.826:113) : proctitle=/bin/true
type=PATH msg=audit(10/12/23 14:00:12.826:113) : item=1 name=/lib64/ld-linux-
x86-64.so.2 inode=8389810 dev=fc:04 mode=file,755 uid=root ogid=root rdev=00:00
  obj=system_u:object_r:ld_so_t:s0 nametype=NORMAL cap_fp=none cap_hi=none cap_fe=0
  cap_fver=0 cap_frootid=0
type=PATH msg=audit(10/12/23 14:00:12.826:113) : item=0 name=/bin/true
  inode=1241 dev=fc:04 mode=file,755 uid=root ogid=root rdev=00:00
  obj=system_u:object_r:bin_t:s0 nametype=NORMAL cap_fp=none cap_hi=none cap_fe=0
  cap_fver=0 cap_frootid=0
type=CWD msg=audit(10/12/23 14:00:12.826:113) : cwd=/root
type=EXECVE msg=audit(10/12/23 14:00:12.826:113) : argc=1 a0=/bin/true
type=SYSCALL msg=audit(10/12/23 14:00:12.826:113) : arch=x86_64 syscall=execve
  success=yes exit=0 a0=0x5595f85a5d20 a1=0x5595f8554e80 a2=0x5595f85a6550 a3=0x8
  items=2 ppid=1186 pid=1214 auid=student uid=root gid=root euid=root suid=root
  fsuid=root egid=root sgid=root fsgid=root tty=pts0 ses=1 comm=true exe=/usr/bin/
  true subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 key=privileged-
  execution

```

Chapter 6 | Recording System Events with Audit

- 3. Persistently add an Audit rule to your system that audits all system calls for all users with an Audit UID parameter of the 1000 value or higher. Give these rules the `delete` identifier key.

- `unlink`
- `unlinkat`
- `rename`
- `renameat`
- `rmdir`

- 3.1. Add the following line to the `/etc/audit/rules.d/audit.rules` file. The rule in the example does not specify the architecture for simplicity.

```
-a exit,always -S unlink -S unlinkat -S rename -S renameat -S rmdir -F auid>=1000  
-k delete
```

- 3.2. Regenerate the Audit rules by using the `augenrules --load` command.

```
[root@servera ~]# augenrules --load  
No rules  
...output omitted...
```

- 3.3. As the `student` user, create and then immediately delete an empty `/tmp/testfile` file.

```
[root@servera ~]# logout  
[student@servera ~]$ touch /tmp/testfile; rm /tmp/testfile
```

- 3.4. As the `root` user, search for all Audit messages with the `delete` key and the path name `/tmp/testfile` for this year.

```
[student@servera ~]$ sudo -i  
[sudo] password for student: student  
[root@servera ~]# ausearch --start this-year -i -k delete \  
-f /tmp/testfile  
+---+  
type=PROCTITLE msg=audit(10/12/23 14:13:39.273:208) : proctitle=rm /tmp/testfile  
type=PATH msg=audit(10/12/23 14:13:39.273:208) : item=1 name=/tmp/testfile  
inode=16802086 dev=fc:04 mode=file,644 uid=student ogid=student rdev=00:00  
obj=unconfined_u:object_r:user_tmp_t:s0 nametype=DELETE cap_fp=none cap_fi=none  
cap_fe=0 cap_fver=0 cap_frootid=0  
type=PATH msg=audit(10/12/23 14:13:39.273:208) : item=0 name=/tmp/  
inode=16798505 dev=fc:04 mode=dir,sticky,777 uid=root ogid=root rdev=00:00  
obj=system_u:object_r:tmp_t:s0 nametype=PARENT cap_fp=none cap_fi=none cap_fe=0  
cap_fver=0 cap_frootid=0  
type=CWD msg=audit(10/12/23 14:13:39.273:208) : cwd=/home/student
```

Chapter 6 | Recording System Events with Audit

```
type=SYSCALL msg=audit(10/12/23 14:13:39.273:208) : arch=x86_64 syscall=unlinkat  
success=yes exit=0 a0=AT_FDCWD a1=0x55f742fa8c20 a2=0x0 a3=0x200 items=2  
ppid=1160 pid=1335 uid=student uid=student gid=student euid=student suid=student  
fsuid=student egid=student sgid=student fsgid=student tty pts0 ses=1 comm=rm  
exe=/usr/bin/rm subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023  
key=delete  
+----+  
...output omitted...
```

3.5. When done, log out of the servera machine.

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

Finish

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

```
[student@workstation ~]$ lab finish audit-custom
```

Enabling Prepackaged Audit Rule Sets

Objectives

- Enable standard Audit rule sets that are provided with Red Hat Enterprise Linux, and identify potentially useful rule sets.

Sample Audit Rule Sets

The audit package comes with a set of sample Audit rule sets. This rule set is a good starting point for a practical security policy implementation. These rules are available in the `/usr/share/audit/sample-rules` directory as files with the suffix `.rules`. The `/usr/share/audit/sample-rules/README.rules` file provides basic instructions for using the rules, and the individual rule files contain comments about how they should be used and whether additional rule files should also be loaded.

Each name of a rule file starts with a number to help ensure that the rules are loaded in the correct order. Remember that order is important when loading rules.

The following rule files are included:

- `30-nispom.rules`, which is intended to meet the requirements of the Information System Security chapter of the *National Industrial Security Program Operating Manual*.
- `30-pci-dss-v31.rules`, which is intended to meet the requirements set by the Payment Card Industry Data Security Standard (PCI DSS) v3.1.
- `30-stig.rules`, which is intended to meet the requirements set by the US Department of Defense Security Technical Implementation Guides (STIGs).

Each rule file contains several individual rules. A number sign (#) marks the start of a comment. Only the uncommented rules are enabled for loading. You must not enable all the rules in the file at the same time because some rules might conflict with others.

Enabling Sample Rule Sets

To use one of these sample rule sets, copy the `.rules` file or files to the `/etc/audit/rules.d` directory, and run the `augenrules --load` command to reload the Audit rules. These example files do not guarantee full compliance as written, but the files do give you a starting point to configure your environment.

After copying one of these default rule sets, you must review the file and follow any instructions to enable or disable certain rules for your environment. For example, the `99-finalize.rules` file contains a commented-out control rule to make the rule configuration immutable. You need to enable that control rule for production.

Full Terminal Keystroke Logging

Some auditing policies require that every keystroke a user makes is logged. Audit provides this functionality with the `pam_tty_audit` PAM module. Every keystroke is then recorded in the Audit log (`/var/log/audit/audit.log`).

To enable keystroke logging, you must add the `pam_tty_audit` module to the `/etc/pam.d/system-auth` and the `/etc/pam.d/password-auth` files. This configuration ensures that all daemons started by the system that implement some form of terminal functionality have their keystrokes logged as well, unless explicitly disabled in their PAM configuration.

**Note**

The `pam_tty_audit.so` module only implements session functionality. Adding the module to any other section in PAM prevents any user from logging in at all.

The `pam_tty_audit` module takes either the `enable` or the `disable` options. Both options take as arguments a comma-separated list of patterns for user names to enable and disable, respectively. The following example enables keystroke logging for the `demo` user, and disables it for all other users. Note that you must create a custom `authselect` profile to enable the module.

```
[root@host ~]# authselect create-profile minimal-with-tty-audit \
    -b minimal --symlink-meta --symlink-pam
...output omitted...
[root@host ~]# echo "session required pam_tty_audit.so disable=student
enable=devops log_passwd" \
    >> /etc/authselect/custom/minimal-with-tty-audit/system-auth
[root@host ~]# echo "session required pam_tty_audit.so disable=student
enable=devops log_passwd" \
    >> /etc/authselect/custom/minimal-with-tty-audit/password-auth
[root@host ~]# authselect select custom/minimal-with-tty-audit --force
```

If both an `enable=` pattern and a `disable=` pattern match a user, the last one on the command line is the pattern that applies.

To convert the data logged in the Audit system to a more readable format, you can use the `aureport --tty` command.

**Important**

Keystroke logging can require a large amount of storage on the system. You must consider this issue before enabling this functionality.

In addition, there might be certain legal restrictions or requirements on the use of keystroke logging in your location or the location of your data centers and users. You must discuss these questions with your legal counsel before implementing keystroke logging.

**References**

`pam_tty_audit(8)` man page

For more information, refer to the *Pre-configured Audit Rules Files for Compliance with Standards* section in the *Auditing the System* chapter in the *Red Hat Enterprise Linux 9 Security Hardening* guide at

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html/security_hardening/auditing-the-system_security-hardening#pre-configured-audit-rules-files-for-compliance-with-standards_auditing-the-system

► Guided Exercise

Enabling Prepackaged Audit Rule Sets

Configure prepackaged Audit rules, and configure the Audit subsystem to log terminal activity.

Outcomes

- Enable prepackaged Audit rules.
- Audit TTY with the `pam_tty_audit` PAM module.

Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

```
[student@workstation ~]$ lab start audit-rulesets
```

Instructions

► 1. Enable the prepackaged STIG Audit rules on the `servera` machine.

- 1.1. Log in to the `servera` machine as the `student` user. No password is required.

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

- 1.2. Use the `sudo -i` command to switch identity to the `root` user. Use `student` as the password.

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

- 1.3. Copy the `/usr/share/audit/sample-rules/30-stig.rules` file with the STIG Audit rules to the `/etc/audit/rules.d/` directory.

```
[root@servera ~]# cp /usr/share/audit/sample-rules/30-stig.rules \  
/etc/audit/rules.d/
```

- 1.4. Load the STIG Audit rules with the `augenrules --load` command.

```
[root@servera ~]# augenrules --load  
...output omitted...
```

► 2. Verify that the STIG Audit rules are working correctly.

- 2.1. Find the STIG Audit rules that use the `identity` key.

```
[root@servera ~]# grep identity /etc/audit/rules.d/30-stig.rules
-w /etc/group -p wa -k identity
-w /etc/passwd -p wa -k identity
-w /etc/gshadow -p wa -k identity
-w /etc/shadow -p wa -k identity
-w /etc/security/opasswd -p wa -k identity
```

- 2.2. Create a user called `testuser` to test the previous STIG Audit rules. Creating a new user modifies the files that are associated with the rules that use the `identity` key (for example, `/etc/passwd`), and triggers the STIG Audit rules.

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

- 2.3. Search the Audit log for the `identity` key to verify that the previous STIG Audit rules are active.

```
[root@servera ~]# ausearch -k identity
...output omitted...
type=PATH msg=audit(1697528559.373:1784): item=0 name="/etc/
passwd" inode=8622253 dev=fc:04 mode=0100644 ouid=0 ogid=0
rdev=00:00 obj=system_u:object_r:passwd_file_t:s0 nametype=NORMAL cap_fp=0
cap_hi=0 cap_fe=0 cap_fver=0 cap_frootid=0
...output omitted...
type=PATH msg=audit(1697528559.396:1787): item=0 name="/etc/shadow" inode=8391348
dev=fc:04 mode=0100000 ouid=0 ogid=0 rdev=00:00 obj=system_u:object_r:shadow_t:s0
nametype=NORMAL cap_fp=0 cap_hi=0 cap_fe=0 cap_fver=0 cap_frootid=0
...output omitted...
type=SYSCALL msg=audit(1697528559.441:1793): arch=c000003e syscall=82
success=yes exit=0 a0=7ffc0f270c00 a1=56423929d880 a2=7ffc0f270b70 a3=100
items=5 ppid=3303 pid=3351 auid=0 uid=0 gid=0 euid=0 suid=0 egid=0
sgid=0 fsgid=0 tty pts=63 comm="useradd" exe="/usr/sbin/useradd"
subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 key="identity"
...output omitted...
```

If you find Audit events with the correct key, then you have successfully configured the prepackaged Audit rules.

- 3. Now that you have configured prepackaged Audit rules, you will configure keystroke logging.

Configure the `pam_tty_audit` PAM module to enable auditing of TTY for the `student` user on the `servera` machine.

- 3.1. Create a custom `authselect` profile that adds the `pam_tty_audit` module to the PAM configuration.

```
[root@servera ~]# authselect create-profile minimal-with-tty-audit \
    -b minimal --symlink-meta --symlink-pam
...output omitted...
[root@servera ~]# echo "session required pam_tty_audit.so disable=student" \
    " enable=devops log_passwd" \
    >> /etc/authselect/custom/minimal-with-tty-audit/system-auth
[root@servera ~]# echo "session required pam_tty_audit.so disable=student" \
    " enable=devops log_passwd" \
    >> /etc/authselect/custom/minimal-with-tty-audit/password-auth
```

3.2. Enable the custom authselect profile.

```
[root@servera ~]# authselect select custom/minimal-with-tty-audit --force
```

- 3.3. Log in to the servera machine as the devops user and run the ls /tmp command to test that the auditing of TTY is working. When done, log off and then log in to the servera machine as the root user.



Note

Do not use tab completion to type the following commands. Typing the commands ensures that the actual keystrokes to enter those commands are logged for the ls /tmp and logout commands, instead of displaying the first few characters and a Tab character.

This example demonstrates something to keep in mind about keystroke logging. The keystrokes that are typed are logged, but you might need to reconstruct the effect of those keystrokes, based on the programs that are used.

```
[root@servera ~]# logout
[student@servera ~]$ logout
[student@workstation ~]$ ssh devops@servera
[devops@servera ~]$ ls /tmp
...output omitted...
[devops@servera ~]$ logout
[student@workstation ~]$ ssh root@servera
```

- 3.4. Verify the Audit logs for the previous commands with the aureport --tty command. When done, log off from the servera machine.

```
[root@servera ~]# aureport --tty
TTY Report
=====
# date time event auid term sess comm data
=====
1. 10/17/23 04:00:15 1925 1001 pts0 69 bash "ls /tmp",<ret>,"exit",<ret>
[root@servera ~]# logout
[student@workstation ~]$
```

Finish

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

```
[student@workstation ~]$ lab finish audit-rulesets
```

▶ Lab

Recording System Events with Audit

Configure remote Audit logs, enable prepackaged STIG Audit rules, and record terminal activity.

Outcomes

- Configure remote Audit logs.
- Enable prepackaged Audit rules.
- Enable auditing of TTY.

Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

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

Instructions

- Configure the Audit service on the `servera` machine to send Audit messages to the Audit service on the `serverb.lab.example.com` host. After configuration, use the `service auditd restart` command to restart the daemon and load the new configuration.



Note

The `service` command is deprecated; do not use it production environments. The `systemctl restart` command cannot be used with the `auditd` service due to the interaction between the daemon and the Linux kernel. In production environments, reboot the machine to ensure that the new configuration is loaded.

- Configure the Audit service on the `serverb` machine to accept the Audit messages from the Audit service on the `servera` machine. After configuration, use the `service auditd restart` command to restart the daemon and load the new configuration.
- Log in to the `servera` machine as the `student` user to verify that remote logging for Audit is working. Use the `auditctl -m` command to send the Audit message: `This is a test message from servera.`
- Enable the prepackaged STIG Audit rules on the `servera` machine.
- On the `servera` machine, some STIG Audit rules mark events that affect identity, user, or group information with the `identity` key string. Verify that those rules work correctly by creating the `labuser` user and confirming that Audit creates records marked with the `identity` key string.
- Create and select the `custom/lab-minimal-with-tty-audit` security profile based on the `minimal` security profile to enable the `pam_tty_audit` PAM module for the `student` user.

7. Return to the **workstation** machine as the **student** user.

```
[root@servera ~]# logout  
[student@servera ~]$ logout  
Connection to servera closed.  
[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 audit-review
```

Finish

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

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

► Solution

Recording System Events with Audit

Configure remote Audit logs, enable prepackaged STIG Audit rules, and record terminal activity.

Outcomes

- Configure remote Audit logs.
- Enable prepackaged Audit rules.
- Enable auditing of TTY.

Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

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

Instructions

- Configure the Audit service on the `servera` machine to send Audit messages to the Audit service on the `serverb.lab.example.com` host. After configuration, use the `service auditd restart` command to restart the daemon and load the new configuration.



Note

The `service` command is deprecated; do not use it in production environments. The `systemctl restart` command cannot be used with the `auditd` service due to the interaction between the daemon and the Linux kernel. In production environments, reboot the machine to ensure that the new configuration is loaded.

- Log in to the `servera` machine as the student user.

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

- Change to the `root` user. Use `student` as the password.

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

- Install the `audispd-plugins` package.

```
[root@servera ~]# dnf -y install audispd-plugins  
...output omitted...
```

- 1.4. In the `/etc/audit/plugins.d/au-remote.conf` file, set the value for the `active` variable to `yes` to enable remote logging.

```
[root@servera ~]# cat /etc/audit/plugins.d/au-remote.conf  
...output omitted...  
active = yes  
...output omitted...
```

- 1.5. In the `/etc/audit/audisp-remote.conf` file, set the `remote_server` variable to the `serverb.lab.example.com` hostname. Also, set the port to be used in the remote logging server, which is `60` by default.

```
[root@servera ~]# cat /etc/audit/audisp-remote.conf  
...output omitted...  
remote_server = serverb.lab.example.com  
port = 60  
...output omitted...
```

- 1.6. Restart the `auditd` service to update its configuration. When done, return to the `workstation` machine as the `student` user.

```
[root@servera ~]# service auditd restart  
Stopping logging:  
Redirecting start to /bin/systemctl start auditd.service  
[root@servera ~]# logout  
[student@servera ~]$ logout  
[student@workstation ~]$
```

2. Configure the Audit service on the `serverb` machine to accept the Audit messages from the Audit service on the `servera` machine. After configuration, use the `service auditd restart` command to restart the daemon and load the new configuration.

- 2.1. Log in to the `serverb` machine as the `student` user.

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

- 2.2. Change to the `root` user. Use `student` as the password.

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

- 2.3. In the `/etc/audit/auditd.conf` file, uncomment the `tcp_listen_port` variable, and set its value to `60` so that the Audit service listens on the `60` TCP port.

```
[root@serverb ~]# vim /etc/audit/auditd.conf  
...output omitted...  
tcp_listen_port = 60  
...output omitted...
```

- 2.4. Open the 60 TCP port to enable access to the Audit server.

```
[root@serverb ~]# firewall-cmd --zone=public --add-port=60/tcp \  
--permanent  
success  
[root@serverb ~]# firewall-cmd --reload  
success
```

- 2.5. Restart the `auditd` service to update its configuration. When done, return to the `workstation` machine as the `student` user.

```
[root@serverb ~]# service auditd restart  
Stopping logging:  
Redirecting start to /bin/systemctl start auditd.service  
[root@serverb ~]# logout  
[student@serverb ~]$ logout  
[student@workstation ~]$
```

3. Log in to the `servera` machine as the `student` user to verify that remote logging for Audit is working. Use the `auditctl -m` command to send the Audit message: `This is a test message from servera`.

- 3.1. Log in to the `servera` machine as the `student` user.

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

- 3.2. Change to the `root` user. Use `student` as the password.

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

- 3.3. Use the `auditctl -m` command to write a test message to the Audit log: `This is a test message from servera`. When done, return to the `workstation` machine as the `student` user.

```
[root@servera ~]# auditctl -m 'This is a test message from servera'  
[root@servera ~]# logout  
[student@servera ~]$ logout  
[student@workstation ~]$
```

- 3.4. Log in to the `serverb` machine as the `student` user.

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

- 3.5. Change to the **root** user. Use **student** as the password.

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

- 3.6. Verify that a new entry in the Audit log file exists for the message that was created on the **servera** machine. When done, return to the **workstation** machine as the **student** user.

```
[root@serverb ~]$ grep servera /var/log/audit/audit.log  
type=USER msg=audit(1697216044.752:241): pid=26638 uid=0 auid=0 ses=7  
subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 msg='text=This is  
a test message from servera exe="/usr/sbin/auditctl" hostname=servera addr=?  
terminal=pts/0 res=success'UID="root" AUID="root"  
type=USER msg=audit(1697216087.366:284): pid=26679 uid=0 auid=0 ses=8  
subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 msg='text=This is  
a test message from servera exe="/usr/sbin/auditctl" hostname=servera addr=?  
terminal=pts/0 res=success'UID="root" AUID="root"  
[root@serverb ~]# logout  
[student@serverb ~]$ logout  
[student@workstation ~]$
```

4. Enable the prepackaged STIG Audit rules on the **servera** machine.

- 4.1. Log in to the **servera** machine as the **student** user.

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

- 4.2. Change to the **root** user. Use **student** as the password.

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

- 4.3. Copy the **/usr/share/audit/sample-rules/30-stig.rules** file with the STIG Audit rules into the **/etc/audit/rules.d/** directory.

```
[root@servera ~]# cp /usr/share/audit/sample-rules/30-stig.rules \  
/etc/audit/rules.d/
```

- 4.4. Load the STIG Audit rules with the **augenrules --load** command.

```
[root@servera ~]# augenrules --load  
...output omitted...
```

5. On the servera machine, some STIG Audit rules mark events that affect identity, user, or group information with the `identity` key string. Verify that those rules work correctly by creating the `labuser` user and confirming that Audit creates records marked with the `identity` key string.

5.1. Verify which STIG Audit rules use the `identity` key.

```
[root@servera ~]# grep identity /etc/audit/rules.d/30-stig.rules  
## Things that affect identity  
-w /etc/group -p wa -k identity  
-w /etc/passwd -p wa -k identity  
-w /etc/gshadow -p wa -k identity  
-w /etc/shadow -p wa -k identity  
-w /etc/security/opasswd -p wa -k identity
```

- 5.2. Create the `labuser` user to test the previous STIG Audit rules. Creating a new user modifies the files that are associated with the rules that use the `identity` key (for example `/etc/passwd`), and triggers those STIG Audit rules.

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

- 5.3. Search the Audit log for the `identity` key to verify that the previous STIG Audit rules are active.

```
[root@servera ~]# ausearch -k identity  
...output omitted...  
----  
time->Fri Oct 13 18:16:20 2023  
type=PROCTITLE msg=audit(1697235380.471:468):  
    proctitle=75736572616464007465737475736572  
type=PATH msg=audit(1697235380.471:468): item=4 name="/etc/gshadow" inode=8393349  
    dev=fc:04 mode=0100000 ouid=0 ogid=0 rdev=00:00 obj=system_u:object_r:shadow_t:s0  
    nametype=CREATE cap_fp=0 cap_hi=0 cap_fe=0 cap_fver=0 cap_frootid=0  
type=PATH msg=audit(1697235380.471:468): item=3 name="/etc/gshadow" inode=8393365  
    dev=fc:04 mode=0100000 ouid=0 ogid=0 rdev=00:00 obj=system_u:object_r:shadow_t:s0  
    nametype=DELETE cap_fp=0 cap_hi=0 cap_fe=0 cap_fver=0 cap_frootid=0  
type=PATH msg=audit(1697235380.471:468): item=2 name="/etc/gshadow+" inode=8393349  
    dev=fc:04 mode=0100000 ouid=0 ogid=0 rdev=00:00 obj=system_u:object_r:shadow_t:s0  
    nametype=DELETE cap_fp=0 cap_hi=0 cap_fe=0 cap_fver=0 cap_frootid=0  
type=PATH msg=audit(1697235380.471:468): item=1 name="/etc/" inode=8388736  
    dev=fc:04 mode=040755 ouid=0 ogid=0 rdev=00:00 obj=system_u:object_r:etc_t:s0  
    nametype=PARENT cap_fp=0 cap_hi=0 cap_fe=0 cap_fver=0 cap_frootid=0
```

Chapter 6 | Recording System Events with Audit

```
type=PATH msg=audit(1697235380.471:468): item=0 name="/etc/" inode=8388736
dev=fc:04 mode=040755 uid=0 ogid=0 rdev=00:00 obj=system_u:object_r:etc_t:s0
nametype=PARENT cap_fp=0 cap_fi=0 cap_fe=0 cap_fver=0 cap_frootid=0
type=CWD msg=audit(1697235380.471:468): cwd="/root"
type=SYSCALL msg=audit(1697235380.471:468): arch=c000003e syscall=82
success=yes exit=0 a0=7fff46201d30 a1=5586f7ef0880 a2=7fff46201ca0 a3=100
items=5 ppid=27033 pid=27103 auid=0 uid=0 gid=0 euid=0 suid=0
egid=0 sgid=0 fsgid=0 tty pts0 ses=14 comm="useradd" exe="/usr/sbin/useradd"
subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 key="identity"
```

- 6.** Create and select the `custom/lab-minimal-with-tty-audit` security profile based on the `minimal` security profile to enable the `pam_tty_audit` PAM module for the student user.

- 6.1. Create the `lab-minimal-with-tty-audit` security profile.

```
[root@servera ~]# authselect create-profile lab-minimal-with-tty-audit \
-b minimal --symlink-meta --symlink-pam
New profile was created at /etc/authselect/custom/lab-minimal-with-tty-audit
```

- 6.2. Add entries to the `/etc/authselect/custom/lab-minimal-with-tty-audit/system-auth` and `/etc/authselect/custom/lab-minimal-with-tty-audit/password-auth` files to enable the `pam_tty_audit.so` module for the student user.

```
[root@servera ~]# echo "session required pam_tty_audit.so enable=student" \
>> /etc/authselect/custom/lab-minimal-with-tty-audit/system-auth
[root@servera ~]# echo "session required pam_tty_audit.so enable=student" \
>> /etc/authselect/custom/lab-minimal-with-tty-audit/password-auth
```

- 6.3. Enable the `custom/lab-minimal-with-tty-audit` security profile.

```
[root@servera ~]# authselect select custom/lab-minimal-with-tty-audit --force
...output omitted...
```

- 7.** Return to the workstation machine as the student user.

```
[root@servera ~]# logout
[student@servera ~]$ logout
Connection to servera closed.
[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 audit-review
```

Finish

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

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

Summary

- The Linux Audit system collects and logs security-related events based on a list of Audit rules. Audit is managed by the kernel.
- The kernel sends the Audit messages that it collects to a user-space daemon (`auditd`), which is responsible for recording them.
- The `auditd` daemon can save messages to a local log, or relay them to a remote `auditd` or `syslog` service.
- You can use the `ausearch` and `aureport` commands to analyze the Audit log.
- You can define Audit rules persistently by editing files in the `/etc/audit/rules.d/` directory that have a `.rules` suffix.
- There are three types of rules: file system rules (watch rules), system call rules, and control rules.
- The `auditctl` command might be used to edit Audit rules temporarily.
- The `audit` package includes some sample Audit rule files to help you implement common security requirements.
- If a control rule has been set to make the Audit rules immutable, then the Audit rules cannot be changed until the system is rebooted.

Chapter 7

Monitoring File-system Changes

Goal

Detect and analyze changes to a server's file systems and their contents by using AIDE.

Sections

- Detecting File-system Changes with AIDE (and Guided Exercise)
- Investigating File-system Changes with AIDE (and Guided Exercise)

Lab

- Monitoring File-system Changes

Detecting File-system Changes with AIDE

Objectives

- Detect and identify changes to files on a system that has AIDE installed, and manage AIDE checks and the AIDE detection database.

Analyzing File-system Changes with AIDE

On an operating server, files are commonly added, removed, and modified in its file systems. However, unexpected changes to certain files, such as executable programs and configuration files, might indicate unauthorized modifications or other security issues. Therefore, Red Hat recommends that you monitor those files for changes to their content, permissions, or other characteristics.

Red Hat Enterprise Linux provides *Advanced Intrusion Detection Environment* (AIDE), a user-space utility that can help with this monitoring. AIDE monitors files for various changes including permission or ownership changes, content changes, and time stamp changes (modification or access time stamps).

Installing AIDE

The `aide` package is typically not installed by default. AIDE must be installed and configured, and its initial database built, before it can verify file systems. The following command installs AIDE on the system:

```
[root@host]# dnf install aide
```

Configuring AIDE

The configuration file for AIDE is the `/etc/aide.conf` file. This file controls which files AIDE monitors for changes, and what characteristics are monitored for each file. For example, by default, AIDE monitors most files in the `/etc` directory for permission changes only, but specific files are monitored more closely. As the security administrator, you might want to adjust exactly what AIDE monitors for different parts of your computer's file system.



Important

AIDE ships with a reasonably well-configured default `/etc/aide.conf` file that you might use to build an initial database. If you want or need to adjust exactly what AIDE monitors, you should modify the file before building or updating your AIDE database.

You can edit the `/etc/aide.conf` file to adjust the operational behavior of AIDE. Each line in the file is a directive. The file contains three types of lines: configuration lines, selection lines, and macro lines. Any line that starts with a number sign (#) is a comment and has no effect.

Configuration Lines

The configuration lines adjust the configuration parameters of AIDE. These lines either adjust the functional behavior of AIDE globally or set a *group definition*. Group definitions are used by selection lines to specify what characteristics of a file AIDE should monitor when detecting file-system changes.

The syntax of a configuration line is `parameter = value`. Configuration parameters include the following examples:

`database`

This configuration sets the location from which AIDE reads its database when it runs checks. This database is typically a local file.

`database_out`

This configuration sets the location to which AIDE writes its database when it is updated. This database is also usually a local file, and must be different from the input database.

`gzip_dbout`

If this parameter is set to yes, then AIDE creates a new database and compresses it with gzip compression.

A configuration line can also create a group definition. Group definitions are used with selection lines to set the characteristics of the file to monitor. For example, the default `/etc/aide.conf` file has the following group definition:

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

This group definition creates the PERMS group. If a selection line uses this group definition, then files selected by that line are monitored for changes to permissions (p), user (u), group (g), Access Control List permissions (acl), SELinux context (selinux), and file system Extended Attributes (xattrs).

The default configuration file has other important predefined group definitions. For example, the NORMAL group definition also monitors for changes in the SHA256 and SHA512 checksums, the size of the file, the inode or number of links to the file, the time stamp, and whether the file is growing. The default configuration file and the `aide.conf(5)` man page have more details on the built-in groups that are available for group definitions.

Selection Lines

The selection lines specify the files and directories that AIDE monitors, and the changes for which AIDE watches. Selection lines can be *regular*, *equals*, or *negative*.

A regular selection line is a regular expression that matches the absolute path to a file or directory, followed by the name of a group definition. Files and directories that match that regular expression are added to the AIDE database, with checks performed as specified by the line's group definition. This effectively means that if the regular expression (regex) is `/etc`, then regex also recursively matches all files and directories in the `/etc` directory.

An equals selection line starts with an equal sign (=) followed by a regex and a group definition. AIDE records the files that match the regular expression, considering all the checks that the line's group definition mentions. However, an equals selection line only matches the children of directories if the regex ends with a forward slash (/). The children of subdirectories are not recursively matched.

Chapter 7 | Monitoring File-system Changes

A negative selection line starts with an exclamation point (!) followed by a regular expression that matches the absolute path to a file or directory. AIDE does not monitor files or directories that match a negative selection line.

The following lines are examples of selection lines:

```
/etc    PERMS  
=/testdir    PERMS  
!/etc/mtab
```

The first line is a regular selection line that matches the /etc directory and recursively matches all files and directories in the /etc directory. This line applies the group definition PERMS to those files.

The second line is an equals selection line that matches exactly the /testdir directory, but does not match its subdirectories. This line also applies the group definition PERMS to that directory, but not to the files or subdirectories in that directory.

The third line is a negative selection line that tells AIDE not to monitor the /etc/mtab file.

Macro Lines

Macro lines set or clear variables that are useful for referring to lengthy URLs or file-system paths in multiple occurrences throughout the AIDE configuration file.

The following snippet from the AIDE configuration file shows sample macro lines:

```
@@define DBDIR /var/lib/aide  
@@define LOGDIR /var/log/aide
```

These macro lines define the DBDIR and LOGDIR variables that substitute the /var/lib/aide and /var/log/aide directories, respectively.

The following example shows how to use a macro line to expand the variable.

```
database=file:@@{DBDIR}/aide.db.gz
```

This macro line sets the database parameter to the file:/var/lib/aide/aide.db.gz value.

Initializing the AIDE Database

After installing AIDE, ensure that AIDE is aware of the current status of the file system. AIDE uses the known state of the file system as a reference point to detect and report on file-system changes. Use the aide --init command to generate an initial AIDE database.

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



Important

Ideally, build your AIDE database as soon as you can after installation, possibly as part of the provisioning process.

AIDE operates by comparing the current state of files to information about their expected state that is stored in the AIDE database. If there is no baseline database, AIDE has nothing it can use to determine the expected state of the system.

Verifying Integrity with AIDE

After you initialize AIDE, it can detect file-system changes (if any) by comparing against the known status. To manually perform an integrity check, run the following command as the `root` user:

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

This command uses the AIDE configuration to compare the state of the system's files to the saved database. A report is printed on standard output and to the `/var/log/aide/aide.log` file by default.

In production environments, you should periodically run AIDE checks. If the AIDE database is kept on the local system, you might choose to use a cron job, a `systemd` timer unit, or another system to automatically run the AIDE checks.

For example, you can set a cron job to run the AIDE integrity check every day at 5:00 PM by adding a file that contains the following line to the `/etc/cron.d` directory:

```
0 17 * * * root /usr/sbin/aide --check
```

You can additionally configure the `crontab` file to email the report to an administrator.

Updating the AIDE Database

You must update the AIDE database after expected changes are made to the system. For example, a package update or an authorized change to a configuration file might change time stamps, permissions, or checksums on monitored files. To prevent AIDE from reporting false positives, you must update the database to reflect these authorized changes. After confirming that all remaining changes reported by AIDE are authorized, run the following command to update the AIDE database:

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



Important

Do not forget to replace the earlier database file with the updated file. Otherwise, AIDE continues to use the earlier database file as its baseline for checks.

The locations of these files are specified in your `/etc/aide.conf` file, as discussed earlier in this section. The database that is used for checks defaults to the `/var/lib/aide/aide.db.gz` file. By default, the `--update` option writes an updated database to the `/var/lib/aide/aide.db.new.gz` file.



References

aide(1) and aide.conf(5) man pages

For more information, refer to the *Checking Integrity with Aide* chapter in the *Red Hat Enterprise Linux 9 Security Hardening* guide at

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html/security_hardening/checking-integrity-with-aide_security-hardening

► Guided Exercise

Detecting File-system Changes with AIDE

Install AIDE, perform an initial baseline scan, and then changes files on a monitored file system to explore how AIDE detects and reports those changes.

Outcomes

- Install the `aide` package.
- Modify the `/etc/aide.conf` configuration file.
- Perform an initial baseline scan.
- Detect changes made to files after the baseline scan with a subsequent scan.
- Update the AIDE database to accept approved changes to the file system.

Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

```
[student@workstation ~]$ lab start aide-detecting
```

Instructions

- 1. From the `workstation` machine, log in to the `servera` machine as the `student` user.

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

- 2. Change to the `root` user and install the `aide` package.

- 2.1. Change to the `root` user. Use `student` as the password.

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

- 2.2. Install the `aide` package.

```
[root@servera ~]# dnf install aide
Last metadata expiration check: 2:09:21 ago on Fri Oct 13 16:30:57 2023.
Dependencies resolved.
=====
 Package      Architecture Version      Repository      Size
 =====
```

```
Installing:  
 aide x86_64 0.16-100.el9  rhel-9.2-for-x86_64-appstream-rpms      154 k  
  
Transaction Summary  
=====...  
...output omitted...  
Is this ok [y/N]: y  
...output omitted...  
Complete!
```

- 3. Edit the `/etc/aide.conf` configuration file so that AIDE monitors the existing `/etc` directory and a new `/testdir/` directory for changes to permissions or file content. Use regular selection lines so that the contents of those directories are also monitored. Remove all other selection lines that are not associated with the mentioned directories.

The `CONTENT_EX` group definition is provided in the default `/etc/aide.conf` file. Any selection line that uses the `CONTENT_EX` definition monitors the selected files for any change in content, Linux file type, number of links, ownership, permissions, SELinux context, and extended attributes.

The two selection lines configure AIDE to monitor any object that is within the hierarchy of the `/etc` and `/testdir` directories in the file system by using the checks specified by the `CONTENT_EX` group definition.



Note

You remove the other selection lines to speed up preparation of the AIDE database for this exercise. With this configuration change, AIDE analyzes only a handful of the files on the system. In practice, you might decide to keep or adjust the default selection lines, rather than remove them.

```
[root@servera ~]# cat /etc/aide.conf  
...output omitted...  
# Extended content + file type + access.  
CONTENT_EX = sha256+ftype+p+u+g+n+acl+selinux+xattrs  
  
# Some files get updated automatically, so the inode/ctime/mtime change  
# but we want to know when the data inside them changes  
DATAONLY = p+n+u+g+s+acl+selinux+xattr+sha256  
  
# Next decide what directories/files you want in the database.  
  
/etc          CONTENT_EX  
/testdir      CONTENT_EX
```

- 4. Create the `/testdir` directory.

```
[root@servera ~]# mkdir /testdir
```

Chapter 7 | Monitoring File-system Changes

- ▶ 5. Initialize the AIDE database by using the `aide --init` command. The command performs an initial baseline scan, which generates a new AIDE database that records the current state of the file-system hierarchy.

```
[root@servera ~]# aide --init
Start timestamp: 2023-10-16 16:19:50 -0400 (AIDE 0.16)
AIDE initialized database at /var/lib/aide/aide.db.new.gz

Number of entries: 969

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

/var/lib/aide/aide.db.new.gz
MD5      : ABeCgEmptsRtz8E9ATuD5Q==
SHA1     : iPFfNp932eaD8ZzN6YKJEi6SNqU=
RMD160   : 3p70Yvccf1dpF/dT0LeYJ2LGWNE=
TIGER    : YveonafhD6gSwiz2mCs+3iv/j5HplTny
SHA256   : dU6dXIsP556LyrqirQWfw8WSJb+r/N2L
          eAl47C37qXU=
SHA512   : ir/ETdTvPy/ctA+vNCSyddQ1c/yP8nh5
          nrl34sYPAmVbvh6bETZfCVSACbE1+WI
          MBBN4CSLlzSAXiqJXswIlQ==

End timestamp: 2023-10-16 16:19:51 -0400 (run time: 0m 1s)
```

- ▶ 6. Rename the new AIDE database file from `/var/lib/aide/aide.db.new.gz` to `/var/lib/aide/aide.db.gz` so that AIDE uses the newly generated file as the current database. The file name of any new database that is generated, and that of the database used by AIDE, are set in the `/etc/aide.conf` configuration file by default.

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

- ▶ 7. Determine the current status of the machine's file systems.

AIDE reports that there are no changes to the files and directories that it monitors.

```
[root@servera ~]# aide --check
Start timestamp: 2023-10-16 16:21:42 -0400 (AIDE 0.16)
AIDE found NO differences between database and filesystem. Looks okay!!

Number of entries: 969

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

/var/lib/aide/aide.db.gz
MD5      : ABeCgEmptsRtz8E9ATuD5Q==
SHA1     : iPFfNp932eaD8ZzN6YKJEi6SNqU=
```

```
RMD160 : 3p70Yvccf1dpF/dT0LeYJ2LGWNE=
TIGER   : YveonafhD6gSwiz2mCs+3iv/j5HplTny
SHA256   : du6dXIsP556LyrqirQWfw8WSJb+r/N2L
           eAl47C37qXU=
SHA512   : ir/ETdTvpPy/ctA+vNCSyddQ1c/yP8nh5
           nrl34sYPAmVbawah6bETZfCVSACbE1+WI
           MBBN4CSLlzSAXiqJXswI1Q==
```

```
End timestamp: 2023-10-16 16:21:42 -0400 (run time: 0m 0s)
```

- 8. Create the /testdir/testfile file.

```
[root@servera ~]# touch /testdir/testfile
```

- 9. View the changed status of the machine's file systems.

AIDE reports that the /testdir/testfile file has been added. No files have been removed or changed.

```
[root@servera ~]# aide --check
Start timestamp: 2023-10-16 16:24:59 -0400 (AIDE 0.16)
AIDE found differences between database and filesystem!!
```

Summary:

```
Total number of entries: 970
Added entries: 1
Removed entries: 0
Changed entries: 0
```

```
-----
```

Added entries:

```
-----
```

```
f+++++++/: /testdir/testfile
```

```
-----
```

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

```
-----
```

```
/var/lib/aide/aide.db.gz
MD5      : ABeCgEMptsRtz8E9ATuD5Q==
SHA1     : iPFfNp932eaD8ZzN6YKJEi6SNqU=
RMD160   : 3p70Yvccf1dpF/dT0LeYJ2LGWNE=
TIGER    : YveonafhD6gSwiz2mCs+3iv/j5HplTny
SHA256   : du6dXIsP556LyrqirQWfw8WSJb+r/N2L
           eAl47C37qXU=
SHA512   : ir/ETdTvpPy/ctA+vNCSyddQ1c/yP8nh5
           nrl34sYPAmVbawah6bETZfCVSACbE1+WI
           MBBN4CSLlzSAXiqJXswI1Q==
```

Chapter 7 | Monitoring File-system Changes

- ▶ 10. Modify the permissions of the /etc/shadow file to 644 (which is read and write for the user, and read-only for the group and other).

```
[root@servera ~]# chmod 644 /etc/shadow
```

- ▶ 11. Determine the current status of the machine's file systems.

AIDE reports that the permissions of the /etc/shadow file have changed and indicates what those changes are. AIDE still reports about the newly created /testdir/testfile file.

```
[root@servera ~]# aide --check
Start timestamp: 2023-10-16 16:27:35 -0400 (AIDE 0.16)
AIDE found differences between database and filesystem!

Summary:
Total number of entries: 970
Added entries: 1
Removed entries: 0
Changed entries: 1

-----
Added entries:
-----
f+++++++: /testdir/testfile

-----
Changed entries:
-----
f p... .A.. : /etc/shadow

-----
Detailed information about changes:
-----
File: /etc/shadow
Perm      : -----          | -rw-r--r--
ACL       : A: user:----   | A: user::rw-
                  A: group:---- | A: group::r--
                  A: other:---- | A: other::r--

-----
The attributes of the (uncompressed) database(s):
-----
/var/lib/aide/aide.db.gz
MD5      : ABeCgEmptsRtz8E9ATuD5Q==
SHA1     : iPFfNp932eaD8ZzN6YKJEi6SNqU=
RMD160   : 3p70Yvccf1dpF/dT0LeYJ2LGWNE=
TIGER    : YveonafhD6gSwiZ2mCs+3iv/j5HplTny
SHA256   : dU6dXIsP556LyrqirQWfw8WSJb+r/N2L
```

Chapter 7 | Monitoring File-system Changes

```
eAl47C37qXU=
SHA512 : ir/ETdTvPy/ctA+vNCSyddQ1c/yP8nh5
          nrl34sYPAmVbvah6bETZfCVSACbE1+WI
          MBBN4CSLlzSAXiqJXswI1Q==
```

End timestamp: 2023-10-16 16:27:35 -0400 (run time: 0m 0s)

- ▶ 12. Restore the file permissions of the /etc/shadow file to 000 (which is no access for the user, group, or other).

```
[root@servera ~]# chmod 000 /etc/shadow
```

- ▶ 13. Determine the current status of the file system.

After restoring the file permissions of the /etc/shadow file, AIDE no longer reports that it has changed. AIDE still reports about the newly created /testdir/testfile file, because that change has not been reverted.

**Note**

If the CONTENT_EX group definition included the c group to monitor ctime (status change time stamp) updates, then the /etc/shadow file would still report a change, because ctime is updated when permissions are changed.

That time stamp is separate from the mtime (modification time stamp) that AIDE monitors with the m group, and which shows up in default ls -l listings to indicate when the contents of the file last changed.

```
[root@servera ~]# aide --check
Start timestamp: 2023-10-16 16:32:10 -0400 (AIDE 0.16)
AIDE found differences between database and filesystem!!
```

Summary:

```
Total number of entries: 970
Added entries:    1
Removed entries:  0
Changed entries:  0
```

Added entries:

```
-----+
f+++++++: /testdir/testfile
```

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

```
/var/lib/aide/aide.db.gz
MD5      : ABeCgEmptsRtz8E9ATuD5Q==
SHA1     : iPFfNp932eaD8ZzN6YKJEi6SNqU=
RMD160   : 3p70Yvccf1dpF/dT0LeYJ2LGWNE=
```

Chapter 7 | Monitoring File-system Changes

```
TIGER      : YveonafhD6gSwiZ2mCs+3iv/j5HplTny
SHA256     : dU6dXIsP556LyrqirQWfw8WSJb+r/N2L
              eAl47C37qXU=
SHA512     : ir/ETdTVPy/ctA+vNCSyddQ1c/yP8nh5
              nrl34sYPAmVbavah6bETZfcVSACbE1+WI
              MBBN4CSLlzSAXiqJXswIlQ==
```

End timestamp: 2023-10-16 16:32:10 -0400 (run time: 0m 0s)

- ▶ **14.** Implement a cron job to automatically run AIDE to check the current status of the file systems and to log the results. The job should run as the `root` user at midnight weekly, and log to the `/var/log/aide.log` file. Create a file named `/etc/cron.d/aide` file that contains the following entry.

Use the automated approach for verifying the file system's current status, which allows administrators to periodically monitor the file system for changes and to merge the accepted changes with the AIDE database.

**Note**

Alternatively, you can use a `systemd` timer unit to automate AIDE checks. This guided exercise uses a `cron` job for simplicity.

```
[root@servera ~]# cat /etc/cron.d/aide
0 0 */7 * * root /sbin/aide --check >> /var/log/aide.log
```

- ▶ **15.** Generate an updated AIDE database to accept the changes that you made to the machine's file systems. Rename the newly generated `aide.db.new.gz` file to `aide.db.gz` to ensure that AIDE uses the correct database to detect file-system changes.

```
[root@servera ~]# aide --update
...output omitted...
[root@servera ~]# mv /var/lib/aide/aide.db.new.gz \
/var/lib/aide/aide.db.gz
mv: overwrite '/var/lib/aide/aide.db.gz'? y
```

- ▶ **16.** Confirm that you successfully updated the AIDE database to reflect the changes to the machine's file systems.

```
[root@servera ~]# aide --check
Start timestamp: 2023-10-16 17:07:04 -0400 (AIDE 0.16)
AIDE found NO differences between database and filesystem. Looks okay!!

Number of entries: 970

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

/var/lib/aide/aide.db.gz
MD5      : y9vF6r0VgqjaXGoR6GBkZQ==
```

Chapter 7 | Monitoring File-system Changes

```
SHA1      : hoxPwkyBmq5p4eJkpV0fGt0b0ns=
RMD160    : hABCAXJLUYb8EkniRLV3yNKz/CQ=
TIGER     : Nnm8ZtrSD0lnJWQ2ee1hH/3/mreszTo3
SHA256    : LPRnIC4br2m+9fNBYiZnQd3p3X2cFSIz
              Hsav+nIsrbg=
SHA512    : 6JEfse5zl37j9f4J57h/GNfUXVq1+3dp
              YAWGM6oU8xy26CX7y5Sja+YDzA8ojMeH
              wPcvQm97nrudBhB51ZdCCw==
```

```
End timestamp: 2023-10-16 17:07:04 -0400 (run time: 0m 0s)
```

- 17. Return to the **workstation** machine as the **student** user.

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

Finish

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

```
[student@workstation ~]$ lab finish aide-detecting
```

Investigating File-system Changes with AIDE

Objectives

- Investigate causes of file-system changes that are reported by AIDE by using Linux Audit tools.

Combining AIDE and Audit

Advanced Intrusion Detection Environment (AIDE) detects changes on a computer's file systems, and reports to administrators about specific changes. However, one limitation of AIDE is that it does not tell you what caused the changes.

For example, you can configure AIDE to report changes to the /etc/passwd and /etc/shadow files, but AIDE does not provide details such as who made the changes, at what time, or which tool they used. This is where Audit rules can help you to get the details that you need about what happened.

The Linux Audit system monitors files for activity that might cause a change, and it logs useful information about the time of the activity, and the process and user that are involved. The Audit system logs these changes, but it is not straightforward to set up Audit to report on key events by itself.

You can solve this problem by combining AIDE and Audit. By configuring both systems to monitor key files and directories, you can use AIDE to report periodically on unexpected changes in your systems. Then you can review the Audit log from the last AIDE report before the change to help identify what activity might be responsible.



Note

One limitation of this approach is that AIDE does not tell you in real time that there has been a change. It only detects changes when a check is run.

If you are monitoring for changes to the contents of files, then an AIDE check can be expensive in terms of system resources. Setting the frequency of AIDE checks too high can increase the load on the system in CPU time and disk I/O. Likewise, large amounts of Audit logging, especially by using system call rules, can have an impact on system performance.

Configuring AIDE and Audit

For this approach to be effective, you must configure both AIDE and Audit to monitor the same files. Audit records activity that affects a file; AIDE detects and reports that there has been a change to a file. As a result, you must set up the AIDE database and watch rules on files in advance to collect the relevant information for detection and investigation.

Start by making sure that AIDE is properly configured. Edit the /etc/aide.conf file to configure AIDE to monitor the correct files, initialize the AIDE database, and set up the cron job to periodically run AIDE checks and to send the report to the system administrators.

The following example shows the configuration of the /etc/aide.conf file to monitor the changes of all the files in the /usr directory and its subdirectories. The default /etc/aide.conf

Chapter 7 | Monitoring File-system Changes

file settings set a regular selection line to monitor the `/usr` directory for changes to content, permissions, and ownership:

```
CONTENT_EX = sha256+ftype+p+u+g+n+acl+selinux+xattr  
...output omitted...  
/usr/    CONTENT_EX
```

You must also set an Audit rule to monitor the `/usr` directory recursively for attempts to write files or change permissions. One method is to set a file-system rule in a `.rules` file in the `/etc/audit/rules.d` directory.

The following example shows a minimal rule to set a watch for changes to the `/usr` directory:

```
-w /usr -p wa -k usr-modification
```



Important

If you do not need to collect information for auditing purposes, you should probably avoid doing so. This reduces extraneous information in the logs, reduces the size of the logs, and might improve performance, depending on the structure of your rules.

In the previous watch example, the `-p rwx` option is not set. This option creates a log entry every time something opens a file in the `/usr` directory to read or execute it. Opening a file in the `usr` directory occurs often and the information is less relevant for this analysis. Additional information could make your analysis simpler, and you might care about that information for other reasons. But additional information could also make it harder for you to find the relevant event. You should carefully consider the tradeoffs of what and how much information you log.

Investigating File-system Changes

Now you can search for Audit events that are relevant to AIDE reports. For example, if AIDE reports that the `/etc/group` file has unexpectedly changed, then you can use the `ausearch` command to look for relevant Audit logs:

```
[root@host]# ausearch -i -f /etc/group  
...output omitted...
```

Use the `-f` option to select events relevant to the file that you are investigating. Use the `-i` option to help interpret the Audit event, such as by converting numbers to more human-readable names.



Note

In some cases you might want to omit `-i` to see the uninterpreted Audit log. For example, if there is a misconfiguration where multiple UID numbers map to the same username, the raw log might be useful.

To further narrow down the output of the Audit search to events of your interest, you can use the `-ts` option to search for events from the date and time of the last AIDE report that did not report a change.

```
[root@host ~]# ausearch -ts -i -f /etc/group
...output omitted...
type=PROCTITLE msg=audit(10/15/23 23:48:59.672:158) : proctitle=groupadd test-
group1
type=PATH msg=audit(10/15/23 23:48:59.672:158) : item=0 name=/etc/group
inode=8393479 dev=fc:04 mode=file,644 uid=root ogid=root rdev=00:00
obj=system_u:object_r:passwd_file_t:s0 nametype=NORMAL cap_fp=none cap_fi=none
cap_fe=0 cap_fver=0 cap_frootid=0
type=CWD msg=audit(10/15/23 23:48:59.672:158) : cwd=/root
type=SYSCALL msg=audit(10/15/23 23:48:59.672:158) : arch=x86_64 syscall=openat
success=yes exit=4 a0=AT_FDCWD a1=0x7f3b467bbe0c a2=0_RDONLY|0_CLOEXEC a3=0x0
items=1 ppid=1221 pid=26503 auid=student uid=root gid=root euid=root suid=root
fsuid=root egid=root sgid=root fsgid=root tty=pts0 ses=3 comm=groupadd exe=/
usr/sbin/groupadd subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
key=group_watch
...output omitted...
```

Depending on the Audit rules that you have set, you might have several Audit events to look through. Next, you need to determine which of those events are involved in the change.

Interpreting Audit Events

You must review the Audit events that the ausearch command finds to determine what happened. There might be multiple Audit events involved in sequence. If you are logging other activity on the files that are involved, then some irrelevant events might be logged that are not involved in whatever changed that file. Alternatively, there might be events before the actual change that were triggered by the same command that made the change.

For this reason, you need to carefully review the Audit events to understand what happened. Often, the logged event is a system call to the kernel. The events might help you understand what arguments the system call takes so that you can better interpret the SYSCALL Audit record to identify the relevant event.

To illustrate this scenario, examine two different Audit events involving the /etc/group file. Assume that an AIDE report states that the modification time and contents of the /etc/group file have changed.

The following example shows one Audit event:

```
...output omitted...
type=PROCTITLE msg=audit(10/16/23 00:01:26.949:344) : proctitle=touch /etc/group
type=PATH msg=audit(10/16/23 00:01:26.949:344) : item=1 name=/etc/group
inode=8622249 dev=fc:04 mode=file,644 uid=root ogid=root rdev=00:00
obj=system_u:object_r:passwd_file_t:s0 nametype=NORMAL cap_fp=none cap_hi=none
cap_fe=0 cap_fver=0 cap_frootid=0
type=PATH msg=audit(10/16/23 00:01:26.949:344) : item=0 name=/etc/
inode=8388736 dev=fc:04 mode=dir,755 uid=root ogid=root rdev=00:00
obj=system_u:object_r:etc_t:s0 nametype=PARENT cap_fp=none cap_hi=none cap_fe=0
cap_fver=0 cap_frootid=0
type=CWD msg=audit(10/16/23 00:01:26.949:344) : cwd=/home/student
type=SYSCALL msg=audit(10/16/23 00:01:26.949:344) :
arch=x86_64 syscall=openat success=no exit=EACCES(Permission denied) a0=AT_FDCWD
a1=0x7ffdc8d99687 a2=0_WRONLY|0_CREAT|0_NOCTTY|0_NONBLOCK a3=0x1b6 items=2
ppid=26589 pid=26731 auid=student uid=student gid=student euid=student
suid=student fsuid=student egid=student sgid=student fsgid=student tty=pts1 ses=5
comm=touch exe=/usr/bin/touch subj=unconfined_u:unconfined_r:unconfined_t:s0-
s0:c0.c1023 key=group_watch
...output omitted...
```

The SYSCALL record for this event indicates that the `openat` system call is being used to open the file in `O_WRONLY` mode. The `syscall=openat` field indicates the system call used, and the `a2` field contains the third argument to the system call, counting from zero. The `openat(2)` man page indicates that `O_WRONLY` mode in that location is a flag for write-only mode. Review the man page for the system call that you are analyzing to determine which arguments contain what data.

One of the PATH records for this event indicates that the `/etc/group` file is involved.

The PROCTITLE record for this event suggests that `touch /etc/group` was the command that triggered this event, which should update the modification time of the `/etc/group` file, but should not affect the file's contents. The metadata update would still require write access to the file's inode.

However, carefully examine the SYSCALL record again. This system call failed as indicated by the `success=no` and `exit=EACCES(Permission denied)` records. This is not the right Audit event.

The following example is a second Audit event:

```
...output omitted...
type=PROCTITLE msg=audit(10/16/23 00:04:25.441:363) : proctitle=vigr
type=PATH msg=audit(10/16/23 00:04:25.441:363) : item=4 name=/etc/
group inode=8622254 dev=fc:04 mode=file,644 uid=root ogid=root rdev=00:00
obj=system_u:object_r:passwd_file_t:s0 nametype=CREATE cap_fp=none cap_hi=none
cap_fe=0 cap_fver=0 cap_frootid=0
type=PATH msg=audit(10/16/23 00:04:25.441:363) : item=3 name=/etc/group
inode=8622249 dev=fc:04 mode=file,644 uid=root ogid=root rdev=00:00
obj=system_u:object_r:passwd_file_t:s0 nametype=DELETE cap_fp=none cap_hi=none
cap_fe=0 cap_fver=0 cap_frootid=0
type=PATH msg=audit(10/16/23 00:04:25.441:363) : item=2 name=/etc/
group.edit inode=8622254 dev=fc:04 mode=file,644 uid=root ogid=root rdev=00:00
obj=system_u:object_r:passwd_file_t:s0 nametype=DELETE cap_fp=none cap_hi=none
cap_fe=0 cap_fver=0 cap_frootid=0
```

Chapter 7 | Monitoring File-system Changes

```

type=PATH msg=audit(10/16/23 00:04:25.441:363) : item=1 name=/etc/
inode=8388736 dev=fc:04 mode=dir,755 uid=root ogid=root rdev=00:00
obj=system_u:object_r:etc_t:s0 nametype=PARENT cap_fp=none cap_fi=none cap_fe=0
cap_fver=0 cap_frootid=0
type=PATH msg=audit(10/16/23 00:04:25.441:363) : item=0 name=/etc/
inode=8388736 dev=fc:04 mode=dir,755 uid=root ogid=root rdev=00:00
obj=system_u:object_r:etc_t:s0 nametype=PARENT cap_fp=none cap_fi=none cap_fe=0
cap_fver=0 cap_frootid=0
type=CWD msg=audit(10/16/23 00:04:25.441:363) : cwd=/root
type=SYSCALL msg=audit(10/16/23 00:04:25.441:363) :
arch=x86_64 syscall=rename success=yes exit=0 a0=0x7ffc431008c0 a1=0x56270e8e4440
a2=0x0 a3=0x0 items=5 ppid=26740 pid=26762 auid=student uid=root gid=root
euid=root suid=root fsuid=root egid=root sgid=root fsgid=root tty=pts1 ses=5
comm=vigr exe=/usr/sbin/vipw subj=unconfined_u:unconfined_r:unconfined_t:s0-
s0:c0.c1023 key=group_watch
...output omitted...

```

The output shows the PROCTITLE record, five PATH records, a CWD record, and a SYSCALL record.

The SYSCALL record shows that this event was a `rename` system call that was successful (`success=yes`), which means the file was renamed. (That system call is documented by the `rename(2)` man page.)

Examine the PATH records, paying close attention to the inode numbers and the `objtype` fields. The 8622254 inode has two records: `item=2` and `item=4`. The inode had the `/etc/group.edit` name in the `item=2` record, but that was deleted. The inode was created with the `/etc/group` name in the `item=4` record, which looks like the file that was renamed. In addition, inode 8622254 had the `/etc/group` name and was deleted. This file is the original `/etc/group` file that is being overwritten.

Reviewing the SYSCALL record again, the file was renamed by the `root` (`uid=root`) user. But, the user originally logged in as the `student` (`auid=student`) user and changed their real UID somehow. The command used was `vigr`, which executed the `/usr/sbin/vipw` command. The `/usr/sbin/vigr` file is a symlink to the `/usr/sbin/vipw` file. The command was run from terminal `pts/1` (`tty=pts1`), which is a pseudoterminal (most likely a remote login session or graphical terminal window). This is all useful data for further investigation.

**Important**

This event certainly changed the contents, modification time, and inode belonging to the `/etc/group` file, so this is an event of interest. Remember, there might be more than one event of interest in the period of time. You should investigate any events that the `ausearch` command helps you find.

The following table lists some record types that you might see when determining the cause of a file change.

Common Audit Record Types

Record type	Description
CWD	The current working directory that is relevant to this event.

Record type	Description
PATH	Information about the path to a file that is involved in this event. Watch in particular for its <code>obj type</code> field, which records why this record is involved in an event, including a SYSCALL record. There might be multiple PATH records in the same event, for files that are involved for different reasons.
PROCTITLE	The complete command line that triggered this event.
SYSCALL	The system call that was made to the kernel that triggered this event. Particular system calls are likely to be involved in a file change event, such as the <code>open()</code> , <code>fchmod()</code> , and <code>setxattr()</code> calls. This record is also most likely to have the most information about which user and process is involved in the operation.

The following table lists some useful fields that you might see in an Audit record.

Audit Record Fields

Field	Description
a0	First argument of the system call.
a1	Second argument of the system call. This is particularly relevant with the <code>open(2)</code> system call on x86_64 because it records the access mode that is being used to open the file. The <code>O_WRONLY</code> or <code>O_RDWR</code> modes indicate that the system call is trying to open the file in a writable mode.
a2	Third argument of the system call. The <code>openat(2)</code> system call records the access mode that is used to open the file in this argument.
a3	Fourth argument of the system call.
auid	The Audit UID. This is the user ID that was used to log in to the system initially. It stays the same even after the user uses the <code>su</code> command or some other command to change their real UID.
cwd	The current working directory that is relevant to this event.
euid	The effective UID of the user who started the analyzed process.
egid	The effective GID of the group that the process has for permission checks.

Field	Description
proctitle	The process title. The record shows the full command line along with the arguments of the executed command. This record is encoded in hexadecimal notation. Use <code>-i</code> option of the <code>ausearch</code> command to decode it.
syscall	The system call that is sent to the kernel.
success	Shows whether the system call was successful.
tty	The terminal or pseudoterminal that is controlling the process. If the command was run as part of an interactive session and the user had several in progress, then this field can be used to help identify which session was used to run the command.
uid	The real UID. The user ID that started the process. This might not be the same as the initial login ID of the user. For example, a user might use the <code>su</code> command to switch from one user to another, which changes their real UID. Likewise, if the process was started by the <code>setuid</code> or <code>setgid</code> executables, the effective UID or GID of the process might be different.



References

`auditctl(8)`, `ausearch(8)`, `audit.rules(7)`, and `auditd.conf(5)` man pages

For more information, refer to the *Checking Integrity with AIDE* chapter in the *Red Hat Enterprise Linux 9 Security Hardening* guide at

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html/security_hardening/checking-integrity-with-aide_security-hardening

For more information, refer to the *Understanding Audit Log Files* chapter in the *Red Hat Enterprise Linux 9 Security Hardening* guide at

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html/security_hardening/auditing-the-system_security-hardening#understanding-audit-log-files_auditing-the-system

► Guided Exercise

Investigating File-system Changes with AIDE

Use AIDE and Audit to detect changes on a file system, and use Audit tools to identify the cause of those changes.

Outcomes

- Monitor the /etc/passwd and /etc/shadow files with AIDE.
- Set Linux Audit rules to monitor the files.
- Change the files, and then use Audit tools to identify which user and process made the changes.

Before You Begin

As the student user on the workstation machine, use the lab command to prepare your environment for this exercise, and to ensure that all required resources are available.

