

RED HAT[®]
CONSULTING



RH415 - Red Hat Security

Linux in Physical, Virtual, and Cloud

Travis Michette

Version 1.0

Table of Contents

Introduction	1
Environment Overview and Launching an Instance	1
Accessing the System Externally	3
1. Managing Security and Risk	4
1.1. Managing Security and Risk	4
1.1.1. Risk Management	4
1.1.2. Managing Security	4
1.1.3. How Red Hat Can Help Manage Security	6
1.1.4. Red Hat Security Reporting	6
1.1.4.1. Red Hat Security Severity Ratings	6
1.1.5. Backporting Security Fixes	8
1.2. Reviewing Recommended Security Practices	9
1.2.1. Baseline Standard Operating Environment	9
1.2.2. Securing Services	9
1.2.3. Configuring SSH Key-Based authentication	9
1.2.4. Customizing Your SSH Service Configuration	9
1.2.5. Escalating User Privileges	9
2. Automating Configuration and Remediation with Ansible	10
2.1. Configuring Ansible for Security Automation	10
2.1.1. Ansible Concepts and Architecture	10
2.1.2. Installing Ansible	10
2.1.3. Managing a Host Inventory	11
2.1.4. Configuring Ansible Operation	11
2.1.5. Testing Ansible with Ad Hoc Commands	11
2.2. Remediating Issues with Ansible Playbooks	11
2.2.1. Reading Ansible playbooks	11
2.3. Managing Playbooks with Red Hat Ansible Tower	12
3. Protecting Data with LUKS and NBDE	14
3.1. Managing File System Encryption with LUKS	14
3.1.1. Encrypting Storage with Linux Unified Key Setup (LUKS)	14
3.1.2. Creation of Encrypted Devices at Installation	14
3.1.3. Encrypting with LUKS after Installation	14
3.2. Controlling File System Decryption with NBDE	14
3.2.1. Introducing Network-Bound Disk Encryption	14
3.2.2. Persistently Mounting LUKS File Systems	14
3.2.3. Unattended Device Decryption at Boot Time	14

3.2.4. Configuring Clevis and Tang	14
4. Restricting USB Device Access	16
4.1. Controlling USB Access with USBGuard	16
4.1.1. Introduction to USBGuard	16
4.1.1.1. Installing USBGuard	16
4.1.2. Using USBGuard	17
4.1.2.1. Using the USBGuard Command-Line Interface (CLI)	17
4.1.2.2. Creating an Initial Rule Set	18
4.1.2.3. Dynamically and Persistently Allowing/Blocking Devices with USBGuard	18
4.1.3. Whitelisting and Blacklisting Devices	20
4.1.3.1. Securing Access to the USBGuard IPC	22
4.1.4. Applying Rules to Specific Devices and Classes of Device	22
5. Controlling Authentication with PAM	24
5.1. Auditing the PAM Configuration	24
5.1.1. Describing PAM	24
5.1.2. Configuring PAM	25
5.2. Modifying the PAM Configuration	26
5.2.1. Preparing for Configuration update	26
5.2.2. Using authconfig to Configure pam	27
5.2.3. Manually Configuring PAM	27
5.3. Configuring Password Quality Requirements	27
5.3.1. Describing the pam_pwquality Module	27
5.4. Limiting Access After Failed Logins	27
5.4.1. Locking Accounts with Multiple Failed Logins	27
6. Recording System Events with Audit	28
6.1. Configuring Audit to Record System Events	28
6.1.1. The Linux Audit System	28
6.1.2. Auditing Your System with auditd	28
6.1.3. Configuring auditd	28
6.1.3.1. Adjusting auditd Settings to Manage Storage	29
6.1.3.2. Adjusting auditd Settings to Tune Performance	29
6.1.4. Remote Logging with auditd	30
6.2. Inspecting Audit Logs	31
6.2.1. Interpreting Audit Messages	31
6.2.2. Searching for Events	31
6.2.3. Reporting on Audit Messages	32
6.2.4. Tracing a Program	32
6.3. Writing Custom Audit Rules	33