```
[student@workstation ~]$ lab start aide-investigating
```

Instructions

- 1. From the workstation machine, log in to the servera machine as the student user.

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

- 2. Change to the root user. Use student as the password.

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

- 3. Edit the /etc/aide.conf file to direct AIDE to use the CONTENT_EX group definition for monitoring for any changes to the /etc/shadow and /etc/passwd files. Comment out the other selection lines in the configuration file. Usually, you would not modify these lines from your /etc/aide.conf file. The other selection lines are commented out to speed up AIDE scans for the purpose of this exercise.

```
...output omitted...  
# Extended content + file type + access.  
CONTENT_EX = sha512+ftype+p+u+g+n+acl+selinux+xattrs  
  
# Some files get updated automatically, so the inode/ctime/mtime change
```

Chapter 7 | Monitoring File-system Changes

```
# but we want to know when the data inside them changes
DATAONLY = p+n+u+g+s+acl+selinux+xattr+sha512

# Next decide what directories/files you want in the database

#/boot      CONTENT_EX
#/opt       CONTENT

/etc/passwd CONTENT_EX
/etc/shadow CONTENT_EX
...output omitted...
```

- ▶ 4. Initialize the AIDE database with the results of a baseline scan.

```
[root@servera ~]# aide --init
Start timestamp: 2023-10-16 01:36:16 -0400 (AIDE 0.16)
AIDE initialized database at /var/lib/aide/aide.db.new.gz

Number of entries: 52938

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

/var/lib/aide/aide.db.new.gz
MD5      : Srt/ufvcsTXSVth6usbGXQ==
SHA1     : FPUaTNXkq5x72FLK2FwXZOF417k=
RMD160   : DV3dBmQyHM+eo99C+XHEeCWSCgA=
TIGER    : 6A1ky0dKHLmjGll7LA6HcRTKSiNksyvN
SHA256   : NKZfQFk8b5mxL3PbsLrpq4CLFsU+N+6C1
          wi7LpMEkzBA=
SHA512   : +7YUEfIp7mgWeDdh1ItQE1B+fHkBicAJ
          asJ4VbX5jCIriPwqvcPy3f9QwZiu8ULS
          e2lj+XiIVTqVE4p3UwscmQ==

End timestamp: 2023-10-16 01:36:52 -0400 (run time: 0m 36s)
```

- ▶ 5. Rename the new AIDE database file from `/var/lib/aide/aide.db.new.gz` to `/var/lib/aide/aide.db.gz` so that AIDE uses the newly generated file as the current database.

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

- ▶ 6. Determine the current status of the machine's file systems.

AIDE reports no changes to the files and directories that it monitors, because you have not changed any files or directories after initializing the AIDE database.

```
[root@servera ~]# aide --check
Start timestamp: 2023-10-16 01:38:48 -0400 (AIDE 0.16)
AIDE found NO differences between database and filesystem. Looks okay!!
```

Chapter 7 | Monitoring File-system Changes

```
Number of entries: 52938

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

/var/lib/aide/aide.db.gz
MD5      : Srt/ufvcstXSVth6usbGXQ==
SHA1     : FPUaTNXkq5x72FlK2FwXZ0F417k=
RMD160   : DV3dBmQyHM+eo99C+XHeeCWSCgA=
TIGER    : 6A1ky0dKHLmjGll7LA6HcRTKSiNksyvN
SHA256   : NKZfQFk8b5mxL3PbsLrpq4CLFsU+N+6C1
           wi7LpMEkzBA=
SHA512   : +7YUEfIp7mgWeDdh1ItQE1B+fHkBicAJ
           asJ4VbX5jCIriPwqvCPy3f9QwZiu8ULS
           e2lj+XiIVTqVE4p3UwscmQ==

End timestamp: 2023-10-16 01:39:20 -0400 (run time: 0m 32s)
```

- ▶ 7. AIDE reports files that have changed, but not what or who changed them. You can add Audit rules to monitor your files so that when AIDE reports a change, you can look in the Audit log to determine who or what might have caused that change.
Add a persistent Audit watch rule to generate Audit log entries whenever there is an attempt to read, write, execute, or change an attribute of the /etc/shadow file. Use `group_watch` as the filter key on the Audit rule.

```
[root@servera ~]# echo "-w /etc/shadow -p rwx -k group_watch" \
>> /etc/audit/rules.d/audit.rules
```

- ▶ 8. Apply the changes for the newly added Audit rules to take effect.

```
[root@servera ~]# augenrules --load
...output omitted...
```

- ▶ 9. List the Audit rules and verify that the newly added Audit rule is currently in effect.

```
[root@servera ~]# auditctl -l
-w /etc/shadow -p rwx -k group_watch
```

- ▶ 10. Change the password from student to redhat for the student user on the servera machine. This update causes a change in the /etc/shadow file.

```
[root@servera ~]# passwd student
Changing password for user student.
New password: redhat
BAD PASSWORD: The password is shorter than 8 characters
Retype new password: redhat
passwd: all authentication tokens updated successfully.
```

- 11. Check the status of the machine's file systems that are monitored by AIDE.

The output of the command should report that the SHA512 checksum (and therefore the content) of the /etc/shadow file has changed. Use the ausearch command to view the activity that caused the change in the /etc/shadow file.

The values of the SHA512 checksums that you see in the following output might differ from what you see in your system.

```
[root@servera ~]# aide --check
Start timestamp: 2023-10-16 01:49:28 -0400 (AIDE 0.16)
AIDE found differences between database and filesystem!

Summary:
Total number of entries: 52939
Added entries: 1
Removed entries: 0
Changed entries: 3
...output omitted...

-----
Changed entries:
-----

f ... .C... : /etc/audit/audit.rules
f ... .C... : /etc/audit/rules.d/audit.rules
f ... .C... : /etc/shadow

-----
Detailed information about changes:
-----

...output omitted...

File: /etc/shadow
SHA512   : azDXLAUX+i9BVvvyk5zzJx+VmNQ/fDJ0 | 00YQkoRwN2epjEA8Il9df4SAZhfppe2rM
           6BXU+zLxtR0Q6JyLzE0t/8XoVE3fVw3u | KZoIaSDEvVppLAYuHVuWlywvcnx0tipU/
           yXSxhBpX+20KwCXgVX9x8A==           | rHXbHn8S14hLtZuF+CYXCw==

...output omitted...
```

- 12. Use the ausearch command to search the Audit log for further investigation of the changes to the /etc/shadow file that AIDE reported. Use the group_watch key that is used by your watch rule to filter for the events that are relevant to the /etc/shadow

file. Use the `ausearch` command with its `-i` option to convert numbers for UIDs and time stamps into names and more human-readable text.

The values of the timestamp and the inode that you see in the following output might differ from what you see in your system.

```
[root@servera ~]# ausearch -i -f /etc/shadow -k group_watch
...output omitted...
type=PROCTITLE msg=audit(10/16/23 01:48:26.698:354) : proctitle=passwd student
type=PATH msg=audit(10/16/23 01:48:26.698:354) : item=0 name=/etc/shadow
inode=8391344 dev=fc:04 mode=file,000 uid=root ogid=root rdev=00:00
obj=system_u:object_r:shadow_t:s0 nametype=NORMAL cap_fp=none cap_fi=none
cap_fe=0 cap_fver=0 cap_frootid=0
type=CWD msg=audit(10/16/23 01:48:26.698:354) : cwd=/root
type=SYSCALL msg=audit(10/16/23 01:48:26.698:354) :
    arch=x86_64 syscall=openat success=yes exit=4 a0=AT_FDCWD
    a1=0x7f842c67c12d a2=0_RDONLY a3=0x0 items=1 ppid=27023
    pid=27126 auid@student uid=root gid=root euid=root suid=root fsuid=root
    egid=root sgid=root fsgid=root tty pts0 ses=9 comm=passwd exe=/usr/bin/passwd
    subj=unconfined_u:unconfined_r:passwd_t:s0-s0:c0.c1023 key=group_watch
...output omitted...
```

There are several events recorded by the Audit log. Each event is separated by four dashes in the output, and the event number should appear in the output after the final colon in the time stamp (the `msg` field in each Audit record). This is Audit event 354.

This event consists of four records: PROCTITLE, PATH, CWD, and SYSCALL.

- The PROCTITLE record shows the `passwd student` command that ran on the `/etc/shadow` file shown in the PATH record.
- The PATH record shows that the `/etc/shadow` file was changed and provides information about the file such as its inode number, the major and minor numbers of the device its file system is on, its ownership and permissions, that it is a regular file, and more information.
- The CWD record gives the current working directory that shows the command was run from the `/root` directory.
- The SYSCALL record indicate that this event is logging an attempt to open the `/etc/shadow` file in read-only mode as shown in the `a2=0_RDONLY` item representing the third argument to the system call (counting from zero). The access was successful (`success=yes`). The user was on the `pts/0` pseudoterminal (possibly a graphical terminal window or ssh login) at the time. The `uid` and `euid` fields indicate that the process was running as the `root` user. However, the `auid` field indicates that the user that ran this command originally logged in as the `student` user.

This event is not the Audit event that caused the change to the `/etc/shadow` file, because in this event the `passwd` command opens the file as read-only. However, this event might be part of the sequence of Audit events that led to the change. You need to look further at the output to find the correct event.

- 13. Further down in the output of the ausearch command, another event occurred at almost the same time, which looks similar to the following output.

```
[root@servera ~]# ausearch -i -f /etc/shadow -k group_watch
...output omitted...
type=PROCTITLE msg=audit(10/16/23 01:48:26.725:358) : proctitle=passwd student
type=PATH msg=audit(10/16/23 01:48:26.725:358) : item=4 name=/etc/shadow
inode=8393250 dev=fc:04 mode=file,000 uid=root ogid=root rdev=00:00
obj=system_u:object_r:shadow_t:s0 nametype=CREATE cap_fp=none cap_fi=none
cap_fe=0 cap_fver=0 cap_frootid=0
type=PATH msg=audit(10/16/23 01:48:26.725:358) : item=3 name=/etc/shadow
inode=8391344 dev=fc:04 mode=file,000 uid=root ogid=root rdev=00:00
obj=system_u:object_r:shadow_t:s0 nametype=DELETE cap_fp=none cap_fi=none
cap_fe=0 cap_fver=0 cap_frootid=0
type=PATH msg=audit(10/16/23 01:48:26.725:358) : item=2 name=/etc/nshadow
inode=8393250 dev=fc:04 mode=file,000 uid=root ogid=root rdev=00:00
obj=system_u:object_r:shadow_t:s0 nametype=DELETE cap_fp=none cap_fi=none
cap_fe=0 cap_fver=0 cap_frootid=0
type=PATH msg=audit(10/16/23 01:48:26.725:358) : item=1 name=/etc/
inode=8388736 dev=fc:04 mode=dir,755 uid=root ogid=root rdev=00:00
obj=system_u:object_r:etc_t:s0 nametype=PARENT cap_fp=none cap_fi=none cap_fe=0
cap_fver=0 cap_frootid=0
type=PATH msg=audit(10/16/23 01:48:26.725:358) : item=0 name=/etc/
inode=8388736 dev=fc:04 mode=dir,755 uid=root ogid=root rdev=00:00
obj=system_u:object_r:etc_t:s0 nametype=PARENT cap_fp=none cap_fi=none cap_fe=0
cap_fver=0 cap_frootid=0
type=CWD msg=audit(10/16/23 01:48:26.725:358) : cwd=/root
type=SYSCALL msg=audit(10/16/23 01:48:26.725:358) : arch=x86_64 syscall=rename
success=yes exit=0 a0=0x7f842c67c38e a1=0x7f842c67c12d a2=0x558ebaa4ed53 a3=0x7
items=5 ppid=27023 pid=27126 auid=student uid=root gid=root euid=root suid=root
fsuid=root egid=root sgid=root fsgid=root tty=pts0 ses=9 comm=passwd exe=/usr/
bin/passwd subj=unconfined_u:unconfined_r:passwd_t:s0-s0:c0.c1023 key=group_watch
...output omitted...
```

The PROCTITLE record indicates that the `passwd student` command was run.

The PATH records indicate that the `/etc/shadow` file that had the 8391344 inode was deleted (item 3). In addition, the 8393250 inode was renamed from the `/etc/nshadow` file to the `/etc/shadow` file (items 2 and 4).

The CWD record gives the current working directory, and shows that the command was run from the `/root` directory.

The SYSCALL record indicates that a rename system call was successfully executed. The `passwd` command was executed by the root user, who originally logged in as the `student` user, by running the `/usr/bin/passwd` command on the `pts/0` terminal. The `passwd` command creates the `/etc/nshadow` temporary file that contains the changes, and then replaces the previous shadow password file with the temporary file. This is the event that changed the `/etc/shadow` file.

- 14. Return to the workstation machine as the `student` user.

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

Finish

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

```
[student@workstation ~]$ lab finish aide-investigating
```

▶ Lab

Monitoring File-system Changes

Configure AIDE to check for changes to file systems, and use Audit watches to identify the causes of those changes.

Outcomes

- Use AIDE to detect changes that are made to the /etc directory and its contents.
- Configure Audit to log events for changes to file access permissions in the /etc/ssh directory, and to label the log entries with a key.
- Change the /etc/ssh file, and detect those changes with AIDE.
- Use Audit tools to locate a record that shows which user and process made the changes.

Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

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

Instructions

1. Install the `aide` package as the `root` user on the `serverb` machine.
2. Edit the `/etc/aide.conf` file to detect changes that are made in the `/etc/ssh` directory. Add the line to the `ssh` section in the `/etc/aide.conf` file.

```
aide --init
```
3. Initialize the baseline AIDE database.
4. Rename the new AIDE database file from `/var/lib/aide/aide.db.new.gz` to `/var/lib/aide/aide.db.gz` so that AIDE uses the newly generated file as the current database.
5. Determine the current status of the machine's file systems.
AIDE reports no changes to the files and directories that it monitors, because you have not changed any files or directories after initializing the AIDE database.
6. Add a persistent Audit watch rule to generate Audit log entries whenever there is an attempt to read, write, execute, or change an attribute of the `/etc/ssh` directory. Use `sshd_config_monitor` as the filter key on the Audit rule.

```
auditctl -w /etc/ssh -k sshd_config_monitor
```
7. Apply the changes for the newly added Audit rules to take effect.
8. List the Audit rules and verify that the newly added Audit rule is currently in effect.

```
ausearch -k sshd_config_monitor
```
9. Make a change in the `/etc/ssh` directory by modifying the `/etc/ssh/sshd_config` file. In the `/etc/ssh/sshd_config` file, uncomment the `PasswordAuthentication` directive and change the `yes` value to the `no` value.

```
sed -i 's/PasswordAuthentication yes/PasswordAuthentication no/g' /etc/ssh/sshd_config
```

**Important**

If you make a mistake here, then you might create issues with SSH authentication on the **serverb** machine, preventing future logins that use the **ssh** service. If this mistake happens, then rebuild your lab environment and begin this exercise again from the beginning to ensure that you do not encounter further issues in future exercises.

10. Restart the **sshd** daemon to apply the new changes in the SSH service configuration file.
11. Verify the current status of the machine's file systems with AIDE to ensure that AIDE detects the change in the **/etc/ssh/sshd_config** file.
12. Investigate the Audit log to determine what changed the **/etc/ssh/sshd_config** file. Use the **sshd_config_monitor** key to limit the output.
13. Return to the **workstation** machine as the **student** user.

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 aide-review
```

Finish

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

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

► Solution

Monitoring File-system Changes

Configure AIDE to check for changes to file systems, and use Audit watches to identify the causes of those changes.

Outcomes

- Use AIDE to detect changes that are made to the /etc directory and its contents.
- Configure Audit to log events for changes to file access permissions in the /etc/ssh directory, and to label the log entries with a key.
- Change the /etc/ssh file, and detect those changes with AIDE.
- Use Audit tools to locate a record that shows which user and process made the changes.

Before You Begin

As the **student** user on the **workstation** machine, use the **lab** command to prepare your environment for this exercise, and to ensure that all required resources are available.

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

Instructions

1. Install the **aide** package as the **root** user on the **serverb** machine.
 - 1.1. From the **workstation** machine, log in to the **serverb** machine as the **student** user.

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

- 1.2. Change to the **root** user. Use **student** as the password.

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

- 1.3. Install the **aide** package.

```
[root@serverb ~]# dnf install aide
...output omitted...
Dependencies resolved.
=====
 Package Arch Version Repository Size
=====
 Installing:
```

Chapter 7 | Monitoring File-system Changes

```
aide      x86_64      0.16-100.el9      rhel-9.2-for-x86_64-appstream-rpms      154 k

Transaction Summary
=====
...output omitted...
Is this ok [y/N]: y
...output omitted...
Complete!
```

2. Edit the /etc/aide.conf file to detect changes that are made in the /etc/ssh directory. Add the line to the ssh section in the /etc/aide.conf file.

```
[root@serverb ~]# cat /etc/aide.conf
# ssh
/etc/ssh CONTENT_EX
/etc/ssh/sshd_config$ CONTENT_EX
/etc/ssh/ssh_config$ CONTENT_EX
```

3. Initialize the baseline AIDE database.

```
[root@serverb ~]# aide --init
Start timestamp: 2023-10-16 15:17:51 -0400 (AIDE 0.16)
AIDE initialized database at /var/lib/aide/aide.db.new.gz

Number of entries: 53262

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

/var/lib/aide/aide.db.new.gz
MD5      : lUxp0jxdbnY3+x1sKq8PA==
SHA1     : ImCD2brbbbRI1GxGqrmaE5To/Ww=
RMD160   : b+JFmhCsHYkdfjzbC5eYajNSI+c=
TIGER    : RUZqbTT+px0bAahjG15Zub84zsZHLzY
SHA256   : 0vQpy7LTDRNv1Ig5oSwxngJzInYQDRVc
          rqEEks9JELM=
SHA512   : Ymsu3peZuMVQyUT/+jrGD2voEZMaQYg
          GIT10AntZZV4dDdr6AOjEt4ELVXQub
          gimVw46Z1PkwfRG3jxp3DQ==

End timestamp: 2023-10-16 15:18:37 -0400 (run time: 0m 46s)
```

4. Rename the new AIDE database file from /var/lib/aide/aide.db.new.gz to /var/lib/aide/aide.db.gz so that AIDE uses the newly generated file as the current database.

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

5. Determine the current status of the machine's file systems.

AIDE reports no changes to the files and directories that it monitors, because you have not changed any files or directories after initializing the AIDE database.

```
[root@serverb ~]# aide --check
Start timestamp: 2023-10-16 15:22:29 -0400 (AIDE 0.16)
AIDE found NO differences between database and filesystem. Looks okay!!

Number of entries: 53262

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

/var/lib/aide/aide.db.gz
MD5      : lUxp0jxdbnY3+x1sKq8PA==
SHA1     : ImCD2brbbbRI1GxGqrmaE5To/Ww=
RMD160   : b+JFmhCShYkdfjzbC5eYajNSI+c=
TIGER    : RUZqBTT+pxf0bAahjG15Zub84zsZHLzY
SHA256   : 0vQpy7LTDRNv1Ig5oSwxngJzInYQDRvC
          rqEEkS9JELM=
SHA512   : Ymsu3peZuMVQyUT/+jrGD2voEZMaQYg
          GIT10AntZzV4dDdr6AOPjcEt4ELVXQUb
          gimVw46Z1PkwfRG3jxp3DQ==

End timestamp: 2023-10-16 15:23:11 -0400 (run time: 0m 42s)
```

6. Add a persistent Audit watch rule to generate Audit log entries whenever there is an attempt to read, write, execute, or change an attribute of the /etc/ssh directory. Use `sshd_config_monitor` as the filter key on the Audit rule.

```
[root@serverb ~]# cat /etc/audit/rules.d/audit.rules
...output omitted...
-w /etc/ssh -p wa -k sshd_config_monitor
```

7. Apply the changes for the newly added Audit rules to take effect.

```
[root@serverb ~]# augenrules --load
...output omitted...
```

8. List the Audit rules and verify that the newly added Audit rule is currently in effect.

```
[root@serverb ~]# auditctl -l
-w /etc/ssh -p wa -k sshd_config_monitor
```

9. Make a change in the /etc/ssh directory by modifying the /etc/ssh/sshd_config file. In the /etc/ssh/sshd_config file, uncomment the PasswordAuthentication directive and change the yes value to the no value.



Important

If you make a mistake here, then you might create issues with SSH authentication on the serverb machine, preventing future logins that use the ssh service. If this mistake happens, then rebuild your lab environment and begin this exercise again from the beginning to ensure that you do not encounter further issues in future exercises.

```
[root@serverb ~]# cat /etc/ssh/sshd_config
...output omitted...
PasswordAuthentication no
...output omitted...
```

10. Restart the sshd daemon to apply the new changes in the SSH service configuration file.

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

11. Verify the current status of the machine's file systems with AIDE to ensure that AIDE detects the change in the /etc/ssh/sshd_config file.

```
[root@serverb ~]# aide --check
Start timestamp: 2023-10-16 15:33:03 -0400 (AIDE 0.16)
AIDE found differences between database and filesystem!!

Summary:
  Total number of entries: 53263
  Added entries:  1
  Removed entries:  0
  Changed entries:  3
  ...output omitted...

Changed entries:
-----
f    ...    .C... : /etc/audit/audit.rules
f    ...    .C... : /etc/audit/rules.d/audit.rules
f    ...    .C... : /etc/ssh/sshd_config

-----
Detailed information about changes:
-----
...output omitted...
File: /etc/ssh/sshd_config
  SHA512   : Cs01Y1exozdL381/r0KJ7UwoPqu08LYe | 3uQTXYd1wnvJNs+GuLQc84Q9z16yzd0c
              Us+4qpsVDYJMSfePesfYZDHvSKzAEsRt | HKzXcq5/tNmufAF2V+JxG/dQQNu61h4I+
              uLiLiWwiIcrl/8R/0dfocKg==           | uABk/lSwCr0pSrxBMwX/4Q==
  ...output omitted...
```

Chapter 7 | Monitoring File-system Changes

12. Investigate the Audit log to determine what changed the /etc/ssh/sshd_config file. Use the sshd_config_monitor key to limit the output.

```
[root@serverb ~]# ausearch -i -f /etc/ssh/sshd_config -k sshd_config_monitor
...output omitted...
type=PROCTITLE msg=audit(10/16/23 15:30:47.188:257) : proctitle=vim /etc/ssh/
sshd_config
type=PATH msg=audit(10/16/23 15:30:47.188:257) : item=3 name=/etc/ssh/sshd_config~
inode=16809537 dev=fc:04 mode=file,600 uid=root ogid=root rdev=00:00
obj=system_u:object_r:etc_t:s0 nametype=CREATE cap_fp=none cap_fi=none cap_fe=0
cap_fver=0 cap_frootid=0
type=PATH msg=audit(10/16/23 15:30:47.188:257) : item=2 name=/etc/ssh/sshd_config
inode=16809537 dev=fc:04 mode=file,600 uid=root ogid=root rdev=00:00
obj=system_u:object_r:etc_t:s0 nametype=DELETE cap_fp=none cap_fi=none cap_fe=0
cap_fver=0 cap_frootid=0
type=PATH msg=audit(10/16/23 15:30:47.188:257) : item=1 name=/etc/ssh/
inode=16797950 dev=fc:04 mode=dir,755 uid=root ogid=root rdev=00:00
obj=system_u:object_r:etc_t:s0 nametype=PARENT cap_fp=none cap_fi=none cap_fe=0
cap_fver=0 cap_frootid=0
type=PATH msg=audit(10/16/23 15:30:47.188:257) : item=0 name=/etc/ssh/
inode=16797950 dev=fc:04 mode=dir,755 uid=root ogid=root rdev=00:00
obj=system_u:object_r:etc_t:s0 nametype=PARENT cap_fp=none cap_fi=none cap_fe=0
cap_fver=0 cap_frootid=0
type=CWD msg=audit(10/16/23 15:30:47.188:257) : cwd=/root
type=SYSCALL msg=audit(10/16/23 15:30:47.188:257) :
arch=x86_64 syscall=rename success=yes exit=0 a0=0x556420fce810 a1=0x556421251630
a2=0xfffffffffffffe98 a3=0x0 items=4 ppid=26526 pid=27113 auid@student uid=root
gid=root euid=root suid=root egid=root sgid=root fsgid=root tty=pts0
ses=5 comm=vim exe=/usr/bin/vim subj=unconfined_u:unconfined_r:unconfined_t:s0-
s0:c0.c1023 key:sshd_config_monitor
...output omitted...
```

13. Return to the workstation machine as the student user.

```
[root@serverb ~]# logout
[student@serverb ~]$ logout
Connection to serverb closed.
[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 aide-review
```

Finish

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

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

Summary

- AIDE detects changes that are made to a machine's file systems by using a database of baseline information.
- An AIDE check can be run manually, or it can be scheduled with a tool such as crontab.
- You can configure AIDE checks against specific files and directories by using group definitions, selection lines, and macros in the /etc/aide.conf file.
- You must rebuild the AIDE database file to accept authorized changes to files and to apply new settings from the configuration file.
- You can use Audit with AIDE to help you determine what process or user caused a change that AIDE reported.

Chapter 8

Mitigating Risk with SELinux

Goal

Improve security and confinement between processes by using SELinux and advanced SELinux techniques and analysis.

Sections

- Enabling SELinux from the Disabled State (and Guided Exercise)
- Controlling Access with Confined Users (and Guided Exercise)
- Auditing the SELinux Policy (and Guided Exercise)
- Mitigating Risk with SELinux

Lab

Enabling SELinux from the Disabled State

Objectives

- Configure SELinux in enforcing mode on a server that has been running with SELinux disabled.

SELinux Operating Modes

SELinux can work in one of three modes: `enforcing`, `permissive`, and `disabled`.

Enforcing

In this mode, SELinux enforces the currently loaded security policy on the system. SELinux logs attempted SELinux access violations and protects the system from them. This mode is the default, and Red Hat recommends that you use it.

Permissive

In permissive mode, SELinux does not block access violations. However, SELinux is still enabled and it otherwise operates according to the loaded security policy; it assigns SELinux labels to files, ports, and processes as usual, and SELinux access violations are logged.

Disabled

SELinux can be disabled. When it is disabled, nothing enforces the SELinux policy, and SELinux labels are not assigned to new files, to ports, or to processes. This mode can make it difficult to enable SELinux in the future. Disabled mode is strongly discouraged.

Use the `getenforce` command to display the SELinux mode that is in effect.

```
[root@host ~]# getenforce  
Enforcing
```

Use the `setenforce` command to switch between the `enforcing` and `permissive` modes. Changes made with the `setenforce` command do not persist across reboots.

```
[root@host ~]# setenforce  
usage: setenforce [ Enforcing | Permissive | 1 | 0 ]  
[root@host ~]# setenforce 0  
[root@host ~]# getenforce  
Permissive  
[root@host ~]# setenforce 1  
[root@host ~]# getenforce  
Enforcing
```

To persistently set the mode at boot time, edit the `/etc/selinux/config` configuration file.

```
# This file controls the state of SELinux on the system.  
# SELINUX= can take one of these three values:  
#       enforcing - SELinux security policy is enforced.  
#       permissive - SELinux prints warnings instead of enforcing.  
#       disabled - No SELinux policy is loaded.  
# See also:
```

```
# https://docs.fedoraproject.org/en-US/quick-docs/getting-started-with-selinux/
#getting-started-with-selinux-selinux-states-and-modes
#
# NOTE: In earlier Fedora kernel builds, SELINUX=disabled would also
# fully disable SELinux during boot. If you need a system with SELinux
# fully disabled instead of SELinux running with no policy loaded, you
# need to pass selinux=0 to the kernel command line. You can use grubby
# to persistently set the bootloader to boot with selinux=0:
#
#     grubby --update-kernel ALL --args selinux=0
#
# To revert back to SELinux enabled:
#
#     grubby --update-kernel ALL --remove-args selinux
#
SELINUXTYPE=enforcing
# SELINUXTYPE= can take one of these three values:
#     targeted - Targeted processes are protected,
#     minimum - Modification of targeted policy. Only selected processes are
protected.
#     mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

Disabling SELinux

When SELinux is disabled, SELinux policy is not loaded at all; the security policy is not enforced and Access Vector Cache (AVC) messages are not logged. Therefore, all the benefits of running SELinux are lost.



Warning

If you choose not to use enforcing mode, then Red Hat strongly recommends that you use permissive mode instead of disabling SELinux.

Install the grubby package:

```
[root@host ~]# dnf install grubby
...output omitted...
```

Configure your bootloader to add the `selinux=0` parameter to the kernel command line:

```
[root@host ~]# grubby --update-kernel ALL --args selinux=0
```

After reboot, confirm that the `getenforce` command returns the `Disabled` output:

```
[root@host ~]# getenforce
Disabled
```

Enabling SELinux from Disabled Mode

When your systems have been running with SELinux disabled and you want to enable SELinux, you must proceed with care to avoid having problems booting the system or executing normal application operations.



Warning

When systems run SELinux in permissive mode, users and processes might label various file-system objects incorrectly. File-system objects that are created when SELinux is disabled are not labeled at all. This behavior causes problems when changing to enforcing mode because SELinux relies on file-system objects being correctly labeled.

To prevent incorrectly labeled and unlabeled files from causing problems, SELinux automatically relabels file systems when changing from the disabled state to permissive or enforcing mode.

Before rebooting the system for relabeling, ensure that the system boots in permissive mode, for example by using the `enforcing=0` kernel option. This mode prevents the system from failing to boot in case the system contains unlabeled files that are required by the `systemd` daemon before it starts the `selinux-autorelabel` service.

The following steps provide an overview of how to successfully enable SELinux on a system that has been operating with SELinux disabled.

Configure SELinux to use permissive mode by editing the `/etc/selinux/config` file.

```
...output omitted...
SELINUX=permissive
...output omitted...
```

Use the `grubby` command to remove the `selinux` kernel parameter if it is present. If this parameter is present, then this configuration is probably how SELinux was disabled.

```
[root@host ~]# grubby --update-kernel ALL --remove-args selinux
```

Reboot the system to apply the changes. Check for SELinux denial messages.

```
[root@host ~]# ausearch -m AVC,USER_AVC,SELINUX_ERR,USER_SELINUX_ERR -ts recent
<no matches>
[root@host ~]# journalctl -t setroubleshoot
-- No entries --
```

Ensure that files are relabeled upon the next reboot. The `fixfiles` command `-F` option creates the `/.autorelabel` file.

```
[root@host ~]# fixfiles -F onboot
System will relabel on next boot
```

Reboot the system after running the `fixfiles` command.

Check for SELinux denial messages again, and fix any that remain. If there are no denials, then switch to enforcing mode in the `/etc/selinux/config` file.

```
...output omitted...
SELINUX=enforcing
...output omitted...
```

Reboot the system to apply the changes. After the system reboots, confirm that the system still works correctly.

```
[root@host ~]# getenforce
Enforcing
```

Resolving SELinux Issues

As you prepare to move your system from permissive to enforcing mode, you might need to resolve various SELinux access violations so that your applications and the system continue to operate correctly. There are a number of things that you might need to do before putting the system in SELinux enforcing mode.

SELinux Access Violation Audit Events

When SELinux denies access to a resource, it also logs an Audit event. If the `auditd` service is started, then the system logs SELinux Audit events in the `/var/log/audit/audit.log` file. Otherwise, the system sends SELinux messages to the `/var/log/messages` file.



Note

In Red Hat Enterprise Linux, the `audit` package is installed by default unless you explicitly remove it from the package selection during installation.

The following example extracts the SELinux denials from the Audit log file:

```
[root@host ~]# grep denied /var/log/audit/audit.log
...output omitted...
type=AVC msg=audit(1698377265.596:217): avc: denied { getattr } for
pid=27484 comm="httpd" path="/webserver/index.html" dev="vda4" ino=9393827
scontext=system_u:system_r:httpd_t:s0 tcontext=unconfined_u:object_r:default_t:s0
tclass=file permissive=0
```

You can also use the `ausearch` command to verify log events for SELinux. The `ausearch` command `-m AVC` and `-ts boot` options filter the output to display only messages from SELinux, and from the last system boot. With the `-ts` option, you can use the `recent` and `today` parameters to display only messages from the last 10 minutes, or from midnight.

```
[root@host ~]# ausearch -m AVC -ts boot
---
time->Mon Oct 30 06:32:56 2023
type=PROCTITLE msg=audit(1698661976.435:87):
proctitle=2F7573722F7362696E2F6874747064002D44464F524547524F554E44
```

```

type=SYSCALL msg=audit(1698661976.435:87): arch=c000003e syscall=262
  success=no exit=-13 a0=fffffff9c a1=564301854b70 a2=7fcac87f78b0 a3=0 items=0
  ppid=796 pid=830 auid=4294967295 uid=48 gid=48 euid=48 suid=48 fsuid=48 egid=48
  sgid=48 fsgid=48 tty=(none) ses=4294967295 comm="httpd" exe="/usr/sbin/httpd"
  subj=system_u:system_r:httpd_t:s0 key=(null)
type=AVC msg=audit(1698661976.435:87): avc: denied { getattr } for
  pid=830 comm="httpd" path="/webserver/index.html" dev="vda4" ino=8412944
  scontext=system_u:system_r:httpd_t:s0 tcontext=unconfined_u:object_r:default_t:s0
  tclass=file permissive=0
-----
time->Mon Oct 30 06:32:56 2023
type=PROCTITLE msg=audit(1698661976.435:88):
  proctitle=2F7573722F7362696E2F6874747064002D44464F524547524F554E44
type=SYSCALL msg=audit(1698661976.435:88): arch=c000003e syscall=262
  success=no exit=-13 a0=fffffff9c a1=564301854c48 a2=7fcac87f78b0 a3=100 items=0
  ppid=796 pid=830 auid=4294967295 uid=48 gid=48 euid=48 suid=48 fsuid=48 egid=48
  sgid=48 fsgid=48 tty=(none) ses=4294967295 comm="httpd" exe="/usr/sbin/httpd"
  subj=system_u:system_r:httpd_t:s0 key=(null)
type=AVC msg=audit(1698661976.435:88): avc: denied { getattr } for
  pid=830 comm="httpd" path="/webserver/index.html" dev="vda4" ino=8412944
  scontext=system_u:system_r:httpd_t:s0 tcontext=unconfined_u:object_r:default_t:s0
  tclass=file permissive=0

```

The previous output shows two Audit events with an AVC Audit record. Both events have the 830 PID, and they are attempts to access the /webserver/index.html file with the /usr/sbin/httpd daemon. SELinux blocked the `getattr` attempt on that file.

In enforcing mode, unlabeled files or mislabeled context types might cause problems with accessing files or directories. The httpd process has the `httpd_t` SELinux domain, and the httpd process is trying to read a file of the `default_t` type.

Changing SELinux Contexts for Files and Directories

Usually, a newly created file inherits the context of its parent directory.

The following example shows the context of the /etc/httpd/conf.d directory:

```

[root@host ~]# ls -Zd /etc/httpd/conf.d/
system_u:object_r:httpd_config_t:s0 /etc/httpd/conf.d/
[root@host ~]# touch /etc/httpd/conf.d/test.conf
[root@host ~]# ls -Z /etc/httpd/conf.d/test.conf
unconfined_u:object_r:httpd_config_t:s0 /etc/httpd/conf.d/test.conf

```

In addition, the SELinux policy has other file context rules that help to ensure that files are assigned the correct labels when the files are created. Therefore, you rarely need to change directory or file contexts manually.

Sometimes, however, files get the wrong label, or you create a custom directory for your application that requires the correct context. In those situations, you must relabel the files and directories.

The following log message is a typical SELinux denial due to a mislabeled file.

```
[root@host ~]# ausearch -m AVC,USER_AVC
...output omitted...
type=AVC msg=audit(1698377265.596:217): avc: denied { getattr } for
pid=27484 comm="httpd" path="/webserver/index.html" dev="vda4" ino=9393827
scontext=system_u:system_r:httpd_t:s0 tcontext=unconfined_u:object_r:default_t:s0
tclass=file permissive=0
```

The message gives the name, the PID of the process, the path, the device, and the inode of the file or directory that the process tries to access. The `tclass` attribute specifies the type of the targeted object, which is either a file or a directory. The `scontext` attribute, or *source context*, is the context of the process, and `tcontext`, or *target context*, is the actual context of the file or directory.

This message indicates that SELinux denies the `httpd` process, whose PID is 27484 and whose domain is `httpd_t`, access to the `/webserver/index.html` file, whose inode number is 9393827 on the `vda4` device, and with the `default_t` context type.

Defining Custom SELinux File Context Rules

When you run the `fixfiles -F` command and reboot the system, the files on the system are relabeled based on the SELinux policy's file context rules. File context rules already exist in SELinux for the standard system files and directories.

However, sometimes your application might have its own custom files in nonstandard locations, and you might need to add your own custom file context rules to ensure that the files are automatically relabeled correctly.

The following example lists the file context rules that exist.

```
[root@host ~]# semanage fcontext -l
...output omitted...
/var/www/(.*)?                                all files
    system_u:object_r:httpd_sys_content_t:s0
...output omitted...
```

Rules use extended regular expressions to specify the path and file names. The most common extended regular expression is `(/ .*)?`, which means "optionally, match a / directory followed by any number of characters". The regular expression matches the directory listed before the expression and everything in that directory recursively.

When you create a custom directory for your application, you must also add a rule. To add a new rule, use the `semanage fcontext -a` command.

The `restorecon` command -v option lists the relabeled files. The -R option recursively restores the contexts on directories.

With the `restorecon` command, you do not specify the context to apply. The command uses rules in the SELinux policy to determine the context of the file.

```
[root@host ~]# mkdir /virtual/
[root@host ~]# touch /virtual/index.html
[root@host ~]# ls -Zd /virtual/
unconfined_u:object_r:default_t:s0 /virtual/
[root@host ~]# ls -Z /virtual/index.html
```

Chapter 8 | Mitigating Risk with SELinux

```
unconfined_u:object_r:default_t:s0 /virtual/index.html
[root@host ~]# semanage fcontext -a -t httpd_sys_content_t '/virtual(/.)?/*'
[root@host ~]# restorecon -Rv /virtual/
Relabeled /virtual from unconfined_u:object_r:default_t:s0 to
unconfined_u:object_r:httpd_sys_content_t:s0
Relabeled /virtual/index.html from unconfined_u:object_r:default_t:s0 to
unconfined_u:object_r:httpd_sys_content_t:s0
[root@host ~]# ls -Zd /virtual/
unconfined_u:object_r:httpd_sys_content_t:s0 /virtual/
[root@host ~]# ls -Z /virtual/index.html
unconfined_u:object_r:httpd_sys_content_t:s0 /virtual/index.html
```

Labeling SELinux Ports

Whenever a process tries to listen on a port, SELinux verifies whether the label that is associated with that process, the domain, is allowed to bind to that port label. This port label stops a rogue service from taking over ports that are otherwise used by other legitimate network services.

If you run a service on a nonstandard port, then you must update the SELinux port labels.

For example, to control access to the 22 TCP port that is used by the `ssh` service, the port must use the `ssh_port_t` label.

By default, some ports are labeled with a type that you can use. For example, a web server might use the 8080 TCP port, which by default has the `http_cache_port_t` label.

```
[root@host ~]# semanage port -l | grep 8080
http_cache_port_t          tcp      8080, 8118, 8123, 10001-10010
```

The following log message is a typical SELinux denial due to a mislabeled port.

```
[root@host ~]# ausearch -m AVC,USER_AVC
...output omitted...
type=AVC msg=audit(1698380012.436:267): avc: denied { name_bind } for
pid=28420 comm="httpd" src=3131 scontext=system_u:system_r:httpd_t:s0
tcontext=system_u:object_r:unreserved_port_t:s0 tclass=tcp_socket permissive=0
```

The message provides the name and PID of the process, and the port that the process tries to access. The `tclass` attribute specifies the type of the targeted object: a TCP socket in this example. The `src` attribute indicates the port number. SELinux denies the `name_bind` system operation.

To fix such mislabeling issues, use the `semanage port -a` command.

```
[root@host ~]# semanage port -a -t http_port_t -p tcp 3131
[root@host ~]# semanage port -l | grep 3131
http_port_t          tcp      3131, 80, 81, 443, 488, 8008, 8009, 8443,
9000
```

Switching SELinux Booleans

SELinux Booleans are switches that change the behavior of the SELinux policy. These Booleans enable or disable sets of SELinux access rules. Booleans can be used by security administrators

to selectively adjust the policy. You might need to set a Boolean to a particular value to allow processes in a specific SELinux domain to work with files of a certain SELinux type, for example.

Use the `getsebool` command to display SELinux Booleans, and the `setsebool` command to modify SELinux Booleans. The `setsebool -P` command updates the SELinux policy to make the modification persistent.

```
[root@host ~]# getsebool -a
abrt_anon_write --> off
abrt_handle_event --> off
abrt_upload_watch_anon_write --> on
...output omitted...
[root@host ~]# getsebool httpd_enable_homedirs
httpd_enable_homedirs --> off
[root@host ~]# setsebool -P httpd_enable_homedirs on
[root@host ~]# getsebool httpd_enable_homedirs
httpd_enable_homedirs --> on
```

Using Permissive Domains

In permissive mode, SELinux logs the access that it must deny, but allows the access anyway. Switching the system into permissive mode, perhaps to debug a specific process, might have security consequences, because all the other services are now also unprotected by SELinux.

When migrating a system from `disabled` to `enforcing` mode, you must resolve AVC issues for all except one or two SELinux domains.

You can keep your system in `enforcing` mode and only set a specific domain or domains to `permissive` mode. This allows you to protect most of the system with SELinux, but the services using that permissive domain would *not* be protected by SELinux.

Use the `semanage permissive -a <domain>` command to set a specific domain to `permissive` mode.

The following command sets the Apache `default_t` domain to `permissive` mode.

```
[root@host ~]# semanage permissive -a default_t
```

Use the `semanage permissive -l` command to list the domains that are in `permissive` mode.

```
[root@host ~]# semanage permissive -l

Builtin Permissive Types

mptcpd_t
rshim_t

Customized Permissive Types

default_t
insights_client_t
rhcd_t
```

Chapter 8 | Mitigating Risk with SELinux

Use the `semanage permissive -d <domain>` command to switch a domain back to enforcing mode. The following command changes the `default_t` domain to enforcing mode.

```
[root@host ~]# semanage permissive -d default_t
libsemanage.semanage_direct_remove_key: Removing last permissive_default_t module
(no other permissive_default_t module exists at another priority).
```

Automating SELinux Issue Remediation with Ansible

RHEL System Roles is a collection of Ansible roles and modules that provide a consistent configuration interface to remotely manage multiple systems. The `selinux` System Role enables the following actions:

- Cleaning local policy modifications related to SELinux Booleans, file contexts, ports, and logins.
- Setting SELinux policy Booleans, file contexts, ports, and logins.
- Restoring file contexts on specified files or directories.
- Managing SELinux modules.

For a detailed reference on the `selinux` role variables, install the `rhel-system-roles` package, and see the `README.md` or `README.html` files in the `/usr/share/doc/rhel-system-roles/selinux/` directory.

The `/usr/share/doc/rhel-system-roles/selinux/example-selinux-playbook.yml` example playbook installed by the `rhel-system-roles` package demonstrates how to set the targeted policy in enforcing mode. The playbook also applies several local policy modifications and restores file contexts in the `/tmp/test_dir/` directory.

```
---
- name: Manage SELinux policy example
  hosts: all
  vars:
    # Use "targeted" SELinux policy type
    selinux_policy: targeted
    # Set "enforcing" mode
    selinux_state: enforcing
    # Switch some SELinux booleans
    selinux_booleans:
      # Set the 'samba_enable_home_dirs' boolean to 'on' in the current
      # session only
      - {name: 'samba_enable_home_dirs', state: 'on'}
      # Set the 'ssh_sysadm_login' boolean to 'on' permanently
      - {name: 'ssh_sysadm_login', state: 'on', persistent: 'yes'}
    # Map '/tmp/test_dir' and its subdirectories to the 'user_home_dir_t'
    # SELinux file type
    selinux_fcontexts:
      - {target: '/tmp/test_dir(/.*)?', setype: 'user_home_dir_t', ftype: 'd'}
    # Restore SELinux file contexts in '/tmp/test_dir'
    selinux_restore_dirs:
      - /tmp/test_dir
    # Map tcp port 22100 to the 'ssh_port_t' SELinux port type
    selinux_ports:
      - {ports: '22100', proto: 'tcp', setype: 'ssh_port_t', state: 'present'}
```

```

# Map the 'sar-user' Linux user to the 'staff_u' SELinux user
selinux_logins:
  - {login: 'sar-user', seuser: 'staff_u', serange: 's0-s0:c0.c1023',
    state: 'present'}
# Manage modules
selinux_modules:
  # Install the 'localpolicy.cil' module with priority 300
  - {path: "localpolicy.cil", priority: "300", state: "enabled"}
  # Disable the 'unconfineduser' module with priority 100
  - {name: "unconfineduser", priority: "100", state: "disabled"}
  # Remove the 'temporarypolicy' module with priority 400
  - {name: "temporarypolicy", priority: "400", state: "absent"}
roles:
  - rhel-system-roles.selinux

```

Accessing SELinux Documentation

Each SELinux command has a man page. For example, the man page for the `semanage fcontext` command is the `semanage-fcontext(8)` man page, and the `semanage port` command is documented in the `semanage-port(8)` man page.

Some domains, such as the `httpd_t` and `sshd_t` domains, have dedicated man pages that describe all their types and Booleans. You can also build the man pages by installing the `policycoreutils-devel` package and running the `sepolicy manpage` command with the following options:

- `-a` to build all the man pages.
- `-d <domain>` to build the man page of a specific domain. For example:

```
[root@host ~]# sepolicy manpage -d httpd_t
/tmp/apache_selinux.8
/tmp/httpd_selinux.8
```

- `-p <path>` to create the man pages in a specific directory: for example, in the `/usr/share/man/man8` directory. By default, the `sepolicy manpage` command generates the man pages in the `/tmp` directory.



References

The `semanage(8)`, `restorecon(8)`, `getenforce(8)`, `setenforce(8)`, `ausearch(8)`, `grubby(8)`, `getsebool(8)`, and `setsebool(8)` man pages

For more information, refer to the *Getting Started With SELinux* chapter in the *Using SELinux* guide at
https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html/using_selinux/getting-started-with-selinux_using-selinux

For more information, refer to the *Deploying the Same SELinux Configuration on Multiple Systems* chapter in the *Using SELinux* guide at
https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html/using_selinux/deploying-the-same-selinux-configuration-on-multiple-systems_using-selinux

► Guided Exercise

Enabling SELinux from the Disabled State

Enable enforcing mode for SELinux on a server that has been running with SELinux disabled.

Outcomes

- Use the `grubby` command to disable and enable SELinux.
- Modify the SELinux modes.
- Inspect AVC messages by using the Audit system.
- Ensure the correct operation of systems that are using enforcing mode.

Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

```
[student@workstation ~]$ lab start selinux-enabling
```

Instructions

- 1. Log in to the `servere` machine as the `student` user and use the `sudo -i` command to switch to the `root` account. The password for the `student` user account is `student`.

```
[student@workstation ~]$ ssh student@servere
[student@servere ~]$ sudo -i
[sudo] password for student: student
[root@servere ~]#
```

- 2. Confirm that SELinux is disabled on the `servere` machine and verify that the `test.html` file is accessible.

2.1. Confirm that SELinux is in `Disabled` mode.

```
[root@servere ~]# getenforce
Disabled
```

2.2. Use the `curl` command to access the `test.html` file on the `servere` machine.

```
[root@servere ~]# curl http://localhost/test.html
<html>
  <head>SELinux</head>
  <body>
    <h1>Works!</h1>
  </body>
</html>
```

- 3. Enable SELinux in permissive mode, and reboot the server.

- 3.1. Enable SELinux in permissive mode in the /etc/selinux/config file.

```
...output omitted...
SELINUX=permissive
...output omitted...
```

- 3.2. Use the grubby command to enable SELinux by removing the argument that disables SELinux from the kernel command line.

```
[root@servere ~]# grubby --update-kernel ALL --remove-args selinux
```

- 3.3. Reboot the servere machine.

```
[root@servere ~]# reboot
Connection to servere closed.
[student@workstation ~]$
```

- 4. Verify that the SELinux mode on the servere machine is set to Permissive mode.

- 4.1. Log in to the servere machine as the student user.

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

- 4.2. Change to the root user. Use student as the password.

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

- 4.3. Verify that SELinux is in Permissive mode.

```
[root@servere ~]# getenforce
Permissive
```

- 5. Relabel all the files upon the next reboot.

- 5.1. Relabel all the files.

```
[root@servere ~]# fixfiles -F onboot
System will relabel on next boot
```

- 5.2. Reboot the servere machine.

```
[root@servere ~]# reboot
Connection to servere closed.
[student@workstation ~]$
```

► 6. Check for SELinux denial messages.

- 6.1. Log in to the
- servere**
- machine as the
- student**
- user.

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

- 6.2. Use the
- curl**
- command to access the
- test.html**
- file on the
- servere**
- machine.

```
[student@servere ~]$ curl http://localhost/test.html
<html>
  <head>SELinux</head>
  <body>
    <h1>Works!</h1>
  </body>
</html>
```

- 6.3. Change to the
- root**
- user. Use
- student**
- as the password.

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

- 6.4. Check the Audit log file for any SELinux denial messages. The
- curl**
- command in the previous step was successful because SELinux is set to
- permissive**
- mode, but the command did generate an AVC denied message in the log. This denied message means that the
- curl http://localhost/test.html**
- command fails if SELinux is set to
- enforcing**
- mode.

```
[root@servere ~]# ausearch -m AVC,USER_AVC,SELINUX_ERR,USER_SELINUX_ERR -ts recent
...output omitted...
time->Tue Nov 21 15:34:36 2023
type=PROCTITLE msg=audit(1700598876.985:553):
  proctitle=2F7573722F7362696E2F6874747064002D44464F524547524F554E44
type=SYSCALL msg=audit(1700598876.985:553): arch=c000003e syscall=9 success=yes
  exit=140467566579712 a0=0 a1=6c a2=1 a3=1 items=0 ppid=4050 pid=4139
  auid=4294967295 uid=48 gid=48 euid=48 suid=48 fsuid=48 egid=48 sgid=48
  fsgid=48 tty=(none) ses=4294967295 comm="httpd" exe="/usr/sbin/httpd"
  subj=system_u:system_r:httpd_t:s0 key=(null)
type=AVC msg=audit(1700598876.985:553): avc: denied { map } for
  pid=4139 comm="httpd" path="/web/test.html" dev="vda4" ino=25195768
  scontext=system_u:system_r:httpd_t:s0 tcontext=unconfined_u:object_r:default_t:s0
  tclass=file permissive=1
...output omitted...
```

- 6.5. Check the journal log file for any SELinux denial messages.

```
[root@servere ~]# journalctl -t setroubleshoot
-- No entries --
```

► 7. Change SELinux to **enforcing** mode, and reboot the server.

- 7.1. Change SELinux to enforcing mode in the /etc/selinux/config file.

```
...output omitted...
SELINUX=enforcing
...output omitted...
```

- 7.2. Reboot the servere machine.

```
[root@servere ~]$ reboot
Connection to servere closed.
[student@workstation ~]$
```

- 8. Verify that the SELinux mode on servere is set to Enforcing mode and determine whether the test.html file is accessible.

- 8.1. Log in to the servere machine as the student user.

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

- 8.2. Change to the root user. Use student as the password.

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

- 8.3. Verify that the SELinux mode on the servere machine is set to Enforcing mode.

```
[root@servere ~]$ getenforce
Enforcing
```

- 8.4. Use the curl command to determine whether the test.html file is accessible.

```
[root@servere ~]# curl http://localhost/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>
```

- 9. Check for the SELinux denial messages. Also, check the SELinux context for the document root directory of the httpd service.

- 9.1. Check the /var/log/audit/audit.log file for AVC denials. In enforcing mode, the curl command generates an AVC denied entry that is similar to the one from a previous step, but this time, the command fails.

```
[root@servere ~]# grep denied /var/log/audit/audit.log
...output omitted...
type=AVC msg=audit(1698002546.646:386): avc: denied
{ getattr } for pid=28806 comm="httpd" name="test.html"
dev=vda4 ino=151760 scontext=system_u:system_r:httpd_t:s0
tcontext=unconfined_u:object_r:default_t:s0 tclass=file permissive=0
type=AVC msg=audit(1698002546.646:386): avc: denied
{ getattr } for pid=28806 comm="httpd" path="/web/test.html"
dev=vda4 ino=151760 scontext=system_u:system_r:httpd_t:s0
tcontext=unconfined_u:object_r:default_t:s0 tclass=file permissive=0
...output omitted...
```

9.2. Retrieve the document root configuration from the httpd configuration file.

```
[root@servere ~]# grep ^DocumentRoot /etc/httpd/conf/httpd.conf
DocumentRoot "/web"
```

9.3. Check the SELinux context for the /web directory.

```
[root@servere ~]# ls -laz /web
total 4
drwxr-xr-x. 2 root root unconfined_u:object_r:default_t:s0 23 Oct 20 20:09 .
dr-xr-xr-x. 20 root root system_u:object_r:root_t:s0 257 Oct 20 18:39 ..
-rw-r--r--. 1 root root unconfined_u:object_r:default_t:s0 72 Oct 20 19:41
test.html
```

9.4. Verify the SELinux context for the /var/www/html directory.

```
[root@servere ~]# ls -laz /var/www/html/
total 0
drwxr-xr-x. 2 root root system_u:object_r:httpd_sys_content_t:s0 6 Mar 29 2023 .
drwxr-xr-x. 4 root root system_u:object_r:httpd_sys_content_t:s0 33 Oct 20
19:59 ..
```

► 10. Change the SELinux context of the document root directory to the httpd_sys_content_t type.

10.1. Use the semanage fcontext command to add the httpd_sys_content_t type to the /web directory.

```
[root@servere ~]# semanage fcontext -a -t httpd_sys_content_t '/web(/.*)?'
```

10.2. Use the restorecon command to apply the context changes recursively to the /web directory.

```
[root@servere ~]# restorecon -Rv /web
Relabeled /web from unconfined_u:object_r:default_t:s0 to
unconfined_u:object_r:httpd_sys_content_t:s0
Relabeled /web/test.html from unconfined_u:object_r:default_t:s0 to
unconfined_u:object_r:httpd_sys_content_t:s0
```

- 10.3. Verify that the /web directory has the httpd_sys_content_t type.

```
[root@servere ~]# ls -laz /web
total 4
drwxr-xr-x. 2 root root unconfined_u:object_r:httpd_sys_content_t:s0 23 Oct 20
20:09 .
dr-xr-xr-x. 20 root root system_u:object_r:root_t:s0 257 Oct 20
18:39 ..
-rw-r--r--. 1 root root unconfined_u:object_r:httpd_sys_content_t:s0 72 Oct 20
19:41 test.html
```

- 11. Use the curl command to verify that the test.html file is accessible.

```
[root@servere ~]# curl http://localhost/test.html
<html>
  <head>SELinux</head>
  <body>
    <h1>Works!</h1>
  </body>
</html>
```

- 12. Return to the workstation machine as the student user.

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

Finish

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

```
[student@workstation ~]$ lab finish selinux-enabling
```

Controlling Access with Confined Users

Objectives

- Limit user access to the system and the `root` account by configuring those users as confined users.

Defining SELinux Users

The SELinux policy defines its own SELinux users, which are distinct from Linux users. When a Linux user logs in, they are mapped to exactly one SELinux user. Similar to the Audit UID, the SELinux user cannot be changed during a login session. Normally, many Linux users are mapped to the same SELinux user. The policy can place additional SELinux-enforced restrictions on what particular SELinux users can do.

You can list the SELinux users with the `semanage user -l` command.

```
[root@host ~]# semanage user -l

          Labeling  MLS/
SELinux User  Prefix    MCS Level ...  SELinux Roles

guest_u       user      s0        ...  guest_r
root          user      s0        ...  staff_r sysadm_r system_r unconfined_r
staff_u       user      s0        ...  staff_r sysadm_r system_r unconfined_r
sysadm_u     user      s0        ...  sysadm_r
system_u     user      s0        ...  system_r unconfined_r
unconfined_u  user      s0        ...  system_r unconfined_r
user_u        user      s0        ...  user_r
xguest_u     user      s0        ...  xguest_r
```

Each SELinux user has access to a set of SELinux roles, which are shown in the last column. These roles ultimately define which programs an SELinux user can run. For example, the `sysadm_r` role allows the use of the `su` and `sudo` commands. The `xguest_r` role restricts the commands that the user can use and only allows network access through the Firefox web browser.

The goal of the SELinux user feature is to more tightly control the programs that a user can run.

Mapping Linux Users to SELinux Users

At login time, SELinux maps the Linux user to an SELinux user. This way, the Linux user inherits the restrictions that are assigned to their associated SELinux user.

The `semanage login -l` command displays the table that SELinux uses for this mapping.

[root@host ~]# semanage login -l			
Login Name	SELinux User	MLS/MCS Range	Service
__default__	unconfined_u	s0-s0:c0.c1023	*
root	unconfined_u	s0-s0:c0.c1023	*
system_u	system_u	s0-s0:c0.c1023	*

As shown in this output, SELinux maps the Linux `root` user to the `unconfined_u` SELinux user. The `__default__` entry instructs SELinux to map all the Linux users that are not explicitly mapped to an SELinux user to the `unconfined_u` SELinux user. Linux users that are mapped to the `unconfined_u` (the *unconfined user*) SELinux user do not have additional user-based SELinux restrictions. Other SELinux policy restrictions still apply.

SELinux uses the `system_u` user for system services. Do not use it for your Linux users.



Note

By default, on a new Red Hat Enterprise Linux installation, Linux users are mapped to the `unconfined_u` SELinux user. This SELinux user does not have additional user-specific rules confining it.

Linux users who are logged in can retrieve their associated SELinux user with the `id -Z` command.

```
[root@host ~]# id -Z  
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

To map an existing Linux user to an SELinux user, use the `semanage login -a -s SELinux_user Linux_user` command.

```
[root@host ~]# semanage login -a -s sysadm_u operator1
```

To remove the mapping, use the `semanage login -d -s SELinux_user Linux_user` command.

```
[root@host ~]# semanage login -d -s sysadm_u operator1
```

To modify the default mapping, use the `semanage login -m -s SELinux_user -r s0 __default__` command.

```
[root@host ~]# semanage login -m -s user_u -r s0 __default__
```



Important

The `__default__` name has two underscores on either side of the word "default".

You can also map a new Linux user at creation time by using the `-Z` option of the `useradd` command.

```
[root@host ~]# useradd -z staff_u developer1
```

Comparing the SELinux Users

On Red Hat Enterprise Linux, SELinux comes with a set of predefined SELinux users. You rarely have to create new ones because these existing users cover most use cases.

The following table lists the most useful confined SELinux users for system administration.

user_u

This account is for standard, nonadministrative users. SELinux prevents Linux users that are mapped to the `user_u` SELinux user from becoming the `root` Linux user by using the `su` or `sudo` commands, or from executing most *set user ID (setuid)* programs.

sysadm_u

The `sysadm_u` SELinux user is for system administration. SELinux allows Linux users that are mapped to the `sysadm_u` SELinux user to use the `su` and `sudo` commands. (Whether the user can do anything useful with the `sudo` command depends on its configuration, in the usual way.) In addition, users that are mapped to the `sysadm_u` SELinux user cannot log in by using the `ssh` command unless the `ssh_sysadm_login` Boolean is set to the `on` value.

staff_u

Linux users that are mapped to the `staff_u` SELinux user can use the `sudo` command but not the `su` command. The `staff_u` SELinux user is for regular users who need to use the `sudo` command for specific tasks. For example, web developers might need to use the `sudo` command to restart the `httpd` service. By mapping a Linux user to the `staff_u` SELinux user, these developers can use the `sudo` command, but cannot get full `root` user access by using the `su` command.

SELinux User Booleans

A number of Booleans can adjust the restrictions set on confined users. The preceding list includes one Boolean: `ssh_sysadm_login`, which controls whether users that are mapped to the `sysadm_u` SELinux user can log in by using the `ssh` command.

Another set of Booleans restricts whether these users can run executables in their home directory or in the `/tmp/` directory. These are the `user_exec_content`, `sysadm_exec_content`, and `staff_exec_content` Booleans for the `user_u`, `sysadm_u`, and `staff_u` SELinux users, respectively.

Confining User Accounts

Use the following guidelines to implement user confinement on your systems.

Update the default SELinux mapping to associate your Linux users to the `user_u` SELinux user.

```
[root@host ~]# semanage login -m -s user_u -r s0 __default__
```

Map the system administrators to the `sysadm_u` SELinux user.

```
[root@host ~]# semanage login -a -s sysadm_u operator1
[root@host ~]# semanage login -a -s sysadm_u operator2
[root@host ~]# semanage login -a -s sysadm_u operator3
```

Chapter 8 | Mitigating Risk with SELinux

Optionally, map the Linux users who need sudo command access to the staff_u SELinux user, and configure the sudo command.

```
[root@host ~]# semanage login -a -s staff_u developer1  
[root@host ~]# semanage login -a -s staff_u developer2
```

Confining Different User Accounts

When you decide to confine all your Linux users, you usually start by modifying the default mapping to the user_u SELinux user.

```
[root@host ~]# semanage login -m -s user_u -r s0 __default__
```

This way, SELinux confines all your Linux users to an SELinux user with minimal privileges by default. If you use the default mapping, no further configuration is needed; SELinux automatically confines all existing users that do not have a mapping on their next login.

For extra protection, you can also prevent users with the user_u SELinux user mapping from executing programs in their home directories and the /tmp/ directory. To do that, set the user_exec_content Boolean to the off value.

```
[root@host ~]# setsebool -P user_exec_content off
```

Confining System Administrators

To confine your system administrators, map their Linux account to the sysadm_u SELinux user. For existing accounts, use the semanage login -a command:

```
[root@host ~]# semanage login -a -s sysadm_u operator1
```

For new users, you can use the -Z option of the useradd command to do the mapping at user creation time.

```
[root@host ~]# useradd -G wheel -Z sysadm_u operator2
```

The previous command uses the -G option to add the new account to the Linux wheel group to benefit from the existing sudo rule for this group.

By default, and for extra protection, users that are mapped to the sysadm_u SELinux user cannot use SSH to log in. Set the ssh_sysadm_login Boolean to the on value if you must allow SSH access.

```
[root@host ~]# setsebool -P ssh_sysadm_login on
```

Remember to remove the mapping when deleting a Linux user account.

```
[root@host ~]# userdel operator1  
[root@host ~]# semanage login -d -s sysadm_u operator1
```

You can also use the userdel -Z command to remove the mapping at the same time that you delete the user account.

```
[root@host ~]# userdel -z operator2
```

Confining Staff Users

Some standard Linux users might need to run specific commands as the `root` user. Map them to the `staff_u` SELinux user account and configure the `sudo` command.

For existing accounts, use the `semanage login -a` command:

```
[root@host ~]# semanage login -a -s staff_u developer1
```

For new users, you can use the `-Z` option of the `useradd` command to do the mapping at user creation time.

```
[root@host ~]# useradd -Z staff_u developer2
```

The next step is to configure the `sudoers` files to list the commands that these users can run as the `root` user.

For extra protection, you can also prevent the `staff_u` SELinux users from executing programs in their home directories and the `/tmp/` directory. For that level of protection, set the `staff_exec_content` Boolean to the `off` value.

```
[root@host ~]# setsebool -P staff_exec_content off
```

Remember to remove the mapping when deleting the Linux user account.

```
[root@host ~]# userdel -z developer1
```



Note

Additional confined users, such as the `xguest_u` and `guest_u` SELinux users, are more restricted than the `user_u` SELinux user. For more information, refer to the *Using SELinux* guide.



References

The `semanage-login(8)` and `semanage-user(8)` man pages

For more information, refer to the *Managing Confined and Unconfined Users* chapter in the *Using SELinux* guide at

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html-single/using_selinux/index#managing-confined-and-unconfined-users_using-selinux

► Guided Exercise

Controlling Access with Confined Users

Configure users as SELinux confined users to limit the mechanisms that they might use to access the system and the superuser account.

Outcomes

- Prevent Linux users from switching to a different SELinux confined user.
- Control the methods that administrators are permitted to use to switch user with SELinux confined users.
- Examine the effects of confined SELinux users on the sudo, su, and ssh commands, and on SUID execution.
- Explore the Booleans that control how SELinux controls confined users.

Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

```
[student@workstation ~]$ lab start selinux-controlling
```

Instructions

- 1. On the `serverc` machine, confine all Linux users except the `root` user by mapping them to the `user_u` SELinux user at login.
- 1.1. Log in to the `serverc` machine as the `student` user. No password is required.

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

- 1.2. Display the SELinux user that is associated with the `student` user.

```
[student@serverc ~]$ id -Z
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

- 1.3. Change to the `root` user. Use `student` as the password.

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

- 1.4. Confirm that SELinux is in Enforcing mode.

```
[root@serverc ~]# getenforce  
Enforcing
```

- 1.5. Retrieve the mapping between the Linux user and the SELinux user.

```
[root@serverc ~]# semanage login -l
```

Login Name	SELinux User	MLS/MCS Range	Service
<u>default</u>	unconfined_u	s0-s0:c0.c1023	*
root	unconfined_u	s0-s0:c0.c1023	*

By default, SELinux maps the Linux users to the `unconfined_u` SELinux user. Changing this mapping has no impact on the `root` user because it has its own rule.

- 1.6. Change the default mapping to map the Linux users to the `user_u` SELinux user. The `semanage` command might take up to one minute to complete.

```
[root@serverc ~]# semanage login -m -s user_u -r s0 __default__
```

- 1.7. Use the `semanage login -l` command again to verify your work.

```
[root@serverc ~]# semanage login -l
```

Login Name	SELinux User	MLS/MCS Range	Service
<u>default</u>	user_u	s0	*
root	unconfined_u	s0-s0:c0.c1023	*

- 1.8. Log out of the `serverc` machine, and then log in again as the `student` user.

```
[root@serverc ~]# logout  
[student@serverc ~]$ logout  
[student@workstation ~]$ ssh student@serverc
```

- 1.9. Use the `id -Z` command to confirm that SELinux maps the `student` user to the `user_u` SELinux user.

```
[student@serverc ~]$ id -Z  
user_u:user_r:user_t:s0
```

- 1.10. Confirm that the `student` user cannot use the `sudo` or `su` commands anymore. For the `su` command, use `redhat` as the password for the `root` user.

```
[student@serverc ~]$ sudo -i  
sudo: PERM_SUDOERS: setresuid(-1, 1, -1): Operation not permitted  
sudo: no valid sudoers sources found, quitting  
sudo: setresuid() [0, 0, 0] -> [1000, -1, -1]: Operation not permitted  
sudo: error initializing audit plugin sudoers_audit  
[student@serverc ~]$ su -  
Password: redhat  
su: Authentication failure
```

- ▶ 2. Set the `user_exec_content` SELinux Boolean to the `off` value to prevent users who are mapped to the `user_u` SELinux user from executing programs in the `/tmp` directory or in their home directories.
- 2.1. Verify that the `student` user can execute programs in their home directory and in the `/tmp` directory on the `serverc` machine.

```
[student@serverc ~]$ /tmp/runme  
Tested  
[student@serverc ~]$ ./runme  
Tested
```

The `runme` program was deployed by the `lab_start` command.

- 2.2. The new mapping to the `user_u` SELinux user prevents Linux users from using the `sudo -i` command. To use the `root` account, log out of the `serverc` machine and log in again as the `root` user.

```
[student@serverc ~]$ logout  
[student@workstation ~]$ ssh root@serverc  
[root@serverc ~]#
```

- 2.3. Set the SELinux `user_exec_content` Boolean to the `off` value. The `setsebool` command might take up to one minute to complete.

```
[root@serverc ~]# setsebool -P user_exec_content off
```

- 2.4. Log out of the `serverc` machine and log in again as the `student` user. No password is required.

```
[root@serverc ~]# logout  
[student@workstation ~]$ ssh student@serverc  
[student@serverc ~]$
```

- 2.5. Confirm that the `student` user can no longer execute programs in the `/tmp` directory or their home directory.

```
[student@serverc ~]$ /tmp/runme  
-bash: /tmp/runme: Permission denied  
[student@serverc ~]$ ./runme  
-bash: ./runme: Permission denied
```

► 3. Create the `operator1` Linux user as follows:

- The user is a system administrator and must be able to use the `sudo` and `su` commands. Map the user to the `sysadm_u` SELinux user.
- Add the user to the `wheel` group, to benefit from the existing `sudo` rule that grants permission to run any command as any user.
- Set the user's password to `redhat`.
- The `operator1` user must be able to log in by using the `ssh` command. The `sysadm_u` SELinux user does not allow this by default, but you can meet this requirement by allowing users that are mapped to the `sysadm_u` SELinux user to use the `ssh` command to log in.

3.1. Log out of the `serverc` machine and log in again as the `root` user.

```
[student@serverc ~]$ logout  
[student@workstation ~]$ ssh root@serverc  
[root@serverc ~]#
```

3.2. Create the `operator1` Linux user account, map it to the `sysadm_u` SELinux user, and add it to the `wheel` group. The `useradd` command might take up to one minute to complete.

```
[root@serverc ~]# useradd -G wheel -Z sysadm_u operator1  
[root@serverc ~]#
```

3.3. Confirm that SELinux maps the `operator1` user to the `sysadm_u` SELinux user.

```
[root@serverc ~]# semanage login -l
```

Login Name	SELinux User	MLS/MCS Range	Service
<code>__default__</code>	<code>user_u</code>	<code>s0</code>	*
<code>operator1</code>	<code>sysadm_u</code>	<code>s0-s0:c0.c1023</code>	*
<code>root</code>	<code>unconfined_u</code>	<code>s0-s0:c0.c1023</code>	*

3.4. Set the password for the `operator1` user to `redhat`.

```
[root@serverc ~]# echo redhat | passwd --stdin operator1  
Changing password for user operator1.
```

3.5. Log out of the `serverc` machine and try to use SSH to log in as the `operator1` user.

```
[root@serverc ~]# logout  
[student@workstation ~]$ ssh operator1@serverc  
client_loop: send disconnect: Broken pipe  
[student@workstation ~]$
```

By default, SELinux denies access to `sysadm_u` accounts over SSH.

3.6. Log in as the `root` user and set the `ssh_sysadm_login` Boolean to the `on` value.

```
[student@workstation ~]$ ssh root@serverc  
[root@serverc ~]# setsebool -P ssh_sysadm_login on  
[root@serverc ~]#
```

- 3.7. Log out of the `serverc` machine and log in as the `operator1` user. This time, the connection succeeds.

```
[root@serverc ~]# logout  
[student@workstation ~]$ ssh operator1@serverc  
...output omitted...  
[operator1@serverc ~]$
```

- 3.8. To confirm that the `operator1` user can administer the system, use the `sudo -i` command to switch identity to the `root` user and restart the `sshd` service. Use `redhat` as the password.

```
[operator1@serverc ~]$ sudo -i  
[sudo] password for operator1: redhat  
[root@serverc ~]# systemctl restart sshd  
[root@serverc ~]# systemctl is-active sshd  
active
```

- ▶ 4. Create the `developer1` Linux user account for your developer. The `developer1` user is not a system administrator, but they must be able to restart the `sshd` service with the `sudo` command.

Create the user account as follows:

- Map the `developer1` user to the `staff_u` SELinux user. Remember that if you do not explicitly map the user account, it is mapped to the `user_u` SELinux user, which does not allow the `sudo` command.
- Set the user's password to `redhat`.
- Create a `sudo` configuration file, `/etc/sudoers.d/developers`, and add a rule to permit the `developer1` user to run the `systemctl restart sshd` command as the `root` user.

- 4.1. Create the `developer1` Linux user account and map it to the `staff_u` SELinux user. The `useradd` command might take up to one minute to complete.

```
[root@serverc ~]# useradd -Z staff_u developer1
```

- 4.2. Confirm that SELinux maps the `developer1` user to the `staff_u` SELinux user.

```
[root@serverc ~]# semanage login -l
```

Login Name	SELinux User	MLS/MCS Range	Service
<code>_default_</code>	<code>user_u</code>	<code>s0</code>	*
developer1	staff_u	<code>s0-s0:c0.c1023</code>	*
<code>operator1</code>	<code>sysadm_u</code>	<code>s0-s0:c0.c1023</code>	*
<code>root</code>	<code>unconfined_u</code>	<code>s0-s0:c0.c1023</code>	*

- 4.3. Set the password for the developer1 user to redhat.

```
[root@serverc ~]# echo redhat | passwd --stdin developer1
Changing password for user developer1.
```

- 4.4. Create the /etc/sudoers.d/developers file and add a rule to permit the developer1 user to run the systemctl restart sshd command as the root user.

```
[root@serverc ~]# cat /etc/sudoers.d/developers
developer1  ALL=/bin/systemctl restart sshd
```

- 4.5. Log out of the serverc machine and log in as the developer1 user. No password is required.

```
[root@serverc ~]# logout
[operator1@serverc ~]$ logout
[student@workstation ~]$ ssh developer1@serverc
[developer1@serverc ~]$
```

- 4.6. Use the id -Z command to confirm that SELinux maps the developer1 user to the staff_u SELinux user.

```
[developer1@serverc ~]$ id -Z
staff_u:staff_r:staff_t:s0-s0:c0.c1023
```

Notice that the current role is the staff_r role.

- 4.7. Only the sysadm_r role allows the execution of commands that use the sudo command. Confirm that the staff_r role does not allow the user to restart the sshd service by using the sudo command. Use redhat as the password.

```
[developer1@serverc ~]$ sudo systemctl restart sshd
[sudo] password for developer1: redhat
Failed to add a watch for /run/systemd/ask-password: Permission denied
```

- 4.8. To run commands with the sudo command, the developer1 user must change their current role to the sysadm_r role. Log out of the serverc machine and log in as the root user. Use the semanage user -l command to confirm that SELinux allows staff_u users to change role to the sysadm_r role.

```
[developer1@serverc ~]$ logout  
[student@workstation ~]$ ssh root@serverc  
[root@serverc ~]# semanage user -l  
  
          Labeling  MLS/  
SELinux User  Prefix    MCS Level ...  SELinux Roles  
  
guest_u       user     s0      ...  guest_r  
root          user     s0      ...  staff_r sysadm_r system_r unconfined_r  
staff_u       user     s0      ...  staff_r sysadm_r system_r unconfined_r  
sysadm_u      user     s0      ...  sysadm_r  
system_u       user     s0      ...  system_r unconfined_r  
unconfined_u   user     s0      ...  system_r unconfined_r  
user_u         user     s0      ...  user_r  
xguest_u      user     s0      ...  xguest_r
```

- 4.9. Configure the sudo command to perform the role change before running the command. Edit the /etc/sudoers.d/developers file and insert the ROLE variable before the command.

```
[root@serverc ~]# cat /etc/sudoers.d/developers  
developer1  ALL= ROLE=sysadm_r /bin/systemctl restart sshd
```

- 4.10. Log out of the serverc machine and log in as the developer1 user.

```
[root@serverc ~]# logout  
[student@workstation ~]$ ssh developer1@serverc  
[developer1@serverc ~]$
```

- 4.11. Confirm that the developer1 user can restart the sshd service. Use redhat as the password. Return to the workstation machine when done.

```
[developer1@serverc ~]$ sudo /bin/systemctl restart sshd  
[sudo] password for developer1: redhat  
[developer1@serverc ~]$ logout  
[student@workstation ~]$
```

The systemctl restart command does not display anything when successful.

Finish

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

```
[student@workstation ~]$ lab finish selinux-controlling
```

Auditing the SELinux Policy

Objectives

- Examine a system's SELinux policy to evaluate the access that it permits, and to troubleshoot and resolve issues.

Overview of the SELinux Policy

The SELinux policy is a set of rules that guides the SELinux security engine. It defines the SELinux users, Booleans, types for files and processes, and many other objects. Red Hat Enterprise Linux ships with a standardized policy that is adjustable for many use cases.

You can use the `seinfo` command, which is part of the `setools-console` package, to list all the objects in the policy.

```
[user@host ~]$ seinfo
Statistics for policy file: /sys/fs/selinux/policy
Policy Version:          33 (MLS enabled)
Target Policy:           selinux
Handle unknown classes: allow
Classes:                 135   Permissions:        457
Sensitivities:          1      Categories:         1024
Types:                   5066   Attributes:         253
Users:                   8      Roles:              14
Booleans:                348   Cond. Expr.:       378
Allow:                   63338  Neverallow:        0
Auditallow:              163   Dontaudit:        8432
Type_trans:              249598 Type_change:       87
Type_member:             35    Range_trans:       6161
Role allow:              37    Role_trans:        418
Constraints:             70    Validatetrans:     0
MLS Constrain:          72    MLS Val. Tran:     0
Permissives:             4     Polcap:            6
Defaults:                7     Typebounds:        0
Allowxperm:              0     Neverallowxperm:   0
Auditallowxperm:         0     Dontauditxperm:    0
Ibendportcon:           0     Ibpkeycon:         0
Initial SIDs:            27    Fs_use:            35
Genfscon:                109   Portcon:          660
Netifcon:                0     Nodecon:          0

[user@host ~]$ seinfo --type
Types: 5066
NetworkManager_dispatcher_chronyc_script_t
NetworkManager_dispatcher_chronyc_t
NetworkManager_dispatcher_cloud_script_t
NetworkManager_dispatcher_cloud_t
NetworkManager_dispatcher_console_script_t
```

```
NetworkManager_dispatcher_console_t  
NetworkManager_dispatcher_console_var_run_t  
NetworkManager_dispatcher_custom_t  
. . . output omitted...
```

The policy also includes the rules that determine how each domain might access each type. Recall that a domain is what a type is called when it is applied to a process. These rules are discussed in more detail in a later section.

SELinux types have names that end in the _t suffix, and for clarity, they usually include the kind of object that they apply to. For example, types that are associated with service executable files end in the _exec_t suffix:

- The httpd_exec_t type corresponds to the /usr/sbin/httpd binary.
- The crond_exec_t type corresponds to the /usr/sbin/crond binary.
- The postfix_master_exec_t type corresponds to the /usr/sbin/postfix binary.

Types for log files end in the _log_t suffix. The following examples are log files and their associated SELinux types:

- The httpd_log_t type corresponds to log files in the /var/log/httpd/ directory.
- The cron_log_t type corresponds to the /var/log/cron log file.

Types that are used to label ports end in the _port_t suffix, such as the http_port_t or smtp_port_t types.

For categorization and organization, SELinux associates *attributes* with types. Attributes are a way to group types. The following examples show some attributes and the types that they group:

- The **logfile** attribute includes the httpd_log_t and cron_log_t types.
- The **exec_type** attribute contains the httpd_exec_t, crond_exec_t, and postfix_master_exec_t types.
- The **domain** attribute groups all the domains (types for processes), such as the httpd_t, sshd_t, or crond_t types.

Attributes help reduce and simplify the access rules. Because all processes need access to the libraries in the /usr/lib64/ directory, a single rule exists to allow the **domain** attribute to read directories with the lib_t type, which is the type that is associated with the /usr/lib64/ directory.

You can list all attributes with the **seinfo --attribute** command, and you can list the types in an attribute with the **seinfo --attribute=attribute -x** command.

```
[user@host ~]$ seinfo --attribute  
  
Type Attributes: 253  
abrt_domain  
admindomain  
afs_domain  
antivirus_domain  
application_domain_type  
application_exec_type  
base_file_type  
base_ro_file_type  
. . . output omitted...  
[user@host ~]$ seinfo --attribute=exec_type -x
```

```
Type Attributes: 1
attribute exec_type;
NetworkManager_dispatcher_chronyc_script_t
NetworkManager_dispatcher_cloud_script_t
NetworkManager_dispatcher_console_script_t
NetworkManager_dispatcher_ddclient_script_t
NetworkManager_dispatcher_dhclient_script_t
NetworkManager_dispatcher_dnssec_script_t
NetworkManager_dispatcher_exec_t
...output omitted...
```

Analyzing the Targeted Policy

In Red Hat Enterprise Linux 9, Red Hat supports three policies: *targeted*, *Multi-Level Security (MLS)*, and *minimum*.

The *targeted* policy is the default policy. This policy controls all the daemons that listen on the network, and the services that start at boot. Processes started by users, however, run in the *unconfined_t* domain. For processes in the *unconfined_t* domain, the SELinux policy still enforces restrictions, but the rules that are in effect allow almost complete access, so these processes are mostly restricted by standard permissions.



Note

Unconfined domains might still perform domain transitions (causing the child process to become confined), and some restrictions on executable memory might apply depending on Boolean settings.

The targeted policy is flexible enough to fit into enterprise infrastructures. The daemons that are part of the policy run in their own domains and SELinux controls every operation that they perform on the system. This way, daemons that are defective or exploited are limited in the damage that they can do.

The *MLS (Multi-Level Security)* policy contains the differentiation of security levels and categories. This policy is used in military applications, and allows for classifications such as unclassified, secret, or top secret.

The *minimum* policy is based on the targeted policy, but nearly everything runs under the *unconfined_t* domain. You can activate SELinux for a specific domain by loading its SELinux module.

Interpreting the Allow Rules

The targeted policy includes rules that determine how each domain might access each type. For example, the following rule allows the `httpd` service (`httpd_t` domain) to access and read its configuration files (`httpd_config_t`).

```
allow httpd_t httpd_config_t:file { getattr ioctl lock map open read };
```

- The `httpd_t` element is a source or domain type.
- The `httpd_config_t` element is the target type.
- The `file` element is the class of the target type. For example, this element can be `file`, `dir`, `lnk_file`, or `socket`.

- The `{ setattr ioctl lock map open read }` element is the list of operations that SELinux allows. Because the `write` operation is not listed, SELinux does not allow the `httpd` process to write its configuration files.

To list the rules, use the `sesearch` command with the `-A` option. The `sesearch` command is part of the `setools-console` package.

```
[root@host ~]# sesearch -A
allow NetworkManager_dispatcher_chronyc_script_t
NetworkManager_dispatcher_chronyc_script_t:filesystem associate;
allow NetworkManager_dispatcher_chronyc_t
NetworkManager_dispatcher_chronyc_script_t:file { entrypoint execute setattr
ioctl lock map open read };
allow NetworkManager_dispatcher_chronyc_t
NetworkManager_dispatcher_chronyc_t:association sendto;
allow NetworkManager_dispatcher_chronyc_t NetworkManager_dispatcher_chronyc_t:dir
{ setattr ioctl lock open read search watch };
allow NetworkManager_dispatcher_chronyc_t
NetworkManager_dispatcher_chronyc_t:fifo_file { append setattr ioctl lock open
read write };
allow NetworkManager_dispatcher_chronyc_t
NetworkManager_dispatcher_chronyc_t:fifo_file { create link rename setattr
unlink }; [ fips_mode ]:True
allow NetworkManager_dispatcher_chronyc_t NetworkManager_dispatcher_chronyc_t:file
{ append setattr ioctl lock open read write };
allow NetworkManager_dispatcher_chronyc_t
NetworkManager_dispatcher_chronyc_t:lnk_file { setattr ioctl lock read };
allow NetworkManager_dispatcher_chronyc_t NetworkManager_dispatcher_chronyc_t:peer
recv;
allow NetworkManager_dispatcher_chronyc_t
NetworkManager_dispatcher_chronyc_t:process { fork getcap getsched sigchld
sigkill signal signull sigstop };
...output omitted...
```

The `sesearch` command accepts options to filter its output. The following command displays only the rule that allows the `httpd_t` source type (`-s`) to access files (`-c file`) with the `httpd_config_t` target type (`-t`).

```
[root@host ~]# sesearch -A -s httpd_t -t httpd_config_t -c file
allow domain file_type:file map; [ domain_can_mmap_files ]:True
allow httpd_t httpd_config_t:file { setattr ioctl lock map open read };
```

The following table lists some of the most useful options for the `sesearch -A` command.

Option	Description
<code>-s name</code>	Source domain type or attribute
<code>-t name</code>	Target type or attribute
<code>-c name</code>	Class of the target object. Use <code>seinfo -c</code> to get the full list. Use the <code>-c</code> option with the <code>-x</code> option to print a list of permissions for each displayed object class.

Chapter 8 | Mitigating Risk with SELinux

The `-s` and `-t` options also accept SELinux attributes. The following command lists the rules that allow domains to access directories with the `lib_t` type, which is the type of the `/usr/lib64` directory.

```
[root@host ~]# sesearch -A -s domain -t lib_t -c dir
allow NetworkManager_t lib_t:dir watch;
allow abrt_dump_oops_t non_security_file_type:dir { add_name create setattr ioctl
    link lock open read remove_name rename reparent rmdir search setattr unlink watch
    watch_reads write };
allow abrt_t file_type:dir { setattr open search };
allow aide_t file_type:dir { setattr ioctl lock open read search };
allow amanda_t file_type:dir { setattr ioctl lock open read search };
allow antivirus_domain file_type:dir { setattr ioctl lock open read search };
[ antivirus_can_scan_system ]:True
allow antivirus_domain file_type:dir { setattr ioctl lock open read search };
[ antivirus_can_scan_system ]:True
allow antivirus_domain file_type:dir { setattr open search };
[ antivirus_can_scan_system ]:True
allow antivirus_domain file_type:dir { setattr open search };
[ antivirus_can_scan_system ]:True
allow auditctl_t file_type:dir { setattr open search };
...output omitted...
```

Remember that the `domain` attribute groups all the types that are associated with domains, such as the `httpd_t` type. Distinguishing attributes and types might be challenging at first, but remember that types always end with the `_t` suffix, which is never the case for attributes.

Selectively Disabling Audits

In addition to `allow` rules, the SELinux policy also contains `dontaudit` rules. When SELinux does not allow an action, but a `dontaudit` rule exists for that action, then SELinux does not log the denial.

A `dontaudit` rule is useful because some applications probe for files or directories: for example, to identify some features or devices that might be provided by the operating system. If those files or directory are missing, or the access is not allowed, then the application continues without problems. But these access attempts still might trigger SELinux violations, which would normally be added to the log.

By setting `dontaudit` rules, the log file does not store logs for access violations that are expected, and which do not have significant security implications.

You can list all the `dontaudit` rules on a system by using the `--dontaudit` option with the `sesearch` command.

```
[user@host ~]$ sesearch --dontaudit
dontaudit NetworkManager_dispatcher_chronyc_t
    NetworkManager_dispatcher_chronyc_t:capability { net_admin sys_module };
dontaudit NetworkManager_dispatcher_chronyc_t
    NetworkManager_dispatcher_chronyc_t:create;
dontaudit NetworkManager_dispatcher_chronyc_t
    NetworkManager_dispatcher_chronyc_t:udp_socket listen;
dontaudit NetworkManager_dispatcher_cloud_t
    NetworkManager_dispatcher_cloud_t:capability { net_admin sys_module sys_ptrace };
```

```
dontaudit NetworkManager_dispatcher_cloud_t NetworkManager_dispatcher_cloud_t::file  
create;  
dontaudit NetworkManager_dispatcher_cloud_t  
NetworkManager_dispatcher_cloud_t::process setrlimit;  
dontaudit NetworkManager_dispatcher_cloud_t  
NetworkManager_dispatcher_cloud_t::udp_socket listen;  
dontaudit NetworkManager_dispatcher_console_t  
NetworkManager_dispatcher_console_t::capability { net_admin sys_module };  
dontaudit NetworkManager_dispatcher_console_t  
NetworkManager_dispatcher_console_t::file create;  
dontaudit NetworkManager_dispatcher_console_t  
NetworkManager_dispatcher_console_t::udp_socket listen;  
...output omitted...
```

Sometimes, for troubleshooting purposes, you might want to disable the `dontaudit` rules to record all SELinux denials in the log. Use the `semodule -DB` command to disable the `dontaudit` rules.

```
[root@host ~]# semodule -DB
```

To re-enable the `dontaudit` rules, use the `semodule -B` command.

```
[root@host ~]# semodule -B
```

Creating Custom Policy Modules

Developing a custom policy for your application is beyond the scope of this course. However, the `audit2allow` command can generate a policy module for you by analyzing the denials in the `audit.log` file.

For example, the `audit2allow` command can generate a custom rule based on the following message in the `/var/log/audit/audit.log` file:

```
[root@host ~]# grep denied /var/log/audit/audit.log  
...output omitted...  
type=AVC msg=audit(1697823181.181:205): avc: denied { getattr }  
for pid=27012 comm="httpd" path="/var/www/html/index.html"  
dev="vda4" ino=25541492 scontext=system_u:system_r:httpd_t:s0  
tcontext=unconfined_u:object_r:admin_home_t:s0 tclass=file permissive=0  
...output omitted...
```

Run `audit2allow -a` to print the rule to allow the access.

```
[root@host ~]# audit2allow -a  
  
===== httpd_t =====  
allow httpd_t admin_home_t::file getattr;  
allow httpd_t self:capability net_admin;
```



Important

Use `audit2allow` with caution: it generates `allow` rules based on the denials in the log file. Ultimately, you must decide whether the rule is safe to add in the policy. Before changing the policy, consider whether the violation is being reported for a good reason, and why that access might be blocked by default.

In the previous example, a better solution might be to relabel the `index.html` file, rather than adding a new rule that gives the `httpd` daemon access to all the files that are labeled with the `admin_home_t` type.

To generate a new SELinux policy module, add the `-M modulename` option to the previous `audit2allow` command. Use the `semodule` command to persistently load the new module in SELinux.

```
[root@host ~]# audit2allow -a -M mymodule
***** IMPORTANT *****
To make this policy package active, execute:

semodule -i mymodule.pp

[root@host ~]# semodule -i mymodule.pp
```

In the example, the first operation that the `httpd` daemon performs on the `index.html` file is to retrieve the file's attributes (`getattr`). Because SELinux is in enforcing mode, the operation is denied and the `httpd` daemon does not access the file further. The new rule that allows the `getattr` operation is not sufficient because the `httpd` daemon also needs to open and read the file, and those operations are still not allowed.

Therefore, before using the `audit2allow` command, put SELinux in permissive mode to collect all the denials in one operation.

```
[root@host ~]# setenforce 0
[root@host ~]# curl http://localhost/index.html
Hello, World!
[root@host ~]# setenforce 1
[root@host ~]# audit2allow -a

===== httpd_t =====
allow httpd_t admin_home_t:file { getattr open read };

#!!!! This avc can be allowed using the boolean 'domain_can_mmap_files'
allow httpd_t admin_home_t:file map;
allow httpd_t self:capability net_admin;
```

This time, the rule also includes the `open` and `read` operations.

Domain Transitions

New processes inherit the context type of their parent. The following example shows that the `vim` process with the 27393 PID has the same context as its parent.

```
[root@host ~]# pstree -Z 27393
bash(`unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023')
└─vim(`unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023')
```

Occasionally, processes must change their security contexts. This transition is fairly common with daemons that are started by the `systemd` daemon. When the `systemd` daemon starts a service, SELinux must confine the service in its own domain. For example, when the `systemd` daemon starts the `httpd` service, the `httpd` service transitions to the `httpd_t` domain.

```
[root@host ~]# pstree -Z | grep -e ^systemd -e httpd
systemd(`system_u:system_r:init_t:s0')
├─httpd(`system_u:system_r:httpd_t:s0')
│ ├─httpd(`system_u:system_r:httpd_t:s0')
│ ├─httpd(`system_u:system_r:httpd_t:s0')
│ │ └─64*[{httpd}(`system_u:system_r:httpd_t:s0')]
│ ├─httpd(`system_u:system_r:httpd_t:s0')
│ │ └─64*[{httpd}(`system_u:system_r:httpd_t:s0')]
│ └─httpd(`system_u:system_r:httpd_t:s0')
│   └─80*[{httpd}(`system_u:system_r:httpd_t:s0')]
```

The previous output shows that even though the `systemd` process is running in the `init_t` domain, SELinux confines the child `httpd` process in its own `httpd_t` domain.

These transition rules are part of the SELinux policy.

The following transition rule specifies that if a process with the `init_t` context type executes a binary file that has the `httpd_exec_t` context type, then the resulting child process has the `httpd_t` context type.

```
type_transition init_t httpd_exec_t : process httpd_t ;
```

- The `init_t` element is the source or domain type of the parent process.
- The `httpd_exec_t` element is the context type of the program file.
- The `httpd_t` element is the domain of the resulting child process.

To list the transition rules, use the `sestatus` command with the `-T` option.

```
[root@host ~]# sestatus -T -s init_t -t httpd_exec_t
type_transition init_t httpd_exec_t:process httpd_t;
```



Note

Policies commonly set domain transition rules so that when an unconfined domain like the `init_t` domain runs an executable that has a type like the `something_exec_t` type, the resulting process ends up with a confined domain such as the `example_t` type.

The domain transition rules for the `something_t` domain are generally much more strict, which helps enforce confinement. If there is not a rule to transition the `example_t` domain back to an unconfined domain, then the executable remains confined.

Chapter 8 | Mitigating Risk with SELinux

As a complement to the `sestatus -T` command, the `sepolicy transition` command, from the `policycoreutils-devel` package, can analyze the policy and list all the intermediary types for the transition from one domain to another.

The following example lists all the paths of sequential transitions that can get from the `httpd_t` domain to the `unconfined_t` domain. In other words, the example shows whether and how a subprocess of the `httpd` service might be unconfined through domain transitions. The output does not indicate whether or not executables exist on the system to accomplish this set of transitions.

```
[root@host ~]# sepolicy transition -s httpd_t -t unconfined_t
httpd_t ... abrt_retrace_worker_t ... mock_t ... mount_t ... glusterd_t ...
initrc_t ... pegasus_t ... rpm_t ... rpm_script_t ... openshift_initrc_t ...
virt_qemu_ga_t ... system_cronjob_t ... logrotate_t ... cluster_t ...
condor_schedd_t ... condor_startd_t ... inetd_t ... sshd_t @ shell_exec_t -->
unconfined_t -- Allowed True [ ssh_sysadm_login=0 || unconfined_login=1 ]
...output omitted...
```

Run the `sepolicy transition` command on each transition to get more details.

```
[root@host ~]# sepolicy transition -s httpd_t -t abrt_retrace_worker_t
httpd_t @ abrt_retrace_worker_exec_t --> abrt_retrace_worker_t

[root@host ~]# sepolicy transition -s abrt_retrace_worker_t -t mock_t
abrt_retrace_worker_t @ mock_exec_t --> mock_t

[root@host ~]# sepolicy transition -s mock_t -t mount_t
mock_t @ fusermount_exec_t --> mount_t
mock_t @ mount_exec_t --> mount_t
```

Some domains have no way to transition to other domains under the domain transition policy, such as the `postfix_master_t` domain to the `unconfined_t` domain in the following example:

```
[root@host ~]# sepolicy transition -s postfix_master_t -t unconfined_t
[root@host ~]#
```

File Transitions

New files and directories inherit the context of their parent directory. Sometimes this inherited context is not correct and you must relabel the object.

You can get the expected context of an object with the `matchpathcon` command. The file or directory that you specify does not need to exist because the command uses the file context rules database, the same database that the `restorecon` command uses, to retrieve the context.

```
[root@host ~]# matchpathcon /var/www/html/myimage.png
/var/www/html/myimage.png system_u:object_r:httpd_sys_content_t:s0
```

You can list the default file context rules with the `semanage fcontext -l` command and add new rules with the `semanage fcontext -a` command.

```
[root@host ~]# semanage fcontext -l
...output omitted...
/var/www(/.*)?    all files    system_u:object_r:httpd_sys_content_t:s0
...output omitted...
[root@host ~]# semanage fcontext -a -t httpd_sys_content_t '/virtual(/.*)?'
```

Sometimes, new files and directories must have a specific context at creation time. For example, the `/var/log/cron` file has a specific context that allows the `crond` daemon to write to it. The context of this file is not the same as its parent directory.

```
[root@host ~]# ls -zd /var/log /var/log/cron
drwxr-xr-x. root root system_u:object_r:`var_log_t`:s0  /var/log
-rw-r--r--. root root system_u:object_r:`cron_log_t`:s0  /var/log/cron
```

The SELinux policy contains file transition rules that allow SELinux to automatically set the context of specific files and directories at creation time. For example, the following transition rule specifies that if a process in the `crond_t` domain (`crond`) creates a file in a directory with the `var_log_t` type, then the resulting file has the `cron_log_t` type.

```
type_transition crond_t var_log_t : file cron_log_t ;
```

- The `crond_t` element is the source or domain type of the process.
- The `var_log_t` element is the context type of the parent directory.
- The `file` element is the type of the object that is being created.
- The `cron_log_t` element is the resulting context type of the object.

To list the file transition rules, use the `sesearch -T` command.

```
[root@host ~]# sesearch -T -s crond_t -t var_log_t -c file
type_transition crond_t var_log_t:file NetworkManager_var_lib_t
wpa_supplicant.log;
type_transition crond_t var_log_t:file cron_log_t;
type_transition crond_t var_log_t:file devicekit_var_log_t pm-powersave.log;
type_transition crond_t var_log_t:file devicekit_var_log_t pm-suspend.log;
type_transition crond_t var_log_t:file faillog_t btmp;
type_transition crond_t var_log_t:file faillog_t faillog;
type_transition crond_t var_log_t:file faillog_t tallylog;
type_transition crond_t var_log_t:file lastlog_t lastlog;
type_transition crond_t var_log_t:file plymouthd_var_log_t boot.log;
type_transition crond_t var_log_t:file rpm_log_t dnf.librepo.log;
...output omitted...
```

These file transition rules can include an extra parameter to specify the object name. The following rule sets the `rpm_log_t` type on the `dnf.log` file when an `unconfined_t` domain creates it in a directory with the `var_log_t` type.

```
[root@host ~]# sesearch -T -s unconfined_t -t var_log_t -c file | \
grep dnf.log
type_transition unconfined_t var_log_t : file rpm_log_t "dnf.log";
```



References

The `seinfo(1)`, `sesearch(1)`, `semodule(8)`, `matchpathcon(8)`, `sepolicy-transition(8)`, and `audit2allow(1)` man pages

For more information on troubleshooting SELinux denials, refer to the *Troubleshooting Problems Related to SELinux* chapter in the *Using SELinux* guide at https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html-single/using_selinux/index#troubleshooting-problems-related-to-selinux_using-selinux

► Guided Exercise

Auditing the SELinux Policy

Use system tools to examine the system's current SELinux policy, and to interpret whether specific SELinux domains that are used by processes have access to files and ports that are labeled with specific SELinux types.

Outcomes

- Use policy tools to predict which SELinux domain the process will have when it is run, based on the SELinux type on an executable.
- Determine which SELinux types might be accessed by that process, and what access is permitted or blocked.
- Determine whether a particular SELinux domain can transition to the `unconfined_t` type, and whether the `unconfined_t` type can transition to that domain.

Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

```
[student@workstation ~]$ lab start selinux-auditing
```

Instructions

- 1. Use the SELinux policy tools to predict the SELinux domain type for the `httpd` daemon when the `systemd` daemon starts the service.

1.1. Log in to the `serverc` machine as the `student` user. No password is required.

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

1.2. Change to the `root` user. Use `student` as the password.

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

1.3. Install the `policycoreutils-devel` and `setools-console` packages to provide the `sepolicy` and `seseach` commands, respectively.

```
[root@serverc ~]# dnf install policycoreutils-devel setools-console
...output omitted...
Is this ok [y/N]: y
...output omitted...
Complete!
```

- 1.4. When using the `systemctl` command to start a service, the command forwards the request to the `systemd` daemon. Retrieve the SELinux domain type of the `systemd` daemon.

```
[root@serverc ~]# ps -Z -C systemd
LABEL PID TTY TIME CMD
system_u:system_r:init_t:s0 1 ? 00:00:01 systemd
...output omitted...
```

- 1.5. The `systemd` daemon starts the service by executing the `httpd` binary file. Retrieve the SELinux context type of the `httpd` executable.

```
[root@serverc ~]# which httpd
/sbin/httpd
[root@serverc ~]# ls -Z /sbin/httpd
system_u:object_r:httpd_exec_t:s0 /sbin/httpd
```

- 1.6. Use the `sesearch` command to retrieve the SELinux domain transition rule for when a daemon of the `init_t` type executes a program of the `httpd_exec_t` type.

```
[root@serverc ~]# sesearch -T -s init_t -t httpd_exec_t
type_transition init_t httpd_exec_t:process httpd_t;
```

The SELinux domain type of the resulting process is the `httpd_t` type.

- 1.7. Use the `sepolicy transition` command as another way to list domain transitions.

```
[root@serverc ~]# sepolicy transition -s init_t -t httpd_t
init_t @ httpd_exec_t --> httpd_t
init_t ... initrc_t @ httpd_exec_t --> httpd_t
init_t ... initrc_t ... pegasus_t ... rpm_t ... rpm_script_t ... bootloader_t ...
kmod_t ... mount_t ... glusterd_t @ httpd_exec_t --> httpd_t
init_t ... initrc_t ... pegasus_t ... rpm_t ... rpm_script_t ... bootloader_t ...
kmod_t ... mount_t ... glusterd_t ... svc_start_t ... svc_run_t @ httpd_exec_t
--> httpd_t
...output omitted...
```

The `sepolicy transition` command displays all the transition paths between a source and a target domain. The first line indicates that a direct transition occurred through the execution of a binary with the `httpd_exec_t` type.

- 1.8. Confirm your observation by starting the `httpd` service and retrieving the domain type of the resulting `httpd` processes.

```
[root@serverc ~]# systemctl start httpd
```

```
[root@serverc ~]# ps -Z -C httpd
LABEL PID TTY TIME CMD
system_u:system_r:httpd_t:s0 27970 ? 00:00:00 httpd
system_u:system_r:httpd_t:s0 27971 ? 00:00:00 httpd
system_u:system_r:httpd_t:s0 27972 ? 00:00:00 httpd
system_u:system_r:httpd_t:s0 27973 ? 00:00:00 httpd
system_u:system_r:httpd_t:s0 27974 ? 00:00:00 httpd
```

- 2. Manually start the `httpd` daemon, without using the `systemctl` command. Observe the resulting SELinux domain type.

- 2.1. Stop the `httpd` service.

```
[root@serverc ~]# systemctl stop httpd
```

- 2.2. Directly start the `httpd` daemon, without using the `systemctl` command.

```
[root@serverc ~]# httpd
```

Red Hat recommends starting services by using the `systemctl` command.

- 2.3. Retrieve the SELinux domain type of the `httpd` daemon.

```
[root@serverc ~]# ps -Z -C httpd
LABEL PID TTY TIME CMD
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 28192 ? 00:00:00 httpd
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 28193 ? 00:00:00 httpd
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 28194 ? 00:00:00 httpd
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 28195 ? 00:00:00 httpd
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 28196 ? 00:00:00 httpd
```

The `httpd` domain type is the `unconfined_t` type.

- 2.4. The source domain that started the `httpd` daemon is your Bash shell. Retrieve the SELinux type of your shell.

```
[root@serverc ~]# ps -Z $$
LABEL PID TTY STAT TIME COMMAND
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 26503 pts/0 Ss 0:00 -bash
```

The `$$` token is the PID of the current process.

- 2.5. Look for a domain transition rule from the `unconfined_t` source type when executing a program with the `httpd_exec_t` type.

```
[root@serverc ~]# sesearch -T -s unconfined_t -t httpd_exec_t
```

There is no such rule. Therefore, the daemon that results from the execution of the `/usr/sbin/httpd` binary inherits the domain type of the program that launches the command: your shell, in this example.

- 2.6. Terminate the `httpd` processes and restart the service by using the `systemctl` command.

```
[root@serverc ~]# pkill httpd
[root@serverc ~]# systemctl start httpd
```

- ▶ 3. Create a test HTML page, and locate the rule that allows the `httpd` daemon to read that file.
- 3.1. Create the `index.html` test page in the `/var/www/html/` directory. Use the `curl` command to confirm that you can access the new page.

```
[root@serverc ~]# echo "Hello World" > /var/www/html/index.html
[root@serverc ~]# curl http://localhost
Hello World
```

- 3.2. Retrieve the SELinux domain type of the `httpd` daemon, and the type of the `index.html` file.

```
[root@serverc ~]# ps -Z -C httpd
LABEL PID TTY TIME CMD
system_u:system_r:httpd_t:s0 28415 ?
system_u:system_r:httpd_t:s0 28416 ?
system_u:system_r:httpd_t:s0 28417 ?
system_u:system_r:httpd_t:s0 28418 ?
system_u:system_r:httpd_t:s0 28419 ?
[root@serverc ~]# ls -Z /var/www/html/index.html
unconfined_u:object_r:httpd_sys_content_t:s0 /var/www/html/index.html
```

- 3.3. Retrieve the rule that allows the `httpd_t` domain type to read files with the `httpd_sys_content_t` type.

```
[root@serverc ~]# sesearch -A -s httpd_t -t httpd_sys_content_t -c file
allow domain file_type:file map; [ domain_can_mmap_files ]:True
allow httpd_t httpd_content_type:file { setattr ioctl lock map open read };
allow httpd_t httpdcontent:file { append create setattr ioctl link lock open read
    rename setattr unlink watch watch_reads write }; [ ( httpd_builtin_scripting &&
    httpd_unified && httpd_enable_cgi ) ]:True
allow httpd_t httpdcontent:file { execute execute_no_trans setattr ioctl map open
    read }; [ ( httpd_builtin_scripting && httpd_unified && httpd_enable_cgi ) ]:True
```

Four rules are returned that include `httpd_t` or an SELinux attribute that includes `httpd_t` as the source domain, and `httpd_sys_content_t` or an SELinux attribute that includes `httpd_sys_content_t` as the target type, for SELinux objects of class `file` (regular files).

The second rule that matched is the critical one. It allows processes with the domain `httpd_t` to use the `open` and `read` system calls, among others, on regular files of an SELinux type that is in the `httpd_content_type` SELinux attribute. That includes the `httpd_sys_content_t` SELinux type.

The first rule applies because the `domain_can_mmap_files` SELinux Boolean is currently turned on (True) in your system. The last two rules also apply because the `httpd_unified` SELinux Boolean is currently turned on on your system. (Both SELinux Booleans are turned on by default.)

**Note**

To prove that the `httpd_sys_content_t` SELinux type is part of the `httpd_content_type` SELinux attribute in the policy, other than the fact that it matches your search, you can run one of the following commands:

To see all the SELinux attributes that include the `httpd_sys_content_t` type:

```
[root@serverc ~]# seinfo -t httpd_sys_content_t -x
```

To see all the SELinux types in the `httpd_content_type` SELinux attribute:

```
[root@serverc ~]# seinfo --attribute=httpd_content_type -x
```

- 4. Locate the rule that allows the `httpd` daemon to execute CGI scripts in the `/var/www/cgi-bin/` directory.

- 4.1. Retrieve the SELinux context type of the `/var/www/cgi-bin/` directory.

```
[root@serverc ~]# ls -Zd /var/www/cgi-bin/
system_u:object_r:httpd_sys_script_exec_t:s0 /var/www/cgi-bin/
```

CGI scripts in this directory also inherit that context.

- 4.2. Use the `sesearch` command to retrieve the rule that allows the `httpd_t` domain type to execute files with the `httpd_sys_script_exec_t` type.

```
[root@serverc ~]# sesearch -A -s httpd_t \
-t httpd_sys_script_exec_t -c file
allow domain file_type:file map; [ domain_can_mmap_files ]:True
allow httpd_t httpd_content_type:file { getattr ioctl lock map open read };
allow httpd_t httpd_script_exec_type:file { getattr ioctl lock open read };
allow httpd_t httpd_sys_script_exec_t:file { execute execute_no_trans };
[ httpd_enable_cgi ]:True
```

The previous rule depends on the `httpd_enable_cgi` Boolean. The `httpd_enable_cgi` Boolean is currently on (True), therefore the rule applies.

- 5. Locate the rule that allows the `httpd` daemon to bind to TCP port 80.

- 5.1. Retrieve the SELinux type that is associated with TCP port 80.

```
[root@serverc ~]# semanage port -l | grep 80
...output omitted...
http_port_t          tcp      80, 81, 443, 488, 8008, 8009, 8443, 9000
...output omitted...
```

- 5.2. Use the `sesearch` command to get the rule that allows the `httpd_t` domain type to bind to the ports with the `http_port_t` type.

```
[root@serverc ~]# sesearch -A -s httpd_t -t http_port_t  
allow httpd_t http_port_t:tcp_socket name_bind;  
...output omitted...
```

- 6. Return to the workstation machine as the student user.

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

Finish

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

```
[student@workstation ~]$ lab finish selinux-auditing
```

▶ Lab

Mitigating Risk with SELinux

Configure confined users and address some SELinux denials.

Outcomes

- Ensure that SELinux is in enforcing mode.
- Configure confined users and administrators.
- Audit the SELinux policy to explain the context of a process.
- Resolve SELinux AVC denials.

Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

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

Instructions

1. On the `serverd` machine, change the default SELinux mode to `enforcing`. The change must persist across reboots.
2. On the `serverd` machine, try to start the `httpd` service. Diagnose the problem and fix the issue.
When done, try to access the `index.html` page. Fix the issue.
3. Confine the users on the `serverd` machine to prevent them from using the `sudo` and `su` commands. Further, they must not be able to run programs in the `/tmp/` directory or in their home directory. These restrictions do not apply to the `root` user.
You can test your restrictions by logging in as the `student` user and trying to use the `sudo -i` command. You can also try to execute the `runme` program in the `/tmp/` and `/home/student/` directories to verify your work.
4. Create the following system administrator account:
 - Username: `operator2`
 - Password: `redhat`
 - Supplementary group: `wheel`Map the `operator2` user to a confined SELinux user that has access to the `su` and `sudo` commands. Additionally, the `operator2` user must be able to log in by using SSH.
5. Locate the `type_transition` SELinux rule that explains the following behavior:
 - The `ps -Z $$` command shows that the `student` user's `bash` process is running under the `user_t` domain type.

Chapter 8 | Mitigating Risk with SELinux

- The `ls -Z /usr/bin/passwd` command shows that the `passwd` binary file has the `passwd_exec_t` type.
- When the `student` user runs the `passwd` command to change their password, the resulting process is running under the `passwd_t` domain type.

After you have located the rule, store it in the `/home/student/rule` file on the `serverd` machine. The grading script at the end of this exercise uses that file to verify your work.

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 selinux-review
```

Finish

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

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

► Solution

Mitigating Risk with SELinux

Configure confined users and address some SELinux denials.

Outcomes

- Ensure that SELinux is in enforcing mode.
- Configure confined users and administrators.
- Audit the SELinux policy to explain the context of a process.
- Resolve SELinux AVC denials.

Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

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

Instructions

1. On the `serverd` machine, change the default SELinux mode to `enforcing`. The change must persist across reboots.

- 1.1. Log in to the `serverd` machine as the `student` user. No password is required.

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

- 1.2. Use the `sudo -i` command to switch identity to the `root` user. Use `student` as the password.