6.3.1. Adding Rules	33
6.3.1.1. Setting File System Rules (Watches)	33
6.3.1.2. Setting System Call Rules	34
6.3.1.3. Setting Control Rules	35
6.3.2. Removing Rules	35
6.3.3. Inspecting Rules	35
6.3.4. Making Rules Immutable	35
6.3.5. Persistent Rules	35
6.4. Enabling Prepackaged Audit Rulesets	35
6.4.1. Prepackaged Audit Rulesets	35
6.4.2. Full Terminal Keystroke Logging	36
7. Monitoring File System Changes	37
7.1. Detecting File System Changes with AIDE	37
7.1.1. Analyzing File System Changes with AIDE	37
7.1.1.1. Installing AIDE	37
7.1.1.2. Configuring AIDE	37
7.1.1.3. Initializing the AIDE Database	38
7.1.1.4. Verifying Integrity with AIDE	38
7.1.1.5. Updating the AIDE Database	38
7.2. Investigating File System Changes with AIDE	39
7.2.1. Combining AIDE and Audit	39
7.2.2. Configuring AIDE and Audit	39
7.2.3. Investigating File System Changes	39
7.2.3.1. Interpreting Audit Events	39
8. Mitigating Risk with SELinux	42
8.1. Enabling SELinux from Disabled State	42
8.1.1. Reviewing Basic SELinux Concepts	42
8.1.1.1. Changing SELinux Contexts for Files and Directories	43
8.1.1.2. Defining SELinux Default File Context Rules	43
8.1.1.3. Labeling SELinux Ports	43
8.1.1.4. Using SELinux Booleans	43
8.1.1.5. Accessing the Documentation	43
8.1.2. Configuring SELinux Modes	43
8.1.3. Enabling SELinux from Disabled Mode	43
8.1.3.1. Reviewing SELinux Access Violation Audit Events	43
8.1.3.2. Using Permissive Domains	43
8.1.3.3. Using Ansible for Remediation	43
8.2. Controlling Access with Confined Users	43

8.2.1. Defining SELinux Users	43
8.2.1.1. Mapping Linux Users to SELinux Users	44
8.2.1.2. Comparing the SELinux Users	44
8.2.1.3. SELinux User Booleans	44
8.2.2. Confining User Accounts	44
8.2.2.1. Confining Different User Accounts	44
8.2.2.2. Confining System Administrators	44
8.2.2.3. Confining Staff Users	44
8.3. Auditing the SELinux Policy	45
8.3.1. Introducing the SELinux Policy	45
8.3.2. Analyzing the Targeted Policy	45
8.3.2.1. Interpreting the Allow Rules	45
8.3.2.2. Disabling and Enabling the "dontaudit" Rules	46
8.3.2.3. Creating Custom Policy Modules with audit2allow	46
8.3.2.4. Analyzing Domain Transitions	46
8.3.2.5. Analyzing File Transitions	46
9. Managing Compliance with OpenSCAP	47
9.1. Installing OpenSCAP	47
9.1.1. OpenSCAP and Security Compliance in Red Hat Enterprise Linux	47
9.1.1.1. Security Compliance Tools	47
9.1.2. SCAP Security Guide	47
9.1.3. SCAP Workbench	48
9.1.3.1. Local System OpenSCAP Scan	48
9.2. Scanning and Analyzing Compliance	49
9.2.1. Introducing the oscap Command	49
9.2.2. Scanning a System for Compliance	50
9.2.2.1. Generating the HTML Report	51
9.3. Customizing OpenSCAP Policy	51
9.3.1. Customizing a SCAP Security Guide Profile	51
9.3.1.1. Creating a Tailoring File	51
9.3.2. Scanning a System Using a Profile Customized with a Tailoring File	51
9.4. Remediating OpenSCAP Issues with Ansible	52
9.4.1. Generating a Remediation Ansible Playbook	52
9.4.1.1. Creating an Ansible Playbook for a Profile	52
9.4.1.2. Creating an Ansible Playbook from a Result XML File	53
9.4.1.3. Adjusting Variables in the Remediation Ansible Playbook	53
9.4.2. Running a Remediation Ansible Playbook	53
9.4.2.1. Filtering Tasks	54