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

- 1.3. Get the current SELinux mode.

```
[root@serverd ~]# getenforce
Permissive
```

- 1.4. Edit the `/etc/selinux/config` file and set the `SELINUX` variable to the `enforcing` value.

```
...output omitted...
SELINUX=enforcing
...output omitted...
```

- 1.5. Reboot the `serverd` machine to load the new configuration. Wait for the reboot to complete and verify your work with the `getenforce` command.

```
[root@serverd ~]# reboot
Connection to serverd closed.
[student@workstation ~]$ ssh student@serverd
[student@serverd ~]$ sudo -i
[sudo] password for student: student
[root@serverd ~]# getenforce
Enforcing
```

2. On the `serverd` machine, try to start the `httpd` service. Diagnose the problem and fix the issue.

When done, try to access the `index.html` page. Fix the issue.

- 2.1. Try to start the `httpd` service.

```
[root@serverd ~]# systemctl start httpd
Job for httpd.service failed because the control process exited with error code.
See "systemctl status httpd.service" and "journalctl -xeu httpd.service" for
details.
```

- 2.2. As instructed by the error message, run the `systemctl status httpd.service` command to get more information.

```
[root@serverd ~]# systemctl status httpd.service
× httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; preset:
disabled)
     Active: failed (Result: exit-code) since Thu 2023-10-19 17:46:24 EDT; 34s ago
       Docs: man:httpd.service(8)
    Process: 1122 ExecStart=/usr/sbin/httpd $OPTIONS -DFOREGROUND (code=exited,
status=1/FAILURE)
      Main PID: 1122 (code=exited, status=1/FAILURE)
        Status: "Reading configuration..."
         CPU: 40ms

Oct 19 17:46:24 serverd systemd[1]: Starting The Apache HTTP Server...
Oct 19 17:46:24 serverd httpd[1122]: (13)Permission denied: AH00091: httpd: could
not open error log file /etc/httpd/logs/error_log.
Oct 19 17:46:24 serverd httpd[1122]: AH00015: Unable to open logs
Oct 19 17:46:24 serverd systemd[1]: httpd.service: Main process exited,
code=exited, status=1/FAILURE
Oct 19 17:46:24 serverd systemd[1]: httpd.service: Failed with result 'exit-code'.
Oct 19 17:46:24 serverd systemd[1]: Failed to start The Apache HTTP Server.
...output omitted...
```

- 2.3. Collect more details by inspecting the `/etc/httpd/logs` directory.

```
[root@serverd ~]# ls -ld /etc/httpd/logs  
lrwxrwxrwx. 1 root root 19 Oct 19 17:33 /etc/httpd/logs -> /custom/httpd_logs/
```

The `/etc/httpd/logs` symlink is a link to a custom directory; the default `httpd` log directory is usually the `/var/log/httpd` directory.

- 2.4. Temporarily enable SELinux denial auditing and reattempt to start the `httpd` service.

```
[root@serverd ~]# semodule -DB  
[root@serverd ~]# systemctl start httpd  
Job for httpd.service failed because the control process exited with error code.  
See "systemctl status httpd.service" and "journalctl -xeu httpd.service" for  
details.
```

- 2.5. Search for SELinux denials by using the `ausearch` command.

```
[root@serverd ~]# ausearch -m AVC,USER_AVC,SELINUX_ERR,USER_SELINUX_ERR -ts recent  
----  
time->Thu Oct 19 17:57:26 2023  
type=PROCTITLE msg=audit(1697752646.877:89):  
    proctitle=2F7573722F7362696E2F6874747064002D44464F524547524F554E44  
type=SYSCALL msg=audit(1697752646.877:89): arch=c000003e syscall=54 success=no  
    exit=-1 a0=3 a1=1 a2=20 a3=7ffe8bad4734 items=0 ppid=1 pid=1171 auid=4294967295  
    uid=0 gid=0 euid=0 suid=0 egid=0 sgid=0 fsgid=0 tty=(none) ses=4294967295  
    comm="httpd" exe="/usr/sbin/httpd" subj=system_u:system_r:httpd_t:s0 key=(null)  
type=AVC msg=audit(1697752646.877:89): avc: denied { net_admin } for  
    pid=1171 comm="httpd" capability=12 scontext=system_u:system_r:httpd_t:s0  
    tcontext=system_u:system_r:httpd_t:s0 tclass=capability permissive=0  
----  
time->Thu Oct 19 17:57:26 2023  
type=PROCTITLE msg=audit(1697752646.891:90):  
    proctitle=2F7573722F7362696E2F6874747064002D44464F524547524F554E44  
type=SYSCALL msg=audit(1697752646.891:90): arch=c000003e syscall=257  
    success=no exit=-13 a0=fffffff9c a1=564d51b1c1f0 a2=80441 a3=1b6 items=0  
    ppid=1 pid=1171 auid=4294967295 uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0  
    sgid=0 fsgid=0 tty=(none) ses=4294967295 comm="httpd" exe="/usr/sbin/httpd"  
    subj=system_u:system_r:httpd_t:s0 key=(null)  
type=AVC msg=audit(1697752646.891:90): avc: denied { write } for  
    pid=1171 comm="httpd" name="httpd_logs" dev="vda4" ino=8582288  
    scontext=system_u:system_r:httpd_t:s0 tcontext=unconfined_u:object_r:default_t:s0  
    tclass=dir permissive=0  
----  
...output omitted...
```

The message indicates that SELinux denies the `httpd` process write access to the `httpd_logs` directory, whose inode number is 8582288. This inode number is probably different on your system.

The `httpd_logs` directory has the wrong SELinux context.

- 2.6. You probably have only one directory named `httpd_logs` on your system, but because the SELinux message does not give you the full path, you cannot be sure. Use the `find` command with the `-inum` option to locate the directory by its inode number.

Chapter 8 | Mitigating Risk with SELinux

```
[root@serverd ~]# find / -inum 8582288  
/custom/httpd_logs
```

Remember to replace the inode number in the command with the one that you retrieved in the previous step.

- 2.7. To find out what context type to set on the `/custom/httpd_logs/` directory, retrieve the context type of the `/var/log/httpd/` default log directory.

```
[root@serverd ~]# ls -Zd /var/log/httpd  
system_u:object_r:httpd_log_t:s0 /var/log/httpd
```

- 2.8. Use the `semanage fcontext` command to add the new rule for the `/custom/httpd_logs` directory.

```
[root@serverd ~]# semanage fcontext -a -t httpd_log_t \  
'/custom/httpd_logs(/.*)?'  
[root@serverd ~]#
```

- 2.9. Remember to restore the context of the `/custom/httpd_logs` directory.

```
[root@serverd ~]# restorecon -Rv /custom/httpd_logs  
Relabeled /custom/httpd_logs from unconfined_u:object_r:default_t:s0 to  
unconfined_u:object_r:httpd_log_t:s0
```

- 2.10. Verify that you can now start the `httpd` service.

```
[root@serverd ~]# systemctl start httpd  
[root@serverd ~]# systemctl is-active httpd  
active
```

- 2.11. Use the `curl` command to access the `index.html` page.

```
[root@serverd ~]# curl http://localhost/index.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>
```

The command output indicates that the `httpd` service cannot access the `index.html` file.

- 2.12. Verify that the `index.html` file exists and has the correct Linux access rights.

```
[root@serverd ~]# ls -l /var/www/html/index.html  
-rw-r--r--. 1 root root 15 Jul 25 04:28 /var/www/html/index.html
```

The file exists and has the correct access rights.

2.13. Search for SELinux denials by using the ausearch command.

```
[root@serverd ~]# ausearch -m AVC,USER_AVC,SELINUX_ERR,USER_SELINUX_ERR -ts recent
...output omitted...
-----
time->Thu Oct 19 18:25:46 2023
type=PROCTITLE msg=audit(1697754346.334:116):
    proctitle=2F7573722F7362696E2F6874747064002D44464F524547524F554E44
type=SYSCALL msg=audit(1697754346.334:116): arch=c000003e syscall=262 success=no
    exit=-13 a0=fffffff9c a1=7f3f4c045d78 a2=7f3f517e18b0 a3=0 items=0 ppid=1248
    pid=1251 auid=4294967295 uid=48 gid=48 euid=48 suid=48 egid=48
    sgid=48 fsgid=48 tty=(none) ses=4294967295 comm="httpd" exe="/usr/sbin/httpd"
    subj=system_u:system_r:httpd_t:s0 key=(null)
type=AVC msg=audit(1697754346.334:116): avc: denied { setattr }
    for pid=1251 comm="httpd" path="/var/www/html/index.html"
    dev="vda4" ino=16857024 scontext=system_u:system_r:httpd_t:s0
    tcontext=unconfined_u:object_r:admin_home_t:s0 tclass=file permissive=0
-----
time->Thu Oct 19 18:25:46 2023
type=PROCTITLE msg=audit(1697754346.334:117):
    proctitle=2F7573722F7362696E2F6874747064002D44464F524547524F554E44
type=SYSCALL msg=audit(1697754346.334:117): arch=c000003e syscall=262 success=no
    exit=-13 a0=fffffff9c a1=7f3f4c045e58 a2=7f3f517e18b0 a3=100 items=0 ppid=1248
    pid=1251 auid=4294967295 uid=48 gid=48 euid=48 suid=48 egid=48
    sgid=48 fsgid=48 tty=(none) ses=4294967295 comm="httpd" exe="/usr/sbin/httpd"
    subj=system_u:system_r:httpd_t:s0 key=(null)
type=AVC msg=audit(1697754346.334:117): avc: denied { setattr }
    for pid=1251 comm="httpd" path="/var/www/html/index.html"
    dev="vda4" ino=16857024 scontext=system_u:system_r:httpd_t:s0
    tcontext=unconfined_u:object_r:admin_home_t:s0 tclass=file permissive=0
-----
time->Thu Oct 19 18:27:23 2023
type=PROCTITLE msg=audit(1697754443.937:123):
    proctitle=2F7573722F7362696E2F6874747064002D44464F524547524F554E44
type=SYSCALL msg=audit(1697754443.937:123): arch=c000003e syscall=262 success=no
    exit=-13 a0=fffffff9c a1=7f3f4c03dd38 a2=7f3f4bff8b0 a3=0 items=0 ppid=1248
    pid=1251 auid=4294967295 uid=48 gid=48 euid=48 suid=48 egid=48
    sgid=48 fsgid=48 tty=(none) ses=4294967295 comm="httpd" exe="/usr/sbin/httpd"
    subj=system_u:system_r:httpd_t:s0 key=(null)
type=AVC msg=audit(1697754443.937:123): avc: denied { setattr }
    for pid=1251 comm="httpd" path="/var/www/html/index.html"
    dev="vda4" ino=16857024 scontext=system_u:system_r:httpd_t:s0
    tcontext=unconfined_u:object_r:admin_home_t:s0 tclass=file permissive=0
-----
time->Thu Oct 19 18:27:23 2023
type=PROCTITLE msg=audit(1697754443.937:124):
    proctitle=2F7573722F7362696E2F6874747064002D44464F524547524F554E44
```

Chapter 8 | Mitigating Risk with SELinux

```
type=SYSCALL msg=audit(1697754443.937:124): arch=c000003e syscall=262 success=no
exit=-13 a0=fffff9c a1=7f3f4c03de18 a2=7f3f4bff8b0 a3=100 items=0 ppid=1248
pid=1251 auid=4294967295 uid=48 gid=48 euid=48 suid=48 egid=48
sgid=48 fsgid=48 tty=(none) ses=4294967295 comm="httpd" exe="/usr/sbin/httpd"
subj=system_u:system_r:httpd_t:s0 key=(null)
type=AVC msg=audit(1697754443.937:124): avc: denied { getattr }
for pid=1251 comm="httpd" path="/var/www/html/index.html"
dev="vda4" ino=16857024 scontext=system_u:system_r:httpd_t:s0
tcontext=unconfined_u:object_r:admin_home_t:s0 tclass=file permissive=0
-----
...output omitted...
```

The last message indicates that SELinux denies the httpd process access to the /var/www/html/index.html file.

- 2.14. Review the context type of the /var/www/html/index.html file and its parent directory.

```
[root@serverd ~]# ls -az /var/www/html/
system_u:object_r:httpd_sys_content_t:s0 .
system_u:object_r:httpd_sys_content_t:s0 ..
unconfined_u:object_r:admin_home_t:s0 index.html
```

The index.html file does not have the correct context.

- 2.15. You do not need to add a new file context rule because the /var/www/html directory is the default DocumentRoot directory and already has a rule. You only need to relabel the index.html file.

```
[root@serverd ~]# restorecon -v /var/www/html/index.html
Relabeled /var/www/html/index.html from unconfined_u:object_r:admin_home_t:s0 to
unconfined_u:object_r:httpd_sys_content_t:s0
```

- 2.16. Confirm that you can now access the index.html page.

```
[root@serverd ~]# curl http://localhost/index.html
<html><body>Hello, World! from serverd</body></html>
```

- 2.17. Disable SELinux denial auditing.

```
[root@serverd ~]# semodule -B
```

3. Confine the users on the serverd machine to prevent them from using the sudo and su commands. Further, they must not be able to run programs in the /tmp/ directory or in their home directory. These restrictions do not apply to the root user.

You can test your restrictions by logging in as the student user and trying to use the sudo -i command. You can also try to execute the runme program in the /tmp/ and /home/student/ directories to verify your work.

- 3.1. Change the default mapping between the Linux users and the SELinux users. Map the Linux users to the user_u SELinux user.

```
[root@serverd ~]# semanage login -m -s user_u -r s0 __default__  
[root@serverd ~]#
```

- 3.2. To prevent users from running programs in the `/tmp/` directory or their home directory, set the SELinux `user_exec_content` Boolean to the `off` value.

```
[root@serverd ~]# setsebool -P user_exec_content off  
[root@serverd ~]#
```

- 3.3. Log out of the `serverd` machine and log in again as the `student` user.

```
[root@serverd ~]# logout  
[student@serverd ~]$ logout  
[student@workstation ~]$ ssh student@serverd  
[student@serverd ~]$
```

- 3.4. Confirm that you can no longer use the `sudo -i` command.

```
[student@serverd ~]$ sudo -i  
sudo: PERM_SUDOERS: setresuid(-1, 1, -1): Operation not permitted  
sudo: no valid sudoers sources found, quitting  
sudo: setresuid() [0, 0, 0] -> [1000, -1, -1]: Operation not permitted  
sudo: error initializing audit plugin sudoers_audit  
[student@serverd ~]$
```

- 3.5. Confirm that you cannot execute programs in the `/tmp` or `/home/student` directories. Use the `runme` program for that test.

```
[student@serverd ~]$ /tmp/runme  
-bash: /tmp/runme: Permission denied  
[student@serverd ~]$ ./runme  
-bash: ./runme: Permission denied
```

4. Create the following system administrator account:

- Username: `operator2`
- Password: `redhat`
- Supplementary group: `wheel`

Map the `operator2` user to a confined SELinux user that has access to the `su` and `sudo` commands. Additionally, the `operator2` user must be able to log in by using SSH.

- 4.1. Log out of the `serverd` machine and log in as the `root` user.

```
[student@serverd ~]$ logout  
[student@workstation ~]$ ssh root@serverd  
[root@serverd ~]#
```

- 4.2. Create the `operator2` account according to the requirements, and map it to the `sysadm_u` SELinux user.

```
[root@serverd ~]# useradd -G wheel -z sysadm_u operator2
[root@serverd ~]# echo redhat | passwd --stdin operator2
Changing password for user operator2.
passwd: all authentication tokens updated successfully.
```

- 4.3. Set the `ssh_sysadm_login` SELinux Boolean to the `on` value so that `sysadm_u` SELinux users can log in by using SSH.

```
[root@serverd ~]# setsebool -P ssh_sysadm_login on
[root@serverd ~]#
```

- 4.4. Log out of the `serverd` machine and log in as the `operator2` user.

```
[root@serverd ~]# logout
[student@workstation ~]$ ssh operator2@serverd
[operator2@serverd ~]$
```

- 4.5. Confirm that you can use the `sudo -i` command to become the `root` user. Use `redhat` for the password. When done, log out of the `serverd` machine.

```
[operator2@serverd ~]$ sudo -i
[sudo] password for operator2: redhat
[root@serverd ~]# logout
[operator2@serverd ~]$ logout
[student@workstation ~]$
```

5. Locate the `type_transition` SELinux rule that explains the following behavior:

- The `ps -Z $$` command shows that the `student` user's bash process is running under the `user_t` domain type.
- The `ls -Z /usr/bin/passwd` command shows that the `passwd` binary file has the `passwd_exec_t` type.
- When the `student` user runs the `passwd` command to change their password, the resulting process is running under the `passwd_t` domain type.

After you have located the rule, store it in the `/home/student/rule` file on the `serverd` machine. The grading script at the end of this exercise uses that file to verify your work.

- 5.1. Replicate the scenario to confirm the described behavior.

Log in to the `serverd` machine as the `student` user. No password is required.

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

- 5.2. Retrieve the domain type of the bash process.

```
[student@serverd ~]$ ps -Z $$
```

LABEL	PID	TTY	STAT	TIME	COMMAND
user_u:user_r:user_t:s0	3624	pts/0	Ss	0:00	-bash

- 5.3. Retrieve the context type of the /usr/bin/passwd binary file.

```
[student@serverd ~]$ ls -Z /usr/bin/passwd
system_u:object_r:passwd_exec_t:s0 /usr/bin/passwd
```

- 5.4. Run the passwd command but do not enter any password. Leave the command running. You do not want to change your password; you just want to inspect the context of the process.

```
[student@serverd ~]$ passwd
Changing password for user student.
Current password:
```

- 5.5. Open a new terminal on the workstation machine and log in to the serverd machine as the **root** user.

```
[student@workstation ~]$ ssh root@serverd
[root@serverd ~]#
```

- 5.6. Retrieve the context type of the running passwd process.

```
[root@serverd ~]# ps -Z -C passwd
LABEL PID TTY TIME CMD
user_u:object_r:passwd_exec_t:s0 3766 pts/0 00:00:00 passwd
```

- 5.7. Install the setools-console package to get the sesearch command.

```
[root@serverd ~]# dnf install setools-console
...output omitted...
Is this ok [y/N]: y
...output omitted...
Complete!
```

- 5.8. Use the sesearch command with the following options:

- **-T**: search for transition rules.
- **-s user_t**: the source domain is the domain of the **student** user's Bash shell.
- **-t passwd_exec_t**: the target is the context of the **passwd** binary file.

```
[root@serverd ~]# sesearch -T -s user_t -t passwd_exec_t
type_transition user_t passwd_exec_t:process passwd_t;
```

- 5.9. Redirect the output of the previous command to the /home/student/rule file. When done, log out of the **serverd** machine.

```
[root@serverd ~]# sesearch -T -s user_t -t passwd_exec_t > /home/student/rule
[root@serverd ~]# logout
[student@workstation ~]$
```

In the other terminal, quit the `passwd` command and log out of the `serverd` machine.

```
^C  
[student@serverd ~]$ logout  
[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 selinux-review
```

Finish

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

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

Summary

- To migrate a system that has SELinux disabled to **enforcing** mode: first, switch to **permissive** mode; then, review the audit log, relabel files, and resolve issues; and finally, switch to **enforcing** mode.
- By setting confined SELinux users, you can restrict users from using the `sudo` or `su` commands to switch user, logging in by using the `ssh` command, or running some commands on the system.
- You can use the `sesearch` command to look up the access rules and transition rules that SELinux enforces.
- You can use the `sepolicy transition` command to analyze whether a process running in one domain can potentially use one or more domain transitions to run a process in another domain.
- You can use the `matchpathcon` command to determine the expected context of a file that is created in a particular location, even if the file does not exist.

Chapter 9

Managing Compliance with OpenSCAP

Goal

Evaluate and remediate a server's compliance with security policies by using OpenSCAP.

Sections

- Installing OpenSCAP (and Guided Exercise)
- Scanning and Analyzing Compliance (and Guided Exercise)
- Customizing OpenSCAP Policy (and Guided Exercise)
- Remediating OpenSCAP Issues with Ansible (and Guided Exercise)

Lab

- Managing Compliance with OpenSCAP

Installing OpenSCAP

Objectives

- Explain basic OpenSCAP concepts, tools, and profiles, and prepare a system for a local OpenSCAP scan.

OpenSCAP and Security Compliance in Red Hat Enterprise Linux

Enterprise computing environments might consist of hundreds or thousands of interconnected computer systems that are running many applications and services, and which are accessed by a large and diverse set of users and applications. To maintain control over the security of this vast environment, a standard way to scan systems for compliance with security policies is needed.

The *National Institute of Standards and Technology (NIST)*, along with other authorities, developed a standard compliance system called *Security Content Automation Protocol (SCAP)*. The SCAP standard is a framework of security specifications. SCAP standards support automated configuration, and vulnerability and patch checking measurement. The OpenSCAP project is an open source project that develops tools for implementing and enforcing security policies by using the SCAP standard.

The OpenSCAP project also provides a number of predefined and customizable compliance policies in the SCAP format for use with OpenSCAP tools. The needs and risk profiles of each organization are different, and often require the ability to customize the compliance policy checklist. As a result, the compliance policy varies across organizations. To verify whether a given object follows a rule in a *compliance policy*, perform a *compliance audit*.

Red Hat Enterprise Linux provides tools that are based on the SCAP standard and that enable administrators to run a fully automated compliance audit. In addition to natively providing OpenSCAP tooling, Red Hat provides the underlying development libraries. This approach allows independent software vendors (ISVs) to embed NIST-certified configuration and vulnerability scanning into their applications.

Security Compliance Tools

The following security compliance tools are provided on Red Hat Enterprise Linux 9:

OpenSCAP

The `oscap` command-line utility, provided by the `openscap-scanner` package, performs configuration and vulnerability scans, and generates reports and guidance based on these scans.

SCAP Security Guide (SSG)

This predefined collection of security policies for Linux systems is provided in the `scap-security-guide` package. The guide provides a catalog of hardening advice that is linked to various government requirements to help define and customize security policies according to the organization's needs. This guide is not just documentation; rather, this guide provides rules and scripts that are used by the `oscap` command.

Script Check Engine (SCE)

The `openscap-engine-sce` package provides the SCE extension that enables you to write security content by using Bash, Python, or Ruby.

SCAP Workbench

This graphical utility performs scans on a single local or a remote system, and generates security reports based on these scans. The utility can also be used to customize compliance policies.

The SCAP Security Guide

The SCAP Security Guide is a collection of security policies for Linux systems, in the form of SCAP documents. The guide consists of rules with detailed descriptions and proven remediation scripts and Ansible Playbooks. The SCAP Security Guide can be used with OpenSCAP tools to automate the auditing of a Linux system.

SCAP Security Guide transforms the security guidelines that are recommended by different authorities into a machine-readable format that can be used by OpenSCAP to audit your system. The guide builds multiple security baselines from the high-quality SCAP content. If your system must comply with one of the provided baselines, then you can select the appropriate profile from the SCAP Security Guide. However, most real world deployments require adjustments to the profile based on the organization's security requirements.

Various security policies are available in the SCAP Security Guide, such as Fedora Linux, Red Hat Enterprise Linux, Mozilla Firefox, and others. Red Hat recommends that you write a security policy in a proactive way that balances security risk against business needs. Security policies should be regularly updated and maintained, and must incorporate any government and industry requirements.

For ease of use, all the available security policies are broken into *profiles*. A profile can be defined as a grouping of security settings that correlate to a known policy.

Use the `dnf install scap-security-guide` command to install the SCAP Security Guide. This command automatically installs the `openscap-scanner` package as a dependency. The `openscap-scanner` package contains the OpenSCAP command-line tool called `oscap`.

```
[root@host ~]# dnf install scap-security-guide
```

The `scap-security-guide` package installs a predefined data stream file in the `/usr/share/xml/scap/ssg/content/` directory. The SCAP source data stream file (`ssg-rhel9-ds.xml`) contains all the data that were contained in the XCCDF file (`ssg-rhel9-xccdf.xml`) in previous versions of RHEL. The SCAP source data stream is a container file that includes all the components that are needed to perform a compliance scan. Using the SCAP source data stream instead of XCCDF files has been recommended since Red Hat Enterprise Linux 7. In previous versions of Red Hat Enterprise Linux, the data in the XCCDF file and SCAP source data stream was duplicated. In Red Hat Enterprise Linux 9, this duplication is removed to reduce the RPM package size. If your scenario requires using separate files instead of the data stream, then you can split the data stream file by using the following command:

```
[root@host ~]# oscap ds sds-split \
/usr/share/xml/scap/ssg/content/ssg-rhel9-ds.xml output_directory
```

The SCAP source data stream supports document generation, information interchange, automation of compliance testing, compliance scoring, and situational tailoring. The data stream

file contains the SCAP profiles with all the rules that are required to run an evaluation or a scan. You learn how to use this file in upcoming sections of this chapter.

```
[root@host ~]# ls -l /usr/share/xml/scap/ssg/content/
total 22548
-rw-r--r--. 1 root root 23088822 Feb 14 2023 ssg-rhel9-ds.xml
```

To review all the security rules that are associated with a profile, you can consult the data stream file. However, you can also use the `oscap` command to generate a user-friendly HTML version of the security guide for a specific profile.

To use the `oscap` command to generate the HTML security guide for a specific profile, you must provide the profile's unique `id` attribute. You can use the `oscap info` command to parse the data stream file and to display the profiles, along with their `id` attributes.

The following example uses the `oscap info` command to inspect the security content:

```
[root@host ~]# oscap info /usr/share/xml/scap/ssg/content/ssg-rhel9-ds.xml
Document type: Source Data Stream
Imported: 2023-02-14T07:34:39

Stream: scap_org.open-scap_datastream_from_xccdf_ssg-rhel9-xccdf.xml
Generated: (null)
Version: 1.3
Checklists:
  Ref-Id: scap_org.open-scap_cref_ssg-rhel9-xccdf.xml
    Status: draft
    Generated: 2023-02-14
    Resolved: true
  Profiles:
    Title: ANSSI-BP-028 (enhanced)
      Id: xccdf_org.ssgproject.content_profile_anssi_bp28_enhanced
    Title: ANSSI-BP-028 (high)
      Id: xccdf_org.ssgproject.content_profile_anssi_bp28_high
    Title: ANSSI-BP-028 (intermediary)
      Id: xccdf_org.ssgproject.content_profile_anssi_bp28_intermediary
    Title: ANSSI-BP-028 (minimal)
      Id: xccdf_org.ssgproject.content_profile_anssi_bp28_minimal
    Title: CIS Red Hat Enterprise Linux 9 Benchmark for Level 2 - Server
      Id: xccdf_org.ssgproject.content_profile_cis
    Title: CIS Red Hat Enterprise Linux 9 Benchmark for Level 1 - Server
      Id: xccdf_org.ssgproject.content_profile_cis_server_l1
    Title: CIS Red Hat Enterprise Linux 9 Benchmark for Level 1 -
  Workstation
    Id: xccdf_org.ssgproject.content_profile_cis_workstation_l1
    Title: CIS Red Hat Enterprise Linux 9 Benchmark for Level 2 -
  Workstation
    Id: xccdf_org.ssgproject.content_profile_cis_workstation_l2
    Title: [DRAFT] Unclassified Information in Non-federal Information
Systems and Organizations (NIST 800-171)
      Id: xccdf_org.ssgproject.content_profile_cui
...output omitted...
```

Chapter 9 | Managing Compliance with OpenSCAP

The output contains the available configuration profiles. To generate the HTML security guide, choose the appropriate profile and use the `oscap xccdf generate guide` command. The following command generates the HTML security guide for the DISA STIG for Red Hat Enterprise Linux 9 profile.

```
[root@host ~]# oscap xccdf generate guide \
--profile xccdf_org.ssgproject.content_profile_stig \
/usr/share/xml/scap/ssg/content/ssg-rhel9-ds.xml > guide.html
```

Use a web browser to open the final `guide.html` security guide. The HTML security guide contains all the available rules in an organized form.

SCAP Workbench

SCAP Workbench is a graphical tool that enables users to perform configuration scans, remediate the system, and generate reports based on the evaluations. The tool can scan the local system, or it can use SSH to connect to and scan a single remote system.

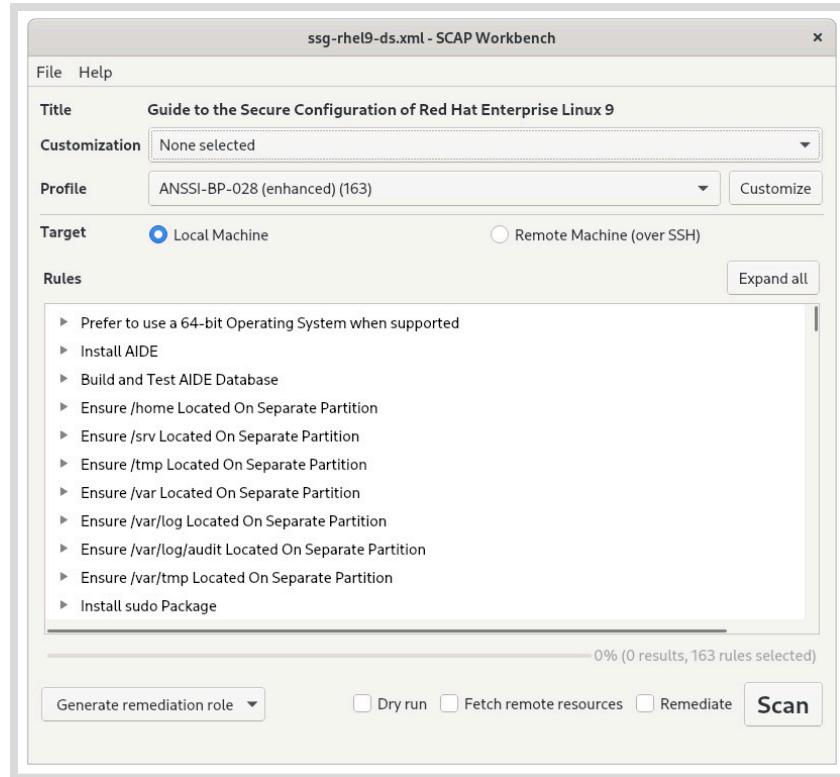


Figure 9.1: SCAP Workbench interface

Use the `dnf install scap-workbench` command to install SCAP Workbench.

```
[root@host ~]# dnf install scap-workbench
```

When you launch SCAP Workbench, you can choose the security content that you want to use. The `scap-security-guide` package is installed as a dependency of SCAP Workbench. You can choose from the predefined content that the `scap-security-guide` package installs in the `/usr/share/xml/scap/ssg/content/` directory.

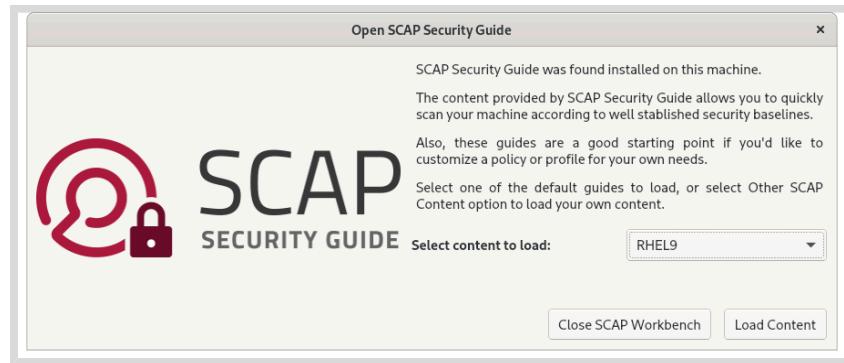


Figure 9.2: Choosing SCAP Workbench content

OpenSCAP Scans

To scan a file system, use the `oscap` command-line utility, which is provided as part of the OpenSCAP project. If SCAP Workbench is installed on the system, then the `oscap` utility is installed as a dependency. If you are scanning a remote host that does not have SCAP Workbench installed, then you must install the `oscap` tool separately.

To prepare your remote system for an OpenSCAP scan, install the `openscap-scanner` package, which contains the `oscap` command-line utility.

```
[root@host ~]# dnf install openscap-scanner
```

The `oscap` command uses security policies that are defined as SCAP documents. Red Hat recommends using the `scap-security-guide` package, which provides the SCAP Security Guide.

```
[root@host ~]# dnf install scap-security-guide
```

You can also start by installing only the SCAP Security Guide, which installs the `openscap-scanner` as a dependency.

With both packages installed, your system is ready for an OpenSCAP scan. In the next section of this course, you learn how to perform and evaluate the results of a scan.



References

`oscap(8)` man page

For more information, refer to the *Scanning the System for Configuration Compliance and Vulnerabilities* chapter in the *Red Hat Enterprise Linux 9 Security Hardening* guide at

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html-single/security_hardening/index#scanning-the-system-for-configuration-compliance-and-vulnerabilities_security-hardening

► Guided Exercise

Installing OpenSCAP

Install OpenSCAP tools and the SCAP Security Guide on a server and examine the files that they provide.

Outcomes

- Install OpenSCAP tools and the SCAP Security Guide on a server.

Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

```
[student@workstation ~]$ lab start openscap-installing
```

Instructions

- 1. On the `serverc` machine, install the `oscap` command-line tool and the SCAP Security Guide.

1.1. Log in to the `serverc` machine as the `student` user. No password is required.

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

1.2. Use the `sudo -i` command to switch identity to the `root` user. Use `student` as the password.

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

1.3. The `openscap-scanner` package provides the `oscap` command-line utility. Install that package.

```
[root@serverc ~]# dnf install openscap-scanner  
...output omitted...  
Complete!
```

1.4. Run the `oscap -V` command to confirm that the tool is now available.

```
[root@serverc ~]# oscap -V  
OpenSCAP command line tool (oscap) 1.3.7  
Copyright 2009--2021 Red Hat Inc., Durham, North Carolina.  
...output omitted...
```

15. The oscap command needs some security content in order to work. Install the scap-security-guide package, which provides the SCAP Security Guide. The SCAP Security Guide contains some standard security policies for Linux systems.

```
[root@serverc ~]# dnf install scap-security-guide  
...output omitted...  
Complete!
```

► 2. Review the available profiles in the SCAP Security Guide.

- 2.1. The scap-security-guide package installs the scap-security-guide(8) man page. On this man page, review the **Red Hat Enterprise Linux 9 PROFILES** section, which describes the available profiles.

```
[root@serverc ~]# man scap-security-guide
```

- 2.2. Another way to list the available profiles is to directly review the XCCDF XML files. The scap-security-guide package deploys those files to the /usr/share/xml/scap/ssg/content/ directory. In that directory, extract the profile list from the ssg-rhel9-ds.xml file.

```
[root@serverc ~]# grep 'content_profile' \  
/usr/share/xml/scap/ssg/content/ssg-rhel9-ds.xml  
<xccdf-1.2:Profile  
id="xccdf_org.ssgproject.content_profile_anssi_bp28_enhanced">  
<xccdf-1.2:Profile  
id="xccdf_org.ssgproject.content_profile_anssi_bp28_high">  
<xccdf-1.2:Profile  
id="xccdf_org.ssgproject.content_profile_anssi_bp28_intermediary">  
<xccdf-1.2:Profile  
id="xccdf_org.ssgproject.content_profile_anssi_bp28_minimal">  
<xccdf-1.2:Profile id="xccdf_org.ssgproject.content_profile_cis">  
<xccdf-1.2:Profile id="xccdf_org.ssgproject.content_profile_cis_server_l1">  
<xccdf-1.2:Profile  
id="xccdf_org.ssgproject.content_profile_cis_workstation_l1">  
<xccdf-1.2:Profile  
id="xccdf_org.ssgproject.content_profile_cis_workstation_l2">  
<xccdf-1.2:Profile id="xccdf_org.ssgproject.content_profile_cui">  
<xccdf-1.2:Profile id="xccdf_org.ssgproject.content_profile_e8">  
<xccdf-1.2:Profile id="xccdf_org.ssgproject.content_profile_hipaa">  
<xccdf-1.2:Profile id="xccdf_org.ssgproject.content_profile_ism_o">  
<xccdf-1.2:Profile id="xccdf_org.ssgproject.content_profile_ospp">  
<xccdf-1.2:Profile id="xccdf_org.ssgproject.content_profile_pci-dss">  
<xccdf-1.2:Profile id="xccdf_org.ssgproject.content_profile_stig">  
<xccdf-1.2:Profile id="xccdf_org.ssgproject.content_profile_stig_gui">
```

The id attribute provides a unique identifier for each profile. You use this identifier with the oscap command to indicate which profile to use during a system scan.

- 2.3. The oscap info command can also parse this XML file and display the profiles.

```
[root@serverc ~]# oscap info /usr/share/xml/scap/ssg/content/ssg-rhel9-ds.xml
WARNING: Datastream component 'scap_org.open-scap_cref_security-data-oval-com.redhat.rhsa-RHEL9.xml.bz2' points out to the remote 'https://access.redhat.com/security/data/oval/com.redhat.rhsa-RHEL9.xml.bz2'. Use '--fetch-remote-resources' option to download it.
WARNING: Skipping 'https://access.redhat.com/security/data/oval/com.redhat.rhsa-RHEL9.xml.bz2' file which is referenced from datastream
Document type: Source Data Stream
Imported: 2023-02-14T07:34:39

Stream: scap_org.open-scap_datastream_from_xccdf_ssg-rhel9-xccdf.xml
Generated: (null)
Version: 1.3
Checklists:
  Ref-Id: scap_org.open-scap_cref_ssg-rhel9-xccdf.xml
    Status: draft
    Generated: 2023-02-14
    Resolved: true
  Profiles:
    Title: ANSSI-BP-028 (enhanced)
      Id: xccdf_org.ssgproject.content_profile_anssi_bp28_enhanced
    Title: ANSSI-BP-028 (high)
      Id: xccdf_org.ssgproject.content_profile_anssi_bp28_high
    Title: ANSSI-BP-028 (intermediary)
      Id: xccdf_org.ssgproject.content_profile_anssi_bp28_intermediary
    Title: ANSSI-BP-028 (minimal)
      Id: xccdf_org.ssgproject.content_profile_anssi_bp28_minimal
    Title: CIS Red Hat Enterprise Linux 9 Benchmark for Level 2 - Server
      Id: xccdf_org.ssgproject.content_profile_cis
    Title: CIS Red Hat Enterprise Linux 9 Benchmark for Level 1 - Server
      Id: xccdf_org.ssgproject.content_profile_cis_server_l1
    Title: CIS Red Hat Enterprise Linux 9 Benchmark for Level 1 - Workstation
      Id: xccdf_org.ssgproject.content_profile_cis_workstation_l1
    Title: CIS Red Hat Enterprise Linux 9 Benchmark for Level 2 - Workstation
      Id: xccdf_org.ssgproject.content_profile_cis_workstation_l2
    Title: [DRAFT] Unclassified Information in Non-federal Information Systems and
          Organizations (NIST 800-171)
      Id: xccdf_org.ssgproject.content_profile_cui
...output omitted...
```

The warnings indicate that there are remote resources that the `oscap` command can download. For the purposes of the guided exercise, you can ignore the warning.

- ▶ 3. Generate the HTML security guide for the DISA STIG for Red Hat Enterprise Linux 9 profile (`xccdf_org.ssgproject.content_profile_stig`) and review the security rules that are included in that profile.
 - 3.1. One way to retrieve the rules that are associated with a profile is to consult the `/usr/share/xml/scap/ssg/content/ssg-rhel9-ds.xml` XCCDF XML file. However, the `oscap` command can generate a more readable HTML version of the security guide for a specific profile.
Use the `oscap xccdf generate guide` command to generate the HTML security guide for the DISA STIG for Red Hat Enterprise Linux 9 profile.

```
[root@serverc ~]# oscap xccdf generate guide \
--profile xccdf_org.ssgproject.content_profile_stig \
/usr/share/xml/scap/ssg/content/ssg-rhel9-ds.xml > guide.html
[root@serverc ~]#
```

- 3.2. Use scp to copy the `guide.html` file to workstation so that you can use Firefox to display it. Use `student` as the password.

```
[root@serverc ~]# scp guide.html student@workstation:
The authenticity of host 'workstation (172.25.250.9)' can't be established.
ED25519 key fingerprint is SHA256:Lavlr4HiIVFIWK+0Et/1swi7gAyFe5W05vc7HP4VhGo.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'workstation' (ED25519) to the list of known hosts.
student@workstation's password: student
guide.html                                         100% 7493KB 171.6MB/s   00:00
```

- 3.3. Log off from serverc.

```
[root@serverc ~]# logout
[student@serverc ~]$ logout
[student@workstation ~]$
```

- 3.4. Use Firefox to display the `guide.html` file. Browse through the page and review some rules. Close Firefox when you are done.

```
[student@workstation ~]$ firefox guide.html
```

Navigate to Table of Contents > System Settings. Click the **Installing and Maintaining Software** link to review the **Updating Software** group information and **Ensure gpgcheck Enabled In Main dnf Configuration** rule details.

- ▶ 4. On `workstation`, use the SCAP Workbench graphical utility to review the DISA STIG for Red Hat Enterprise Linux 9 profile from the SCAP Security Guide.

- 4.1. Install the `scap-workbench` package.

```
[student@workstation ~]$ sudo dnf install scap-workbench
...output omitted...
Complete!
```

- 4.2. Use the `scap-workbench` command to start SCAP Workbench.

```
[student@workstation ~]$ scap-workbench
```

The SCAP Workbench detects that the SCAP Security Guide is already installed on the system and asks you to select the content to use.

In the **Select content to load** field, select RHEL9 and click **Load Content**.

- 4.3. Locate the **Profile** field and select **DISA STIG for Red Hat Enterprise Linux 9**. The lower part of the window displays the rules that are associated with

that profile. Review the rules but do not initiate a scan at this time. Close SCAP Workbench when you are done exploring.

Finish

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

```
[student@workstation ~]$ lab finish openscap-installing
```

Scanning and Analyzing Compliance

Objectives

- Evaluate a server's compliance with the requirements that are specified by a policy from the SCAP Security Guide by using OpenSCAP tools.

Overview of the OpenSCAP Command-line Tool

You can use the `oscap` command-line tool to scan a system for compliance with a SCAP policy, to generate remediation scripts, and to create reports and guides.

The `oscap` command needs some security content in order to work. The `scap-security-guide` package provides the SCAP Security Guide, which contains some standard security policies for Linux systems. The package installs content files in the `/usr/share/xml/scap/ssg/content/` directory.

The files with names that end with `-ds.xml` in that directory are XCCDF data stream files.

```
[user@host ~]$ ls /usr/share/xml/scap/ssg/content/
ssg-rhel9-ds.xml
```

A data stream file can also define multiple profiles that you can choose when scanning a system for compliance. To list the available profiles, use the `oscap info` command.

```
[user@host ~]$ oscap info /usr/share/xml/scap/ssg/content/ssg-rhel9-ds.xml
WARNING: Datastream component 'scap_org.open-scap_cref_security-data-oval-com.redhat.rhsa-RHEL9.xml.bz2' points out to the remote 'https://access.redhat.com/security/data/oval/com.redhat.rhsa-RHEL9.xml.bz2'. Use '--fetch-remote-resources' option to download it.
WARNING: Skipping 'https://access.redhat.com/security/data/oval/com.redhat.rhsa-RHEL9.xml.bz2' file which is referenced from datastream
Document type: Source Data Stream
Imported: 2023-02-14T07:34:39

Stream: scap_org.open-scap_datastream_from_xccdf_ssg-rhel9-xccdf.xml
Generated: (null)
Version: 1.3
Checklists:
  Ref-Id: scap_org.open-scap_cref_ssg-rhel9-xccdf.xml
  Status: draft
  Generated: 2023-02-14
  Resolved: true
Profiles:
  Title: ANSSI-BP-028 (enhanced)
  Id: xccdf_org.ssgproject.content_profile_anssi_bp28_enhanced
  Title: ANSSI-BP-028 (high)
  Id: xccdf_org.ssgproject.content_profile_anssi_bp28_high
  Title: ANSSI-BP-028 (intermediary)
```

```
Id: xccdf_org.ssgproject.content_profile_anssi_bp28_intermediary
Title: ANSSI-BP-028 (minimal)
  Id: xccdf_org.ssgproject.content_profile_anssi_bp28_minimal
...output omitted...
```

Scanning a System for Compliance

To scan a system, install the `openscap-scanner` and `scap-security-guide` packages on that system, and select the XCCDF data stream file and the profile that you intend to use.

Run the `oscap xccdf eval` command to scan the system. Provide the data stream file as an argument, and the identifier of the profile to use with the `--profile` option. The command displays the result of each test on the standard output, and you can save these results in an XML file with the `--results` option. With that file, you can later generate reports and remediation scripts.

The following example uses the `oscap xccdf eval` command to scan the local system. The example uses the `/usr/share/xml/scap/ssg/content/ssg-rhel9-ds.xml` data stream file for Red Hat Enterprise Linux 9. The security profile that is applied from the data stream is the DISA STIG for Red Hat Enterprise Linux 9 profile (`xccdf_org.ssgproject.content_profile_stig`). The command in the example saves the results to the `/root/results.xml` file.

```
[root@host ~]# oscap xccdf eval \
  --profile xccdf_org.ssgproject.content_profile_stig \
  --results /root/results.xml \
  /usr/share/xml/scap/ssg/content/ssg-rhel9-ds.xml
WARNING: Datastream component 'scap_org.open-scap_cref_security-data-oval-com.redhat.rhsa-RHEL9.xml.bz2' points out to the remote 'https://access.redhat.com/security/data/oval/com.redhat.rhsa-RHEL9.xml.bz2'. Use '--fetch-remote-resources' option to download it.
WARNING: Skipping 'https://access.redhat.com/security/data/oval/com.redhat.rhsa-RHEL9.xml.bz2' file which is referenced from datastream
WARNING: Skipping ./security-data-oval-com.redhat.rhsa-RHEL9.xml.bz2 file which is referenced from XCCDF content
--- Starting Evaluation ---

Title  Install AIDE
Rule   xccdf_org.ssgproject.content_rule_package_aide_installed
Ident  CCE-90843-4
Result fail

Title  Configure AIDE to Verify the Audit Tools
Rule   xccdf_org.ssgproject.content_rule_aide_check_audit_tools
Ident  CCE-87757-1
Result fail

Title  Configure Periodic Execution of AIDE
Rule   xccdf_org.ssgproject.content_rule_aide_periodic_cron_checking
Ident  CCE-83437-4
Result fail

Title  Configure Notification of Post-AIDE Scan Details
Rule   xccdf_org.ssgproject.content_rule_aide_scan_notification
```

```
Ident  CCE-90844-2
Result fail

Title  Configure AIDE to Verify Access Control Lists (ACLS)
Rule   xccdf_org.ssgproject.content_rule_aide_verify_acls
Ident  CCE-90837-6
Result fail

Title  Configure AIDE to Verify Extended Attributes
Rule   xccdf_org.ssgproject.content_rule_aide_verify_ext_attributes
Ident  CCE-83439-0
Result fail

Title  Audit Tools Must Be Group-owned by Root
Rule   xccdf_org.ssgproject.content_rule_file_audit_tools_group_ownership
Ident  CCE-86240-9
Result pass

...output omitted...
```

You must run the command as the `root` user because the scan process might need to evaluate files that are only accessible to the `root` user.

Notice the warning messages in the command output. The scan skips some files because the scan requires an up-to-date list of patches to control. If the system can access the internet, then the `oscap` command can download that list from Red Hat. In that case, add the `--fetch-remote-resources` option.

Generating and Viewing the HTML Report

After the scan is complete, you can use the resulting XML file to generate a complete report in HTML format.

Run the `oscap xccdf generate report` command.

```
[root@host ~]# oscap xccdf generate report results.xml > results.html
```

After you generate the `results.html` file, use Mozilla Firefox or another web browser to view it.

```
[root@host ~]# firefox results.html
```

In Firefox, navigate to the **Evaluation Characteristics** section to view initial details related to the target system of the compliance scan. This section includes scan characteristics, such as the name of the system that is targeted for evaluation, start and finish times, network addresses, and other scan results.

Evaluation Characteristics	
Evaluation target	serverc.lab.example.com
Benchmark URL	/usr/share/xml/scap/ssg/content/ssg-rhel9-ds.xml
Benchmark ID	xccdf_org.ssgproject.content_benchmark_RHEL-9
Benchmark version	0.1.66
Profile ID	xccdf_org.ssgproject.content_profile_stig
Started at	2023-10-23T10:29:47-05:00
Finished at	2023-10-23T10:30:09-05:00
Performed by	student
Test system	cpe:/a:redhat:openscap:1.3.7

CPE Platforms

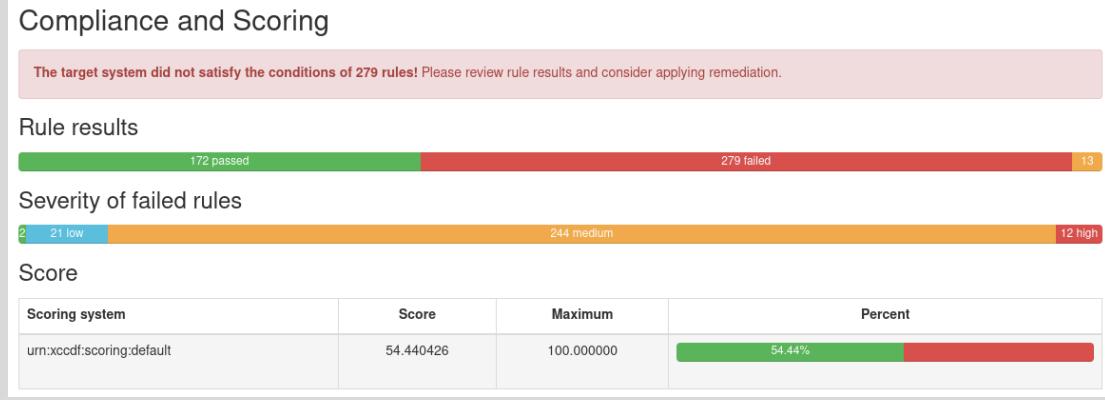
- cpe:/o:redhat:enterprise_linux:9

Addresses

- IPv4 127.0.0.1
- IPv4 172.25.250.12
- IPv6 0:0:0:0:0:0:1
- IPv6 fe80:0:0:5054:ff:fe00:fa0c
- MAC 00:00:00:00:00:00
- MAC 52:54:00:00:FA:0C

Figure 9.3: Evaluation characteristics for the OpenSCAP scan

Navigate to the **Compliance and Scoring** section to view a chart of the total number of rules that either passed or failed compliance, and the totals of severity levels for failed rules.

**Figure 9.4: Compliance and scoring totals of the OpenSCAP scan results**

Navigate to the **Rule Overview** section to view rule groups. The selection for the **Group rules by** list is currently set to **Default**. You can choose the **Severity** or **Result** options to change the grouping view to meet specific needs.

The default view groups rules by title. Rules can have several types of results but the most common are **pass** and **fail**, which indicate whether that particular security control has passed or failed the scan.

Chapter 9 | Managing Compliance with OpenSCAP

The screenshot shows the 'Rule Overview' page of the OpenSCAP interface. At the top, there are several filter checkboxes: 'pass' (checked), 'fail' (unchecked), 'fixed' (checked), 'error' (unchecked), 'informational' (checked), 'notchecked' (checked), and 'notapplicable' (checked). Below these are search fields for 'Search through XCCDF rules' and 'Group rules by:' with a dropdown set to 'Default'. The main area displays a hierarchical tree of rules under 'Guide to the Secure Configuration of Red Hat Enterprise Linux 9'. The tree includes categories like 'System Settings', 'Installing and Maintaining Software', 'System and Software Integrity', 'Disk Partitioning', 'GNOME Desktop Environment', 'Sudo', 'System Tooling / Utilities', and 'Updating Software'. Each node shows its status: '239x fail', '12x notchecked', '279x fail', '13x notchecked', '14x fail', '6x fail', '1x notchecked', '3x fail', '5x fail', '1x fail', '49x fail', and '6x notchecked'. To the right of the tree is a table with columns 'Title', 'Severity', and 'Result'. The table lists specific rules such as 'Ensure dnf Removes Previous Package Versions' (low, pass), 'Ensure gpgcheck Enabled In Main dnf Configuration' (high, pass), 'Ensure gpgcheck Enabled for Local Packages' (high, fail), 'Ensure gpgcheck Enabled for All dnf Package Repositories' (high, pass), 'Ensure Red Hat GPG Key Installed' (high, pass), and 'Ensure Software Patches Installed ()' (medium, notchecked).

Figure 9.5: Rule overview group views

Clicking a rule title, such as **Ensure gpgcheck Enabled for All dnf Package Repositories**, opens a dialog where you can examine why a particular OpenSCAP security rule failed or passed.

The screenshot shows the 'Individual rule details' dialog for the rule 'Ensure gpgcheck Enabled for All dnf Package Repositories'. The dialog has the following fields:

- Rule ID:** xccdf_org.ssgproject.content_rule_ensure_gpgcheck_never_disabled
- Result:** pass
- Multi-check rule:** no
- OVAL Definition ID:** oval:ssg-ensure_gpgcheck_never_disabled:def:1
- Time:** 2023-10-23T10:29:47-05:00
- Severity:** high
- Identifiers and References:** CCE-83464-8
References: BP28(R15), 11, 2, 3, 9, 5.10.4.1, APO01.06, BAI03.05, BAI06.01, BAI10.01, BAI10.02, BAI10.03, BAI10.05, DSS06.02, 3.4.8, CCI-001749, 164.308(a)(1)(ii)(D), 164.312(b), 164.312(c)(1), 164.312(c)(2), 164.312(e)(2)(i), 4.3.4.3.2, 4.3.4.3.3, 4.3.4.4.4, SR 3.1, SR 3.3, SR 3.8, SR 7.6, A.11.2.4, A.12.1.2, A.12.2.1, A.12.5.1, A.12.6.2, A.14.1.2, A.14.1.3, A.14.2.2, A.14.2.3, A.14.2.4, CM-5(i), SI-7, SC-12, SC-12(3), CM-6(a), SA-12, SA-12(10), CM-11(a), CM-11(b), PR.DS-6, PR.DS-8, PR.IP-1, FPT.TUD_EXT.1, FPT.TUD_EXT.2, Req-6.2, SRG-OS-000366-GPOS-00153, SRG-OS-000370-VMM-001430, SRG-OS-000404-VMM-001650
- Description:** To ensure signature checking is not disabled for any repos, remove any lines from files in `/etc/yum.repos.d` of the form:
`gpgcheck=0`
- Rationale:** Verifying the authenticity of the software prior to installation validates the integrity of the patch or upgrade received from a vendor. This ensures the software has not been tampered with and that it has been provided by a trusted vendor. Self-signed certificates are disallowed by this requirement. Certificates used to verify the software must be from an approved Certificate Authority (CA).*

Figure 9.6: Individual rule details



References

oscap(8) man page

For more information, refer to the *Assessing Configuration Compliance with a Specific Baseline* section in the *Scanning the System for Configuration Compliance and Vulnerabilities* chapter in the *Security Hardening* guide at

https://access.redhat.com/documentation/es-es/red_hat_enterprise_linux/9/html-single/security_hardening/index#assessing-configuration-compliance-with-a-specific-baseline_configuration-compliance-scanning

For more information from the OpenSCAP project about the policies that are available in the SCAP Security Guide and how to select them, refer to *Choosing a Policy* at

<https://www.open-scap.org/security-policies/choosing-policy/>

► Guided Exercise

Scanning and Analyzing Compliance

Scan one of your servers for compliance with the OSPP profile for Red Hat Enterprise Linux 9, which is provided with the SCAP Security Guide's content.

Outcomes

- Scan a system with the DISA STIG for Red Hat Enterprise Linux 9 profile.
- Review and interpret the results.

Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

```
[student@workstation ~]$ lab start openscap-scanning
```

Instructions

- 1. On the `serverc` machine, retrieve the identifier of the DISA STIG for Red Hat Enterprise Linux 9 profile and scan the system for compliance with that profile.
- 1.1. Log in to the `serverc` machine as the `student` user. No password is required.

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

- 1.2. Use the `sudo -i` command to switch identity to the `root` user. Use `student` as the password.

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

- 1.3. Use the `oscap info` command to retrieve the identifier of the DISA STIG for Red Hat Enterprise Linux 9 profile.

```
[root@serverc ~]# oscap info /usr/share/xml/scap/ssg/content/ssg-rhel9-ds.xml  
...output omitted...  
Title: [DRAFT] Unclassified Information in Non-federal Information Systems and  
Organizations (NIST 800-171)  
Id: xccdf_org.ssgproject.content_profile_cui  
Title: Australian Cyber Security Centre (ACSC) Essential Eight  
Id: xccdf_org.ssgproject.content_profile_e8  
Title: Health Insurance Portability and Accountability Act (HIPAA)  
Id: xccdf_org.ssgproject.content_profile_hipaa  
Title: Australian Cyber Security Centre (ACSC) ISM Official
```

```
Id: xccdf_org.ssgproject.content_profile_ism_o
Title: Protection Profile for General Purpose Operating Systems
Id: xccdf_org.ssgproject.content_profile_ospp
Title: PCI-DSS v3.2.1 Control Baseline for Red Hat Enterprise Linux 9
Id: xccdf_org.ssgproject.content_profile_pci-dss
Title: [DRAFT] DISA STIG for Red Hat Enterprise Linux 9
Id: xccdf_org.ssgproject.content_profile_stig
Title: [DRAFT] DISA STIG with GUI for Red Hat Enterprise Linux 9
Id: xccdf_org.ssgproject.content_profile_stig_gui
...output omitted...
```

- 1.4. Scan the system for compliance with the DISA STIG for Red Hat Enterprise Linux 9 profile. Save the result in the /root/results.xml file.

```
[root@serverc ~]# oscap xccdf eval \
--profile xccdf_org.ssgproject.content_profile_stig \
--results /root/results.xml \
/usr/share/xml/scap/ssg/content/ssg-rhel9-ds.xml
--- Starting Evaluation ---

Title Install AIDE
Rule xccdf_org.ssgproject.content_rule_package_aide_installed
Ident CCE-90843-4
Result fail

Title Configure AIDE to Verify the Audit Tools
Rule xccdf_org.ssgproject.content_rule_aide_check_audit_tools
Ident CCE-87757-1
Result fail

Title Configure Periodic Execution of AIDE
Rule xccdf_org.ssgproject.content_rule_aide_periodic_cron_checking
Ident CCE-83437-4
Result fail

Title Configure Notification of Post-AIDE Scan Details
Rule xccdf_org.ssgproject.content_rule_aide_scan_notification
Ident CCE-90844-2
Result fail

...output omitted...
```

- ▶ 2. When the scan is complete, generate an HTML report for your scan of the serverc machine and copy it to the workstation machine.

- 2.1. Convert the /root/results.xml file to HTML. Write the HTML report to the /root/results.html file.

```
[root@serverc ~]# oscap xccdf generate report results.xml > results.html
```

- 2.2. Use the scp command to copy the results.html file to the workstation machine so that you can use the Firefox web browser to display the report. Use student as the password.

```
[root@serverc ~]# scp results.html student@workstation:  
The authenticity of host 'workstation (172.25.250.9)' can't be established.  
ED25519 key fingerprint is SHA256:4JZy3aivfx4NsDAE8rKOYCEp8A38DrCbXrp65yl6DK4.  
This key is not known by any other names  
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes  
Warning: Permanently added 'workstation' (ED25519) to the list of known hosts.  
student@workstation's password: student  
results.html                                              100% 7669KB 214.8MB/s   00:00
```

2.3. Return to the workstation machine.

```
[root@serverc ~]# logout  
[student@serverc ~]$ logout  
[student@workstation ~]$
```

► 3. On the workstation machine, use Firefox to review the scan report.

3.1. Use Firefox to display the `results.html` file.

```
[student@workstation ~]$ firefox results.html
```

3.2. Browse through the page and view the following:

- The number of passed and failed rules
- The severity of the failed rules
- The pass result of the `Ensure gpgcheck Enabled for All dnf Package Repositories` rule
- The fail result of the `Ensure gpgcheck Enabled for Local Packages` rule

Close Firefox when you are done exploring the scan results, but keep the `results.html` file so you that can refer to it later.

Finish

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

```
[student@workstation ~]$ lab finish openscap-scanning
```

Customizing OpenSCAP Policy

Objectives

- Create a tailoring file to adjust the policy's security checks so that they are relevant and correct for a specific system and its use case.

Customizing a SCAP Security Guide Profile

The SCAP Security Guide provides profiles for verifying system compliance against standards that are established by governmental or other organizations. For example, the *PCI-DSS* profile tests compliance against the rules mandated by the Payment Card Industry Security Standards Council.

However, in the real world, situations vary. Most profiles in the SCAP Security Guide are meant as a catalog, not a checklist, and satisfying every item might not be sensible or even possible in many operational scenarios.

Your organization, auditors, or other stakeholders might not require your systems to comply with every item that is specified in a particular profile. Instead, you might be required to comply with a specific subset of the profile. Some rules might not apply in your environment, and other rules that are not in the profile might be required. You might also need to increase or decrease the values that are used for certain checks; for example, the maximum password age or the minimum password length that are permitted.

With the SCAP Workbench utility, you can create custom profiles. As a starting point, select an existing profile that you adjust by selecting and clearing compliance rules. Then, save your new profile in an XML *tailoring file* that you can copy to the systems to scan.

Creating a Tailoring File

Run the `scap-workbench` command to start SCAP Workbench and to create your profile customization. You can use the SCAP Workbench tool on your local workstation and copy the resulting XML tailoring file to the systems that you want to scan.

From the main SCAP Workbench window, select the profile on which you want to base your customization and then click **Customize**.

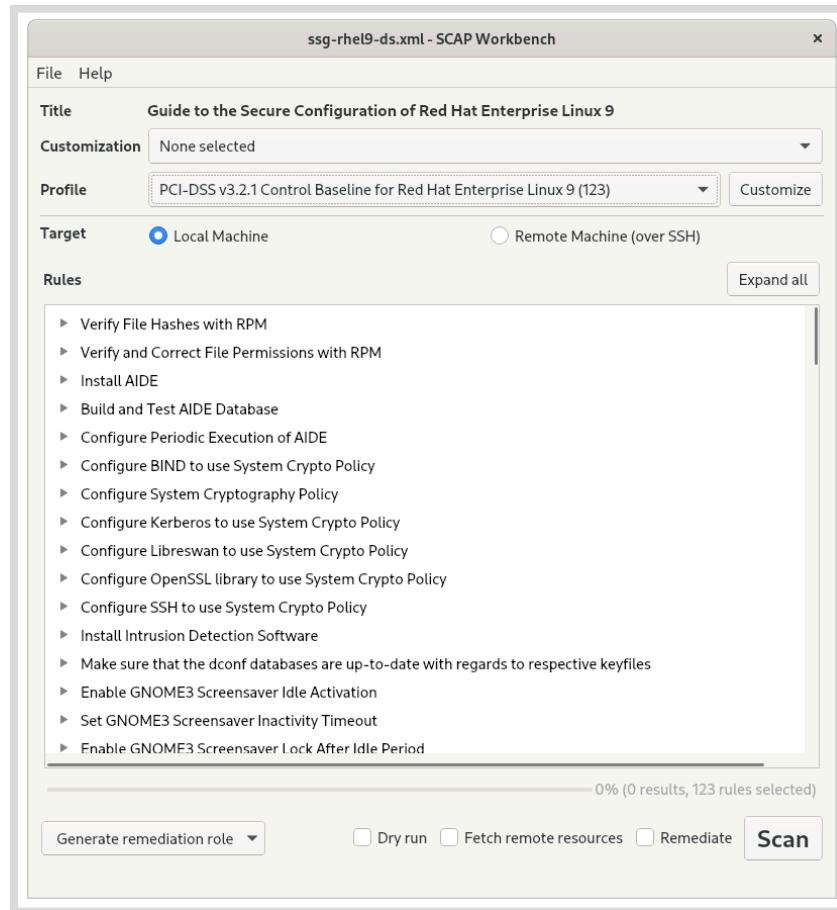


Figure 9.7: Main SCAP Workbench window

Next, give your new profile an identifier. You must use a particular format for the identifier. The **Customize Profile** dialog provides guidance for creating correctly formatted identifiers.

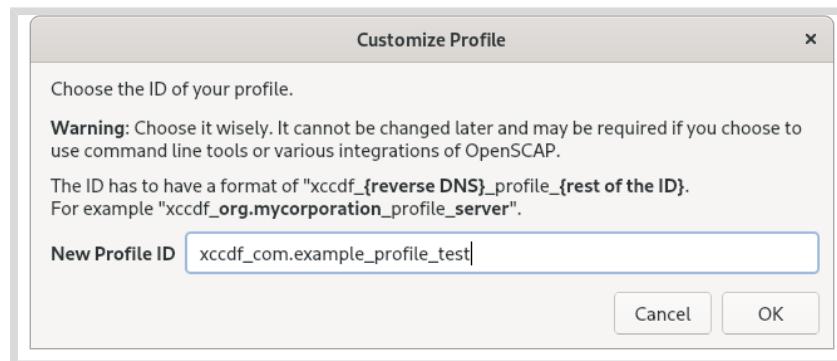


Figure 9.8: Creating a custom profile identifier

In the customization window, select the rules to use for your customized profile. The rules that are already selected come from the profile that you initially chose.

In the left pane, you can make the following adjustments:

- Select or clear rules
- Adjust parameters for some of these rules

Chapter 9 | Managing Compliance with OpenSCAP

The following image highlights the rule selection. Inspect the information that is provided in the right pane: **Selected Item Properties**, **Description**, **Security Identifiers**, and **Depends on Values**. These values represent information that is related to the specific rule that is selected.

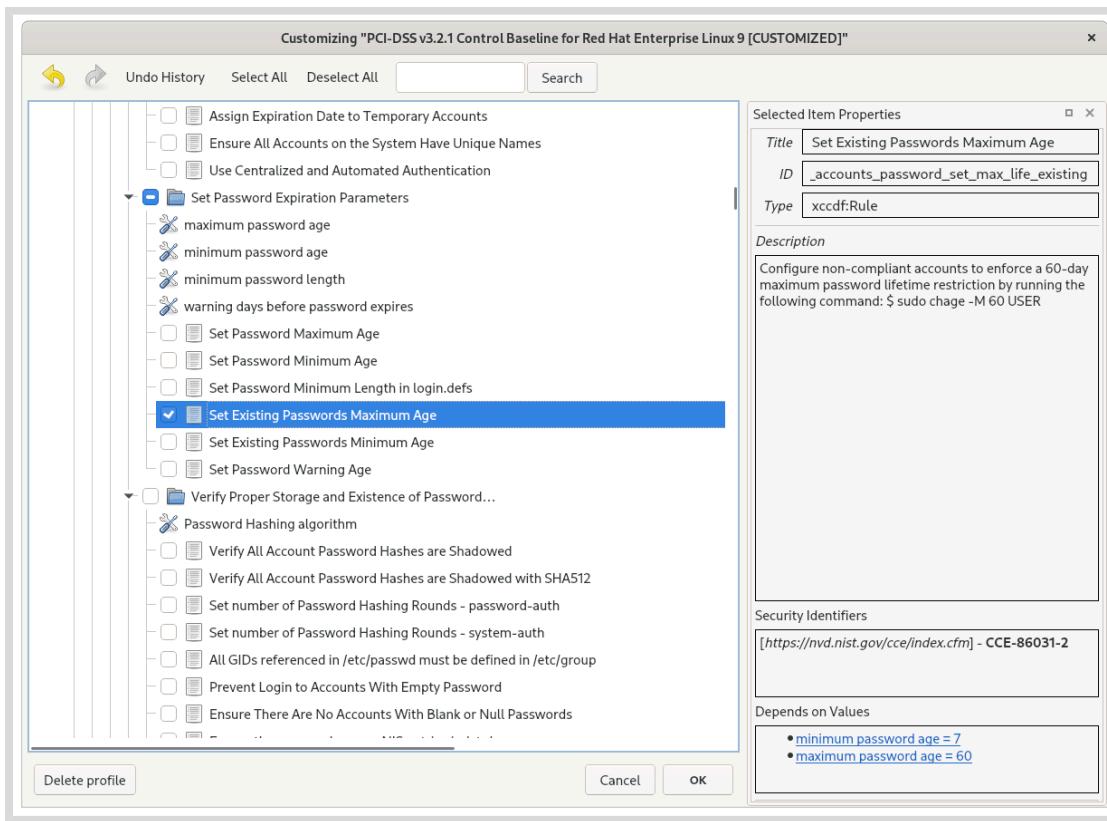


Figure 9.9: Policy customization rule selection

The following image highlights a parameter value that is associated with the selected rule. Inspect the information that is provided in the right pane. In the **Modify Value** section, you can change the value of the highlighted parameter.

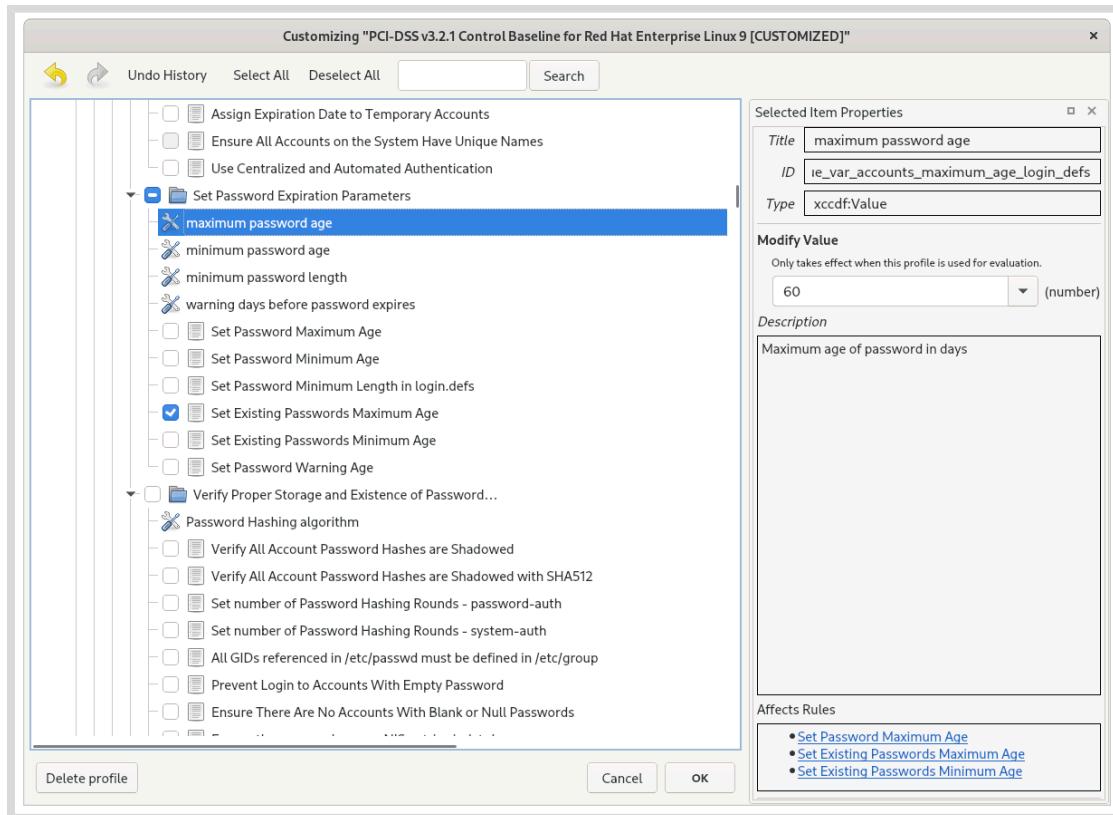


Figure 9.10: Policy customization parameter selection

In the previous image, the **Set Existing Passwords Maximum Age** rule is selected. You can customize this rule by modifying the **maximum password age** setting. In this example, the maximum age for a password is set to 60 days.

When done, click **OK** to return to the main window. Click **File > Save Customization Only** to save your customization file.

Scanning a System by Using a Custom Profile

Before you scan a system with your tailoring file, you must perform the following actions:

- Copy the XML tailoring file to the target system.
- Record the custom profile identifier that you defined when you created the tailoring file. You can use the `oscap info` command on the XML tailoring file to retrieve that identifier.
- Make sure that the `openscap-scanner` and `scap-security-guide` packages are installed on the target system.

To scan the system with your tailoring file, run the `oscap xccdf eval` command as usual, and use the correct data stream file. Use the `--tailoring-file` option to specify your tailoring file. The `--profile` option must specify the identifier of your custom profile.

```
[root@host ~]# oscap xccdf eval \
--profile custom_profile_ID \
--tailoring-file tailoring_file.xml \
--results result_file.xml \
/usr/share/xml/scap/ssg/content/ssg-rhel9-ds.xml
```



References

For more information, refer to the *Customizing a Security Profile with SCAP Workbench* section in the *Scanning the System for Configuration Compliance and Vulnerabilities* chapter in the *Security Hardening* guide at https://access.redhat.com/documentation/es-es/red_hat_enterprise_linux/9/html-single/security_hardening/index#customizing-a-security-profile-with-scap-workbench_scanning-the-system-with-a-customized-profile-using-scap-workbench

► Guided Exercise

Customizing OpenSCAP Policy

Select a subset of checks to perform from a SCAP policy by creating a tailoring file with SCAP Workbench. You will then test the tailored policy by scanning one of your servers and reviewing the results.

Outcomes

- Create a tailoring file that has most checks disabled by using the SCAP Workbench utility.
- Scan a system with the customized policy.
- Review the results.
- Optionally, compare the current results to the results from the previous exercise.

Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

```
[student@workstation ~]$ lab start openscap-customizing
```

Instructions

- 1. On the workstation machine, use the SCAP Workbench graphical utility to customize the DISA STIG for Red Hat Enterprise Linux 9 profile.
- 1.1. Start SCAP Workbench by running the `scap-workbench` command.

```
[student@workstation ~]$ scap-workbench
```

The SCAP Workbench utility detects that the SCAP Security Guide is already installed on the system and asks you to select the content to use.

In the **Select content to load** field, select `rhe19` and click **Load Content**.

- 1.2. Locate the **Profile** field and select the **DISA STIG for Red Hat Enterprise Linux 9** profile.
Click **Customize** to the right of that field.
- 1.3. In the **New Profile ID** field, enter `xccdf_com.example_profile_RH415-rhel9` and click **OK**.
The new window displays all the available rules. The rules that are included in the DISA STIG for Red Hat Enterprise Linux 9 profile are selected.
- 1.4. Click **Deselect All** and select the three following rules in the **Updating Software** section:
 - **Ensure gpgcheck Enabled In Main dnf Configuration**

- Ensure gpgcheck Enabled For All dnf Package repositories
- Ensure dnf Removes Previous Package Versions

In the Set Password Quality Requirements with pam_pwquality section, select the following:

- Select the Ensure PAM Enforces Password Requirements - Minimum Length rule to enable the test.
- Click the minlen rule. In the right pane, select 6 from the Modify Value list. This configuration sets the pam_pwquality_minlen option to six characters.

Click OK.

15. Save the customization in a tailoring file. Select File > Save Customization Only and enter RH415-tailoring.xml for the file name. Keep this file in your home directory so that you can refer to it later.

Close SCAP Workbench.

- 2. Scan the serverc machine for compliance with your customization of the DISA STIG profile.
- 2.1. As the student user, use the scp command to copy the RH415-tailoring.xml tailoring file to the serverc machine. No password is required.

```
[student@workstation ~]$ scp RH415-tailoring.xml student@serverc:  
RH415-tailoring.xml                                         100%   66KB  36.3MB/s  00:00
```

- 2.2. Log in to the serverc machine as the student user. No password is required.

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

- 2.3. Change to the root user. Use student as the password.

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

- 2.4. Use the oscap info command on the /home/student/RH415-tailoring.xml tailoring file to retrieve the associated profile.

```
[root@serverc ~]# oscap info /home/student/RH415-tailoring.xml  
Document type: XCCDF Tailoring  
Imported: 2023-10-24T12:56:41  
Benchmark Hint: /tmp/scap-workbench-ILIHwP/ssg-rhel9-ds.xml  
Profiles:  
Title: [DRAFT] DISA STIG for Red Hat Enterprise Linux 9 [CUSTOMIZED]  
Id: xccdf_com.example_profile_RH415-rhel9
```

- 2.5. Scan the system for compliance with your customization. Save the result in the /root/RH415-results.xml file. Keep this file in your home directory so that you can refer to it later.

```
[root@serverc ~]# oscap xccdf eval \
--profile xccdf_com.example_profile_RH415-rhel9 \
--tailoring-file /home/student/RH415-tailoring.xml \
--results /root/RH415-results.xml \
/usr/share/xml/scap/ssg/content/ssg-rhel9-ds.xml
WARNING: Datastream component 'scap_org.open-scap_cref_security-data-oval-com.redhat.rhsa-RHEL9.xml.bz2' points out to the remote 'https://access.redhat.com/security/data/oval/com.redhat.rhsa-RHEL9.xml.bz2'. Use '--fetch-remote-resources' option to download it.
WARNING: Skipping 'https://access.redhat.com/security/data/oval/com.redhat.rhsa-RHEL9.xml.bz2' file which is referenced from datastream
WARNING: Skipping ./security-data-oval-com.redhat.rhsa-RHEL9.xml.bz2 file which is referenced from XCCDF content
--- Starting Evaluation ---

Title Ensure dnf Removes Previous Package Versions
Rule xccdf_org.ssgproject.content_rule_clean_components_post_updating
Ident CCE-83458-0
Result pass

Title Ensure gpgcheck Enabled In Main dnf Configuration
Rule xccdf_org.ssgproject.content_rule_ensure_gpgcheck_globally_activated
Ident CCE-83457-2
Result pass

Title Ensure gpgcheck Enabled for All dnf Package Repositories
Rule xccdf_org.ssgproject.content_rule_ensure_gpgcheck_never_disabled
Ident CCE-83464-8
Result pass

Title Ensure PAM Enforces Password Requirements - Minimum Length
Rule xccdf_org.ssgproject.content_rule_accounts_password_pam_minlen
Ident CCE-83579-3
Result fail
```

- 2.6. When the scan is complete, convert the /root/RH415-results.xml file to HTML. Save the HTML report in the /root/RH415-results.html file.

```
[root@serverc ~]# oscap xccdf generate report \
RH415-results.xml > RH415-results.html
[root@serverc ~]#
```

- 2.7. Use the scp command to copy the RH415-results.html file to the workstation machine so that you can use Mozilla Firefox to display it. Use student as the password.

```
[root@serverc ~]# scp RH415-results.html student@workstation:
student@workstation's password: student
RH415-results.html                                100% 274KB 99.5MB/s 00:00
```

2.8. Return to the **workstation** machine.

```
[root@serverc ~]# logout  
[student@serverc ~]$ logout  
[student@workstation ~]$
```

2.9. Use Firefox to display the **RH415-results.html** file.

```
[student@workstation ~]$ firefox RH415-results.html
```

2.10. The page only displays the rules from your tailoring file. Optionally, compare the results of this exercise to the results of the previous exercise. Close Firefox when you are done exploring the scan results.

The screenshot shows the 'Rule Overview' section of the OpenSCAP interface. On the left, there are checkboxes for filtering results: 'pass', 'fail', 'fixed', 'error', 'informational', 'notchecked', and 'notapplicable'. To the right is a search bar with placeholder 'Search through XCCDF rules' and a 'Search' button. Below the search bar is a dropdown menu 'Group rules by:' set to 'Default'. The main area is a table with columns 'Title', 'Severity', and 'Result'. The first row shows 'Guide to the Secure Configuration of Red Hat Enterprise Linux 9' with 1x fail. This row has a red background. Below it, the table continues with 'System Settings' (1x fail), 'Installing and Maintaining Software', 'Account and Access Control' (1x fail), 'Protect Accounts by Configuring PAM' (1x fail), 'Set Password Quality Requirements' (1x fail), and 'Set Password Quality Requirements with pam_pwquality' (1x fail). The last row is 'Ensure PAM Enforces Password Requirements - Minimum Length' with 'medium' severity and a red 'fail' result. The entire table has a light gray background.

Figure 9.11: Tailored rules scan results

Finish

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

```
[student@workstation ~]$ lab finish openscap-customizing
```

Remediating OpenSCAP Issues with Ansible

Objectives

- Run Ansible Playbooks, which are provided with the SCAP Security Guide's content, to remediate compliance checks that failed an OpenSCAP scan.

Generating a Remediation Ansible Playbook

The data streams in the SCAP Security Guide include remediation scripts to fix the noncompliant checks. These scripts might take the form of shell scripts, Ansible snippets, Puppet snippets, or even Kickstart commands. Some rules provide multiple remediation scripts in different formats. For example, the rule that verifies whether AIDE is installed provides a remediation shell script, but also an Ansible snippet, a Puppet snippet, and a Kickstart command. Some rules only provide one remediation method, and others do not provide any remediation scripts at all.



Note

Remediation coverage is generally better for shell scripts and Ansible snippets.

By using the `oscap` command, you can generate an Ansible Playbook from a SCAP Security Guide profile, or from a scan result XML file.

Creating an Ansible Playbook from a Profile

From a SCAP Security Guide profile, the `oscap xccdf generate fix` command generates an Ansible Playbook that includes all the tasks for remediation for which Ansible snippets are available.

```
[root@host ~]# oscap xccdf generate fix \
    --profile xccdf_org.ssgproject.content_profile_pci-dss \
    --fix-type ansible \
    /usr/share/xml/scap/ssg/content/ssg-rhel9-ds.xml > pci-dss.yml
```

You can include this Ansible Playbook in a more global Ansible infrastructure; for example, as a postinstallation step. Because Ansible Playbooks are idempotent, you can regularly run the playbook on your systems to keep them compliant; there is no impact on the items that are already compliant.

Some SCAP rules do not have Ansible remediation snippets. For those rules, you might have to develop your own playbook tasks for remediation.

Remember that many profiles are meant as catalogs, not checklists, and not every remediation item makes sense in your environment. If you generate a playbook from a profile, then consider using one that has been tailored for your environment.

Creating an Ansible Playbook from an XML Result File

After scanning a system for compliance, the oscap xccdf generate fix command creates an Ansible Playbook from the XML result file to fix the noncompliant checks.

In the following example, the first oscap command scans the local system, and the second oscap command generates the Ansible Playbook for remediation.

```
[root@host ~]# oscap xccdf eval \
--profile xccdf_org.ssgproject.content_profile_pci-dss \
--results /root/results.xml \
/usr/share/xml/scap/ssg/content/ssg-rhel9-ds.xml
...output omitted...

[root@host ~]# oscap xccdf generate fix \
--profile xccdf_org.ssgproject.content_profile_pci-dss \
--fix-type ansible \
--result-id "" \
/root/results.xml > remediation-playbook.yml
```

The generated Ansible Playbook is only relevant for the scanned system because it includes only the tasks that remediate the failed checks.

Adjusting Variables in the Remediation Ansible Playbook

You can edit the Ansible Playbook that the oscap xccdf generate fix command creates. In particular, the playbook uses variables for some parameters that are associated with rules in the SCAP Security Guide profiles.

The following example shows some variables that are defined in a remediation Ansible Playbook:

```
...output omitted...
- hosts: all
  vars:
    var_accounts_maximum_age_login_defs: 90
    var_account_disable_post_pw_expiration: 90
    var_password_pam_dcredit: -1
    var_password_pam_minlen: 7
    var_password_pam_ucredit: -1
    var_password_pam_lcredit: -1
    var_auditd_space_left_action: email
    var_auditd_admin_space_left_action: single
    sshd_idle_timeout_value: 900
...output omitted...
```

Running a Remediation Ansible Playbook

The oscap xccdf generate fix command creates Ansible Playbooks with the hosts parameter set to all. You can adapt this parameter to specify the targeted group of systems to remediate, or you can create a custom inventory file that lists all these systems.

To specify a custom inventory file, use the `-i inventory_file` option with the `ansible-playbook` command.

```
[root@host ~]# ansible-playbook -i ./myinventory pci-dss.yml
```

**Warning**

The quality of the remediation snippets can vary depending on the content. Review and test the playbook carefully before using it on production systems or at scale.

Filtering Tasks

The remediation Ansible Playbooks can be quite large. Some tasks inside these playbooks might be disruptive, such as performing system updates. Other tasks might remediate low severity issues. Depending on your environment, you might want to skip some playbook tasks.

The tasks in a remediation Ansible Playbook have *tags*. Tags are an Ansible feature that enable you to run only the tasks that have a specific tag.

The SCAP Security Guide XCCDF files specify particular tags for the Ansible snippets. Rather than running the whole playbook, you can specify one or more tags to filter the tasks to play.

The following playbook extract shows two tasks and their associated tags:

```
- name: "Security patches are up to date"
  package:
    name: "*"
    state: "latest"
  tags:
    - security_patches_up_to_date
    - high_severity
    - patch_strategy
    - low_complexity
    - high_disruption
    - CCE-26895-3
    - NIST-800-53-ST-2
    - NIST-800-53-SI-2(c)
    - NIST-800-53-MA-1(b)
    - PCI-DSS-Req-6.2
    - CJIS-5.10.4.1
    - DISA-STIG-RHEL-07-020260

- name: disable prelinking
  lineinfile:
    path: /etc/sysconfig/prelink
    regexp: '^PRELINKING='
    line: 'PRELINKING=no'
  tags:
    - disable_prelink
    - low_severity
    - restrict_strategy
    - low_complexity
    - low_disruption
    - CCE-27078-5
    - NIST-800-53-CM-6(d)
```

- NIST-800-53-CM-6(3)
- NIST-800-53-SC-28
- NIST-800-53-SI-7
- NIST-800-171-3.13.11
- PCI-DSS-Req-11.5
- CJIS-5.10.1.3

The following command runs only the tasks with the `high_severity` tag in the `pci-dss.yml` Ansible Playbook:

```
[root@host ~]# ansible-playbook --tags=high_severity pci-dss.yml
```

You can also specify multiple tags by using a comma-separated list. See the Ansible documentation for more information.

Applying Profiles During Installation

You can call the OpenSCAP installer add-on from a Kickstart file to evaluate system compliance and to automatically remediate deviations during installation. Because the add-on also performs the remediation, your new systems are compliant immediately after installation.

For example, by adding the following lines in your Kickstart file, new systems become compliant with the SCAP Security Guide PCI-DSS profile during installation.

```
%addon org_fedora_oscap
content-type = scap-security-guide
profile = pci-dss
%end
```



References

Ansible Tags

https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_tags.html

For more information, refer to the *Scanning the System for Configuration Compliance and Vulnerabilities* chapter in the *Security Hardening* guide at
https://access.redhat.com/documentation/es-es/red_hat_enterprise_linux/9/html-single/security_hardening/index#scanning-the-system-for-configuration-compliance-and-vulnerabilities_security-hardening

For more information, refer to the *Performing an Automated Installation Using Kickstart* chapter in the *Performing an Advanced RHEL 9 Installation* guide at
https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html-single/performing_an_advanced_rhel_9_installation/index#performing_an_automated_installation_using_kickstart

► Guided Exercise

Remediating OpenSCAP Issues with Ansible

Use an Ansible Playbook that is provided with the system's SCAP content to remediate a failed compliance check.

Outcomes

- Use the output from the customized scan from the previous exercise and an Ansible Playbook provided by the SCAP content to resolve a compliance issue.
- Rescan the system to confirm that the issue is reported as resolved.

Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

```
[student@workstation ~]$ lab start openscap-remediation
```

Instructions

- 1. On the workstation machine, generate a remediation Ansible Playbook. The playbook resolves the compliance issues that you detected in a previous exercise, in a scan that you performed with a customized policy.

To create the remediation Ansible Playbook, the `oscap` command requires:

- The `RH415-tailoring.xml` tailoring file
 - The `RH415-results.xml` results file
- 1.1. Use the `oscap info` command on the `RH415-tailoring.xml` tailoring file to retrieve the associated profile identifier.

```
[student@workstation ~]$ oscap info RH415-tailoring.xml
Document type: XCCDF Tailoring
Imported: 2023-10-24T16:58:41
Benchmark Hint: /tmp/scap-workbench-ILIHW/sgg-rhel9-ds.xml
Profiles:
Title: [DRAFT] DISA STIG for Red Hat Enterprise Linux 9 [CUSTOMIZED]
Id: xccdf_com.example_profile_RH415-rhel9
```

- 1.2. Use the `oscap xccdf generate fix` command to generate the Ansible Playbook. Save the playbook to the `/home/student/playbook.yml` file.

```
[student@workstation ~]$ oscap xccdf generate fix \
--profile xccdf_com.example_profile_RH415-rhel9 \
--tailoring-file RH415-tailoring.xml \
--fix-type ansible \
--result-id "" \
RH415-results.xml > /home/student/playbook.yml
[student@workstation ~]$
```

The command uses an empty string for the `--result-id` option. This way, you do not need to specify the full result identifier.

Notice that you do not need to run this command on the scanned system, provided that the tailoring file, the result file, the SCAP Security Guide, and the `oscap` command are available on your local system.

- 1.3. Review the `/home/student/playbook.yml` Ansible Playbook.

```
[student@workstation ~]$ cat playbook.yml
...output omitted...
```

The tasks fix the issue that is reported in the scan from the previous exercise.

- 2. On the `workstation` machine, run the Ansible Playbook to resolve the compliance issues on the `serverc` machine.

- 2.1. Create an inventory file that contains the `serverc` machine.

```
[student@workstation ~]$ echo serverc > inventory
```

- 2.2. Edit the `/home/student/playbook.yml` file and add the `become: true` line.

```
...output omitted...
- hosts: all
  become: true
  vars:
...output omitted...
```

- 2.3. Use the `ansible-playbook` command to run the playbook. Include the `-K` flag to configure the command to prompt for the become password. Use `student` as the become password.

```
[student@workstation ~]$ ansible-playbook -K -i inventory playbook.yml
BECOME password: student

PLAY [all] ****
TASK [Gathering Facts] ****
ok: [serverc]

TASK [Gather the package facts] ****
ok: [serverc]
```

```
TASK [Ensure PAM Enforces Password Requirements - Minimum Length - Ensure PAM
variable minlen is set accordingly] ****
changed: [serverc]

PLAY RECAP ****
serverc : ok=3    changed=1    unreachable=0    failed=0
      skipped=0   rescued=0    ignored=0
```

- 3. Scan the `serverc` machine again to confirm that the system is now compliant.

- 3.1. Log in to the `serverc` machine as the `student` user. No password is required.

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

- 3.2. Change to the `root` user. Use `student` as the password.

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

- 3.3. Use the `oscap info` command on the `/root/RH415-tailoring.xml` tailoring file to retrieve the associated profile.

```
[root@serverc ~]# oscap info RH415-tailoring.xml
Document type: XCCDF Tailoring
Imported: 2023-10-24T17:56:55
Benchmark Hint: /tmp/scap-workbench-ILIHW_P/ssg-rhel9-ds.xml
Profiles:
  Title: [DRAFT] DISA STIG for Red Hat Enterprise Linux 9 [CUSTOMIZED]
  Id: xccdf_com.example_profile_RH415-rhel9
```

- 3.4. Use the `oscap xccdf eval` command to scan the system for compliance.

```
[root@serverc ~]# oscap xccdf eval \
  --profile xccdf_com.example_profile_RH415-rhel9 \
  --tailoring-file RH415-tailoring.xml \
  /usr/share/xml/scap/ssg/content/ssg-rhel9-ds.xml
...output omitted...
--- Starting Evaluation ---

Title  Ensure dnf Removes Previous Package Versions
Rule   xccdf_org.ssgproject.content_rule_clean_components_post_updating
Ident  CCE-83458-0
Result pass

Title  Ensure gpgcheck Enabled In Main dnf Configuration
Rule   xccdf_org.ssgproject.content_rule_ensure_gpgcheck_globally_activated
Ident  CCE-83457-2
Result pass

Title  Ensure gpgcheck Enabled for All dnf Package Repositories
```

```
Rule    xccdf_org.ssgproject.content_rule_ensure_gpgcheck_never_disabled
Ident   CCE-83464-8
Result  pass

Title   Ensure PAM Enforces Password Requirements - Minimum Length
Rule    xccdf_org.ssgproject.content_rule_accounts_password_pam_minlen
Ident   CCE-83579-3
Result  pass
```

All the tests pass. The system now complies with your custom policy.

3.5. Return to the **workstation** machine when done.

```
[root@serverc ~]# logout
[student@serverc ~]$ logout
[student@workstation ~]$
```

Finish

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

```
[student@workstation ~]$ lab finish openscap-remediation
```

▶ Lab

Managing Compliance with OpenSCAP

Confirm that OpenSCAP tools and SCAP Security Guide content are installed on one of your servers, use SCAP Workbench to create a tailoring file, use OpenSCAP to scan the server with that tailored policy, and use Ansible to remediate a compliance check that failed.

Outcomes

- Install OpenSCAP tools and the SCAP Security Guide.
- Create a tailoring file by using SCAP Workbench.
- Scan the system using the customized policy.
- Generate and use an Ansible Playbook to remediate failed compliance checks.

Before You Begin

As the student user on the **workstation** machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

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

Instructions

1. On the **workstation** machine, install the SCAP Workbench and SCAP Security Guide utilities.
2. On the **workstation** machine, customize the DISA STIG for Red Hat Enterprise Linux 9 profile. Set the new profile identifier to `xccdf_com.example_profile_lab-rhel9`, disable all the rules, and then enable only the following rules:
 - Install AIDE
 - Build and Test AIDE Database
 - Configure AIDE to Verify the Audit Tools
 - Configure Periodic Execution of AIDEStore the resulting tailoring file on the **workstation** machine in the `/home/student/lab-tailoring.xml` file.
3. Scan the **serverd** machine for compliance with your customization of the DISA STIG for Red Hat Enterprise Linux 9 profile. Save the result on the **workstation** machine in the `/home/student/lab-results.xml` file. Generate the HTML report of the scan and store it in the `/home/student/lab-results.html` file on the **workstation** machine.
4. On the **workstation** machine, generate an Ansible Playbook to resolve the compliance issues that were detected in the previous step. Create an inventory file that contains the **serverd** host. Save the Ansible Playbook as `/home/student/fix.yml` and run it to resolve the compliance issues on the **serverd** host.

5. Scan the **serverd** machine again for compliance with your customization of the DISA STIG for Red Hat Enterprise Linux 9 profile. Save the result on the **workstation** machine in the **/home/student/lab-results-fix.xml** file.

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 openscap-review
```

Finish

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

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

► Solution

Managing Compliance with OpenSCAP

Confirm that OpenSCAP tools and SCAP Security Guide content are installed on one of your servers, use SCAP Workbench to create a tailoring file, use OpenSCAP to scan the server with that tailored policy, and use Ansible to remediate a compliance check that failed.

Outcomes

- Install OpenSCAP tools and the SCAP Security Guide.
- Create a tailoring file by using SCAP Workbench.
- Scan the system using the customized policy.
- Generate and use an Ansible Playbook to remediate failed compliance checks.

Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

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

Instructions

1. On the workstation machine, install the SCAP Workbench and SCAP Security Guide utilities.

```
[student@workstation ~]$ sudo dnf install -y scap-workbench scap-security-guide  
[sudo] password for student: student  
...output omitted...  
Complete!
```

2. On the workstation machine, customize the DISA STIG for Red Hat Enterprise Linux 9 profile. Set the new profile identifier to `xccdf_com.example_profile_lab-rhel9`, disable all the rules, and then enable only the following rules:

- Install AIDE
- Build and Test AIDE Database
- Configure AIDE to Verify the Audit Tools
- Configure Periodic Execution of AIDE

Store the resulting tailoring file on the workstation machine in the `/home/student/lab-tailoring.xml` file.

- 2.1. On the workstation machine, start SCAP Workbench by running the `scap-workbench` command.

```
[student@workstation ~]$ scap-workbench
```

SCAP Workbench detects that the SCAP Security Guide is already installed on the system and asks you to select the content to use.

In the **Select content to load** field, select RHEL9 and click **Load Content**.

- 2.2. Locate the **Profile** field and select the DISA STIG for Red Hat Enterprise Linux 9 profile.

Click **Customize** to the right of that field.

- 2.3. In the **New Profile ID** field, enter xccdf_com.example_profile_lab-rhel9 and click **OK**.

The new window displays all the available rules.

- 2.4. Click **Deselect All** and select the following rules in the **System and Software Integrity** section:

- Install AIDE
- Build and Test AIDE Database
- Configure AIDE to Verify the Audit Tools
- Configure Periodic Execution of AIDE

Click **OK**.

- 2.5. Save the customization in a tailoring file. Select **File > Save Customization Only** and enter **lab-tailoring.xml** for the file name in the /home/student/ directory.

Close SCAP Workbench.

3. Scan the **serverd** machine for compliance with your customization of the DISA STIG for Red Hat Enterprise Linux 9 profile. Save the result on the **workstation** machine in the /home/student/lab-results.xml file. Generate the HTML report of the scan and store it in the /home/student/lab-results.html file on the **workstation** machine.

- 3.1. Copy the **lab-tailoring.xml** tailoring file to the **serverd** machine. You need this file to scan the system.

```
[student@workstation ~]$ scp lab-tailoring.xml student@serverd:  
...output omitted...
```

- 3.2. Log in to the **serverd** machine as the **student** user. No password is required.

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

- 3.3. Change to the **root** user. Use **student** as the password.

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

- 3.4. Install the **scap-security-guide** package.

```
[root@serverd ~]# dnf install -y scap-security-guide  
...output omitted...  
Complete!
```

- 3.5. Scan the system for compliance with your customization. Save the result in the /root/lab-results.xml file.

```
[root@serverd ~]# oscap xccdf eval \
--profile xccdf_com.example_profile_lab-rhel9 \
--tailoring-file /home/student/lab-tailoring.xml \
--results /root/lab-results.xml \
/usr/share/xml/scap/ssg/content/ssg-rhel9-ds.xml
...output omitted...
```

- 3.6. When the scan is complete, convert the /root/lab-results.xml file to HTML. Save the HTML report as /root/lab-results.html.

```
[root@serverd ~]# oscap xccdf generate report \
lab-results.xml > lab-results.html
[root@serverd ~]#
```

- 3.7. Use the scp command to copy the two files to the workstation machine. Use student as the password.

```
[root@serverd ~]# scp lab-results.* student@workstation:
student@workstation's password: student
...output omitted...
```

- 3.8. Return to the workstation machine.

```
[root@serverd ~]# logout
[student@serverd ~]$ logout
[student@workstation ~]$
```

4. On the workstation machine, generate an Ansible Playbook to resolve the compliance issues that were detected in the previous step. Create an inventory file that contains the serverd host. Save the Ansible Playbook as /home/student/fix.yml and run it to resolve the compliance issues on the serverd host.

- 4.1. Use the oscap xccdf generate fix command to generate the Ansible Playbook. Save the playbook as /home/student/fix.yml.

```
[student@workstation ~]$ oscap xccdf generate fix \
--profile xccdf_com.example_profile_lab-rhel9 \
--tailoring-file lab-tailoring.xml \
--fix-type ansible \
--result-id "" \
lab-results.xml > fix.yml
[student@workstation ~]$
```

- 4.2. Create an inventory file that contains the serverd host.

```
[student@workstation ~]$ echo serverd > inventory
```

- 4.3. Edit the fix.yml file to set the become: true option.

```
...output omitted...
- hosts: all
  become: true
  vars:
  tasks:
  ...output omitted.
```

- 4.4. Use the `ansible-playbook` command to run the playbook. The AIDE database build might take several minutes to complete. Use the `-K` flag and the `student` become password.

```
[student@workstation ~]$ ansible-playbook -K -i inventory fix.yml
BECOME password: student
...output omitted...
PLAY RECAP
*****
serverd : ok=14    changed=5    unreachable=0    failed=0
skipped=1   rescued=0   ignored=0
```

5. Scan the `serverd` machine again for compliance with your customization of the DISA STIG for Red Hat Enterprise Linux 9 profile. Save the result on the `workstation` machine in the `/home/student/lab-results-fix.xml` file.

- 5.1. Log in to the `serverd` machine as the `student` user. No password is required.

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

- 5.2. Change to the `root` user. Use `student` as the password.

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

- 5.3. Scan the system for compliance with your customization of the DISA STIG for Red Hat Enterprise Linux 9 profile. Save the result in the `/root/lab-results-fix.xml` file.

```
[root@serverd ~]# oscap xccdf eval \
  --profile xccdf_com.example_profile_lab-rhel9 \
  --tailoring-file /home/student/lab-tailoring.xml \
  --results /root/lab-results-fix.xml \
  /usr/share/xml/scap/ssg/content/ssg-rhel9-ds.xml
...output omitted...
--- Starting Evaluation ---

Title  Install AIDE
Rule   xccdf_org.ssgproject.content_rule_package_aide_installed
Ident  CCE-90843-4
Result pass

Title  Build and Test AIDE Database
```

Chapter 9 | Managing Compliance with OpenSCAP

```
Rule    xccdf_org.ssgproject.content_rule_aide_build_database
Ident   CCE-83438-2
Result  pass

Title   Configure AIDE to Verify the Audit Tools
Rule    xccdf_org.ssgproject.content_rule_aide_check_audit_tools
Ident   CCE-87757-1
Result  pass

Title   Configure Periodic Execution of AIDE
Rule    xccdf_org.ssgproject.content_rule_aide_periodic_cron_checking
Ident   CCE-83437-4
Result  pass
```

- 5.4. Use the `scp` command to copy the `/root/lab-results-fix.xml` file to the `workstation` machine. Use `student` as the password.

```
[root@serverd ~]# scp lab-results-fix.xml student@workstation:
student@workstation's password: student
lab-results-fix.xml                                         100%    15MB 219.0MB/s
00:00
```

- 5.5. Return to the `workstation` machine as the `student` user.

```
[root@serverd ~]# logout
[student@serverd ~]$ logout
[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 openscap-review
```

Finish

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

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

Summary

- The `openscap-scanner` and `scap-security-guide` packages must be installed on the system that you intend to scan for compliance.
- You can use the SCAP Workbench utility to explore and customize the policies that are provided by the SCAP Security Guide.
- A compliance policy provided by the SCAP Security Guide is usually customized based on the needs of your organization, auditors, and other stakeholders, so that it is relevant and correct for your systems, use cases, and other requirements.
- The `oscap xccdf eval` command scans systems for compliance by using a data stream file, a profile, and (optionally) a tailoring file that contains local customizations.
- The `oscap generate fix` command can generate an Ansible Playbook from a profile or a scan result XML file, which you can use to apply remediations.

Chapter 10

Analyzing and Remediating Issues with Red Hat Insights

Goal

Identify, detect, and correct common issues and security vulnerabilities with Red Hat Enterprise Linux systems by using Red Hat Insights.

Sections

- Red Hat Insights and Security (and Guided Exercise)
- Creating and Reviewing Red Hat Insights Reports (and Guided Exercise)
- Running OpenSCAP Reports from Red Hat Insights (and Guided Exercise)
- Integrating Red Hat Insights and Automation Controller (and Guided Exercise)

Red Hat Insights and Security

Objectives

- Review what Red Hat Insights is, how it is relevant to security, and how to register RHEL systems to use it.

Introducing Red Hat Insights

Red Hat Insights is a predictive analytics tool to help you identify and remediate threats to security, performance, availability, and stability on systems that run Red Hat products in your infrastructure. Insights is delivered as a Software-as-a-Service (SaaS) product, so you can deploy and scale Insights quickly with no additional infrastructure requirements. In addition, you can immediately take advantage of the latest recommendations and updates from Red Hat that are specific to your deployed systems.

Red Hat regularly updates the knowledge base that Insights uses, based on common support risks, security vulnerabilities, insecure configurations, and other issues that are identified by Red Hat. Red Hat validates and verifies the actions to mitigate or remediate these issues. These updates allow you to proactively identify, prioritize, and resolve issues before they become a larger problem.

Insights tailors recommendations for each system that is registered to the service. You can install an agent on client systems that collects metadata about the runtime configuration of your systems. This data is a subset of what you would provide to Red Hat Support by using the `sosreport` utility. You can further limit or obfuscate the data that your clients send, but this obfuscation might prevent certain analytics from operating, depending on what you limit.

You can begin using Insights immediately after the initial steps to register the system and to synchronize its metadata. Depending on how your systems are registered, you can access this interface through the Red Hat Customer Portal or through your Satellite Server. Insights can recommend next actions that are tailored for each of your systems, and even automate tasks with Ansible Playbooks.

OpenSCAP and Insights

OpenSCAP scanning and Insights are complementary tools.

Instead of reactively resolving issues that you find in OpenSCAP scans, you can use Insights to proactively address emerging security threats, misconfigurations, or other risks that are identified by Red Hat. When security researchers identify new threats, such as software configuration issues or even hardware microarchitecture issues like the Spectre and Meltdown vulnerabilities, updates to Insights can help you to quickly detect issues and mitigate or remediate them. Insights recommendations provide materials such as Ansible Playbooks and human-readable recommendations so that you can implement mitigation and remediation. In addition, Insights provides information about other issues with your systems that might impact your system's performance, availability, or stability. Insights also provides estimates of the risk that is presented by those issues.

Details of the Insights Architecture

You can register a client system to Insights through the Customer Portal Subscription Management service, or through a Red Hat Satellite Server that is connected to Insights. When you register a client, the client provides Insights with metadata about the runtime configuration of the system. Client systems send this metadata to Insights by using TLS encryption. The client anonymizes the data and sends it to Insights for analysis. The Customer Portal or Satellite Server web UI displays the recommendations that the Insights rule engine provides.

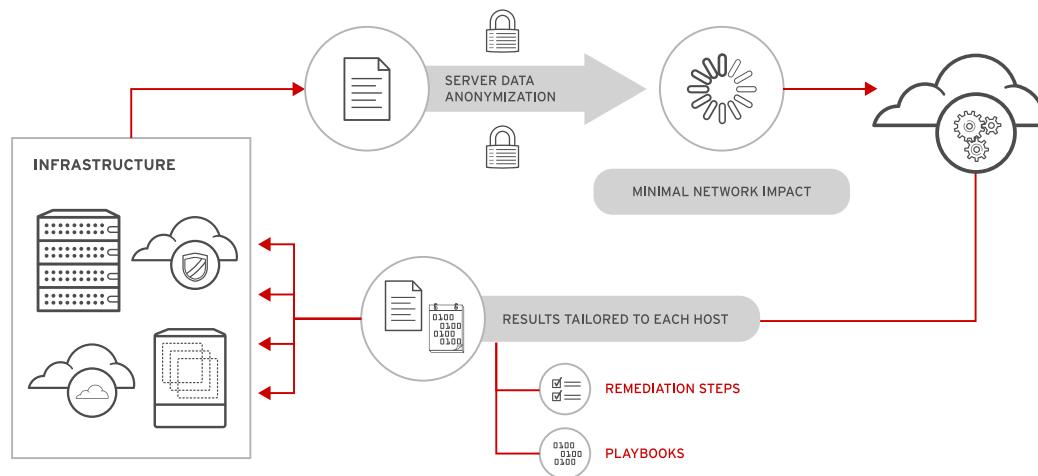


Figure 10.1: Insights architecture

Installing Insights Clients

To configure Insights for Red Hat Enterprise Linux servers, install the `insights-client` package on the system. Red Hat Enterprise Linux 8 and later versions include the client agent preinstalled.

If your system is registered for software entitlements through the Customer Portal Subscription Management service, then you can activate Insights with one command. Use the `insights-client --register` command to register the system.

```
[root@host ~]# insights-client --register
```

In this configuration, your system's Insights reports are accessible by your account at the <https://console.redhat.com/insights/> portal.



Note

To register your system through your Red Hat Satellite Server, you must configure Satellite Server to allow the Insights service, and you must register your client for Subscription Management service through Satellite. This process is discussed in more detail in a later chapter.

The Insights client periodically updates the metadata that is provided to Insights. Use the `insights-client` command to upload the client's metadata at any time.

```
[root@host ~]# insights-client
Starting to collect Insights data for host.lab.example.com
Uploading Insights data.
Successfully uploaded report from 773b351b-dfb1-4393-afa8-915cc2875e06 to
account XXXXX.
```

Controlling Data Sent to Insights

You can configure the Insights client to restrict the data that it sends to Insights. You can exclude specific configuration files, commands, patterns, and keywords. To enable data restriction, first configure the Insights client with an exclusion file that describes the restrictions. Edit the `/etc/insights-client/insights-client.conf` file to include a `remove_file` parameter that specifies the location of the exclusion file, typically the `/etc/insights-client/remove.conf` file.

```
remove_file=/etc/insights-client/remove.conf
```

The Insights client can also filter metadata before uploading it. The `/etc/insights-client/insights-client.conf` file contains two obfuscation options:

- To obfuscate IP addresses and keywords, set the `obfuscate` parameter to the `True` value.
- To obfuscate hostnames, set the `obfuscate_hostname` parameter to the `True` value.

You can provide a comma-separated list of files, commands, patterns, and keywords to exclude in the `remove_file` parameter.

```
[remove]
files=/etc/passwd,/etc/hosts
commands=/bin/dmesg
patterns=password,username
keywords=password$ecret
```

You can also review the data that the client uploads to Insights. Use the `insights-client --no-upload` command to collect the data, but prevent it from being uploaded.

```
[root@host ~]# insights-client --no-upload
Starting to collect Insights data
See Insights data in /var/tmp/oLUbKq/insights-demo-20180810110933.tar.gz
```

The client archives and stores the collected data. To inspect the collected data, extract the archive and review the files.



References

The `insights-client(8)` and `insights-client.conf(5)` man pages

For more information, refer to the *Getting Started with Red Hat Insights* guide at
https://access.redhat.com/documentation/en-us/red_hat_insights/2023/html-single/getting_started_with_red_hat_insights

For more information about exclusions and obfuscations, refer to the *Opting Out of Sending Metadata from Red Hat Insights Client* article at
<https://access.redhat.com/articles/2025273>

For more information about the data collected by the Red Hat Insights client, refer to the *System Information Collected by Red Hat Insights* article at
<https://access.redhat.com/articles/1598863>

► Guided Exercise

Registering Systems with Red Hat Insights

Register systems with Red Hat Insights.

Outcomes

- Use the Insights client to register a RHEL system.

Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

```
[student@workstation ~]$ lab start insights-registering
```

Instructions

- 1. Register the `serverd` machine with Red Hat Insights.

1.1. Log in to the `serverd` machine as the `student` user. No password is required.

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

1.2. Use the `sudo -i` command to switch identity to the `root` user. Use `student` as the password.

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

1.3. Use the `insights-client` command to register the machine.

```
[root@serverd ~]# insights-client --register
```

- 2. Navigate to `Inventory > Systems` to verify that the `serverd` machine is added successfully.

The screenshot shows the 'Systems' page of the Red Hat Insights web interface. At the top, there are search and filter options ('Name', 'Filter by name', 'Delete'). Below the header is a table with columns: Name, Group, Tags, OS, and Last seen. A single row is listed: 'serverd.lab.example.com' under 'Name', 'No group' under 'Group', '0' under 'Tags', 'RHEL 9.2' under 'OS', and 'Just now' under 'Last seen'. At the bottom of the table are navigation links: '1-1 of 1', '<<', '<', '1', 'of 1', '>', and '>>'. The entire interface has a light gray background with blue and black text.

- 3. Use the `insights-client` command to upload the client's information to the Red Hat Insight service.

```
[root@serverd ~]# insights-client
Uploading Insights data.
Successfully uploaded report from serverd.lab.example.com to account 5296548.
View details about this system on console.redhat.com:
https://console.redhat.com/insights/inventory/bcbf902a-dffc-485e-9aee-af36efc53658
```

- 4. When finished, return to the `workstation` machine.