9.4.3. Checking Compliance During Installation	54
10. Automating Compliance with Red Hat Satellite	55
10.1. Configuring Red Hat Satellite for OpenSCAP	55
10.1.1. Security Compliance and Management with Red Hat Satellite	55
10.1.2. Integrating OpenSCAP with Red Hat Satellite	55
10.1.2.1. Installing Satellite Server with the OpenSCAP Plugin	55
10.1.2.2. Uploading OpenSCAP Content to the Satellite Server	55
10.1.2.3. Importing OpenSCAP Puppet Module in Satellite Server	56
10.1.2.4. Initiating a Puppet Agent Run on a Host	56
10.1.2.5. Initiating a Puppet Agent Run using Remote Execution	57
10.2. Scan OpenSCAP Compliance with Red Hat Satellite	57
10.2.1. Performing OpenSCAP Scans using Red Hat Satellite	57
10.2.2. Managing Compliance Policies	57
10.2.2.1. Creating a Compliance Policy	58
10.2.2.2. Creating a SCAP Policy	58
10.2.3. Running Compliance Scans	58
10.2.3.1. Running an OpenSCAP Scan Manually	58
10.2.3.2. Running an OpenSCAP Scan using foreman_scap_client	58
10.2.4. Reviewing OpenSCAP Scan Results in Satellite Server	59
10.2.4.1. Viewing Compliance Policy Dashboard	59
10.2.5. Evaluating OpenSCAP Scan Report	59
10.2.5.1. Viewing Compliance Report	60
10.2.5.1.1. Viewing Compliance Report in Satellite Server	60
10.3. Customize the OpenSCAP Policy in Red Hat Satellite	60
10.3.1. Customizing SCAP Policy in Red Hat Satellite	60
10.3.1.1. Uploading a Tailoring File	60
10.3.1.2. Uploading a Tailoring File Using Satellite WebUI	60
10.3.1.3. Assigning a Tailoring File to a Policy	60
10.3.1.4. Assigning a Tailoring File using the Satellite WebUI	60
10.3.2. Executing a Compliance Scan Using a Customized Policy	60
11. Analyzing and Remediating Issues with Red Hat Insights	61
11.1. Registering Systems with Red Hat Insights	61
11.2. Reviewing Red Hat Insights Reports	61
11.3. Automating Issue Remediation	61
Appendix A: Exam Objectives	62
A.1. Use Red Hat Ansible Engine	63
A.1.1. Install Red Hat Ansible Engine on a control node	63
A.1.2. Configure managed nodes	66

A.1.3. Configure simple inventories	66
A.1.4. Perform basic management of systems	67
A.1.5. Run a provided playbook against specified nodes	67
A.2. Configure intrusion detection	69
A.2.1. Install AIDE	69
A.2.2. Configure AIDE to monitor critical system files	69
A.3. Configure encrypted storage	69
A.3.1. Encrypt and decrypt block devices using LUKS	69
A.3.2. Configure encrypted storage persistence using NBDE	72
A.3.3. Change encrypted storage passphrases	75
A.4. Restrict USB devices	75
A.4.1. Install USBGuard	75
A.4.2. Write device policy rules with specific criteria to manage devices	76
A.4.3. Manage administrative policy and daemon configuration	78
A.5. Manage system login security using pluggable authentication modules (PAM)	78
A.5.1. Configure password quality requirements	81
A.5.2. Configure failed login policy (Chapter 5 Lab)	81
A.5.3. Modify PAM configuration files and parameters	81
A.6. Configure system auditing	82
A.6.1. Write Rules to Log Auditable Events (GE: P222)	82
A.6.2. Enable pre-packaged rules	84
A.6.3. Produce audit reports	85
A.7. Configure SELinux (Chapter 8 of Student Guide)	86
A.7.1. Enable SELinux on a host running a simple application	89
A.7.2. Interpret SELinux violations and determine remedial action	91
A.7.3. Restrict user activity with SELinux user mappings (Guided Exercise - P301)	91
A.7.4. Analyze and correct existing SELinux configurations	95
A.8. Enforce security compliance (Chapter 9)	97
A.8.1. Install OpenSCAP and Workbench	98
A.8.2. Use OpenSCAP and Red Hat Insights to scan hosts for security compliance	100
A.8.3. Use OpenSCAP Workbench to tailor policy	101
A.8.4. Use OpenSCAP Workbench to scan an individual host for security compliance	102
A.8.5. Use Red Hat Satellite server to implement an OpenSCAP policy	102
A.8.6. Apply OpenSCAP remediation scripts to hosts	102

Introduction

VMs running on FoundationX share an external 172.25.250.0/24 network, with a gateway of 172.25.250.254 (**workstation.lab.example.com**). DNS services for the private network are provided by 172.25.250.254 (**workstation**), so the **Workstation** VM must be started first.