```
[root@serverd ~]# logout
[student@serverd ~]$ logout
[student@workstation ~]$
```

Finish

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

```
[student@workstation ~]$ lab finish insights-registering
```

Creating and Reviewing Red Hat Insights Reports

Objectives

- Review and interpret issue reports that are provided by Red Hat Insights.

Red Hat Insights Dashboard

The Red Hat Insights dashboard provides information about potential issues in your environment. The dashboard displays graphs and recommendations for Common Vulnerabilities and Exposures (CVEs), risks, exploits, remediation, optimizations, compliance, and more. You can view specific information about systems or issues by clicking on the resources or graphs.

Viewing Vulnerability CVE Information

You can view the CVEs list by navigating to Security > Vulnerability > CVEs. The CVEs page has a filterable and sortable list of CVEs that affect one or more systems in your environment. You can sort by publish date, severity, Common Vulnerability Scoring System (CVSS) score, and more. Each CVE entry includes a detailed description that you can access by clicking the CVE identifier. Many CVEs also include remediation in the form of Ansible Playbooks, scripts, or manual steps.

CVE Reports

Red Hat Insights can create PDF reports for CVEs from the Security > Vulnerability > Reports page.

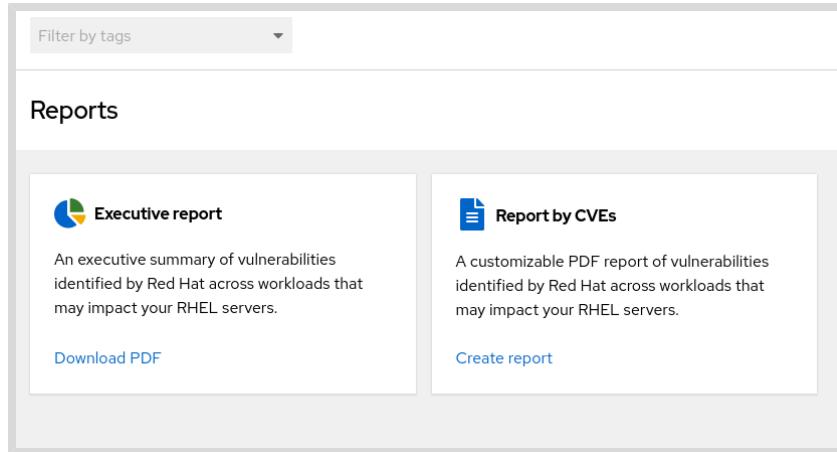


Figure 10.3: Red Hat Insights vulnerability reports

The **Executive report** tool generates a PDF that has general statistics about the CVEs and the security rules that apply to your environment, including detailed information about several of the highest severity CVEs or rules that apply.

The **Report by CVEs** tool enables you to create specific reports that you can filter by various options. You can create a report by clicking **Create report**. By selecting from the filtering options, you can craft a specific report that has only the information that you need. The tool obtains the information that you request and generates a PDF with the results.

The screenshot shows a modal dialog titled 'Report by CVEs'. It contains several sections for configuring the report:

- Report title:** A text input field containing 'Insights Vulnerability CVE Report'.
- Filter CVEs by:** A section with six dropdown menus:
 - Security rule: All
 - Known exploit: All
 - Severity: All
 - CVSS base score: All
 - Business risk: All
 - Status: All
 - Publish date: All
 - Applies to OS: Any
- Advisory:** A dropdown menu showing 'Available'.
- Applying to systems in your inventory meeting these criteria:** A dropdown menu showing 'Tags: All'.
- CVE data to include:** Buttons for 'All columns' and 'Choose columns'.
- Sort CVEs by:** A dropdown menu showing 'CVSS base score: High to Low (Default)'.
- User notes (optional):** A large text area for notes.

At the bottom are two buttons: 'Export report' (blue) and 'Cancel'.

Figure 10.4: Reports by CVEs

Vulnerabilities by System

You can analyze the systems in your environment from the **Security > Vulnerability > Systems** page. This page has a sortable list of each system in your environment. You can select individual systems from this list to view all the CVEs and security rules that apply to that system, along with advisories and remediation information.

Understanding Vulnerability Scores

Vulnerability scores are an objective way to gauge the severity of vulnerabilities. Vulnerability scores are presented as *Common Vulnerability Scoring System* (CVSS) scores. These CVSS scores range from 0.0 to 10.0, with greater values indicating higher severity. CVSS scores supplement Red Hat severity ratings.

The CVSS does not measure risk directly, and instead measures only severity. Depending on your environment, the severity of a vulnerability might not correlate with the risk that it poses. For example, a vulnerability in a web server when processing a specific type of image file might be high severity, but this vulnerability might pose a low risk if your web server has no mechanism to load such a file. Alternatively, a low severity vulnerability that does not allow for privilege escalation on its own might pose a high risk in environments where the vulnerability can be combined with other vulnerabilities. Always perform a comprehensive risk assessment for vulnerabilities that affect your environment.

The CVSS is owned and maintained by the US-based nonprofit, the Forum of Incident Response and Security Teams (FIRST). FIRST develops new CVSS versions over time. The latest version of the CVSS is CVSS 4.0, which was released on November 1, 2023, and which replaces CVSS 3.1. The industry is currently shifting to the new version.

CVSS Version 3.1 Scores

CVSS 3.1 scores use the following factors to calculate a score:

- Attack vector (AV)
- Attack complexity (AC)
- Privileges required (PR)
- User interaction (UI)
- Scope change (S)
- Confidentiality impact (C)
- Integrity impact (I)
- Availability impact (A)

By using a CVSS 3.1 calculator, you can use these factors to find the severity of a vulnerability.

CVSS Version 4.0 Scores

With CVSS 4.0, different metrics include various factors. Depending on the factors that you need to determine risk in your environment, you might use a specific metric.

CVSS-B

This metric is the base score.

CVSS-BE

This metric includes both the base score and environmental factors.

CVSS-BT

This metric includes both the base score and threat factors.

CVSS-BTE

This metric includes the base score, threat factors, and environmental factors.

CVSS 4.0 scores are based on the following base factors, which expand the CVSS 3.1 factors:

- Attack vector (AV)
- Attack complexity (AC)
- Attack requirements (AT)
- Privileges required (PR)
- User interaction (UI)
- Vulnerable system confidentiality (VC)
- Vulnerable system integrity (VI)
- Vulnerable system availability (VA)
- Subsequent system confidentiality (SC)
- Subsequent system integrity (SI)
- Subsequent system availability (SA)

The base factors can be expanded with additional threat, environmental, and supplementary factors.

Red Hat Severity Ratings

In addition to including CVSS scores, Red Hat rates the severity of security issues in Red Hat products on the Red Hat security rating scale. This scale ranges from Critical to Low impact.

Critical impact

This rating is given to flaws that could easily be exploited by a remote unauthenticated attacker, and which could lead to system compromise (arbitrary code execution) without requiring user interaction. Flaws that require authentication, local or physical access to

a system, or an unlikely configuration are not classified as Critical impact. This type of vulnerability can be exploited by worms.

Important impact

This rating is given to flaws that can easily compromise the confidentiality, integrity, or availability of resources. This type of vulnerability allows local or authenticated users to gain additional privileges, allows unauthenticated remote users to view resources that should otherwise be protected by authentication or other controls, allows authenticated remote users to execute arbitrary code, or allows remote users to cause a denial of service.

Moderate impact

This rating is given to flaws that might be more difficult to exploit, but which could still lead to some compromise of the confidentiality, integrity, or availability of resources under certain circumstances. This type of vulnerability could have a Critical or Important impact, but it is less easily exploited based on a technical evaluation of the flaw, or the vulnerability affects unlikely configurations.

Low impact

This rating is given to all other issues that might have a security impact. This type of vulnerability is generally believed to require unlikely circumstances to be able to be exploited, or where a successful exploit would have minimal consequences. This includes flaws that are present in a program's source code but for which no current or theoretically possible, but unproven, exploitation vectors exist or were found during the technical analysis of the flaw.

Red Hat uses the CVSS score along with other factors that are specific to Red Hat products to apply a severity rating for a vulnerability. By using both the CVSS score and the Red Hat severity rating as a starting point, you can analyze how much risk a vulnerability poses to your environment.



References

For more information about FIRST and CVSS scores, refer to
<https://www.first.org/cvss/>

For more information about Red Hat Insights reports, refer to the *Generating Vulnerability Service Reports* guide at
https://access.redhat.com/documentation/en-us/red_hat_insights/2023/html-single/generating_vulnerability_service_reports/index

► Guided Exercise

Creating and Reviewing Red Hat Insights Reports

Review and interpret issue reports for a RHEL system in the Red Hat Insights dashboard.

Outcomes

- View and analyze the results of an Executive report.
- Create a custom Insights report and analyze its results.

Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

```
[student@workstation ~]$ lab start insights-creating
```

Instructions

- ▶ 1. On the workstation machine, open a web browser and navigate to the Hybrid Cloud Console at <https://console.redhat.com/insights/>. Log in with your Red Hat account.
- ▶ 2. Download and view the Executive report.
 - 2.1. Navigate to **Security > Vulnerability > Reports**.
 - 2.2. Click **Download PDF** on the **Executive report** card.
 - 2.3. Wait for the report generation and open the downloaded file.
- ▶ 3. Analyze the results of the Executive report.
 - 3.1. View the CVEs section and determine the CVE with the highest CVSS score.
 - 3.2. View the Insights Security Rules section and determine the highest severity security rule.
- ▶ 4. Generate a report of CVEs that have known exploits.
 - 4.1. Navigate to **Security > Vulnerability > Reports**.
 - 4.2. Click **Create report** on the **Report by CVEs** card.
 - 4.3. Select the **Known Exploit > Has a known exploit** filter option.
 - 4.4. Click **Export report**.
- ▶ 5. Analyze the results of the CVE report.

- 5.1. View the list of CVEs and determine the highest severity CVE.
- 5.2. Click the CVE ID to view more information about it.

Finish

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

```
[student@workstation ~]$ lab finish insights-creating
```

Running OpenSCAP Reports from Red Hat Insights

Objectives

- Use the compliance service in Red Hat Insights to configure OpenSCAP policies and review the resulting reports.

Running OpenSCAP Reports from Red Hat Insights

The Red Hat Insights compliance service enables security and compliance administrators to monitor, assess, and report the compliance of RHEL systems. The compliance service enables the creation, configuration, and management of SCAP policies in a simple interface that includes filtering and context-adding features. You can also use the compliance service to create Ansible Playbooks to resolve compliance issues.

You can create reports to communicate the compliance status to other stakeholders in your organization.

Red Hat Insights Compliance Service Overview

For Red Hat Insights to be able to access or create security policies, the compliance service requires that the `scap-security-guide` package is installed on your RHEL systems.

To access the policies that are available to the compliance service, navigate to **Security > Compliance > SCAP Policies**. On the **SCAP policies** page, you can create or edit security policies, and list the systems that are registered to each policy.

If your Red Hat Insights organization does not have any policies, then you are prompted to create a policy.

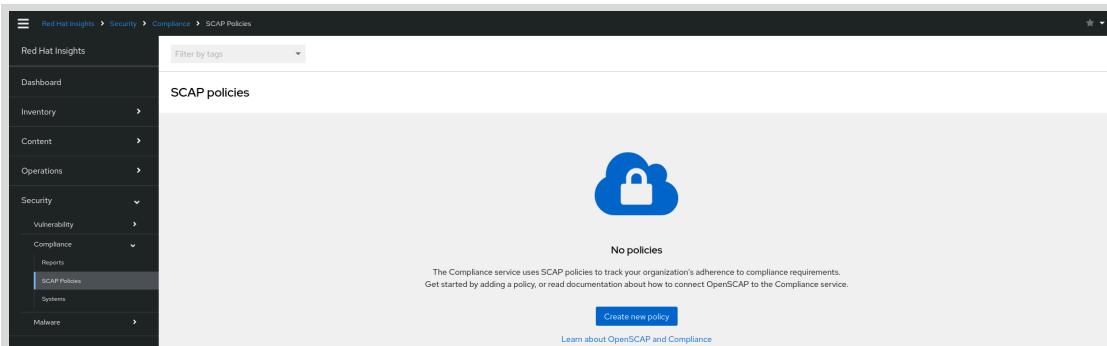


Figure 10.5: SCAP policies

After you create one or more policies, the policy list appears on the **SCAP policies** page.

Name	Operating system	Systems	Business objective	Compliance threshold
CIS Red Hat Enterprise Linux 9 Benchmark for Level 1 - Server CIS Red Hat Enterprise Linux 9 Benchmark for Level 1 - Server	RHEL 9	1	--	100%

Figure 10.6: List of policies

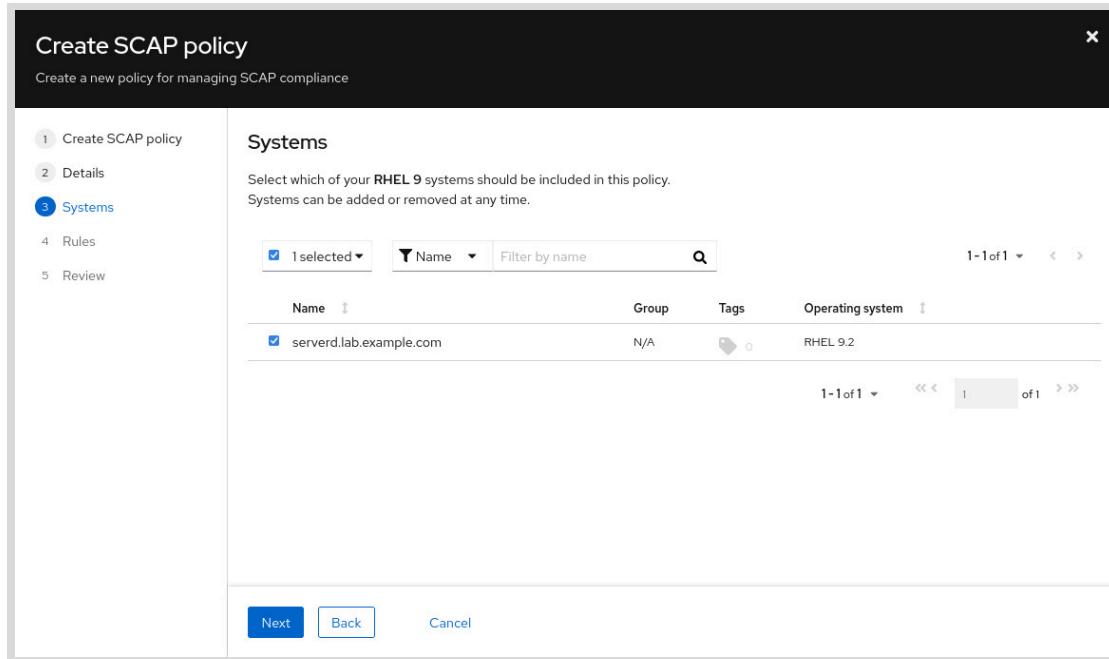
Creating New OpenSCAP Policies

The process of creating a policy requires several steps, which include selecting the RHEL version, adding details (such as name and other attributes), selecting systems, and customizing the policy rules.

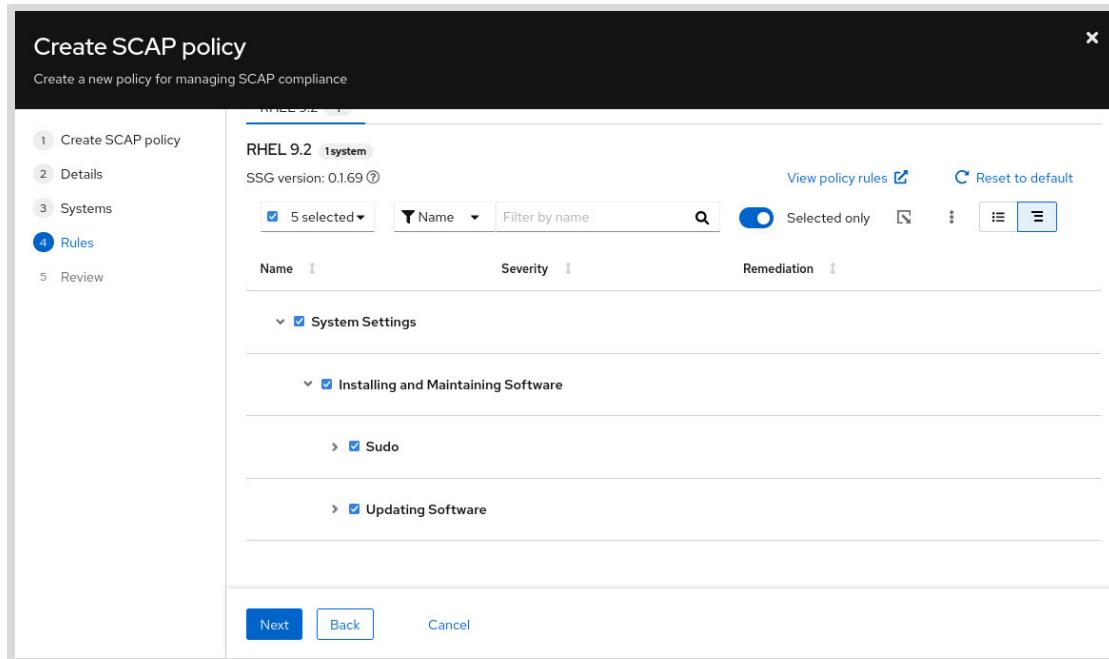
After you select the operating system, you can filter for a policy type by using the **Policy type** search field.

Figure 10.7: Create SCAP policy

After customizing the policy details, you select which systems to include in the policy from a list of available systems.

**Figure 10.8: Select the systems**

During the creation process, you can customize your policy for your environment's needs. Customizing your policy is useful because there are some rules that do not apply to all situations. In the following example, only the `Sudo` and `Updating Software` rules are selected.

**Figure 10.9: SCAP customization**

In the final step, you review the new policy before creating it.

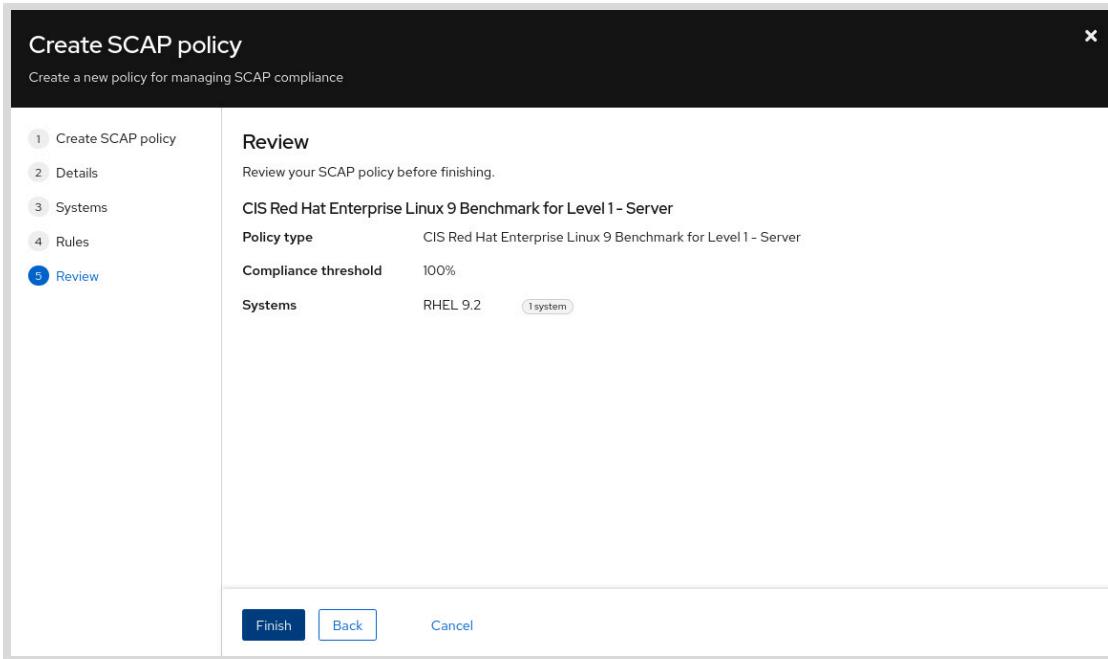


Figure 10.10: Review SCAP policy

Creating Compliance Reports

To create a compliance report, run the `insights-client --compliance` command as a user with `root` privileges. This command runs the scan for the configured policies and uploads the results to the compliance service databases.

To view the reports, navigate to **Security > Compliance > Reports**.

Red Hat Insights does not display any reports until you run the `insights-client --compliance` command.

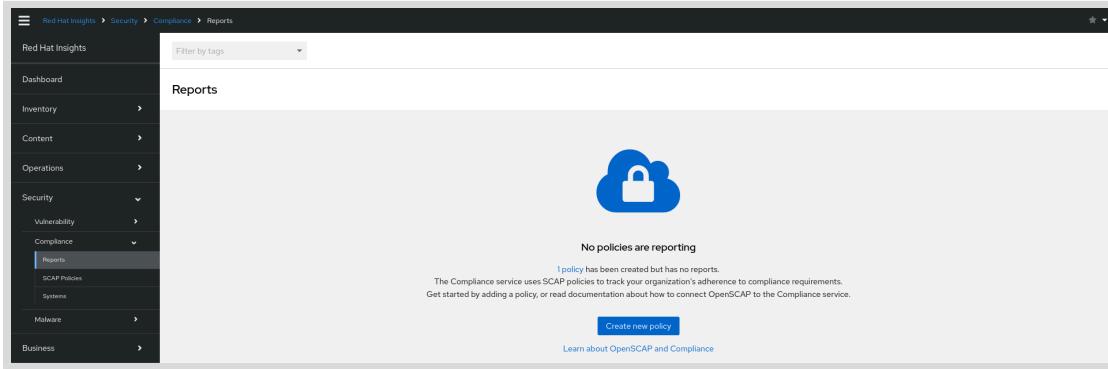


Figure 10.11: SCAP reports

To make compliance reports available in the console, run the `insights-client --compliance` command:

Chapter 10 | Analyzing and Remediating Issues with Red Hat Insights

```
[root@host ~]# insights-client --compliance
System uses SSG version 0.1.66
Saved tailoring file for xccdf_org.ssgproject.content_profile_cis_server_l1 to /
var/tmp/oscap_tailoring_file-
xccdf_org.ssgproject.content_profile_cis_server_l1.fruiolnu.xml
Running scan for xccdf_org.ssgproject.content_profile_cis_server_l1... this may
take a while
Uploading Insights data.
Successfully uploaded report for demo.lab.example.com.
```

After running the `insights-client` command, reload the **Reports** page. The **Reports** page now shows the available reports:

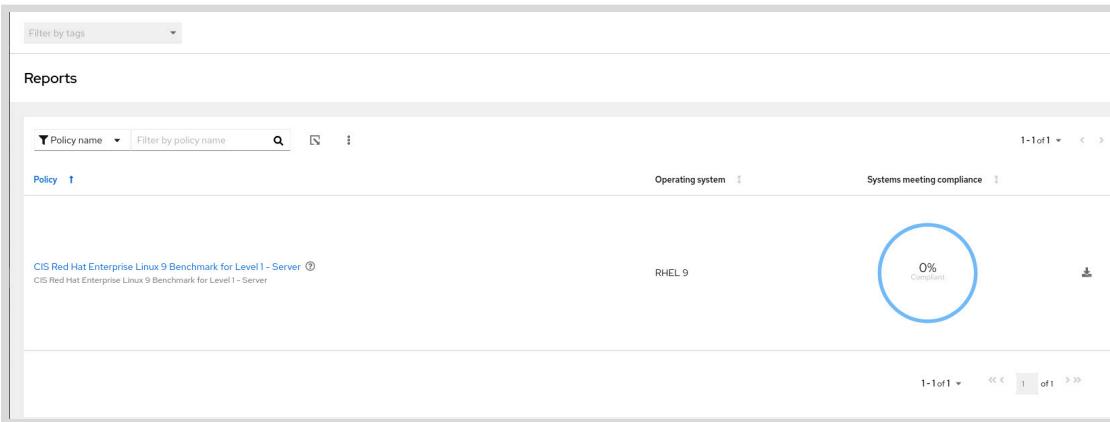


Figure 10.12: SCAP reports

Click a report to see the policy details:

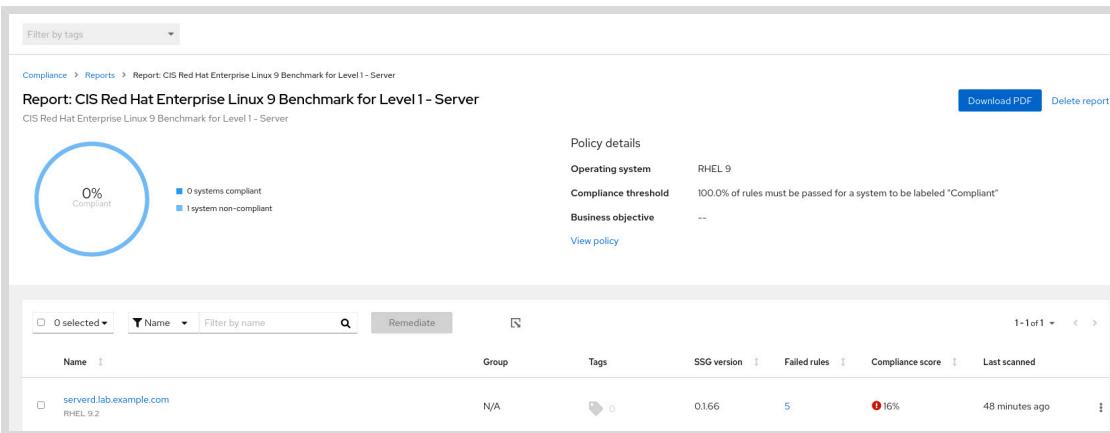


Figure 10.13: SCAP policy details

The **Reports** page displays a list of servers that use this policy. Click a server link to see the report details:

Chapter 10 | Analyzing and Remediating Issues with Red Hat Insights

The screenshot shows the Red Hat Insights Compliance interface. At the top, it displays the server details: 'server.lab.example.com' with a UUID of 'fd466fac-59fd-4635-b500-b477d46a405f'. It was last seen on '07 Nov 2023 21:58 UTC'. Below this, a summary box indicates 'CIS Red Hat Enterprise Linux 9 Benchmark for Level 1 - Server' with a 'Not compliant' status, 3 rules failed (Score: 10%), SSS version 0.1.66, and a scan completed 56 minutes ago. The main table lists compliance rules categorized by policy: System Settings, Installing and Maintaining Software, Account and Access Control, Services, and SSH Server. Each row shows the rule name, severity (e.g., '4x fail'), and remediation status. A 'Remediate' button is available for each row.

Figure 10.14: Server details

To download the report, click **Download PDF**. In the dialog that displays, choose which system data to include.

To generate periodic reports, you might use an automation controller provided by Red Hat Ansible Automation Platform to schedule Ansible Playbooks to periodically run the `redhat.insights.compliance` role on managed hosts that you must monitor.



References

For more information, refer to the *Getting Started Using the Compliance Service* chapter in the *Assessing and Monitoring Security Policy Compliance of RHEL Systems* guide at

https://access.redhat.com/documentation/en-us/red_hat_insights/2023/html-single/assessing_and_monitoring_security_policy_compliance_of_rhel_systems/index#compliance-getting-started_intro-compliance

Getting Started with Red Hat Insights and OpenSCAP for Compliance Reporting

<https://www.redhat.com/en/blog/getting-started-red-hat-insights-and-openscap-compliance-reporting>

► Guided Exercise

Running OpenSCAP Reports from Red Hat Insights

Use the compliance service in Red Hat Insights to configure OpenSCAP policies and review the resulting reports.

Outcomes

- Configure OpenSCAP policies in Red Hat Insights.
- Review RHEL systems compliance by using Red Insights.

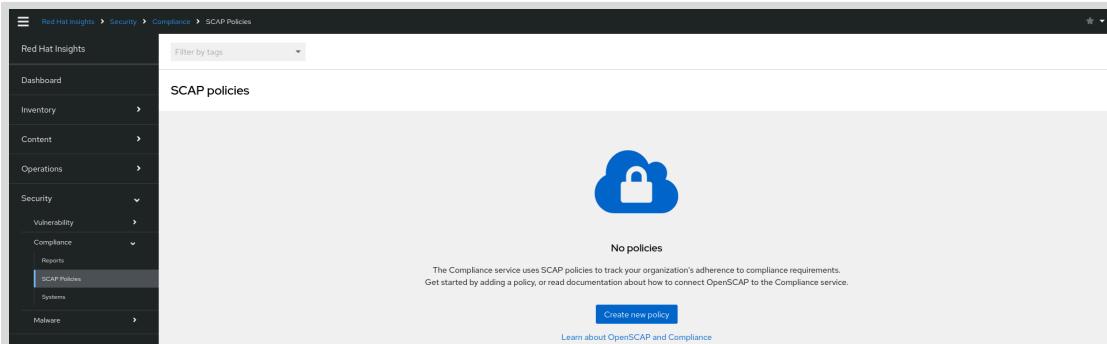
Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

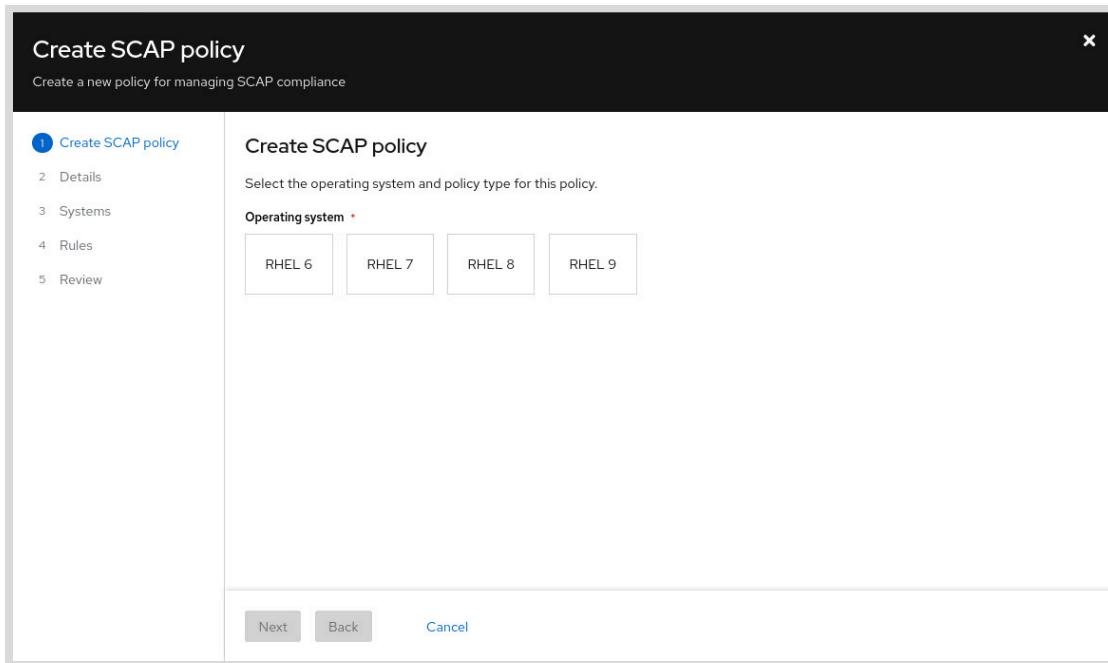
```
[student@workstation ~]$ lab start insights-running
```

Instructions

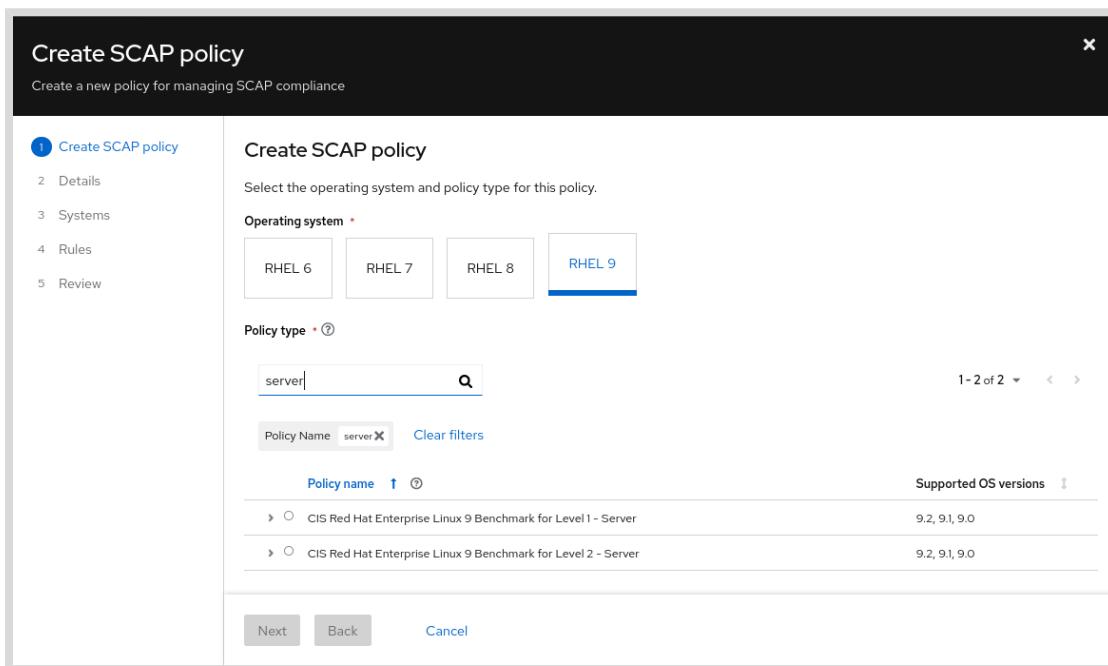
- 1. On the workstation machine, open a web browser and navigate to the Hybrid Cloud Console at <https://console.redhat.com/insights/>. Log in with your Red Hat account. Navigate to **Security > Compliance > SCAP Policies**.



- 2. Click **Create new policy**.

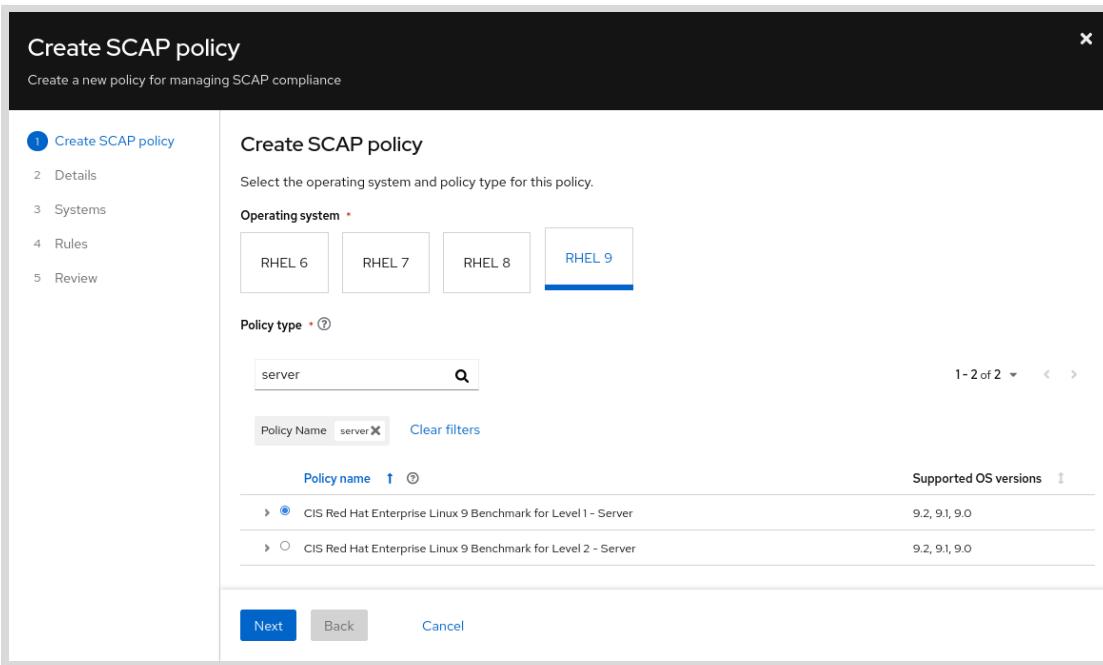


- 3. Select RHEL 9 as the operating system, and filter the policy types by using the **server** keyword in the **Policy type** search field.

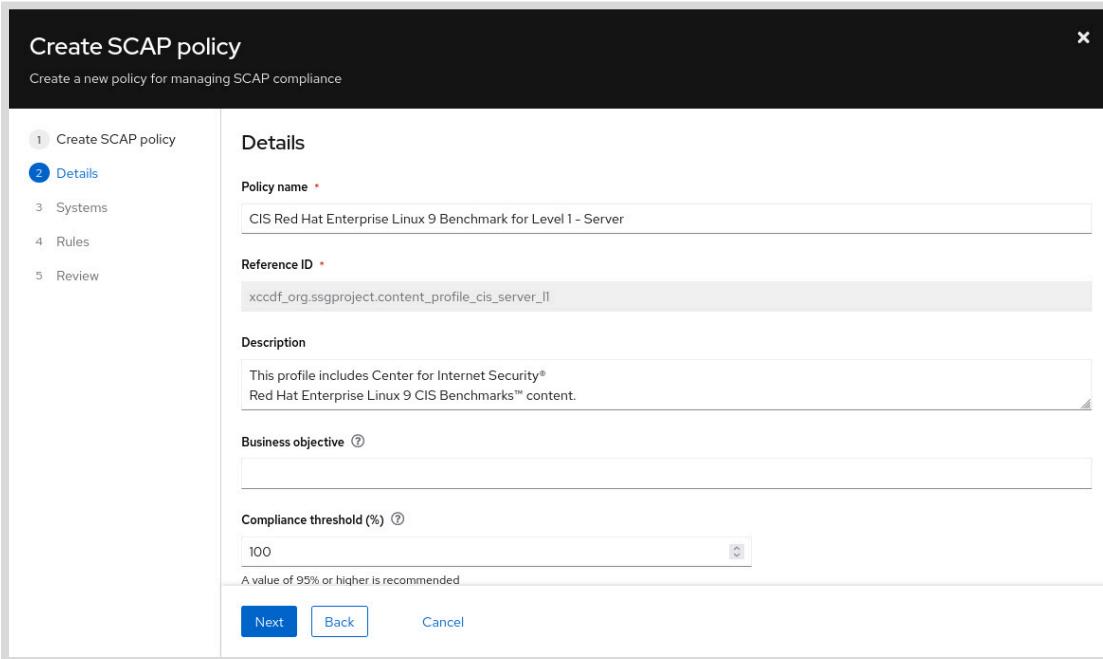


Chapter 10 | Analyzing and Remediating Issues with Red Hat Insights

- 4. Select the CIS Red Hat Enterprise Linux 9 Benchmark for Level 1 - Server policy and click **Next**.



- 5. Review the policy details and click **Next**.



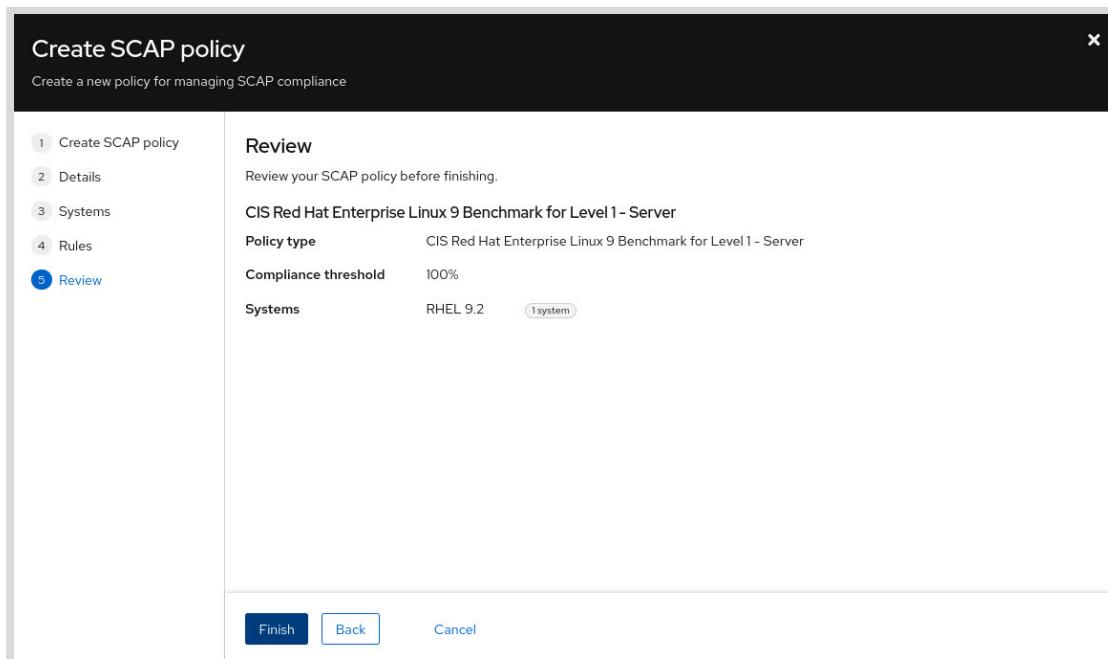
- 6. Select the `serverd.lab.example.com` system to be added to this policy, and click **Next**.

The screenshot shows the 'Create SCAP policy' wizard, step 3: Systems. The left sidebar lists steps 1 through 5. The main area is titled 'Systems' and contains a table with one row. The table columns are Name, Group, Tags, and Operating system. The single row shows 'serverd.lab.example.com' under 'Name', 'N/A' under 'Group', an empty 'Tags' field, and 'RHEL 9.2' under 'Operating system'. Below the table are navigation buttons: 'Next' (highlighted in blue), 'Back', and 'Cancel'.

- 7. Customize the policy for the environment's needs. Select the **Sudo** and **Updating Software** rules, and click **Next**.

The screenshot shows the 'Create SCAP policy' wizard, step 4: Rules. The left sidebar lists steps 1 through 5. The main area shows the 'Rules' section. At the top, it says 'RHEL 9.2 1system' and 'SSG version: 0.1.69'. There are buttons for 'View policy rules' and 'Reset to default'. Below this is a table with columns 'Name', 'Severity', and 'Remediation'. Under 'Name', there are three expandable sections: 'System Settings' (with 'Installing and Maintaining Software' and 'Sudo' checked), 'Sudo' (with 'Updating Software' checked), and another unnamed section. Navigation buttons 'Next' (highlighted in blue), 'Back', and 'Cancel' are at the bottom.

► 8. Review your custom policy, and click **Finish**



► 9. On the serverd machine, install the SCAP Security Guide.

- 9.1. Log in to the serverd machine as the student user. No password is required.

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

- 9.2. Use the sudo -i command to switch identity to the root user. Use student as the password.

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

- 9.3. Install the scap-security-guide package, which provides the SCAP Security Guide.

```
[root@serverd ~]# dnf install scap-security-guide
...output omitted...
Complete!
```

► 10. Run the insights-client --compliance command.

```
[root@serverd ~]# insights-client --compliance
System uses SSG version 0.1.66
Saved tailoring file forxccdf_org.ssgproject.content_profile_cis_server_l1 to /var/tmp/oscap_tailoring_file-
xccdf_org.ssgproject.content_profile_cis_server_l1.fruiolnu.xml
Running scan forxccdf_org.ssgproject.content_profile_cis_server_l1... this may
take a while
Uploading Insights data.
Successfully uploaded report for serverd.lab.example.com.
```

- 11. Navigate to the **Security > Compliance > Reports** page to view the compliance reports.

The screenshot shows the 'Reports' section of the Red Hat Insights interface. A single policy is listed: 'CIS Red Hat Enterprise Linux 9 Benchmark for Level 1 - Server'. The operating system is identified as 'RHEL 9'. A large circular progress bar indicates '0% Compliant'. Below the table, there are navigation links for '1-1 of 1' and 'of 1'.

- 12. Click the **CIS Red Hat Enterprise Linux 9 Benchmark for Level 1 - Server** policy to see the report details:

This screenshot displays the detailed report for the 'CIS Red Hat Enterprise Linux 9 Benchmark for Level1 - Server' policy. It includes policy details like 'Operating system: RHEL 9', 'Compliance threshold: 100.0%', and 'Business objective: --'. The main summary shows '0% Compliant' with a note that '0 systems compliant' and '1 system non-compliant'. Below this, a table lists one system: 'serverd.lab.example.com' (RHEL 9.2), which has a compliance score of 16%. There are buttons for 'Download PDF' and 'Delete report'.

Chapter 10 | Analyzing and Remediating Issues with Red Hat Insights

- 13. Click [serverd.lab.example.com](#) to see the report details for this server.

The screenshot shows the Red Hat Insights Compliance interface. At the top, it displays the path: Compliance > Systems > serverd.lab.example.com. Below this, it shows the server's UUID: fd460fae-59fd-4035-b560-b471d46a405f and last seen date: 07 Nov 2023 21:58 UTC. A prominent red box highlights the 'CIS Red Hat Enterprise Linux 9 Benchmark for Level I - Server' section, which states 'Not compliant' with '5 rules failed (Score: 10%)'. The SSG version is listed as 0.1.66 and last scanned 56 minutes ago. The main content area lists failing rules under various categories:

- System Settings:** 4x fail
- Installing and Maintaining Software:** 3x fail
- Account and Access Control:** 1x fail
- Services:** 1x fail
- SSH Server:** 1x fail

Finish

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

```
[student@workstation ~]$ lab finish insights-running
```

Integrating Red Hat Insights and Automation Controller

Objectives

- Remediate issues that are reported by Red Hat Insights by using automation controller from Red Hat Ansible Automation Platform.

Automate Red Hat Insights with Automation Controller

Integrating automation controller with Red Hat Insights enables you to automate Insights remediation for a group of registered systems. In automation controller, you can create a Source Control Manager (SCM) project that uses Insights as the SCM for the project's Ansible Playbooks. You must create a credential in automation controller of the `Insights` type to store the username and password for your Red Hat Insights account, and then configure your project to use that credential for authentication.

One of the advantages of this configuration is that users can run maintenance plan Ansible Playbooks from automation controller without having direct access to the credentials for the Red Hat Insights account. The credentials are stored in automation controller, and you can use automation controller's permissions system to protect them from being read directly.

Create Credentials for Insights in Automation Controller

The Insights credential stores the username and password that are used to log in to the Red Hat Insights interface in the Customer Portal. The Insights project in automation controller uses the Insights credential to download the Ansible Playbooks that are created by using the Insights planner, and to list the Insights actions that are associated with a registered system.

In the automation controller web UI, use the following steps to create an Insights credential:

- Navigate to Resources > **Credentials** and click **Add**.
- Specify a name for the credential and optionally enter a description.
- Select **Insights** from the **Credential Type** list.
- Use a valid Red Hat Customer Portal account username and password for the **Type Details** credentials.
- Click **Save**.

The screenshot shows the 'Create New Credential' dialog box. At the top left is a 'Credentials' link. The main title is 'Create New Credential'. On the right side is a small circular icon with a question mark. The form has two sections: 'Name *' with a field containing 'My Insights Credential' and a 'Description' field which is empty; and 'Organization' with a search bar containing a magnifying glass icon and a dropdown menu showing 'Insights'. Below this is a section titled 'Type Details' with 'Username *' set to 'my_username' and 'Password *' set to a masked string. There are lock icons next to both fields. At the bottom are 'Save' and 'Cancel' buttons.

Figure 10.26: Create an Insights credential

Create an Insights Project in Automation Controller

Automation controller projects support a special SCM repository type specifically for Red Hat Insights. Upon selecting the Insights source control, the project retrieves its Ansible Playbooks directly from the maintenance plans that you created in Insights.

In the automation controller web UI, use the following steps to create an Insights project:

- Navigate to **Resources > Projects** and click **Add**.
- Specify a name for the project and optionally enter a description.
- Select **Red Hat Insights** from the **Source Control Type** list.
- In the **Insights Credential** search field, select the Insights credential that you previously created.
- Click **Save**.

The screenshot shows the 'Create New Project' dialog box. At the top left is a 'Projects' link. The main title is 'Create New Project'. On the right side is a small circular icon with a 'G' inside. The form contains several input fields:

- Name ***: A text input field containing "My Insights Project".
- Description**: An empty text input field.
- Organization ***: A dropdown menu showing "Default".
- Execution Environment**: A dropdown menu with a search icon and an empty input field.
- Source Control Type ***: A dropdown menu showing "Red Hat Insights".
- Content Signature Validation Credential**: A dropdown menu with a search icon and an empty input field.

Below these fields is a section titled "Type Details". It includes:

- Insights Credential ***: A dropdown menu showing "My Insights Credential".
- Options**: Three checkboxes: "Clean" (unchecked), "Delete" (unchecked), and "Update Revision on Launch" (unchecked).

At the bottom are two buttons: a blue "Save" button and a "Cancel" button.

Figure 10.27: Create an Insights project

Create an Inventory for Insights in Automation Controller

The Ansible Playbooks that are created by Insights specify hosts by name for each play. The hostnames in the inventory that is used by automation controller must match the names that are used for registered systems in the Red Hat Insights Customer Portal. You can manually add the hosts in the inventory, or retrieve them as dynamic inventory from Insights.

In the automation controller web UI, use the following steps to create an inventory:

- Navigate to Resources > Inventories.
- Click Add, and select the Add inventory type.
- Specify a name for the inventory.
- Click Save.

The screenshot shows the 'Create new inventory' dialog box. At the top left is the title 'Create new inventory'. On the right side, there is a small circular icon with a question mark. The main form area contains several input fields and sections:

- Name ***: A text input field containing "My Insights Inventory".
- Description**: An empty text input field.
- Organization ***: A dropdown menu showing "Default".
- Instance Groups**: A search bar with a magnifying glass icon.
- Labels**: A dropdown menu with a single entry "1 ---".
- Options**: A checkbox labeled "Prevent Instance Group Fallback" which is unchecked.
- Variables**: A section with tabs for "YAML" (selected) and "JSON". Below the tabs is a code editor window showing the YAML representation of the variable: "1 ---".
- Buttons**: At the bottom are two buttons: "Save" (blue) and "Cancel".

Figure 10.28: Create an inventory

After creating the inventory, use the following steps to add the host manually:

- Click the **Hosts** tab.
- Click **Add**.
- Specify the name for the host.
- Click **Save**.

Alternatively, you can create an Insights inventory source to dynamically generate a list of hosts that are registered in Insights. After creating the inventory, use the following steps to add the host dynamically:

- Click the **Sources** tab.
- Click **Add**.
- Specify the name for the inventory source.
- Select **Red Hat Insights** from the **Source** list.
- For the **Credential** field under **Source details**, use the Red Hat Insights credential that you previously created.
- Click **Save**.

The screenshot shows the 'Create new source' dialog in the Red Hat Insights interface. The 'Name' field contains 'My Insights Source'. The 'Source' dropdown is set to 'Red Hat Insights'. Under 'Source details', the 'Credential' dropdown shows 'My Insights Credential' and the 'Verbosity' dropdown is set to '1 (Info)'. The 'Host Filter' and 'Enabled Variable' fields are empty. The 'Enabled Value' field contains a single dash ('-'). In the 'Update options' section, none of the three checkboxes ('Overwrite', 'Overwrite variables', 'Update on launch') are selected. The 'Source variables' editor shows two entries: '1 ---' and '2'. At the bottom, there are 'Save' and 'Cancel' buttons.

Figure 10.29: Create an inventory source

Create a Job Template in Automation Controller

To enable automation controller to use the Ansible Playbooks from Insights maintenance plans in a job template, the playbooks must be imported from your Insights account into your project in automation controller.

In the automation controller web UI, use the following steps to create a job template:

- Navigate to Resources > Templates.

Chapter 10 | Analyzing and Remediating Issues with Red Hat Insights

- Click **Add**, and select **Add job template** from the list.
- Specify the name for the job template.
- In the **Job Type** list, select **Run** to execute the job template when it launches.
- In the **Inventory** search field, select the **My Insights Inventory** inventory.
- In the **Project** field, select the **My Insights Project** project.
- In the **Playbook** list, select the Ansible Playbook that is associated with the maintenance plan that you want to run.
- In the **Credentials** search field, select the credentials to authenticate the automation controller.
- Click **Save**.

The screenshot shows the 'Create New Job Template' dialog box. It has several input fields and dropdown menus:

- Name ***: My Insights Job Template
- Description**: (empty)
- Job Type * ?**: Run
- Prompt on launch**: (checkbox)
- Inventory * ?**: My Insights Inventory
- Prompt on launch**: (checkbox)
- Project * ?**: My Insights Project
- Execution Environment ?**: (empty)
- Prompt on launch**: (checkbox)
- Playbook * ?**: my-insights-playbook-1e164032-21f8-4a6e-a... (dropdown menu)
- Credentials ?**: SSH: Classroom Cred... (selected)
- Prompt on launch**: (checkbox)

Figure 10.30: Create a job template

To run the job template, click **Launch** to run the Ansible Playbook.

Alternatively, you can run the job template by navigating to **Resources > Templates**, and then clicking the launch icon (the **rocket** icon).

The screenshot shows the 'Output' tab for a job named 'My Insights Job Template'. The job status is 'Successful'. The output pane displays the command-line interface of the playbook execution. It includes logs for PLAY and TASK steps, and a final PLAY RECAP summary. The recap shows 7 ok, 3 changed, 0 unreachable, 0 failed, 0 skipped, 0 rescued, and 0 ignored tasks across one host.

```
25 PLAY [run insights] ****
** 14:43:04
26
27 TASK [run insights] ****
** 14:43:04
28 ok: [serverc.lab.example.com]
29
30 PLAY RECAP ****
** 14:44:08
31 serverc.lab.example.com : ok=7    changed=3    unreachable=0    failed=0
skipped=0    rescued=0    ignored=0
```

Figure 10.31: Job template result

Automation controller shows the result of the playbook execution. To find the results of the executed jobs, navigate to **Jobs** from the navigation bar.



References

For more information about Red Hat Insights remediations, refer to the *Setting up Insights Remediations* section of the *Automation Controller User Guide* at <https://docs.ansible.com/automation-controller/latest/html/userguide/insights.html>

► Guided Exercise

Integrating Red Hat Insights and Automation Controller

Remediate an issue that is reported by Red Hat Insights by using an Ansible Playbook provided by Insights and automation controller.

Outcomes

- Register a host by using the `insights-client` command.
- Configure Insights and create an Ansible job template.
- Remediate the host by running the Ansible job template.

Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

```
[student@workstation ~]$ lab start insights-integrating
```

Instructions

- 1. From the workstation machine, log in to the `serverd` machine as the `student` user, and switch to the `root` user.
- 1.1. Log in to the `serverd` machine as the `student` user.

```
[student@workstation ~]$ ssh student@serverd
[student@serverd ~]$
```
 - 1.2. Change to the `root` user. Use `student` as the password.

```
[student@serverd ~]$ sudo -i
[sudo] password for student: student
[root@serverd ~]#
```
- 2. Check the `serverd` machine registration status with Red Hat Customer Portal and Red Hat Insights.
- 2.1. Check the `serverd` machine registration status with Red Hat Customer Portal. If the machine is not registered, then register it by using the `subscription-manager register` command. Use a valid Red Hat Customer Portal account.

```
[root@serverd ~]# subscription-manager status
+-----+
 System Status Details
+-----+
Overall Status: Disabled
Content Access Mode is set to Simple Content Access. This host has access to
content, regardless of subscription status.

System Purpose Status: Disabled
```



Important

The preceding output indicates that the client is registered using Simple Content Access for its content access mode, even though the overall subscription status is **Disabled**. Because the client is using Simple Content Access, it does not need to have specific product entitlements enabled.

- 2.2. Check the `serverd` machine registration status with Red Hat Insights. If the machine is not registered, then register it by using the `insights-client --register` command. Use a valid Red Hat Customer Portal account.

```
[root@serverd ~]# insights-client --status
System is registered locally via .registered file. Registered at
2023-12-19T11:17:23.005881
Insights API confirms registration.
```

- ▶ 3. Create a playbook to remediate a CVE vulnerability.
 - 3.1. Log in to the Red Hat Insights web UI at <https://console.redhat.com/insights/>. Use the same Red Hat Customer Portal account that you used in Step 2.
 - 3.2. Navigate to **Inventory > Systems**.
 - 3.3. Click `serverd.lab.example.com`.
 - 3.4. Click the **Vulnerability** tab.
 - 3.5. Select the checkbox for the CVE-2023-3972 CVE identifier, and click **Remediate**.
 - 3.6. In the dialog, enter `CVE remediation playbook` in the **Create new playbook** field, and click **Next**.
 - 3.7. If it is not already selected, then select the `serverd.lab.example.com` value and click **Next**.
 - 3.8. Review the remediation. If Insights requests a reboot, then you can disable it by clicking **Turn off autoreboot**.
 - 3.9. Click **Submit** to create the playbook.
- ▶ 4. Log in to the automation controller web UI at <https://controller.lab.example.com>. Use `admin` as the user and `redhat` as the password.
- ▶ 5. Create a **Machine** credential type named `Classroom servers credential`.

- 5.1. Navigate to **Resources > Credentials** and click **Add**.
 - 5.2. Enter **Classroom servers credential** in the **Name** field.
 - 5.3. In the **Organization** search field, enter the **Default** organization.
 - 5.4. Select **Machine** from the **Credential Type** list.
 - 5.5. Enter **student** in the **Username** field.
 - 5.6. Enter **student** in the **Password** field.
 - 5.7. Select **sudo** from the **Privilege Escalation Method** list.
 - 5.8. Enter **root** in the **Privilege Escalation Username** field.
 - 5.9. Enter **student** in the **Privilege Escalation Password** field.
 - 5.10. Leave the other fields untouched and click **Save**.
- 6. Create an Insights credential type named **RH account credential**. Use the same Red Hat Customer Portal account that you used in Step 2.
- 6.1. Navigate to **Resources > Credentials** and click **Add**.
 - 6.2. Enter **RH account credential** in the **Name** field.
 - 6.3. In the **Organization** search field, enter the **Default** organization.
 - 6.4. Select **Insights** from the **Credential Type** list.
 - 6.5. Enter a valid Red Hat Customer Portal account username and password for the **Type Details** credentials.
 - 6.6. Click **Save**.
- 7. Create a Red Hat Insights project type named **Insights project**.
- 7.1. Navigate to **Resources > Projects** and click **Add**.
 - 7.2. Enter **Insights project** in the **Name** field.
 - 7.3. Select **Red Hat Insights** from the **Source Control Type** list.
 - 7.4. Under **Type details**, enter the **RH account credential** value in the **Insights Credential** search field.
 - 7.5. Leave the other fields untouched and click **Save**.
- 8. Create the **My servers inventory** standard inventory. Add the **serverd.lab.example.com** host.
- 8.1. Navigate to **Resources > Inventories**.
 - 8.2. Click **Add**, and select the **Add inventory** type.
 - 8.3. Enter **My servers inventory** in the **Name** field.
 - 8.4. Leave the other fields untouched and click **Save**.

- 8.5. Under the `My servers inventory` details, select the **Hosts** tab, and click **Add**.
 - 8.6. Enter `serverd.lab.example.com` in the **Name** field
 - 8.7. Leave the other fields untouched and click **Save**.
- ▶ **9.** Create a job template named `Fix Insights servers`.
- 9.1. Navigate to **Resources > Templates**.
 - 9.2. Click **Add**, and select **Add job template** from the list.
 - 9.3. Enter `Fix Insights servers` in the **Name** field.
 - 9.4. Select **Run** from the **Job Type** list.
 - 9.5. Enter `My servers inventory` in the **Inventory** search field.
 - 9.6. Enter `Insights project` in the **Project** field.
 - 9.7. Select the Ansible Playbook that is associated with the maintenance plan that you want to run from the **Playbook** list.
 - 9.8. Enter `Classroom servers credential` in the **Credentials** search field.
 - 9.9. Leave the other fields untouched and click **Save**.
- ▶ **10.** Launch the `Fix Insights servers` job template, and verify the result.
- 10.1. Navigate to **Resources > Templates**.
 - 10.2. Click the **Launch Template** icon for the `Fix Insights servers` job template and wait for the job to complete.
 - 10.3. Verify the results of the job.

```
...output omitted...
PLAY RECAP ****
serverd.lab.example.com      : ok=3      changed=1      unreachable=0      failed=0
    skipped=4      rescued=0      ignored=0
```

- ▶ **11.** Verify that the CVE-2023-3972 CVE identifier is no longer reported in the Red Hat Insights web UI for the `serverd.lab.example.com` system.
- 11.1. Log in to the Red Hat Insights web UI at <https://console.redhat.com/insights/>. Use the same Red Hat Customer Portal account that you used in Step 2.
 - 11.2. Navigate to **Inventory > Systems**.
 - 11.3. Click `serverd.lab.example.com`.
 - 11.4. Click the **Vulnerability** tab.
 - 11.5. Enter the CVE-2023-3972 CVE identifier in the **Search ID or description** search field. Confirm that the CVE is no longer reported for the `serverd.lab.example.com` system.

Finish

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

```
[student@workstation ~]$ lab finish insights-integrating
```

Summary

- Red Hat Insights helps you to identify and to remediate threats to the security, performance, availability, and stability of systems that run Red Hat products.
- Red Hat Insights is a Software-as-a-Service (SaaS) offering that is provided through the Red Hat Customer Portal.
- You can register clients for Red Hat Insights analysis in two ways. You can directly register clients through the Customer Portal, or indirectly register clients by using your Red Hat Satellite Server as a proxy.
- You can configure the Insights client on each host to restrict or obscure the data that is sent to Red Hat Insights for analysis. However, this configuration might result in analysis that is less comprehensive.
- You can review Red Hat Insights reports on the Customer Portal, or through your Red Hat Satellite Server.
- The risk and impact of issues are assessed by the **Likelihood**, **Impact**, **Total Risk**, and **Risk of change** factors to help you understand and prioritize the actions to take to address detected issues.
- You can configure a maintenance plan in Red Hat Insights that creates Ansible Playbooks to address issues with your systems.
- By integrating automation controller with Red Hat Insights, you can automate the remediation of issues that are reported by Red Hat Insights on your registered systems.

Chapter 11

Automating Compliance with Red Hat Satellite

Goal

Automate and scale OpenSCAP compliance checks by using Red Hat Satellite.

Sections

- Configuring Red Hat Satellite for OpenSCAP (and Guided Exercise)
- Scan OpenSCAP Compliance with Red Hat Satellite (and Guided Exercise)
- Customize the OpenSCAP Policy in Red Hat Satellite (and Guided Exercise)

Lab

- Automating Compliance with Red Hat Satellite

Configuring Red Hat Satellite for OpenSCAP

Objectives

- Configure Red Hat Satellite to perform OpenSCAP scans of registered servers.

Security Compliance Management with Red Hat Satellite

A security administrator manages security compliance by defining security policies and auditing hosts for compliance based on the defined policies. Any noncompliant hosts are remediated based on the organization's compliance requirements. These compliance policies need to be flexible, because an organization's policy might vary depending on the services that are provided by the host or the industry to which the organization belongs.

Red Hat Satellite is a systems management solution that provisions systems and provides software updates from Red Hat Customer Portal and other sources. Satellite serves as a local repository of software content and a central point of management for Red Hat entitlements. Red Hat Satellite also provisions and manages system configuration to adhere to predefined standard operating environments.

One of the major benefits of Satellite is that it can scale effectively to meet the demands of large enterprises. With the correct design, Satellite delivers solid performance even with increasing workloads and across geographically distributed environments.

Several options are available for administering Satellite Server. You can manage Satellite by using a web interface, command-line interface, and API. You can use API to create custom workflows or task automation.

Red Hat Satellite Server can use the *Security Content Automation Protocol (SCAP)* to define security policies and monitor Satellite clients for policy compliance. You can use Satellite to schedule recurring compliance auditing and reporting on all registered hosts. SCAP enables security administrators to use a single interface to manage, monitor, and remediate groups of hosts based on the organization's compliance requirements.

Integrating OpenSCAP with Red Hat Satellite

Red Hat Satellite provides default SCAP content for registered hosts based on the Red Hat Enterprise Linux version. The Satellite administrator can either create SCAP content or upload SCAP content from external sources. The SCAP content contains the *Extensible Checklist Configuration Description Format (XCCDF)* profile that defines the rules to be evaluated against a host or host group.

In Red Hat Satellite, a scheduled audit is referred to as a *compliance policy*. A compliance policy is a scheduled task that verifies the specified hosts or host groups for compliance against an XCCDF profile. The schedule is specified in the compliance policy on Satellite Server, but the scans are performed on the hosts. Upon completion of a compliance scan, an *Asset Reporting File (ARF)* is generated in XML format and uploaded to Satellite Server. The security administrator can then view these reports from the compliance policy dashboard.

Installing the OpenSCAP Plug-in for Red Hat Satellite Server

You must install the OpenSCAP plug-in on your Red Hat Satellite Server to integrate OpenSCAP support. The OpenSCAP plug-in provides OpenSCAP controls from the Satellite web interface. These controls are located under the **Hosts** menu in the **Compliance** section.

The default installation of Red Hat Satellite enables the OpenSCAP plug-in.

Uploading OpenSCAP Content to Satellite Server

After configuring the plug-in, but before you create a compliance policy and apply it to a host, you must upload the default OpenSCAP content to your Satellite Server. You can also upload custom SCAP content that is provided by other sources. Note that the available data streams depend on the operating system version on which Satellite runs.

You must run the following command on your Satellite Server to upload the default OpenSCAP content to it.

```
[root@satellite ~]# hammer scap-content bulk-upload --type default
Errors:

Uploaded Scap Contents:

Scap Contents uploaded.
```

Use the `hammer scap-content list` command to list the SCAP contents.

```
[root@satellite ~]# hammer scap-content list --fields Id,Title
---|-----
ID | TITLE
---|-----
1  | Red Hat firefox default content
2  | Red Hat rhel6 default content
3  | Red Hat rhel7 default content
4  | Red Hat rhel8 default content
---|-----
```

To view the SCAP content that is uploaded to Satellite Server by using the web UI, follow these steps:

1. Log in to the Satellite web UI.
2. Navigate to **Hosts > SCAP contents**. The **SCAP Contents** page lists the default SCAP contents.

You can use the Satellite web UI to upload an individual SCAP data stream file as SCAP content. To upload your own SCAP content to the Satellite web UI, follow these steps:

1. Log in to the Satellite web UI.
2. Navigate to **Hosts > SCAP contents**.
3. Click **Upload New SCAP Content**.
4. On the **File Upload** tab, click **Browse** to upload a SCAP data stream file.
5. Click **Submit**.

Preparing Satellite Server for OpenSCAP Scans

Red Hat Satellite Server uses Ansible and Puppet to manage compliance policies on its clients. This course focuses on Ansible Automation Platform for managing compliance policies. Ansible is enabled by default on Satellite Server.

In Satellite, you import Ansible roles to help automate routine tasks. You must import the `theforeman.foreman_scap_client` Ansible role to prepare Satellite Server. You must also import the Ansible variables that are associated with the `theforeman.foreman_scap_client` Ansible role.

Importing OpenSCAP Ansible Roles to Satellite Server

Use the `hammer ansible roles import` command to import the Ansible role:

```
[root@satellite ~]# hammer ansible roles import --organization 'Operations' \
--role-names 'theforeman.foreman_scap_client' --proxy-id 1
Result:
The following ansible roles were changed
Imported:
1) theforeman.foreman_scap_client
```

You can also use the Satellite web UI to import the Ansible role.

1. Select the appropriate Organization and Location.
2. Then click **Configure > Ansible > Roles**.
3. Click **Import from satellite.lab.example.com** and then select the `theforeman.foreman_scap_client` Ansible role.
4. Click **Submit**.

Importing OpenSCAP Ansible Variables to Satellite Server

Use the `hammer ansible variables import` command to import the Ansible variables:

```
[root@satellite ~]# hammer ansible variables import --organization 'Operations' \
--proxy-id 1
Result:
The following ansible variables were changed
Imported:
1) foreman_scap_client_state
2) foreman_scap_client_package
3) foreman_scap_client_server
4) foreman_scap_client_port
5) foreman_scap_client_policies
6) foreman_scap_client_oval_policies
7) foreman_scap_client_ca_cert_path
8) foreman_scap_client_host_cert_path
9) foreman_scap_client_host_private_key_path
10) foreman_scap_client_release
11) foreman_scap_client_repo_url
12) foreman_scap_client_apt_repo_url
13) foreman_scap_client_repo_state
14) foreman_scap_client_repo_key
```

```
15)foreman_scap_client_repo_gpg  
16)foreman_scap_client_cron_template  
17)foreman_scap_client_cron_splay_seed  
18)foreman_scap_client_cron_splay  
19)foreman_scap_client_fetch_remote_resources  
20)foreman_scap_client_http_proxy_server  
21)foreman_scap_client_http_proxy_port  
22)foreman_scap_client_timeout  
23)foreman_scap_client_ciphers
```

You can view the imported Ansible variables in the Satellite web UI.

1. Select the appropriate **Organization** and **Location**.
2. Then click **Configure > Ansible > Variables** to view the list of imported Ansible variables.



References

For more information, refer to the *Managing Security Compliance* chapter in the *Administering Red Hat Satellite* guide at

https://access.redhat.com/documentation/en-us/red_hat_satellite/6.11/html/administering_red_hat_satellite/managing_security_compliance_admin

► Guided Exercise

Configuring Red Hat Satellite for OpenSCAP

Configure an existing Red Hat Satellite Server to perform OpenSCAP scans.

Outcomes

- Configure an existing Satellite Server to import Ansible roles and variables.
- Push OpenSCAP content to the registered host and perform OpenSCAP scans.

Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

```
[student@workstation ~]$ lab start compliance-configuring
```

Instructions

- On the `workstation` machine, open a web browser and navigate to `https://satellite.lab.example.com`. Log in as the `admin` user with `redhat` as the password.
- Verify that a host group named `org-hostgroup1` exists in the `Operations` organization.
 - Set Satellite Server to use the `Operations` organization. Navigate to `Default Organization` and select `Operations`.
 - Navigate to `Configure > Host groups`. Verify that the `org-hostgroup1` host group exists.
 - Click the `org-hostgroup1` link to open the group for editing.Ensure that the following fields are correctly configured:

Ansible Configuration

Field	Value
Content Source	<code>satellite.lab.example.com</code>
Lifecycle Environment	Production
Content View	RHEL9-Content
OpenSCAP Capsule	<code>satellite.lab.example.com</code>

- Import the Ansible role and Ansible variables to Satellite Server.

- 3.1. Navigate to **Configure > Ansible Roles** and click **Import from satellite.lab.example.com**.
 - 3.2. In the **Changed Ansible roles** section, select the **theforeman.foreman_scap_client** role and click **Submit**.
 - 3.3. Navigate to **Configure > Ansible Variables**. Verify that the variables have been imported.
- 4. Import the **theforeman.foreman_scap_client** Ansible role to the **org-hostgroup1** host group.
- 4.1. Navigate to **Configure > Host Groups** and click **org-hostgroup1**.
 - 4.2. On the **Edit org-hostgroup1** page, select the **Ansible Roles** tab.
 - 4.3. Click the plus sign (+) to the right of **theforeman.foreman_scap_client** role to import the Ansible role. Then click **Submit** to import the role.
- 5. Upload the default OpenSCAP content to the Satellite Server database.
- 5.1. Log in to the **satellite** machine as the **student** user. Change to the **root** user. Use **student** as the password.

```
[student@workstation ~]$ ssh student@satellite
[student@satellite ~]$ sudo -i
[sudo] password for student: student
[root@satellite ~]#
```

- 5.2. Use the **hammer scap-content bulk-upload** command to upload the default OpenSCAP content to Satellite Server.

```
[root@satellite ~]# hammer scap-content bulk-upload --type default
Errors:

Uploaded Scap Contents:

Scap Contents uploaded.
```

- 5.3. Verify that the default OpenSCAP content is uploaded to the Satellite Server database. Use the **hammer scap-content list** command to list the OpenSCAP content that is present on the Satellite Server database.

```
[root@satellite ~]# hammer scap-content list --fields ID,Title
--|-----
ID | TITLE
--|-----
1  | Red Hat firefox default content
2  | Red Hat rhel6 default content
3  | Red Hat rhel7 default content
4  | Red Hat rhel8 default content
--|-----
```

- 5.4. Return to the **workstation** machine as the **student** user.

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

- 6. View the OpenSCAP content from the Satellite Server web UI.

Navigate to **Hosts > SCAP Contents**. The **SCAP Contents** page lists the default SCAP content.

Finish

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

```
[student@workstation ~]$ lab finish compliance-configuring
```

Scan OpenSCAP Compliance with Red Hat Satellite

Objectives

- Perform OpenSCAP scans of registered systems from the Red Hat Satellite Server web UI and evaluate the results of those scans.

Performing OpenSCAP Scans with Red Hat Satellite

You can configure Red Hat Satellite to centrally manage, run, and analyze OpenSCAP scans for all hosts in a Satellite host group. A basic workflow might include the following steps:

- Assign roles to the users of Satellite Server to grant them permission to manage compliance policies, run OpenSCAP scans, create OpenSCAP reports, and view OpenSCAP reports.
- Create a compliance policy for the host group that specifies which SCAP content and XCCDF profile to use.
- Create a schedule for when the compliance policy runs.
- Run the first OpenSCAP scan manually, or wait for the automatic scan to complete on all hosts.
- Use the Satellite Server web UI to review the results of the scan in the compliance policy dashboard, and to investigate detailed reports for any noncompliant hosts.
- Remediate issues and periodically review the results of subsequent OpenSCAP scans in the Satellite Server web UI.

Satellite User Permissions for OpenSCAP

Compliance scans from the Satellite Server web UI require the administrator to create users with specific *roles*. Roles define a set of permissions and access levels. Each role contains one or more permission filters that specify the actions that are allowed for the role. Red Hat Satellite provides a set of predefined roles for managing compliance.

The following table describes some predefined roles that are required for OpenSCAP scans:

Predefined Roles in Satellite Server

Role	Permissions provided by role
Compliance manager	View, create, edit, and delete SCAP content files, compliance policies, and tailoring files. View compliance reports.
Compliance viewer	View compliance reports.
Create ARF report	Create compliance reports.
Remote Execution Manager	A role with full remote execution permissions, including modifying job templates. This role is required to manually run an OpenSCAP scan from Satellite Server.

Managing Compliance Policies

In Red Hat Satellite, a compliance policy is a scheduled task to scan a host or host group for compliance with a specific XCCDF profile from SCAP content. The compliance policy is configured on Satellite Server by users with appropriate roles, but the scan is performed locally by each host. After a host runs the compliance scan, it uploads the scan results to Satellite Server in Asset Reporting File (ARF) format by using the `foreman_scap_client` command.

A user with the `Create ARF report` role can instruct a host to create a scan report and transfer it to Satellite Server. A user with the `Compliance manager` or `Compliance viewer` role can view reports from the compliance report dashboard. Only the user with the `Compliance manager` role can manage compliance policies and SCAP content.

Creating Compliance Policies

You can use the Satellite Server web UI to define a compliance policy. A compliance policy includes the following parameters:

- The SCAP content to use
- The XCCDF profile from the SCAP content
- The host groups that should comply with this policy
- The scheduled interval at which the audit shall occur

The following steps describe the process for creating a compliance policy on Satellite Server.

1. In the Satellite Server web UI, log in as a user with a `Compliance manager` role. From the `Organizations` list, select the organization for which you are creating the new policy.



Note

At some resolutions, the `Organizations` list displays in a sidebar menu. If the `Organizations` list does not display at the top of the UI, then navigate to `Organizations > Operations` from the sidebar menu.

2. Navigate to `Hosts > Compliance > Policies`. Click **New Policy**.
3. On the **Deployment Options** tab, select `Ansible` as the deployment option, and then click **Next**.
4. On the **Policy Attributes** tab, enter a name for the policy, an optional description, and then click **Next**.
5. On the **SCAP content** tab, choose the SCAP content and the XCCDF profile to apply, and then click **Next**.
6. On the **Schedule** tab, choose from the following list for **Period**:
 - **Weekly**: Enables you to choose the desired day of the week.
 - **Monthly**: Enables you to choose the desired day of the month.
 - **Custom**: Enables you to choose the desired time based on the cron job.
7. Click **Next**.
8. On the **Locations** tab, select the default location to move it to the **Selected items** list. Click **Next**.
9. On the **Organizations** tab, select the organization to move it to the **Selected items** list. Click **Next**.

10. On the **Hostgroups** tab, select the host group to move it to the **Selected items** list.
11. Click **Submit**.

Running Compliance Scans

Red Hat recommends that you deploy compliance policies by using Ansible. The compliance policy runs automatically based on its cron job.

In some situations, such as when testing a custom policy or verifying a newly added host, you might need to manually deploy a policy.

Use this procedure to deploy a policy on a specific host:

Override the `foreman_scap_client_fetch_remote_resources` Ansible variable with the `true` value:

1. Navigate to **Configure > Ansible > Variables**.
2. Click **Override** to enable editing of the variable value.
3. Set the `foreman_scap_client_fetch_remote_resources` variable to the `true` value and click **Submit**.

Deploy the policy on the host by using the Ansible role.

1. Navigate to **Hosts > Hosts > All Hosts** and select the checkbox of the host that you want to deploy the policy.
2. Click **Select Action** and select **Assign Compliance Policy** from the list.
3. Select **Remember hosts selection for the next bulk action**.
4. Select which policy to deploy and click **Submit**.
5. Click **Select Action** and select **Run all Ansible roles** from the list.
6. Verify the results of the role execution.

Reviewing OpenSCAP Scan Results in Satellite

Red Hat Satellite enables centralized compliance monitoring for all the hosts that it manages through a compliance policy dashboard. The compliance policy dashboard provides an overview of the number of compliant hosts and details for each host based on the rules that passed or failed during the scan. You can use this dashboard to evaluate the risks that are presented by the host, and then take corrective action to bring the host into compliance.

Red Hat Satellite assumes that different users might have different roles in the compliance scanning process. If a particular Satellite user needs to view compliance reports, then you must assign the **Compliance viewer** role to that user. If a user must work with SCAP content and tailoring files, configure compliance policies, and view the compliance reports, then you must assign the **Compliance manager** role.

Viewing the Compliance Policy Dashboard

To view the compliance policy dashboard in the Satellite web UI, navigate to **Hosts > Compliance > Policies**. In the **Actions** column, click **Dashboard** for the compliance policy that you want to verify.

The dashboard provides the following information:

- The **Host Breakdown Chart** shows the number of compliant and noncompliant hosts, based on the compliance policy.
- A statistical breakdown that lists the number of hosts that are compliant, noncompliant, have inconclusive results, or that have never been audited.
- A statistical breakdown of the number of rules that passed or failed for each host, in a tabular format.

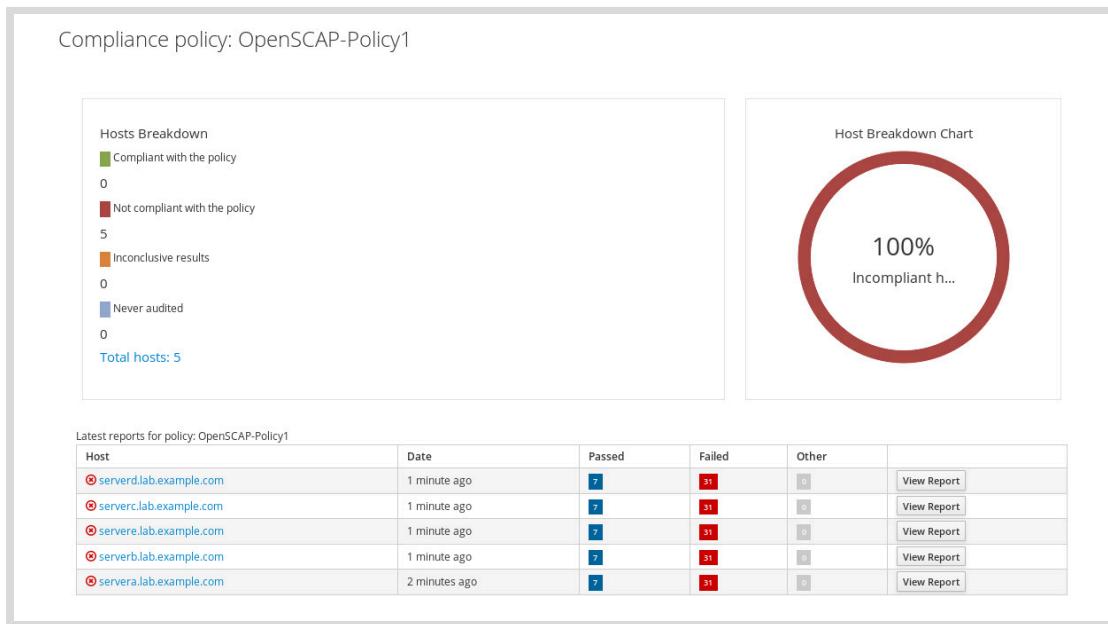


Figure 11.1: Compliance policy dashboard in Red Hat Satellite

Evaluating OpenSCAP Reports

You can access the OpenSCAP reports for every scanned host from the compliance report dashboard. You can use this report to determine and prioritize remediation efforts for any noncompliant hosts.

Viewing Compliance Reports

A compliance report is an OpenSCAP report in ARF format that is uploaded to Satellite Server after an OpenSCAP scan on the host. To list all the available reports from the Satellite web UI, navigate to **Hosts > Reports**. The compliance report dashboard lists the total number of rules that passed or failed during the scan. You can also view the detailed report for a particular host from the compliance report dashboard.

The screenshot shows the Red Hat Satellite web interface. At the top, it displays the title "RED HAT SATELLITE" and the user "org-example@Default Location". The navigation bar includes links for "Monitor", "Hosts", "Infrastructure", and "Administrator". Below the header, the host name "serverd.lab.example.com" is shown. A sub-header "Show log messages:" with a dropdown menu set to "All messages" is followed by a row of buttons: "Back", "Delete" (highlighted in red), "Host details", "View full report", "Download XML in bzip", and "Download HTML". A timestamp "Reported at 2018-08-06 12:59:26 UTC for policy OpenSCAP-Policy1 through satellite.lab.example.com" is displayed. The main content is a table with the following data:

Severity	Message	Resource	Result
Low	Ensure /var/log Located On Separate Partition	xccdf_org.ssgproject.content...	fail
Low	Ensure /var/log/audit Located On Separate Partition	xccdf_org.ssgproject.content...	fail
Medium	Disable the Automounter	xccdf_org.ssgproject.content...	pass
Medium	Ensure rsyslog Is Installed	xccdf_org.ssgproject.content...	pass
Medium	Enable rsyslog Service	xccdf_org.ssgproject.content...	pass
Low	Record attempts to alter time through adjtimex	xccdf_org.ssgproject.content...	fail
Low	Record attempts to alter time through settimofday	xccdf_org.ssgproject.content...	fail
Low	Record Attempts to Alter Time Through stime	xccdf_org.ssgproject.content...	fail
Low	Record Attempts to Alter Time Through clock_settime	xccdf_org.ssgproject.content...	fail
Low	Record Attempts to Alter the localtime File	xccdf_org.ssgproject.content...	fail
Low	Record Events that Modify User/Group Information	xccdf_org.ssgproject.content...	fail

Figure 11.2: Compliance report in Red Hat Satellite

Viewing Compliance Reports in Satellite Server

The following steps outline the process for viewing the compliance report in the Satellite web UI:

1. Log in to the Satellite web UI as a user with either the **Compliance manager** or **Compliance viewer** role.
2. Navigate to **Hosts > Reports**.
3. To open the latest report, click the link in the **Reported At** column to view the number of rules that passed or failed for the latest scan.
4. Click **View full report** to view the detailed report.

The compliance report for each system offers the same information that you get by running the `oscap xccdf eval` command manually on each machine. The advantage of using a Red Hat Satellite compliance policy to manage these scans is the scalability and the central coordination that Satellite provides. You can set up and manage scans for many systems from one central interface. You can use one central interface to review the results of any system's scan. You can delegate authority to auditors to view the results of the latest scans. Finally, you can compare scans to look for patterns of misconfiguration or common issues.



References

For more information, refer to the *Managing Security Compliance* guide at https://access.redhat.com/documentation/en-us/red_hat_satellite/6.14/html-single/managing_security_compliance/index

► Guided Exercise

Scan OpenSCAP Compliance with Red Hat Satellite

Use Red Hat Satellite to perform an OpenSCAP scan of one of your servers, and evaluate the results.

Outcomes

- Create a Red Hat Satellite compliance policy for centralized OpenSCAP scans.
- Manually trigger a compliance policy scan on a Red Hat Satellite client.
- Evaluate the compliance report for that scan in the Red Hat Satellite web UI.

Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

```
[student@workstation ~]$ lab start compliance-scan
```

Instructions

- 1. On the `serverd` machine, install the `scap-security-guide` package to get the supported SCAP content for RHEL 9.
- 1.1. Log in to the `serverd` machine as the `student` user. Change to the `root` user. Use `student` as the password.

```
[student@workstation ~]$ ssh student@serverd
[student@serverd ~]$ sudo -i
[sudo] password for student: student
[root@serverd ~]#
```

- 1.2. Install the `scap-security-guide` package on the `serverd` machine.

```
[root@serverd ~]# dnf install scap-security-guide
...output omitted...
Install 6 Packages

Total download size: 3.2 M
Installed size: 100 M
Is this ok [y/N]: y
...output omitted...
```

- 1.3. Return to the `workstation` machine.

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

- 1.4. Copy the /usr/share/xml/scap/ssg/content/ssg-rhel9-ds.xml file from the serverd machine to the Desktop directory on the workstation machine.

```
[student@workstation ~]$ scp \  
serverd:/usr/share/xml/scap/ssg/content/ssg-rhel9-ds.xml \  
Desktop/ssg-rhel9-ds.xml  
...output omitted..
```

- ▶ 2. Upload the supported SCAP content for RHEL 9 to Satellite.

- 2.1. Log in to the satellite machine as the student user. Change to the root user. Use student as the password.

```
[student@workstation ~]$ ssh student@satellite  
[student@satellite ~]$ sudo -i  
[sudo] password for student: student  
[root@satellite ~]#
```

- 2.2. Create the /usr/share/xml/scap/custom directory. Copy the Desktop/ssg-rhel9-ds.xml file from the workstation machine to the /usr/share/xml/scap/custom directory.

```
[root@satellite ~]# mkdir -p /usr/share/xml/scap/custom  
[root@satellite ~]# scp student@workstation:Desktop:ssg-rhel9-ds.xml \  
/usr/share/xml/scap/custom/  
student@workstation's password: student  
...output omitted...
```

- 2.3. From the satellite machine, use the hammer command to upload the SCAP content.

```
[root@satellite ~]# hammer scap-content bulk-upload --type directory \  
--organization 'Operations' \  
--directory /usr/share/xml/scap/custom  
...output omitted...  
Scap Contents uploaded.
```

- ▶ 3. Log out of the satellite machine.

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

- ▶ **4.** On the workstation machine, open a browser and connect to the Satellite web UI at <https://satellite.lab.example.com>. If required, accept the self-signed certificate and log in as the `admin` user with `redhat` as the password.
- ▶ **5.** In the Satellite web UI, select the **Operations** organization.
 - 5.1. Select **Operations** from the **Organizations** list.

**Note**

At some resolutions, the **Organizations** list displays in a sidebar menu. If the **Organizations** list does not display at the top, then navigate to **Organizations > Operations** from the sidebar menu.

- ▶ **6.** In the Satellite web UI, create a compliance policy named **OpenSCAP-Policy1** by using the default RHEL 9 SCAP content. Configure the policy to run every 10 minutes.
 - 6.1. Navigate to **Hosts > Compliance > Policies** and click **New Policy**.
 - 6.2. Select **Ansible** as the deployment option, and then click **Next**.
 - 6.3. On the **Policy Attributes** tab, enter **OpenSCAP-Policy1** as the name of the policy. The policy description is optional. Click **Next**.
 - 6.4. On the **SCAP Content** tab, select **rhel9** content from the **SCAP Content** list. For **XCCDF Profile**, select **[DRAFT] DISA STIG for Red Hat Enterprise Linux 9**. Click **Next**.
 - 6.5. On the **Schedule** tab, select **Custom** for **Period**. Enter *** /10 * * * *** in the **Cron line** field to run the scan every 10 minutes. Click **Next**.
 - 6.6. On the **Locations** tab, verify that **Default Location** is on the **Selected items** list. Click **Next**.
 - 6.7. On the **Organizations** tab, ensure that **Operations** is the selected organization. Click **Next**.
 - 6.8. On the **Hostgroups** tab, select **org-hostgroup1** to move it to the **Selected items** list. Click **Submit** to create the compliance policy.
- ▶ **7.** Execute the Ansible roles to set up the host for OpenSCAP revisions.
 - 7.1. Return to **Hosts > Hosts > All Hosts** and select the `serverd.lab.example.com` host checkbox.
 - 7.2. Click **Select Action** and select **Run all Ansible roles** from the list.
 - 7.3. Verify the results of the role execution.
- ▶ **8.** Run an OpenSCAP scan for the `serverd` host.
 - 8.1. Navigate to **Hosts > Hosts > All Hosts**.
 - 8.2. Select the checkbox for the `serverd` host.
 - 8.3. Click **Select Action > Schedule Remote Job**.

- 8.4. Select the OpenSCAP job category and the Run OpenSCAP scans job template, and then click **Run on selected hosts**.
- ▶ **9.** View the results of the OpenSCAP-Policy1 OpenSCAP scan.
 - 9.1. Navigate to **Hosts > Compliance > Policies**.
 - 9.2. Click **Dashboard** for the OpenSCAP-Policy1 policy.
 - 9.3. Click **View Report** for the serverd host.
 - 9.4. Browse through the scan results.

Finish

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

```
[student@workstation ~]$ lab finish compliance-scan
```

Customize the OpenSCAP Policy in Red Hat Satellite

Objectives

- Apply a tailoring file to a SCAP profile in Red Hat Satellite, and use the customized SCAP profile to scan registered servers.

Customizing SCAP Profiles in Red Hat Satellite

Organizations sometimes need to adjust a standard security policy, making it stricter or more lenient based on its actual compliance requirements. SCAP tailoring files enable compliance managers to customize a profile without writing new SCAP content. You can create these tailoring files in SCAP Workbench and save them as XCCDF profiles.

Red Hat Satellite 6.3 introduced support for tailoring files. This feature enables users to upload a tailoring file to customize a compliance policy. You can assign the uploaded tailoring file to an existing SCAP profile when creating or updating a compliance policy.



Important

Red Hat Satellite does not have an interface to create or edit tailoring files. A compliance manager should create the tailoring file in SCAP Workbench, and then upload the file to Satellite Server.

Uploading a Tailoring File

After you create a tailoring file in SCAP Workbench, save it in XCCDF Customization format. Then, upload the file to the Red Hat Satellite Server that manages the compliance policy for your scans. You should have already installed the corresponding XCCDF profile for the tailoring file on Satellite Server.

Uploading a Tailoring File to Red Hat Satellite

The following steps outline the process for uploading a tailoring file to Satellite Server:

1. Log in to the Satellite Server web UI as a user with the **Compliance manager** role.
2. Navigate to **Hosts > Tailoring Files** and then click **New Tailoring File**.
3. On the **Upload new Tailoring File** page, enter a name in the **Name** field. Click **Browse** to upload the tailoring file.
4. Click **Submit**.

Assigning a Tailoring File to a Compliance Policy

A tailoring file contains one or more XCCDF profiles. You can assign only one tailoring file to a compliance policy. Any change to a compliance policy propagates to each client when its agent connects to Satellite Server.

Assigning a Tailoring File by using the Satellite Web Interface

The following steps outline the process for assigning a tailoring file to a compliance policy on Satellite Server:

1. Log in to the Satellite web UI as a user with the `Compliance manager` role.
2. Navigate to `Hosts > Compliance > Policies`. Click the name of the policy that you wish to edit.
Alternatively, click `New Policy` or `New Compliance Policy` to create a compliance policy.
3. On the `SCAP Content` tab, choose the required tailoring file from the `Tailoring File` list.
4. Select `XCCDF Profile` in `Tailoring File` from the list. Click `Submit`.

Executing a Compliance Scan with a Customized Compliance Policy

The agent that runs on each host that is managed by Satellite fetches the change in the compliance policy.

The `/etc/foreman_scap_client/config.yaml` file contains information about the tailoring file and the XCCDF profile that OpenSCAP uses for the compliance scan.

```
# DO NOT EDIT THIS FILE MANUALLY
# IT IS MANAGED BY ANSIBLE
# ANY MANUAL CHANGES WILL BE LOST ON THE NEXT ANSIBLE EXECUTION
...output omitted...

# policy (key is id as in Foreman)
1:
:profile: xccdf_com.example_profile_mycustom-rhel9
:content_path: /var/lib/openscap/content/c7ec...4395.xml
# Download path
# A path to download SCAP content from proxy
:download_path: /compliance/policies/1/content/c7ec...4395
:tailoring_path: /var/lib/openscap/tailoring/5013...5164.xml
:tailoring_download_path: /compliance/policies/1/tailoring/5013...5164
```

In the previous output, the `xccdf_com.example_profile_mycustom-rhel9` profile is the XCCDF tailoring profile. The `tailoring_path` variable defines the location of the tailoring file on the SCAP client. The `tailoring_download_path` variable defines the download location of the tailoring file from Satellite Server.

The compliance scan runs based on the cron job that is defined in the compliance policy. To run the scan manually, you can either use remote execution from Satellite Server or use the `foreman_scap_client` command. The agent uploads the result of the scan to Satellite Server.



References

For more information, refer to the *Configuring SCAP Contents* chapter in the *Managing Security Compliance* guide at
https://access.redhat.com/documentation/en-us/red_hat_satellite/6.14/html-single/managing_security_compliance/index#Configuring_SCAP_Contents_security-compliance

► Guided Exercise

Customize OpenSCAP Policy in Red Hat Satellite

Use a tailoring file to customize the compliance policy, re-scan your hosts by using the Red Hat Satellite web UI, and evaluate the results.

Outcomes

- Upload a tailoring file to Satellite Server.
- Assign the tailoring file to a compliance policy.
- Perform compliance scans by using a compliance policy that is customized with a tailoring file.
- View and download the results of a compliance scan.

Before You Begin

As the **student** user on the **workstation** machine, use the **lab** command to prepare your environment for this exercise, and to ensure that all required resources are available.

```
[student@workstation ~]$ lab start compliance-customize
```

Instructions

- ▶ 1. On the **workstation** machine, open a web browser and navigate to `https://satellite.lab.example.com`. Log in as the **admin** user with **redhat** as the password. Use the **Operations** organization as the default organization.
 - 1.1. From the **workstation** machine, open a web browser and navigate to `https://satellite.lab.example.com`. Accept the self-signed certificate and log in as the **admin** user with **redhat** as the password.
 - 1.2. Set Satellite Server to use the **Operations** organization. Navigate to **Organization** and select **Operations**.



Note

At some resolutions, the **Organizations** list displays in a sidebar menu. If the **Organizations** list does not display at the top, then navigate to **Organizations > Operations** from the sidebar menu.

- ▶ 2. Add the `/home/student/RH415-tailoring.xml` file to Satellite. Name the tailoring file `compliance-customize` in Satellite.
 - 2.1. In the Satellite Server web UI, navigate to **Hosts > Compliance > Tailoring Files**. Click **New Tailoring File** to add a new tailoring file.

- 2.2. On the **Upload new Tailoring File** page, enter **compliance-customize** for the **Name**. Click **Browse** to upload the `/home/student/RH415-tailoring.xml` tailoring file, and then click **Submit**.
- ▶ **3.** Create the **compliance-customize** compliance policy and use the **compliance-customize** tailoring file. Use the **rhel9** content SCAP content. Use a monthly period with the first day of the month. Enable the scan for the **org-hostgroup1** hostgroup.
 - 3.1. Navigate to **Hosts > Compliance > Policies**.
 - 3.2. Click **New Compliance Policy**.
 - 3.3. Select the **Ansible** radio button and click **Next**.
 - 3.4. Enter **compliance-customize** in the **Name** field and click **Next**.
 - 3.5. Select the **rhel9** content value for **SCAP Content** and the **compliance-customize** value for **Tailoring File**, and then click **Next**.
 - 3.6. Select the **Monthly** value for **Period**, select the **1** value for **Day of Month**, and then click **Next**.
 - 3.7. Verify that the **Default Location** value is selected and click **Next**.
 - 3.8. Verify that the **Operations** value is selected and click **Next**.
 - 3.9. Select the **org-hostgroup1** hostgroup and click **Submit**.
- ▶ **4.** Execute the Ansible roles to set up the host for OpenSCAP revisions.
 - 4.1. Return to **Hosts > Hosts > All Hosts** and select the `serverd.lab.example.com` host checkbox.
 - 4.2. Click **Select Action** and select **Run all Ansible roles** from the list.
 - 4.3. Verify the results of the role execution.
- ▶ **5.** Run an OpenSCAP scan for the `serverd` host.
 - 5.1. Navigate to **Hosts > Hosts > All Hosts**.
 - 5.2. Select the checkbox for the `serverd` host.
 - 5.3. Click **Select Action > Schedule Remote Job**.
 - 5.4. Select the OpenSCAP job category and the **Run OpenSCAP scans** job template, then click **Run on selected hosts**.
- ▶ **6.** View and download the results of the **compliance-customize** OpenSCAP scan.
 - 6.1. Navigate to **Hosts > Compliance > Policies**.
 - 6.2. Click **Dashboard** for the **compliance-customize** policy.
 - 6.3. Click **View Report** for the `serverd` host.
 - 6.4. Browse through the results to find which rules the `serverd` machine is compliant with.

- 6.5. Click **Download XML in bzip** to download the results.

Finish

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

```
[student@workstation ~]$ lab finish compliance-customize
```

► Lab

Automating Compliance with Red Hat Satellite

Use Red Hat Satellite to scan all of your servers for compliance with a customized OpenSCAP policy and evaluate the results.

Outcomes

- Create a Red Hat Satellite compliance policy that is customized with a tailoring file.
- Initiate an OpenSCAP scan on one or more hosts from Red Hat Satellite by using a compliance policy.
- Evaluate the results of a compliance policy's OpenSCAP scans in the Red Hat Satellite web UI.

Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

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

Instructions

1. From the `workstation` machine, connect to the Satellite web UI at `https://satellite.lab.example.com`. If prompted, accept the self-signed certificate and log in as the `admin` user with `redhat` as the password.
2. Upload a new tailoring file named `ComplianceLab-TailoringFile` to customize the default Standard System Security profile for RHEL 9 SCAP content. Upload the tailoring file from `/home/student/RH415-tailoring.xml` to Satellite.
3. Create a compliance policy named `ComplianceLab-Policy1` by using the default RHEL 9 SCAP content. Choose the [DRAFT] DISA STIG for Red Hat Enterprise Linux 9 XCCDF profile and the `ComplianceLab-TailoringFile` tailoring file. The policy should execute weekly on Sunday and should be deployed by using Ansible. Use the following table to specify the other fields for the compliance policy:

Compliance Policy Parameters

Field	Value
Deployment	Ansible
Locations	Default Location
Organizations	Operations
Hostgroups	org-hostgroup1

4. Assign the `ComplianceLab-Policy1` policy to all hosts.
5. Manually run the compliance scan on the following hosts to update the clients with the new compliance policy.
 - `servera.lab.example.com`
 - `serverb.lab.example.com`
6. View and download the results of the `ComplianceLab-Policy1` OpenSCAP scan.

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 compliance-review
```

Finish

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

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

► Solution

Automating Compliance with Red Hat Satellite

Use Red Hat Satellite to scan all of your servers for compliance with a customized OpenSCAP policy and evaluate the results.

Outcomes

- Create a Red Hat Satellite compliance policy that is customized with a tailoring file.
- Initiate an OpenSCAP scan on one or more hosts from Red Hat Satellite by using a compliance policy.
- Evaluate the results of a compliance policy's OpenSCAP scans in the Red Hat Satellite web UI.

Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

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

Instructions

1. From the workstation machine, connect to the Satellite web UI at `https://satellite.lab.example.com`. If prompted, accept the self-signed certificate and log in as the `admin` user with `redhat` as the password.
2. Upload a new tailoring file named `ComplianceLab-TailoringFile` to customize the default Standard System Security profile for RHEL 9 SCAP content. Upload the tailoring file from `/home/student/RH415-tailoring.xml` to Satellite.
 - 2.1. Select Operations from the Organizations list.



Note

At some resolutions, the **Organizations** list displays in a sidebar menu. If the **Organizations** list does not display at the top, then navigate to **Organizations > Operations** from the sidebar menu.

- 2.2. In the Satellite web UI, navigate to **Hosts > Compliance > Tailoring Files**. Click **Upload New Tailoring file** to upload a new tailoring file.
- 2.3. On the **Upload new Tailoring File** page, enter `ComplianceLab-TailoringFile` in the **Name** field. Click **Browse** to upload the `/home/student/RH415-tailoring.xml` tailoring file. Click **Submit**.
3. Create a compliance policy named `ComplianceLab-Policy1` by using the default RHEL 9 SCAP content. Choose the **[DRAFT] DISA STIG for Red Hat Enterprise Linux 9**

XCCDF profile and the `ComplianceLab-TailoringFile` tailoring file. The policy should execute weekly on Sunday and should be deployed by using Ansible. Use the following table to specify the other fields for the compliance policy:

Compliance Policy Parameters

Field	Value
Deployment	Ansible
Locations	Default Location
Organizations	Operations
Hostgroups	org-hostgroup1

- 3.1. Navigate to **Hosts > Compliance > Policies**.
Click **New Compliance Policy**.
- 3.2. Select **Ansible** as the deployment option, and then click **Next**.
- 3.3. On the **New Compliance Policy** page, enter `ComplianceLab-Policy1` as the name of the policy. The policy description is optional. Click **Next**.
- 3.4. On the **SCAP Content** tab, set the following values:
 - Select `rhel9` content from the **SCAP Content** list.
 - For **Tailoring File**, select `ComplianceLab-TailoringFile`.
 - The **XCCDF Profile in Tailoring File** list automatically sets the **[DRAFT] DISA STIG for Red Hat Enterprise Linux 9 [CUSTOMIZED]** XCCDF profile, because only one profile is included in the tailoring file. Click **Next**.
- 3.5. On the **Schedule** tab, choose **Weekly** for **Period**. For **Weekday**, select **Sunday**. Click **Next**.
- 3.6. On the **Locations** tab, select **Default Location** to move it to the **Selected items** list. Click **Next**.
- 3.7. On the **Organizations** tab, select **Operations** to move it to the **Selected items** list. Click **Next**.
- 3.8. On the **Hostgroups** tab, select `org-hostgroup1` to move it to the **Selected items** list. Click **Submit** to create the compliance policy.
4. Assign the `ComplianceLab-Policy1` policy to all hosts.
 - 4.1. Navigate to **Hosts > Hosts > All Hosts**, and then select the checkboxes for the `servera.lab.example.com` and `serverb.lab.example.com` hosts.
 - 4.2. Click **Select Action** and select **Assign Compliance Policy** from the list.
 - 4.3. Select **Remember hosts selection for the next bulk action**.
 - 4.4. Select `ComplianceLab-Policy1` from the list and click **Submit**.

- 4.5. Select the checkboxes for the `servera.lab.example.com` and `serverb.lab.example.com` hosts. Click **Select Action**, and then select **Run all Ansible roles** from the list.
5. Manually run the compliance scan on the following hosts to update the clients with the new compliance policy.
 - `servera.lab.example.com`
 - `serverb.lab.example.com`
- 5.1. Navigate to **Hosts > Hosts > All Hosts**, and then select the checkboxes for the `servera.lab.example.com` and `serverb.lab.example.com` hosts.
- 5.2. Click **Select Action > Schedule Remote Job**.
- 5.3. Select the **OpenSCAP** job category and the **Run OpenSCAP scans** job template, and then click **Run on selected hosts**.
6. View and download the results of the `ComplianceLab-Policy1` OpenSCAP scan.
 - 6.1. Navigate to **Hosts > Compliance > Policies**.
 - 6.2. Click **Dashboard** for the `ComplianceLab-Policy1` scan.
 - 6.3. Click **View Report** for the `servera` host.
 - 6.4. Browse through the results to find which rules the `servera` machine is compliant with.
 - 6.5. Click **Download XML in bzip** to download the results.

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 compliance-review
```

Finish

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

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

Summary

- You can use Red Hat Satellite compliance policies to centrally manage and review the results of OpenSCAP scans on registered clients.
- A Satellite compliance policy is a named, scheduled task that scans specific hosts for compliance with an OpenSCAP XCCDF profile.
- You must upload OpenSCAP content to Red Hat Satellite Server before you can use it in a compliance policy.
- Clients update their compliance policy configurations with Ansible. The compliance policy that is running on Satellite instructs hosts to run OpenSCAP scans locally, and the hosts upload the results to Satellite.
- The compliance policy dashboard in the Satellite Server web UI provides an overview of compliant and noncompliant hosts, and links to detailed OpenSCAP compliance reports for each host.
- You can customize a compliance policy by creating an OpenSCAP tailoring file by using the SCAP Workbench tool and uploading the tailoring file to Satellite Server.

Chapter 12

Comprehensive Review

Goal

Review tasks from Red Hat Security: Linux in Physical, Virtual, and Cloud.

Sections

- Comprehensive Review

Lab

- Protecting Data with LUKS and NBDE
- Restricting USB Device Access
- Recording Events and Monitoring File-system Changes with PAM, Audit, and AIDE
- Mitigating Risk with SELinux
- Managing Compliance with OpenSCAP and Ansible

Comprehensive Review

Objectives

After completing this section, you should have reviewed and refreshed the knowledge and skills that you learned in *Red Hat Security: Linux in Physical, Virtual, and Cloud*.

Reviewing Red Hat Security: Linux in Physical, Virtual, and Cloud

Before beginning the comprehensive review for this course, you should be comfortable with the topics covered in each chapter. Do not hesitate to ask the instructor for extra guidance or clarification on these topics.

Chapter 1, Managing Security and Risk

Define and implement strategies to manage security on Red Hat Enterprise Linux systems.

Chapter 2, Automating Configuration and Remediation with Ansible

Remediate configuration and security issues automatically with Ansible Playbooks.

Chapter 3, Protecting Data with LUKS and NBDE

Encrypt data on storage devices with Linux Unified Key Setup (LUKS), and use Network-bound Disk Encryption (NBDE) to manage automatic decryption when servers are booted.

Chapter 4, Restricting USB Device Access

Protect systems from rogue USB device access with USBGuard.

Chapter 5, Controlling Authentication with PAM

Manage authentication, authorization, session settings, and password controls by configuring Pluggable Authentication Modules (PAM).

Chapter 6, Recording System Events with Audit

Record and inspect system events relevant to security by using the Linux kernel's Audit system and supporting tools.

Chapter 7, Monitoring File-system Changes

Detect and analyze changes to a server's file systems and their contents by using AIDE.

Chapter 8, Mitigating Risk with SELinux

Improve security and confinement between processes by using SELinux and advanced SELinux techniques and analysis.

Chapter 9, Managing Compliance with OpenSCAP

Evaluate and remediate a server's compliance with security policies by using OpenSCAP.

Chapter 10, Analyzing and Remediating Issues with Red Hat Insights

Identify, detect, and correct common issues and security vulnerabilities with Red Hat Enterprise Linux systems by using Red Hat Insights.

Chapter 11, Automating Compliance with Red Hat Satellite

Automate and scale OpenSCAP compliance checks by using Red Hat Satellite.

► Lab

Protecting Data with LUKS and NBDE

Create and encrypt a storage device with LUKS and configure it to automatically decrypt at boot time securely by using NBDE.

Outcomes

- Encrypt a partition with LUKS.
- Decrypt a LUKS partition by using multiple Tang servers.
- Rotate the keys for Tang servers.

Before You Begin

If you did not reset your **workstation** and **server** machines at the end of the last chapter, then save any work you want to keep from earlier exercises on those machines, and reset them now.

As the **student** user on the **workstation** machine, use the **lab** command to prepare your environment for this exercise, and to ensure that all required resources are available.

```
[student@workstation ~]$ lab start compreview-nbde
```

Specifications

- Create and mount the encrypted `/dev/vdb1` partition and file system on the **serverb** machine with the following parameters:

Parameter	Value
Size	Full Disk (1 GB)
Encryption Password	redhatRHT
Mapper Name	storage
File System	XFS
Mount Point	/encrypted

- Create the `encryption-test.txt` text file in the `/encrypted` directory.
- Edit the `/home/student/RH415/labs/comprevie-nbde/nbde_setup.yml` Ansible Playbook with the following parameters. This playbook associates the LUKS-encrypted partition to the Tang servers.

Parameter	Value
Tang Servers	serverc, serverd

Parameter	Value
Partition	/dev/vdb1
Minimum Threshold	2
Encryption Password	redhatRHT

- Use the /home/student/RH415/labs/comprevew-nbde/inventory inventory file when applying the Ansible Playbook.
- Configure the encrypted partition to automatically decrypt and mount the /encrypted directory on the serverb machine at boot time.
- Rotate the keys for the Tang server on the serverc and serverd machines.

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-nbde
```

Finish

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

```
[student@workstation ~]$ lab finish comprevew-nbde
```

► Solution

Protecting Data with LUKS and NBDE

Create and encrypt a storage device with LUKS and configure it to automatically decrypt at boot time securely by using NBDE.

Outcomes

- Encrypt a partition with LUKS.
- Decrypt a LUKS partition by using multiple Tang servers.
- Rotate the keys for Tang servers.

Before You Begin

If you did not reset your **workstation** and **server** machines at the end of the last chapter, then save any work you want to keep from earlier exercises on those machines, and reset them now.

As the **student** user on the **workstation** machine, use the **lab** command to prepare your environment for this exercise, and to ensure that all required resources are available.

```
[student@workstation ~]$ lab start compreview-nbde
```

- Verify that the **/dev/vdb** disk is available on the **serverb** machine.
 - Open a new terminal window and log in to the **serverb** machine as the **student** user.

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

- Change to the **root** user. Use **student** as the password.

```
[student@serverb ~]$ sudo -i  
[sudo] password for student: student  
[root@serverb ~]#
```
- Verify that the **/dev/vdb** disk is available and has no partitions.

```
[root@serverb ~]# parted -l
Error: /dev/vdb: unrecognised disk label
Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 1074MB
Sector size (logical/physical): 512B/512B
Partition Table: unknown
Disk Flags:
...output omitted...
```

2. Create a 1GB partition on the /dev/vdb disk on the serverb machine.
 - 2.1. Use the `parted` command to create a partition on the /dev/vdb disk on the serverb machine.

```
[root@serverb ~]# parted /dev/vdb \
mklabel msdos mkpart primary xfs 1M 1G
Information: You may need to update /etc/fstab.
```

- 2.2. Verify that the partition is available.

```
[root@serverb ~]# parted /dev/vdb print
Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 1074MB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start   End     Size    Type      File system  Flags
 1       1049kB  1074MB  1073MB  primary
```

3. Encrypt the /dev/vdb1 partition with LUKS. Use `redhatRHT` as the password for the encryption.
 - 3.1. Use the `cryptsetup luksFormat` command to encrypt the /dev/vdb1 partition with LUKS.

```
[root@serverb ~]# cryptsetup luksFormat /dev/vdb1

WARNING!
=====
This will overwrite data on /dev/vdb1 irrevocably.

Are you sure? (Type 'yes' in capital letters): YES
Enter passphrase for /dev/vdb1: redhatRHT
Verify passphrase: redhatRHT
```

4. Use `storage` as the name for the partition.
 - 4.1. Use the `cryptsetup open` command to use `storage` as the name for the storage partition.

```
[root@serverb ~]# cryptsetup open /dev/vdb1 storage  
Enter passphrase for /dev/vdb1: redhatRHT
```

- 4.2. Verify that the /dev/mapper/storage partition is now available.

```
[root@serverb ~]# ls /dev/mapper/storage  
/dev/mapper/storage
```

5. Create an XFS file system on the storage partition, and then mount the new file system on the /encrypted directory. Create the encryption-test.txt text file in the /encrypted directory.

- 5.1. Create an XFS file system on the /dev/mapper/storage partition.

```
[root@serverb ~]# mkfs.xfs /dev/mapper/storage  
meta-data=/dev/mapper/storage     isize=512    agcount=4, agsize=64448 blks  
                                  =                      sectsz=512   attr=2, projid32bit=1  
                                  =                      crc=1       finobt=1, sparse=1, rmapbt=0  
                                  =                      reflink=1  bigtime=1 inobtcount=1  
data     =                      bsize=4096   blocks=257792, imaxpct=25  
        =                      sunit=0      swidth=0 blks  
naming   =version 2           bsize=4096   ascii-ci=0, ftype=1  
log      =internal log        bsize=4096   blocks=1566, version=2  
        =                      sectsz=512  sunit=0 blks, lazy-count=1  
realtime =none                extsz=4096   blocks=0, rtextents=0
```

- 5.2. Create the /encrypted directory.

```
[root@serverb ~]# mkdir /encrypted
```

- 5.3. Mount the /dev/mapper/storage partition on the /encrypted directory.

```
[root@serverb ~]# mount -t xfs /dev/mapper/storage /encrypted
```

- 5.4. Verify that the /dev/vdb1 partition is correctly mounted.

```
[root@serverb ~]# mount | grep /encrypted  
/dev/mapper/storage on /encrypted type xfs  
(rw, relatime, seclabel, attr2, inode64, logbufs=8, logbsize=32k, noquota)
```

- 5.5. Create the encryption-test.txt text file in the /encrypted directory.

```
[root@serverb ~]# touch /encrypted/encryption-test.txt
```

6. Unmount the file system and lock the storage partition.

- 6.1. Unmount the file system that is mounted on the /encrypted directory.

```
[root@serverb ~]# umount /encrypted
```

6.2. Lock the storage partition.

```
[root@serverb ~]# cryptsetup close storage
```

7. Return to the terminal window on the **workstation** machine. Edit the `/home/student/RH415/labs/comprevew-nbde/nbde_setup.yml` Ansible Playbook to associate the LUKS-encrypted partition on the `/dev/vdb1` device with the Tang servers on the `serverc` and `serverd` machines. Use the `/home/student/RH415/labs/comprevew-nbde/inventory` file as the inventory for running the Ansible Playbook. Configure SSS encryption so that at least two Tang servers must be available to decrypt the partition.

- 7.1. Use a text editor to edit the `/home/student/RH415/labs/comprevew-nbde/nbde_setup.yml` Ansible Playbook. Add the following content:

```
---
- hosts: servers
  become: yes
  become_method: sudo

  vars:
    nbde_server_rotate_keys: true
    nbde_server_manage_firewall: true
    nbde_server_manage_selinux: true

  roles:
    - rhel-system-roles.nbde_server

- hosts: clients
  become: yes
  become_method: sudo

  vars:
    nbde_client_bindings:
      - device: /dev/vdb1
        encryption_password: redhatRHT
        servers:
          - http://serverc.lab.example.com
          - http://serverd.lab.example.com
    threshold: 2

  roles:
    - rhel-system-roles.nbde_client
```

- 7.2. Apply the `/home/student/RH415/labs/comprevew-nbde/nbde_setup.yml` Ansible Playbook.

```
[student@workstation ~]$ ansible-playbook \
  -i ~/RH415/labs/comprevew-nbde/inventory \
  --ask-become-pass ~/RH415/labs/comprevew-nbde/nbde_setup.yml
BECOME password: student
...output omitted...
```

Chapter 12 | Comprehensive Review

8. Return to the terminal window of the **serverb** machine. Configure the encrypted partition to automatically decrypt and mount on the **/encrypted** directory at boot time on the **serverb** machine. Reboot the **serverb** machine.

- 8.1. Edit the **/etc/crypttab** file to open the encrypted partition at boot time. Add the following content:

```
storage      /dev/vdb1  none    _netdev
```

- 8.2. Edit the **/etc/fstab** file to mount the encrypted partition on the **/encrypted** directory persistently.

```
...output omitted...
/dev/mapper/storage  /encrypted        xfs    _netdev      1 2
```

- 8.3. Reboot the **serverb** machine.

```
[root@serverb ~]# reboot
[root@serverb ~]# Connection to serverb closed by remote host.
Connection to serverb closed.
[student@workstation ~]$
```

9. After the **serverb** machine reboots, verify that the LUKS-encrypted partition on the **/dev/vdb1** device is decrypted and is mounted automatically on the **/encrypted** directory.

- 9.1. Log in to the **serverb** machine as the **student** user. The **serverb** host might take a few minutes to boot. You can check the boot progress by clicking **Open Console** on the lab control page. If the **serverb** machine fails to boot, then you might need to rebuild your lab environment.

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

- 9.2. Change to the **root** user. Use the **student** sudo password.

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

- 9.3. Verify that the LUKS-encrypted partition is mounted on the **/encrypted** directory.

```
[root@serverb ~]# mount | grep /encrypted
/dev/mapper/storage on /encrypted type xfs
(rw,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=32k,noquota,_netdev)
```

- 9.4. Verify that the **encryption-test.txt** file is present in the **/encrypted** directory.

```
[root@serverb ~]# ls /encrypted
encryption-test.txt
```

9.5. Exit the `serverb` terminal window.

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

10. Rotate the keys for the Tang servers by running the `/home/student/RH415/labs/comprevew-nbde/nbde_setup.yml` Ansible Playbook again.
- 10.1. Apply the `/home/student/RH415/labs/comprevew-nbde/nbde_setup.yml` Ansible Playbook.

```
[student@workstation ~]$ ansible-playbook \  
-i ~/RH415/labs/comprevew-nbde/inventory \  
--ask-become-pass ~/RH415/labs/comprevew-nbde/nbde_setup.yml  
BECOME password: student  
...output omitted...
```

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-nbde
```

Finish

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

```
[student@workstation ~]$ lab finish comprevew-nbde
```

▶ Lab

Restricting USB Device Access

Selectively control which USB devices may access or be accessed by the system by using USBGuard.

Outcomes

- Create a permanent USBGuard policy that allows a specific USB device to interact with the system.
- Generate a base policy that maintains currently defined policies and that ignores any additional USB devices that attempt to connect to the system.
- Use command-line tools to confirm USB device access policies.

Before You Begin

If you did not reset your **workstation** and **server** machines at the end of the last chapter, then save any work you want to keep from earlier exercises on those machines, and reset them now.

As the **student** user on the **workstation** machine, use the **lab** command to prepare your system for this exercise, and to ensure that all required resources are available.

```
[student@workstation ~]$ lab start compreview-usbguard
```

Specifications

Configure the **usbguard** virtual machine that is hosted on the **workstation** machine to use USBGuard.

- The following instructions describe how to use the **usbguard** virtual machine on the **workstation** machine, and how to attach virtual USB devices to the **usbguard** virtual machine:
 - The required XML files to define virtual USB devices for testing are present in the **/home/student/RH415/labs/compreview-usbguard** directory on the **workstation** machine.
 - You can start the virtual machine as the **root** user by using the **virsh start usbguard** command.
 - You can attach a USB device to the virtual machine by using the **virsh attach-device usbguard usb-device.xml** command, where **usb-device.xml** is a USB device configuration file.
 - You can connect to the console of the virtual machine by using the **virsh console usbguard** command.
- The USBGuard service on the **usbguard** virtual machine must be configured to meet the following specifications:

- The `usbguard` virtual machine should allow only the `MRKTG` USB device to access the system.
- The `usbguard` virtual machine should reject any other USB devices from interacting with the system.
- You can use the provided `GREEN` USB device to test whether device rejection is working.
- Make sure that the `USBGuard` service is configured, operating, and starts automatically on boot.

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-usbguard
```

Finish

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

```
[student@workstation ~]$ lab finish compreview-usbguard
```

► Solution

Restricting USB Device Access

Selectively control which USB devices may access or be accessed by the system by using USBGuard.

Outcomes

- Create a permanent USBGuard policy that allows a specific USB device to interact with the system.
- Generate a base policy that maintains currently defined policies and that ignores any additional USB devices that attempt to connect to the system.
- Use command-line tools to confirm USB device access policies.

Before You Begin

If you did not reset your **workstation** and **server** machines at the end of the last chapter, then save any work you want to keep from earlier exercises on those machines, and reset them now.

As the **student** user on the **workstation** machine, use the **lab** command to prepare your system for this exercise, and to ensure that all required resources are available.

```
[student@workstation ~]$ lab start comprevue-usbguard
```

1. On the **workstation** machine, start the **usbguard** virtual machine. Open the **usbguard** VM console and log in as the **student** user. Use **student** as the password.
 - 1.1. Change to the **root** user on the **workstation** machine. Use **student** as the password.

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

- 1.2. Use the **virsh start** command to start the **usbguard** VM. Allow the **usbguard** VM about two minutes to complete the startup process.

```
[root@workstation ~]# virsh start usbguard  
Domain 'usbguard' started
```

- 1.3. Use the **virsh console** command to access the console of the VM. If the console is slow to display the login prompt, then press **Enter** to proceed to the prompt.

```
[root@workstation ~]# virsh console usbguard  
Connected to domain 'usbguard'  
Escape character is ^] (Ctrl + ])
```

```
<Enter>
```

```
localhost login: student
Password: student
...output omitted...
[student@localhost ~]$
```

2. On the usbguard VM, install the RPM packages to configure, control, and manage USB device access.

- 2.1. Change to the `root` user. Use `student` as the password.

```
[student@localhost ~]$ sudo -i
[root@localhost ~]#
```

- 2.2. Install the `usbguard`, `usbutils`, and `udisks2` packages.

```
[root@localhost ~]# dnf install usbguard usbutils udisks2
...output omitted...
=====
Package      Arch      Version       Repository          Size
=====
Installing:
  udisks2    x86_64    2.9.4-7.el9    rhel-9.2-for-x86_64-appstream-rpms  495 k
  usbguard   x86_64    1.0.0-15.el9   rhel-9.2-for-x86_64-appstream-rpms  475 k
  usbutils    x86_64    013-4.el9     rhel-9.2-for-x86_64-baseos-rpms   119 k
...output omitted...
Is this ok [y/N]: y
...output omitted...
Complete!
```

3. Start the `usbguard` service and configure it to persist across reboots. Run the `usbguard list-devices` command to list the default devices.

- 3.1. Configure the `usbguard` service to persist across reboots.

```
[root@localhost ~]# systemctl enable usbguard --now
Created symlink /etc/systemd/system/basic.target.wants/usbguard.service → /usr/
lib/systemd/system/usbguard.service.
...output omitted...
```

- 3.2. Use the `usbguard list-devices` command to list all the USB devices that are recognized by USGuard.

```
[root@localhost ~]# usbguard list-devices
1: allow id 1d6b:0002 serial "0000:00:04.7" name "EHCI Host Controller"
hash "CsKOZ6IY8v3eojsC1fqKDw84V+MMhD6HsjojcZBjSg=" parent-hash
"MhPzffrQEhx5CwP3GXco7JXDbaMzFbD5FPuFFE7nfu0=" via-port "usb1" with-interface
09:00:00 with-connect-type ""
2: allow id 1d6b:0001 serial "0000:00:04.0" name "UHCI Host Controller" hash
"sKXn6PthDDlGgdxZHdnluQ9DR0kH/YSojkBlfpcnsaU=" parent-hash "9Ii0Zm8Mvu2nYz9z/
EgAXJ/ed6bLW8Ctv1iUD5rh6qY=" via-port "usb2" with-interface 09:00:00 with-connect-
type ""
3: allow id 1d6b:0001 serial "0000:00:04.1" name "UHCI Host Controller" hash
"6t6CPSS/v2EqQws6CMq8DVf0hgUGO2f+bEBX7R2yz0=" parent-hash "t7Z0XTvKnMmdqAm1vU
+noU318KzsRQQV+JorpRThQ7c=" via-port "usb3" with-interface 09:00:00 with-connect-
type ""
4: allow id 1d6b:0001 serial "0000:00:04.2" name "UHCI Host Controller"
hash "BSaNQWAaBI31jUqbck0N56uRuh3uVT1VkrdoD0ghs=" parent-hash "UyZQuRI
+gcw41fsM6Kgyty6pgYN0zfyYjqSpJv7na1E=" via-port "usb4" with-interface 09:00:00
with-connect-type ""
```

4. Set a permanent USBGuard policy to allow the MRKTG USB device access to the system. Use the /home/student/RH415/labs/comprevue-usbguard/usb-disk-mrktg.xml file to attach a new MRKTG USB device to the usbguard VM. Create a permanent allow rule target for the MRKTG USB device.
 - 4.1. From the workstation machine, open a second terminal session and attach the MRKTG USB device (usb-disk-mrktg.img) to the usbguard VM.

```
[student@workstation ~]$ sudo virsh attach-device usbguard \
~/RH415/labs/comprevue-usbguard/usb-disk-mrktg.xml
[sudo] password for student: student
Device attached successfully
```

- 4.2. On the virsh console terminal that is connected to the usbguard VM, you can see kernel messages indicating that the MRKTG USB device is not authorized for use. Press Enter to return to the command prompt.

```
[ 318.524538] usb 1-1: new high-speed USB device number 2 using ehci-pci
[ 318.677164] usb 1-1: New USB device found, idVendor=46f4, idProduct=0001,
bcdDevice= 0.00
[ 318.686755] usb 1-1: New USB device strings: Mfr=1, Product=2, SerialNumber=3
[ 318.694659] usb 1-1: Product: QEMU USB HARDDRIVE
[ 318.700177] usb 1-1: Manufacturer: QEMU
[ 318.704553] usb 1-1: SerialNumber: MRKTG
[ 318.725665] usb 1-1: Device is not authorized for usage
<Enter>

[root@localhost ~]#
```

- 4.3. From the usbguard VM, list the blocked USB devices and record the device number for the MRKTG USB device.

The device ID number (5 in the following output) might be different on your system.

```
[root@localhost ~]# usbguard list-devices --blocked
5: block id 46f4:0001 serial "MRKTG" name "QEMU USB HARDDRIVE"
hash "FyMVaBZ38qpDfKMvXvtwKcg0GcD7N1UQ/Kw04tx/coA=" parent-hash
"CsKOZ6IY8v3eojsc1fqKDw84V+MMhD6HsjjojcZBjSg=" via-port "1-1" with-interface
08:06:50 with-connect-type ""
```

- 4.4. On the `usbguard` VM, use the `usbguard allow-device` command to add a permanent `allow` rule target for the MRKTG USB device.

```
[root@localhost ~]# usbguard allow-device -p 5
...output omitted...
<Enter>
[ 876.176533] sd 6:0:0:0: [sda] Attached SCSI disk
```

- 4.5. Restart the `usbguard` services to ensure that the `USBGuard` daemon loads the `/etc/usbguard/rules.conf` file.

```
[root@localhost ~]# systemctl restart usbguard
...output omitted...
<Enter>
```

- 4.6. Run the `usbguard list-rules` command to list persistent rules and verify that the MRKTG USB device is listed. Rule numbers might be different on your system.

```
[root@localhost ~]# usbguard list-rules
1: allow id 46f4:0001 serial "MRKTG" name "QEMU USB HARDDRIVE"
hash "FyMVaBZ38qpDfKMvXvtwKcg0GcD7N1UQ/Kw04tx/coA=" parent-hash
"CsKOZ6IY8v3eojsc1fqKDw84V+MMhD6HsjjojcZBjSg=" with-interface 08:06:50 with-
connect-type ""
```

- 4.7. List the devices to ensure that the MRKTG USB device has a target policy of `allow`.

```
[root@localhost ~]# usbguard list-devices
2: allow id 1d6b:0002 serial "0000:00:04.7" name "EHCI Host Controller"
hash "CsKOZ6IY8v3eojsclfqKD84V+MMhD6HsjojcZBjSg=" parent-hash
"MhPzffrQEhx5CwP3GXco7JXDbmAxFBD5FPuFFE7nfu0=" via-port "usb1" with-interface
09:00:00 with-connect-type ""
3: allow id 1d6b:0001 serial "0000:00:04.0" name "UHCI Host Controller" hash
"sKXn6PthDDlGgdxZHdnluQ9DR0kH/YSojkBlfpcnsaU=" parent-hash "9Ii0Zm8Mvu2nYz9z/
EgAXJ/ed6bLW8Ctv1iUD5rh6qY=" via-port "usb2" with-interface 09:00:00 with-connect-
type ""
4: allow id 1d6b:0001 serial "0000:00:04.1" name "UHCI Host Controller" hash
"6t6CPSS/v2EqQws6CMq8DVf0hgUGO2f+bEBX7R2yz0=" parent-hash "t7Z0XTvKnMmdqAm1vU
+noU318KzsRQQV+JorpRThQ7c=" via-port "usb3" with-interface 09:00:00 with-connect-
type ""
5: allow id 1d6b:0001 serial "0000:00:04.2" name "UHCI Host Controller"
hash "BSaNQWAoBI31jUqbck0N56uRuh3uVT1VkrdoD0ghs=" parent-hash "UyZQuRI
+gcw41fsM6Kgyty6pgYN0zfyYjqSpJv7na1E=" via-port "usb4" with-interface 09:00:00
with-connect-type ""
6: block id 46f4:0001 serial "MRKTG" name "QEMU USB HARDDRIVE"
hash "FyMVaBZ38qpDfKMvXvtwKcg0GcD7N1UQ/Kw04tx/coA=" parent-hash
"CsKOZ6IY8v3eojsclfqKD84V+MMhD6HsjojcZBjSg=" via-port "1-1" with-interface
08:06:50 with-connect-type "unknown"
```

5. From the workstation machine, attach the GREEN USB device to the usbguard VM. Use the provided /home/student/RH415/labs/comprevew-usbguard/usb-disk-green.xml file to attach a new GREEN USB device to the usbguard VM. Then, create a permanent block rule target for the GREEN USB device, and detach the GREEN USB device. Leave only the MRKTG USB device attached.
 - 5.1. From the workstation machine, attach the GREEN USB device (usb-disk-green.img) to the usbguard VM.

```
[student@workstation ~]$ sudo virsh attach-device usbguard \
~/RH415/labs/comprevew-usbguard/usb-disk-green.xml
[sudo] password for student: student
Device attached successfully
```

- 5.2. On the virsh console terminal that is connected to the usbguard VM, you can see kernel messages indicating that the GREEN USB device is not authorized for use. Press Enter to return to the command prompt.

```
[ 417.112529] usb 1-2: new high-speed USB device number 3 using ehci-pci
[ 417.249491] usb 1-2: New USB device found, idVendor=46f4, idProduct=0001,
bcdDevice= 0.00
[ 417.253207] usb 1-2: New USB device strings: Mfr=1, Product=2, SerialNumber=3
[ 417.256561] usb 1-2: Product: QEMU USB HARDDRIVE
[ 417.258714] usb 1-2: Manufacturer: QEMU
[ 417.260685] usb 1-2: SerialNumber: GREEN
[ 417.268334] usb 1-2: Device is not authorized for usage
<Enter>

[root@localhost ~]#
```

- 5.3. From the **usbguard** VM, list the USB devices to confirm that the 5 device ID has the **allow** rule target and that the GREEN USB device has the **block** rule target.

The device numbers (5 and 7 in the following output) might be different on your system.

```
[root@localhost ~]# usbguard list-devices
2: allow id 1d6b:0002 serial "0000:00:04.7" name "EHCI Host Controller"
hash "CsKOZ6IY8v3eojsc1fqKDW84V+MMhD6HsjojcZBjSg=" parent-hash
"MhPzffrQEhx5CwP3GXco7JXDbamZFbD5FPUFFE7nfu0=" via-port "usb1" with-interface
09:00:00 with-connect-type ""
3: allow id 1d6b:0001 serial "0000:00:04.0" name "UHCI Host Controller" hash
"sKXn6PthDDlGgdxZHdnluQ9DR0kH/YSojkBlfpcnsaU=" parent-hash "9Ii0Zm8Mvu2nYz9z/
EgAXJ/ed6bLW8Ctv1iUD5rh6qY=" via-port "usb2" with-interface 09:00:00 with-connect-
type ""
4: allow id 1d6b:0001 serial "0000:00:04.1" name "UHCI Host Controller" hash
"6t6CPSS/v2EqQsws6CMq8DVf0hgUG02f+bEBX7R2yz0=" parent-hash "t7Z0XTvKnMmdqAm1vU
+nou318kzsRQQV+JorpRThQ7c=" via-port "usb3" with-interface 09:00:00 with-connect-
type ""
5: allow id 1d6b:0001 serial "0000:00:04.2" name "UHCI Host Controller"
hash "BSanQWADaBi31juqbck0N56uRuh3uVT1V4rdoD0ghs=" parent-hash "UyZQuRI
+gcw41fsM6Kgyty6pgYn0zfyYjqSpJv7na1E=" via-port "usb4" with-interface 09:00:00
with-connect-type ""
6: block id 46f4:0001 serial "MRKTG" name "QEMU USB HARDDRIVE"
hash "FyMvaBZ38qpDfKMvXvtwKcg0Gcd7N1UQ/Kw04tx/coA=" parent-hash
"CsKOZ6IY8v3eojsc1fqKDW84V+MMhD6HsjojcZBjSg=" via-port "1-1" with-interface
08:06:50 with-connect-type "unknown"
7: block id 46f4:0001 serial "GREEN" name "QEMU USB HARDDRIVE"
hash "Z7yCmQ0vP8sTvvCofXerut6IHbv1JSQpnvQoyElws5E=" parent-hash
"CsKOZ6IY8v3eojsc1fqKDW84V+MMhD6HsjojcZBjSg=" via-port "1-2" with-interface
08:06:50 with-connect-type ""
```

- 5.4. On the workstation machine, use the second terminal session to detach the GREEN USB device from the **usbguard** VM:

```
[student@workstation ~]$ sudo virsh detach-device usbguard \
~/RH415/labs/comprevue-usbguard/usb-disk-green.xml
Device detached successfully
```

6. From the **usbguard** VM, generate a new base policy with a **reject** rule target that ignores any additional USB devices that try to interact with the system. Use the `/home/student/RH415/labs/comprevue-usbguard/usb-disk-green.xml` file to confirm that the GREEN USB device is blocked from interacting with the **usbguard** VM.

- 6.1. Generate a new base policy with a **reject** rule target. Restart the **usbguard** service.

```
[root@localhost ~]# usbguard generate-policy -x -t \
reject > /etc/usbguard/rules.conf
[root@localhost ~]# systemctl restart usbguard.service
...output omitted...
<Enter>
```

- 6.2. Run the `usbguard list-rules` command to confirm an `allow` rule target for the MRKTG USB device followed by a catchall `reject` rule target that applies to any additional USB devices.

```
[root@localhost ~]# usbguard list-rules
1: allow id 1d6b:0002 serial "0000:00:04.7" name "EHCI Host Controller" with-
interface 09:00:00 with-connect-type ""
2: allow id 1d6b:0001 serial "0000:00:04.0" name "UHCI Host Controller" with-
interface 09:00:00 with-connect-type ""
3: allow id 1d6b:0001 serial "0000:00:04.1" name "UHCI Host Controller" with-
interface 09:00:00 with-connect-type ""
4: allow id 1d6b:0001 serial "0000:00:04.2" name "UHCI Host Controller" with-
interface 09:00:00 with-connect-type ""
5: allow id 46f4:0001 serial "MRKTG" name "QEMU USB HARDDRIVE" with-interface
08:06:50 with-connect-type "unknown"
6: reject
```

- 6.3. From the `workstation` machine, attach the GREEN USB device (`usb-disk-green.img`) to the `usbguard` VM.

Although the command output indicates that the GREEN USB device was successfully attached, further investigation on the `usbguard` VM confirms that the attempt to attach a USB device was not authorized. A blocked USB device appears in command-line tool listings but it is not allowed to mount. A rejected USB device is ignored by the system and therefore does not display in command-line tool listings.

```
[student@workstation ~]$ sudo virsh attach-device usbguard \
~/RH415/labs/comprevue-usbguard/usb-disk-green.xml
Device attached successfully
```

- 6.4. The journal records the kernel action as well as the USGuard action. Press `q` to exit the `journalctl` prompt.

```
...output omitted...
<Enter>
[root@localhost ~]# journalctl -b -e
Jan 23 12:22:40 localhost.localdomain kernel: sd 6:0:0:0: [sda] 65536 512-byte >
Jan 23 12:22:40 localhost.localdomain kernel: sd 6:0:0:0: [sda] Write Protect i>
Jan 23 12:22:40 localhost.localdomain kernel: sd 6:0:0:0: [sda] Mode Sense: 63 >
Jan 23 12:22:40 localhost.localdomain kernel: sd 6:0:0:0: [sda] Write cache: en>
Jan 23 12:22:40 localhost.localdomain kernel: sd 6:0:0:0: [sda] Attached SCSI d>
Jan 23 12:23:26 localhost.localdomain systemd[1]: Starting dnf makecache...
Jan 23 12:23:27 localhost.localdomain dnf[11592]: Failed determining last makec>
Jan 23 12:23:27 localhost.localdomain dnf[11592]: Red Hat Enterprise Linux 9.2 >
Jan 23 12:23:27 localhost.localdomain dnf[11592]: Red Hat Enterprise Linux 9.2 >
Jan 23 12:23:27 localhost.localdomain dnf[11592]: Metadata cache created.
Jan 23 12:23:27 localhost.localdomain systemd[1]: dnf-makecache.service: Deacti>
Jan 23 12:23:27 localhost.localdomain systemd[1]: Finished dnf makecache.
Jan 23 12:24:35 localhost.localdomain kernel: usb 1-2: new high-speed USB devic>
Jan 23 12:24:35 localhost.localdomain kernel: usb 1-2: New USB device found, id>
Jan 23 12:24:35 localhost.localdomain kernel: usb 1-2: New USB device strings: >
Jan 23 12:24:35 localhost.localdomain kernel: usb 1-2: Product: QEMU USB HARDDR>
Jan 23 12:24:35 localhost.localdomain kernel: usb 1-2: Manufacturer: QEMU
Jan 23 12:24:35 localhost.localdomain kernel: usb 1-2: SerialNumber: GREEN
```

Chapter 12 | Comprehensive Review

```
Jan 23 12:24:35 localhost.localdomain usbguard-daemon[11576]: uid=0 pid=11574 r>
Jan 23 12:24:35 localhost.localdomain kernel: usb 1-2: Device is not authorized
Jan 23 12:24:35 localhost.localdomain usbguard-daemon[11576]: uid=0 pid=11574 r>
Jan 23 12:24:35 localhost.localdomain kernel: usb 1-2: USB disconnect, device n>
Jan 23 12:24:35 localhost.localdomain usbguard-daemon[11576]: uid=0 pid=11574 r>
...output omitted...
<q>
```

- 6.5. Run the `usbguard list-devices` command to confirm that the MRKTG USB device is listed but the GREEN USB device is ignored and therefore is not listed.

```
[root@localhost ~]# usbguard list-devices
7: allow id 1d6b:0002 serial "0000:00:04.7" name "EHCI Host Controller"
hash "Csk0Z6IY8v3eojsc1fqKDW84V+MMhD6HsjjojcZBjSg=" parent-hash
"MhPzffrQEx5CwP3GXco7JXDbamZFd5FPUffE7nfu0=" via-port "usb1" with-interface
09:00:00 with-connect-type ""
8: allow id 1d6b:0001 serial "0000:00:04.0" name "UHCI Host Controller" hash
"sKXn6PthDDlGgdxZHdnluQ9DR0kH/YSojkBlfpncnsau=" parent-hash "9Ii0Zm8Mvu2nYz9z/
EgAXJ/ed6bLW8Ctv1iUD5rh6qY=" via-port "usb2" with-interface 09:00:00 with-connect-
type ""
9: allow id 1d6b:0001 serial "0000:00:04.1" name "UHCI Host Controller" hash
"6t6CPSS/v2EqQsw6CMq8DVf0hgUG02f+bEBX7R2yz0=" parent-hash "t7Z0XTvKnMmdqAm1VU
+nOU318kZsRQQV+JorpRThQ7c=" via-port "usb3" with-interface 09:00:00 with-connect-
type ""
10: allow id 1d6b:0001 serial "0000:00:04.2" name "UHCI Host Controller"
hash "BSAnQWADaBI31jUqbck0N56uRuh3uVT1Vk4rdoD0ghs=" parent-hash "UyZQuRI
+gcw41fsM6Kgyty6pgYN0zfYjqSpJv7na1E=" via-port "usb4" with-interface 09:00:00
with-connect-type ""
11: allow id 46f4:0001 serial "MRKTG" name "QEMU USB HARDDRIVE"
hash "FyMVaBZ38qpDfKMvXvtwKcg0Gcd7N1UQ/Kw04tx/coA=" parent-hash
"Csk0Z6IY8v3eojsc1fqKDW84V+MMhD6HsjjojcZBjSg=" via-port "1-1" with-interface
08:06:50 with-connect-type "unknown"
```

- 6.6. Return to the student user on the usbguard VM terminal session:

```
[root@localhost ~]# logout
[student@localhost ~]$ logout
```

- 6.7. Exit the virtual machine's console and return to the workstation machine as the **student** user:

```
Red Hat Enterprise Linux Server 7.5 (Maipo)
Kernel 3.10.0-862.3.2.el7.x86_64 on an x86_64

localhost login:
Ctrl+]
[root@workstation ~]# logout
[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-usbguard
```

Finish

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

```
[student@workstation ~]$ lab finish compreview-usbguard
```

▶ Lab

Recording Events and Monitoring File-system Changes with PAM, Audit, and AIDE

Configure password quality requirements, configure system event recording, and monitor file-system changes.

Outcomes

- Configure remote Audit logs.
- Enable prepackaged Audit rules.
- Enable auditing of TTY and enable the `pam_pwquality` module by using an `authselect` security profile.
- Configure Audit to log events for changes to file access permissions in the `/etc/ssh` directory, and to label the log entries with a key.
- Change the `/etc/ssh` file, and detect those changes with AIDE.
- Use Audit tools to locate a record that shows which user and process made the changes.

Before You Begin

If you did not reset your `workstation` and `server` machines at the end of the last chapter, then save any work you want to keep from earlier exercises on those machines, and reset them now.

As the `student` user on the `workstation` machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

```
[student@workstation ~]$ lab start compreview-aide
```

Specifications

- Configure the Audit service on the `servera` machine to send messages to the Audit service on the `serverb.lab.example.com` host. After configuration, use the `service auditd restart` command to restart the daemon and to load the new configuration.
- Configure the Audit service on the `serverb` machine to accept messages from the Audit service on the `servera` machine. After configuration, use the `service auditd restart` command to restart the daemon and to load the new configuration.
- Enable the prepackaged STIG Audit rules on the `servera` machine.
- Create and select the `custom/minimal-with-tty-audit` security profile based on the `minimal` security profile on the `servera` machine to enable the `pam_tty_audit` PAM module.

Chapter 12 | Comprehensive Review

- Adjust the password quality requirements on the `servera` machine to require at least one uppercase letter.
- Install and configure the `aide` package as the `root` user on the `servera` machine to detect changes that are made in the `/etc/ssh` directory.
- Add a persistent Audit watch rule on the `servera` machine to generate Audit log entries whenever there is an attempt to read, write, execute, or change an attribute of the `/etc/ssh` directory. Use `sshd_config_monitor` as the filter key on the Audit rule.
- Make a change in the `/etc/ssh` directory of the `servera` machine by modifying the `/etc/ssh/sshd_config` file. In the `/etc/ssh/sshd_config` file, uncomment the `PasswordAuthentication` directive and change the `yes` value to the `no` value.

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-aide
```

Finish

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

```
[student@workstation ~]$ lab finish compreview-aide
```

► Solution

Recording Events and Monitoring File-system Changes with PAM, Audit, and AIDE

Configure password quality requirements, configure system event recording, and monitor file-system changes.

Outcomes

- Configure remote Audit logs.
- Enable prepackaged Audit rules.
- Enable auditing of TTY and enable the `pam_pwquality` module by using an `authselect` security profile.
- Configure Audit to log events for changes to file access permissions in the `/etc/ssh` directory, and to label the log entries with a key.
- Change the `/etc/ssh` file, and detect those changes with AIDE.
- Use Audit tools to locate a record that shows which user and process made the changes.

Before You Begin

If you did not reset your `workstation` and `server` machines at the end of the last chapter, then save any work you want to keep from earlier exercises on those machines, and reset them now.

As the `student` user on the `workstation` machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

```
[student@workstation ~]$ lab start compreview-aide
```

1. Configure the Audit service on the `servera` machine to send messages to the Audit service on the `serverb.lab.example.com` host. After configuration, use the `service auditd restart` command to restart the daemon and to load the new configuration.



Note

The `service` command is deprecated; do not use it in production environments. The `systemctl restart` command cannot be used with the `auditd` service due to the interaction between the daemon and the Linux kernel. In production environments, reboot the machine to ensure that the new configuration is loaded.

- 1.1. Log in to the `servera` machine as the `student` user.

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

- 1.2. Change to the `root` user. Use `student` as the password.

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

- 1.3. Install the `audispd-plugins` package.

```
[root@servera ~]# dnf -y install audispd-plugins  
...output omitted...
```

- 1.4. In the `/etc/audit/plugins.d/au-remote.conf` file, set the value for the `active` variable to `yes` to enable remote logging.

```
[root@servera ~]# cat /etc/audit/plugins.d/au-remote.conf  
...output omitted...  
active = yes  
...output omitted...
```

- 1.5. In the `/etc/audit/audisp-remote.conf` file, set the `remote_server` variable to the `serverb.lab.example.com` hostname. Also, set the port to be used in the remote logging server, which is `60` by default.

```
[root@servera ~]# cat /etc/audit/audisp-remote.conf  
...output omitted...  
remote_server = serverb.lab.example.com  
port = 60  
...output omitted...
```

- 1.6. Restart the `audited` service to update its configuration. When done, return to the `workstation` machine as the `student` user.

```
[root@servera ~]# service audited restart  
...output omitted...  
[root@servera ~]# logout  
[student@servera ~]$ logout  
[student@workstation ~]$
```

2. Configure the Audit service on the `serverb` machine to accept messages from the Audit service on the `servera` machine. After configuration, use the `service audited restart` command to restart the daemon and to load the new configuration.

- 2.1. Log in to the `serverb` machine as the `student` user.

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

- 2.2. Change to the **root** user. Use **student** as the password.

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

- 2.3. In the **/etc/audit/auditd.conf** file, uncomment the **tcp_listen_port** variable, and set its value to 60 so that the Audit service listens on the 60 TCP port.

```
[root@serverb ~]# cat /etc/audit/auditd.conf  
...output omitted...  
tcp_listen_port = 60  
...output omitted...
```

- 2.4. Open the 60 TCP port to enable access to the Audit server.

```
[root@serverb ~]# firewall-cmd --zone=public --add-port=60/tcp --permanent  
success  
[root@serverb ~]# firewall-cmd --reload  
success
```

- 2.5. Restart the **auditd** service to update its configuration. When done, return to the **workstation** machine as the **student** user.

```
[root@serverb ~]# service auditd restart  
...output omitted...  
[root@serverb ~]# logout  
[student@serverb ~]$ logout  
[student@workstation ~]$
```

3. Enable the prepackaged STIG Audit rules on the **servera** machine.

- 3.1. Log in to the **servera** machine as the **student** user.

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

- 3.2. Change to the **root** user. Use **student** as the password.

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

- 3.3. Copy the **/usr/share/audit/sample-rules/30-stig.rules** file with the STIG Audit rules into the **/etc/audit/rules.d/** directory.

```
[root@servera ~]# cp /usr/share/audit/sample-rules/30-stig.rules \
/etc/audit/rules.d/
```

- 3.4. Load the STIG Audit rules with the augenrules --load command.

```
[root@servera ~]# augenrules --load
...output omitted...
```

4. Create and select the custom/minimal-with-tty-audit security profile based on the minimal security profile to enable the pam_tty_audit PAM module for the student user.

- 4.1. Create the minimal-with-tty-audit security profile.

```
[root@servera ~]# authselect create-profile minimal-with-tty-audit \
-b minimal --symlink-meta --symlink-pam
New profile was created at /etc/authselect/custom/minimal-with-tty-audit
```

- 4.2. Add entries to the /etc/authselect/custom/minimal-with-tty-audit/system-auth and /etc/authselect/custom/minimal-with-tty-audit/password-auth files to enable the pam_tty_audit module for the student user.

```
[root@servera ~]# echo "session required pam_tty_audit.so enable=student" \
>> /etc/authselect/custom/minimal-with-tty-audit/system-auth
[root@servera ~]# echo "session required pam_tty_audit.so enable=student" \
>> /etc/authselect/custom/minimal-with-tty-audit/password-auth
```

- 4.3. Enable the custom/minimal-with-tty-audit security profile.

```
[root@servera ~]# authselect select custom/minimal-with-tty-audit --force
...output omitted...
```

5. Adjust the password quality requirements on the servera machine to require at least one uppercase character.

- 5.1. Use a text editor to edit the /etc/security/pwquality.conf file to uncomment the ucredit entry and change its value to -1.

```
...output omitted...
# The maximum credit for having uppercase characters in the new password.
# If less than 0 it is the minimum number of uppercase characters in the new
# password.
ucredit = -1
...output omitted...
```

6. Install the aide package as the root user on the servera machine.

```
[root@servera ~]# dnf -y install aide
...output omitted...
Dependencies resolved.
=====
 Package Arch Version Repository Size

```

```
=====
Installing:
 aide      x86_64      0.16-100.el9      rhel-9.2-for-x86_64-appstream-rpms      154 k

Transaction Summary
=====
...output omitted...
Is this ok [y/N]: y
...output omitted...
Complete!
```

7. Edit the /etc/aide.conf file to detect changes that are made in the /etc/ssh directory. Add the line to the ssh section in the /etc/aide.conf file.

```
[root@servera ~]# cat /etc/aide.conf
...output omitted...
# ssh
/etc/ssh CONTENT_EX
/etc/ssh/sshd_config$ CONTENT_EX
/etc/ssh/ssh_config$ CONTENT_EX
...output omitted...
```

8. Initialize the baseline AIDE database.

```
[root@servera ~]# aide --init
Start timestamp: 2023-10-16 15:17:51 -0400 (AIDE 0.16)
AIDE initialized database at /var/lib/aide/aide.db.new.gz

Number of entries: 53262

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

/var/lib/aide/aide.db.new.gz
MD5      : lUxp0jxdbnY3+x1sKq8PA==
SHA1     : ImCD2brbbbRI1GxGqrmaE5To/Ww=
RMD160   : b+JFmhCsHYkdfjzbC5eYajNSI+c=
TIGER    : RUZqbTT+pxf0bAahjG15Zub84zsZHLzY
SHA256   : 0vQpy7LTDRNv1Ig5oSwxngJzInYQDRvC
          rqEEkS9JELM=
SHA512   : Ymsu3peZuMVQyUT/+jrGD2voEZMaQYg
          GIT10AntZzV4dDdr6AOjEt4ELVXQub
          gImVw46Z1PkwfRG3jxp3DQ==

End timestamp: 2023-10-16 15:18:37 -0400 (run time: 0m 46s)
```

9. Rename the new AIDE database file from /var/lib/aide/aide.db.new.gz to /var/lib/aide/aide.db.gz so that AIDE uses the newly generated file as the current database.

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

Chapter 12 | Comprehensive Review

10. Determine the current status of the machine's file systems.

AIDE reports no changes to the files and directories that it monitors, because you have not changed any files or directories after initializing the AIDE database.

```
[root@servera ~]# aide --check
Start timestamp: 2023-10-16 15:22:29 -0400 (AIDE 0.16)
AIDE found NO differences between database and filesystem. Looks okay!!

Number of entries: 53262

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

/var/lib/aide/aide.db.gz
MD5      : lUxp0jxdbnY3+x1sKq8PA==
SHA1     : ImCD2brbbbRI1GxGqrmaE5To/Ww=
RMD160   : b+JFmhCsHYkdfjzbC5eYajNSI+c=
TIGER    : RUZqbTT+pxf0bAahjG15Zub84zsZHLzY
SHA256   : 0vQpy7LTDRNv1Ig5oSwxngJzInYQDRvC
                  rqEEkS9JELM=
SHA512   : Ymsu3peZuMVQyUT/+jrGD2voEZMaQYg
                  GIT10AntZzV4dDdr6AOOpjcEt4ELVXQUb
                  gImVw46Z1PkwfRG3jxp3DQ==

End timestamp: 2023-10-16 15:23:11 -0400 (run time: 0m 42s)
```

11. Add a persistent Audit watch rule to generate Audit log entries whenever there is an attempt to read, write, execute, or change an attribute of the /etc/ssh directory. Use `sshd_config_monitor` as the filter key on the Audit rule.

```
[root@servera ~]# cat /etc/audit/rules.d/audit.rules
...output omitted...
-w /etc/ssh -p wa -k sshd_config_monitor
```

12. Apply the changes for the newly added Audit rules to take effect.

```
[root@servera ~]# augenrules --load
...output omitted...
```

13. List the Audit rules and verify that the newly added Audit rule is currently in effect.

```
[root@servera ~]# auditctl -l
...output omitted...
-w /etc/ssh -p wa -k sshd_config_monitor
```

14. Make a change in the /etc/ssh directory by modifying the /etc/ssh/sshd_config file. In the /etc/ssh/sshd_config file, uncomment the PasswordAuthentication directive and change the yes value to the no value.



Important

If you make a mistake here, then you might create issues with SSH authentication on the servera machine, preventing future logins that use the ssh service. If this mistake happens, then rebuild your lab environment and begin this exercise again from the beginning to ensure that you do not encounter further issues in future exercises.

```
[root@servera ~]# cat /etc/ssh/sshd_config
...output omitted...
PasswordAuthentication no
...output omitted...
```

15. Restart the sshd daemon to apply the new changes in the SSH service configuration file.

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

16. Verify the current status of the machine's file systems with AIDE to ensure that AIDE detects the change in the /etc/ssh/sshd_config file.

```
[root@servera ~]# aide --check
Start timestamp: 2023-10-16 15:33:03 -0400 (AIDE 0.16)
AIDE found differences between database and filesystem!

Summary:
Total number of entries: 53263
Added entries: 1
Removed entries: 0
Changed entries: 3
...output omitted...

-----
Changed entries:
-----

f ... .C... : /etc/audit/audit.rules
f ... .C... : /etc/audit/rules.d/audit.rules
f ... .C... : /etc/ssh/sshd_config

-----
Detailed information about changes:
-----
...output omitted...
File: /etc/ssh/sshd_config
SHA512   : Cs01Y1exozdL381/r0KJ7UwoPqu08LYe | 3uQTXYd1wnvJNs+GuLQc84Q9z16yzd0c
          Us+4qpsVDYJMSfePefYZDHvSKzAEsRt | HKzXcq5/tNmuAF2V+JxG/dQQNu61h4I+
          UilLiWwiIcrL/8R/0df0CKg==           | uABk/lSwcR0pSrxBMwX/4Q==
...output omitted...
```

Chapter 12 | Comprehensive Review

17. Investigate the Audit log to determine what changed the /etc/ssh/sshd_config file. Use the sshd_config_monitor key to limit the output.

```
[root@servera ~]# ausearch -i -f /etc/ssh/sshd_config -k sshd_config_monitor
...output omitted...
type=PROCTITLE msg=audit(10/16/23 15:30:47.188:257) : proctitle=vim /etc/ssh/
sshd_config
type=PATH msg=audit(10/16/23 15:30:47.188:257) : item=3 name=/etc/ssh/sshd_config-
inode=16809537 dev=fc:04 mode=file,600 uid=root ogid=root rdev=00:00
obj=system_u:object_r:etc_t:s0 nametype=CREATE cap_fp=none cap_hi=none cap_fe=0
cap_fver=0 cap_frootid=0
type=PATH msg=audit(10/16/23 15:30:47.188:257) : item=2 name=/etc/ssh/sshd_config
inode=16809537 dev=fc:04 mode=file,600 uid=root ogid=root rdev=00:00
obj=system_u:object_r:etc_t:s0 nametype=DELETE cap_fp=none cap_hi=none cap_fe=0
cap_fver=0 cap_frootid=0
type=PATH msg=audit(10/16/23 15:30:47.188:257) : item=1 name=/etc/ssh/
inode=16797950 dev=fc:04 mode=dir,755 uid=root ogid=root rdev=00:00
obj=system_u:object_r:etc_t:s0 nametype=PARENT cap_fp=none cap_hi=none cap_fe=0
cap_fver=0 cap_frootid=0
type=PATH msg=audit(10/16/23 15:30:47.188:257) : item=0 name=/etc/ssh/
inode=16797950 dev=fc:04 mode=dir,755 uid=root ogid=root rdev=00:00
obj=system_u:object_r:etc_t:s0 nametype=PARENT cap_fp=none cap_hi=none cap_fe=0
cap_fver=0 cap_frootid=0
type=CWD msg=audit(10/16/23 15:30:47.188:257) : cwd=/root
type=SYSCALL msg=audit(10/16/23 15:30:47.188:257) :
arch=x86_64 syscall=rename success=yes exit=0 a0=0x556420fce810 a1=0x556421251630
a2=0xfffffffffffffe98 a3=0x0 items=4 ppid=26526 pid=27113 auid=student uid=root
gid=root euid=root suid=root fsuid=root egid=root sgid=root fsgid=root tty=pts0
ses=5 comm=vim exe=/usr/bin/vim subj=unconfined_u:unconfined_r:unconfined_t:s0-
s0:c0.c1023 key=sshd_config_monitor
...output omitted...
```

18. Return to the workstation machine as the student user.

```
[root@servera ~]# logout
[student@servera ~]$ logout
Connection to servera closed.
[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-aide
```

Finish

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

```
[student@workstation ~]$ lab finish compreview-aide
```

► Lab

Mitigating Risk with SELinux

Configure confined users and address some SELinux denials.

Outcomes

- Modify the SELinux mode.
- Use the `grubby` command to disable and enable SELinux.
- Inspect AVC messages by using the Audit system.
- Allow the `httpd` service to listen on a different port for the `http_port_t` port type.
- Prevent Linux users from switching to a different SELinux confined user.
- Enable and configure SELinux to limit user access to `sudo` commands.
- Explore the Booleans that determine how SELinux controls confined users.
- Examine the effects of confined SELinux users on the `sudo`, `su`, and `ssh` commands, and on SUID execution.

Before You Begin

If you did not reset your `workstation` and `server` machines at the end of the last chapter, then save any work you want to keep from earlier exercises on those machines, and reset them now.

As the `student` user on the `workstation` machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

```
[student@workstation ~]$ lab start comprevew-selinux
```

Specifications

- On the `servere` machine, set SELinux to `permissive` mode.
- Use the `grubby` command to enable SELinux.
- Troubleshoot for any SELinux denial messages.
- Set SELinux to `enforcing` mode.
- Allow the `httpd` service to listen on port 2693 for the `http_port_t` port type.
- Confine users on the `servere` machine to prevent them from using the `sudo` and `su` commands. Prevent users from executing binaries within their home directory. Verify this configuration by running the `runme` binary in the `student` home directory. These changes do not apply to the `root` user.

- Create an administrative operator4 user on the servere machine with redhat as the password, and add the user to the wheel Linux group. Map the operator4 user to the sysadm_u SELinux user to allow the use of the su and sudo commands.
- For sysadm_u SELinux users on the servere machine, enable the use of the ssh command to log in.

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-selinux
```

Finish

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

```
[student@workstation ~]$ lab finish compreview-selinux
```

► Solution

Mitigating Risk with SELinux

Configure confined users and address some SELinux denials.

Outcomes

- Modify the SELinux mode.
- Use the `grubby` command to disable and enable SELinux.
- Inspect AVC messages by using the Audit system.
- Allow the `httpd` service to listen on a different port for the `http_port_t` port type.
- Prevent Linux users from switching to a different SELinux confined user.
- Enable and configure SELinux to limit user access to `sudo` commands.
- Explore the Booleans that determine how SELinux controls confined users.
- Examine the effects of confined SELinux users on the `sudo`, `su`, and `ssh` commands, and on SUID execution.

Before You Begin

If you did not reset your `workstation` and `server` machines at the end of the last chapter, then save any work you want to keep from earlier exercises on those machines, and reset them now.

As the `student` user on the `workstation` machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

```
[student@workstation ~]$ lab start comprevew-selinux
```

1. Log in to the `servere` machine as the `student` user. Switch to the `root` user. Use `student` as the password.

```
[student@workstation ~]$ ssh student@servere
[student@servere ~]$ sudo -i
[sudo] password for student: student
[root@servere ~]#
```

2. Confirm that SELinux is in `Disabled` mode.

```
[root@servere ~]# getenforce
Disabled
```

3. Enable SELinux in `permissive` mode, and reboot the server.

- 3.1. Enable SELinux in `permissive` mode in the `/etc/selinux/config` file.

```
...output omitted...
SELINUX=permissive
...output omitted...
```

- 3.2. Use the **grubby** command to enable SELinux by removing the argument that disables SELinux from the kernel command line.

```
[root@servere ~]# grubby --update-kernel ALL --remove-args selinux
```

- 3.3. Reboot the **servere** machine. When ready, log in again to the **servere** machine as the **student** user and switch to the **root** user.

```
[root@servere ~]# reboot
Connection to servere closed.
[student@workstation ~]$ ssh student@servere
[student@servere ~]$ sudo -i
[sudo] password for student: student
[root@servere ~]#
```

4. Use the **curl** command to verify that the web server at <http://servere.lab.example.com:2693> is available.

```
[root@servere ~]# curl http://servere.lab.example.com:2693
<html>
  <head>SELinux</head>
  <body>
    <h1>Works! - Port 2693</h1>
  </body>
</html>
```

5. Verify that SELinux is set to **Permissive** mode, and check for SELinux denial messages.

- 5.1. Verify that SELinux is in **Permissive** mode.

```
[root@servere ~]# getenforce
Permissive
```

- 5.2. Check the Audit log file for any SELinux denial messages. The **curl** command in the previous step was successful because SELinux is set to **permissive** mode, but the command did generate an AVC denied message in the log. This denied message means that the **curl http://servere.lab.example.com:2693** command would fail if SELinux were set to **enforcing** mode.

```
[root@servere ~]# ausearch -m AVC,USER_AVC
...output omitted...
time->Tue Jan 30 20:06:47 2024
type=PROCTITLE msg=audit(1706663207.015:34):
    proctitle=2F7573722F7362696E2F6874747064002D44464F524547524F554E44
type=SYSCALL msg=audit(1706663207.015:34): arch=c000003e syscall=49
    success=yes exit=0 a0=4 a1=556b528be438 a2=1c a3=7ffe0be018ac items=0
    ppid=1 pid=793 auid=4294967295 uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0
    sgid=0 fsgid=0 tty=(none) ses=4294967295 comm="httpd" exe="/usr/sbin/httpd"
    subj=system_u:system_r:httpd_t:s0 key=(null)
type=AVC msg=audit(1706663207.015:34): avc: denied { name_bind } for
    pid=793 comm="httpd" src=2693 scontext=system_u:system_r:httpd_t:s0
    tcontext=system_u:object_r:unreserved_port_t:s0 tclass=tcp_socket permissive=1
...output omitted...
```

- 5.3. Check the journal log file for any SELinux denial messages.

```
[root@servere ~]# journalctl -t setroubleshoot
...output omitted...
Jan 30 20:06:55 servere setroubleshoot[1241]:
SELinux is preventing /usr/sbin/httpd from name_bind access on the tcp_socket port
2693.

***** Plugin bind_ports (92.2 confidence) suggests *****

If you want to allow /usr/sbin/httpd to bind to network port 2693
Then you need to modify the port type.
Do
# semanage port -a -t PORT_TYPE -p tcp 2693
...output omitted...
```

6. Change SELinux to enforcing mode, and reboot the server.

- 6.1. Change SELinux to enforcing mode in the /etc/selinux/config file.

```
...output omitted...
SELINUX=enforcing
...output omitted...
```

- 6.2. Reboot the servere machine. When ready, log in again to the servere machine as the student user and switch to the root user.

```
[root@servere ~]# reboot
Connection to servere closed.
[student@workstation ~]$ ssh student@servere
[student@servere ~]$ sudo -i
[sudo] password for student: student
[root@servere ~]#
```

7. Verify that the SELinux mode is set to Enforcing mode and determine whether the web server at <http://servere.lab.example.com:2693> is available.

- 7.1. Verify that the SELinux mode on the servere machine is set to Enforcing mode.

```
[root@servere ~]$ getenforce  
Enforcing
```

- 7.2. Use the curl command to verify that the web server at <http://servere.lab.example.com:2693> is now unavailable.

```
[root@servere ~]# curl http://servere.lab.example.com:2693  
curl: (7) Failed to connect to servere.lab.example.com port 2693: Connection  
refused
```

8. Allow the httpd service to listen on port 2693 for the http_port_t port type. Verify that the port has been added.

- 8.1. Use the semanage port -a command to allow the httpd service to listen on port 2693 for the http_port_t port type.

```
[root@servere ~]# semanage port -a -t http_port_t -p tcp 2693
```

- 8.2. Use the semanage port -l command to list the ports.

```
[root@servere ~]# semanage port -l | grep 2693  
http_port_t          tcp      2693, 80, 81, 443, 488, 8008, 8009, 8443, 9000
```

9. Restart the httpd service and verify one more time that the web server at <http://servere.lab.example.com:2693> is available.

- 9.1. Use the systemctl restart command to restart the httpd service.

```
[root@servere ~]# systemctl restart httpd.service
```

- 9.2. Use the curl command to verify that the web server at <http://servere.lab.example.com:2693> is available.

```
[root@servere ~]# curl http://servere.lab.example.com:2693  
<html>  
  <head>SELinux</head>  
  <body>  
    <h1>Works! - Port 2693</h1>  
  </body>  
</html>
```

10. Confine users to prevent them from using the sudo and su commands. Prevent users from running programs in their home directory. These restrictions do not apply to the root user.

- 10.1. Change the default mapping to map the Linux users to the user_u SELinux user. The semanage command might take up to one minute to complete.

```
[root@servere ~]# semanage login -m -s user_u -r s0 __default__
```

- 10.2. Use the semanage login -l command to verify your work.

```
[root@servere ~]# semanage login -l
```

Login Name	SELinux User	MLS/MCS Range	Service
default	user_u	s0	*
root	unconfined_u	s0-s0:c0.c1023	*

- 10.3. Set the `user_exec_content` SELinux Boolean to the `off` value to prevent users who are mapped to the `user_u` SELinux user from executing programs in their home directories.

```
[root@servere ~]# setsebool -P user_exec_content off
```

- 10.4. Log out of the `servere` machine and log in again as the `student` user.

```
[root@servere ~]# logout  
[student@servere ~]$ logout  
[student@workstation ~]$ ssh student@servere  
...output omitted...  
[student@servere ~]$
```

- 10.5. Confirm that the `student` user can no longer execute programs in their home directory. Verify that the `student` user can no longer execute the `runme` binary program in the `/home/student/` directory.

```
[student@servere ~]$ ./runme  
-bash: ./runme: Permission denied
```

- 10.6. Confirm that the `student` user can no longer use the `sudo` and `su` commands.

```
[student@servere ~]$ sudo -i  
sudo: PERM_SUDOERS: setresuid(-1, 1, -1): Operation not permitted  
sudo: no valid sudoers sources found, quitting  
sudo: setresuid() [0, 0, 0] -> [1000, -1, -1]: Operation not permitted  
sudo: error initializing audit plugin sudoers_audit  
[student@servere ~]$ su -  
Password: redhat  
su: Authentication failure
```

11. On the `servere` machine, create an administrative `operator4` user with `redhat` as the password. Map that user to a confined SELinux user to use the `su` and `sudo` commands, and to use the `ssh` command to log in.

- 11.1. Log out of the `servere` machine and log in as the `root` user.

```
[student@servere ~]$ logout  
[student@workstation ~]$ ssh root@servere  
[root@servere ~]#
```

- 11.2. Create the `operator4` Linux user account, map it to the `sysadm_u` SELinux user, and add it to the `wheel` group.

```
[root@servere ~]# useradd -G wheel -Z sysadm_u operator4
```

- 11.3. Set the password for the operator4 user to redhat.

```
[root@servere ~]# echo redhat | passwd --stdin operator4
Changing password for user operator4.
passwd: all authentication tokens updated successfully.
```

- 11.4. Confirm that SELinux maps the operator4 user to the sysadm_u SELinux user.

```
[root@servere ~]# semanage login -l
```

Login Name	SELinux User	MLS/MCS Range	Service
__default__	user_u	s0	*
operator4	sysadm_u	s0-s0:c0.c1023	*
root	unconfined_u	s0-s0:c0.c1023	*

- 11.5. Log out of the servere machine and try to use SSH to log in as the operator4 user.

```
[root@servere ~]# logout
[student@workstation ~]$ ssh operator4@servere
client_loop: send disconnect: Broken pipe
[student@workstation ~]$
```

By default, SELinux denies access to sysadm_u accounts over SSH.

- 11.6. Log in as the root user and set the ssh_sysadm_login SELinux Boolean to the on value.

```
[student@workstation ~]$ ssh root@servere
[root@servere ~]# setsebool -P ssh_sysadm_login on
```

- 11.7. Log out of the servere machine and log in as the operator4 user. This time, the connection succeeds.

```
[root@servere ~]# logout
[student@workstation ~]$ ssh operator4@servere
...output omitted...
[operator4@servere ~]$
```

- 11.8. To confirm that the operator4 user can administer the system, use the sudo -i command to switch identity to the root user and restart the sshd service. Use redhat as the password.

```
[operator4@servere ~]$ sudo -i
[sudo] password for operator4: redhat
[root@servere ~]# systemctl restart sshd
[root@servere ~]# systemctl is-active sshd
active
```

12. Return to the workstation machine

```
[root@servere ~]# logout  
[operator4@servere ~]$ logout  
[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-selinux
```

Finish

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

```
[student@workstation ~]$ lab finish compreview-selinux
```

▶ Lab

Managing Compliance with OpenSCAP and Ansible

Manage compliance with a password policy by using OpenSCAP and Ansible.

Outcomes

- Configure and use an Ansible inventory file.
- Install OpenSCAP tools and the SCAP Security Guide.
- Create a tailoring file by using SCAP Workbench.
- Scan a system by using the customized policy.
- Generate and use an Ansible Playbook to remediate failed compliance checks.

Before You Begin

If you did not reset your `workstation` and `server` machines at the end of the last chapter, then save any work you want to keep from earlier exercises on those machines, and reset them now.

As the `student` user on the `workstation` machine, use the `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available.

```
[student@workstation ~]$ lab start compreview-openscap
```

Specifications

- Use a tailoring file to create a custom OpenSCAP profile based on the DISA STIG for Red Hat Enterprise Linux 9 profile that contains only the `Ensure PAM Enforces Password Requirements - Minimum Length` check. Specify a minimum password length of 12 characters.
- Perform an OpenSCAP scan on the `serverd` machine and use the results to generate a remediation playbook.
- Configure Ansible and apply the remediation playbook from the `workstation` machine to remediate the noncompliant `serverd` machine. Use an inventory file named `/home/student/inventory` on the `workstation` machine.

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-openscap
```

Finish

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

```
[student@workstation ~]$ lab finish compreview-openscap
```

► Solution

Managing Compliance with OpenSCAP and Ansible

Manage compliance with a password policy by using OpenSCAP and Ansible.

Outcomes

- Configure and use an Ansible inventory file.
- Install OpenSCAP tools and the SCAP Security Guide.
- Create a tailoring file by using SCAP Workbench.
- Scan a system by using the customized policy.
- Generate and use an Ansible Playbook to remediate failed compliance checks.

Before You Begin

If you did not reset your **workstation** and **server** machines at the end of the last chapter, then save any work you want to keep from earlier exercises on those machines, and reset them now.

As the **student** user on the **workstation** machine, use the **lab** command to prepare your environment for this exercise, and to ensure that all required resources are available.

```
[student@workstation ~]$ lab start compreview-openscap
```

- On the **workstation** machine, install the SCAP Workbench and SCAP Security Guide utilities.

```
[student@workstation ~]$ sudo dnf install -y scap-workbench scap-security-guide
[sudo] password for student: student
...output omitted...
Complete!
```

- On the **workstation** machine, customize the DISA STIG for Red Hat Enterprise Linux 9 profile. Set the new profile identifier to **xccdf_com.example_profile_compreview-rhel9**, disable all the rules, and then enable only the **Ensure PAM Enforces Password Requirements - Minimum Length** check. Specify a minimum password length of 12 characters. Store the resulting tailoring file on the **workstation** machine in the **/home/student/compreview-tailoring.xml** file.
 - On the **workstation** machine, start SCAP Workbench by running the **scap-workbench** command.

```
[student@workstation ~]$ scap-workbench
```

Chapter 12 | Comprehensive Review

SCAP Workbench detects that the SCAP Security Guide is already installed on the system and asks you to select the content to use.

In the **Select content to load** field, select RHEL 9 and click **Load Content**.

- 2.2. In the **Profile** field, select the [DRAFT] DISA STIG for Red Hat Enterprise Linux 9 (496) profile.

Click **Customize** to the right of that field.

- 2.3. In the **New Profile ID** field, enter `xccdf_com.example_profile_compreview-rhel9` and click **OK**.

The new window displays all the available rules.

- 2.4. Click **Deselect All** to clear all the rules.

- 2.5. Use the **Search** field to find and enable the **Ensure PAM Enforces Password Requirements - Minimum Length** rule in the **Set Password Quality Requirements** section. Specify 12 for the **minlen** value. Click **OK**.

- 2.6. Save the customization in a tailoring file. Select **File > Save Customization Only** and enter `compreview-tailoring.xml` for the file name in the `/home/student/` directory.

Close SCAP Workbench.

3. Scan the **serverd** machine for compliance with your customization of the DISA STIG for Red Hat Enterprise Linux 9 profile. Save the result on the **workstation** machine in the `/home/student/compreview-results.xml` file. Generate the HTML report of the scan and store it in the `/home/student/compreview-results.html` file on the **workstation** machine.

- 3.1. Copy the `compreview-tailoring.xml` tailoring file to the **serverd** machine. You need this file to scan the system.

```
[student@workstation ~]$ scp compreview-tailoring.xml student@serverd:  
...output omitted...
```

- 3.2. Log in to the **serverd** machine as the **student** user. No password is required.

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

- 3.3. Change to the **root** user. Use **student** as the password.

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

- 3.4. Install the **scap-security-guide** package.

```
[root@serverd ~]# dnf install -y scap-security-guide  
...output omitted...  
Complete!
```

- 3.5. Scan the system for compliance with your customization. Save the result in the /home/student/comprevew-results.xml file.

```
[root@serverd ~]# oscap xccdf eval \
--profile xccdf_com.example_profile_comprevew-rhel9 \
--tailoring-file /home/student/comprevew-tailoring.xml \
--results /home/student/comprevew-results.xml \
/usr/share/xml/scap/ssg/content/ssg-rhel9-ds.xml
...output omitted...
```

- 3.6. When the scan is complete, convert the /home/student/comprevew-results.xml file to HTML. Save the HTML report as /home/student/comprevew-results.html.

```
[root@serverd ~]# oscap xccdf generate report \
comprevew-results.xml > comprevew-results.html
[root@serverd ~]#
```

- 3.7. Use the scp command to copy the two files to the workstation machine. Use student as the password.

```
[root@serverd ~]# scp /home/student/comprevew-results.* student@workstation:
student@workstation's password: student
...output omitted...
```

- 3.8. Return to the workstation machine.

```
[root@serverd ~]# logout
[student@serverd ~]$ logout
[student@workstation ~]$
```

4. On the workstation machine, generate an Ansible Playbook to resolve the compliance issues that were detected in the previous step. Create an inventory file that contains the serverd host. Save the Ansible Playbook as /home/student/fix.yml and run it to resolve the compliance issues on the serverd host.

- 4.1. Use the oscap xccdf generate fix command to generate the Ansible Playbook. Save the playbook as /home/student/fix.yml.

```
[student@workstation ~]$ oscap xccdf generate fix \
--profile xccdf_com.example_profile_comprevew-rhel9 \
--tailoring-file comprevew-tailoring.xml \
--fix-type ansible \
--result-id "" \
comprevew-results.xml > fix.yml
[student@workstation ~]$
```

- 4.2. Create an inventory file that contains the serverd host.

```
[student@workstation ~]$ echo serverd > inventory
```

- 4.3. Edit the fix.yml file to set the become: true option.

```
...output omitted...
- hosts: all
  become: true
  vars:
  tasks:
  ...output omitted.
```

- 4.4. Use the `ansible-playbook` command to run the playbook. Use the `-K` flag and the `student` become password.

```
[student@workstation ~]$ ansible-playbook -K -i inventory fix.yml
BECOME password: student
...output omitted...
PLAY RECAP
*****
serverd : ok=3    changed=1    unreachable=0    failed=0
 skipped=0   rescued=0   ignored=0
```

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-openscap
```

Finish

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

```
[student@workstation ~]$ lab finish compreview-openscap
```