Environment Overview and Launching an Instance

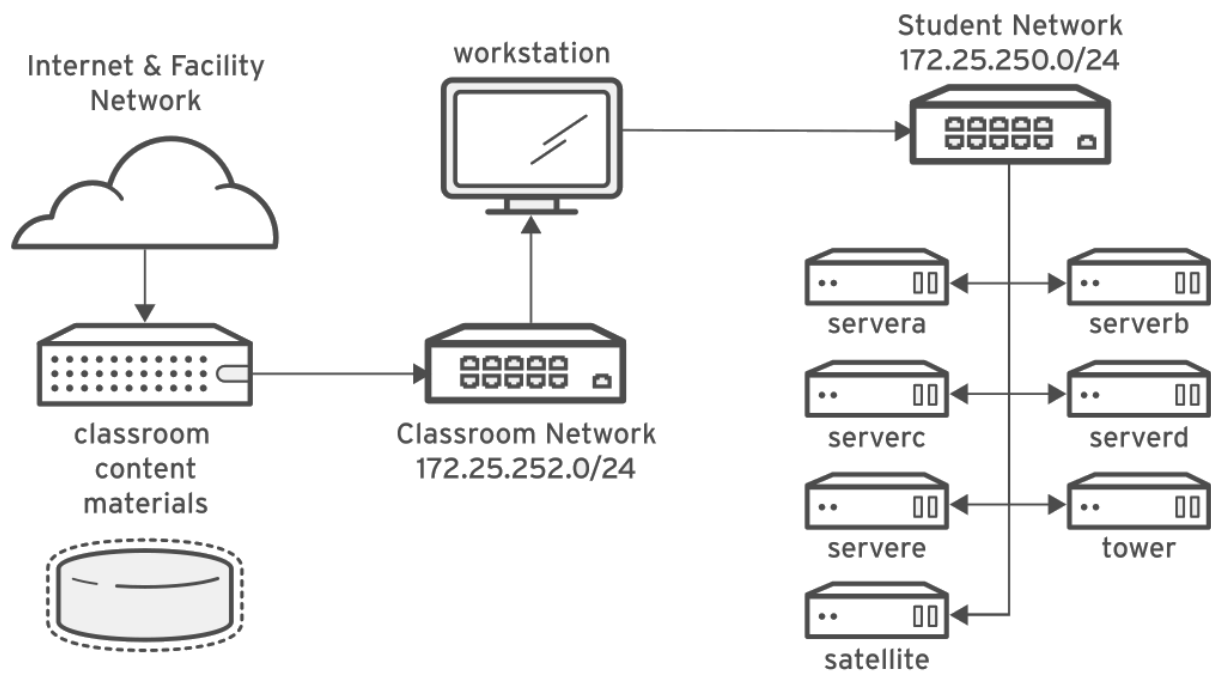


Figure 1. Classroom Environment Layout

There are eight systems used to comprise the entire classroom environment (in addition to **Workstation**). The listing of machines are:

- **servera**
- **serverb**
- **serverc**
- **serverd**
- **servere**
- **satellite**
- **tower**

Table 1. Security Classroom Layout and Information

Machine Name	IP Address	Role
servera.lab.example.com	172.25.250.10	Managed Server "A"
serverb.lab.example.com	172.25.250.11	Managed Server "B"
serverc.lab.example.com	172.25.250.12	Managed Server "C"
serverd.lab.example.com	172.25.250.13	Managed Server "D"
servere.lab.example.com	172.25.250.16	Managed Server "E"
satellite.lab.example.net	172.25.250.15	Red Hat Satellite 6 Server
tower.lab.example.net	172.25.250.14	Red Hat Ansible Tower server
workstation.lab.example.com / workstation0.example.com	172.25.250.254 / 172.25.252.250	Graphical Workstation as Student Desktop
classroom.example.com	172.25.254.254 / 172.25.252.254 / 172.25.253.254	Classroom utility server
foundation0.ilt.example.com / foundationX.ilt.example.com	172.25.254.250 / 172.25.253.250 / 172.25.254.X	Physical System



The **classroom** server acts as a NAT router for the classroom network. It provides DNS, DHCP, HTTP, and other services. It is also known by **content.example.com** and **materials.example.com**.

Classroom Credentials

System(s)	Username	Password
Student Systems (regular user)	student	student
Student Systems (Root user)	root	redhat
Satellite	admin	redhat
Ansible Tower	admin	redhat



The setup scripts are meant to catch up labs between chapters. It should be noted that labs are meant to be successive for this course.



The Workstation VM must be the first machine powered on. After workstation is up, the Satellite machine should be powered on before any of the other machines. After Workstation and Satellite have both been powered on and running, it is safe to start all other VMs with **rht-vmctl start all** command.



Grading/Setup scripts located <http://content/courses/rh415/rhel7.5/grading-scripts/>. The Ansible playbooks are located at <http://content/courses/rh415/rhel7.5/infrastructure/>. Overall classroom files are <http://content/courses/rh415/rhel7.5/>.

Accessing the System Externally

If using a Macbook or another system on the classroom network, it will be assigned an IP address. The way to access workstation is with the **172.25.252.X** IP address. Once on workstation, you can get to other systems. The other method is to access **FoundationX** directly, which can be done with the **172.25.254.X** IP address.



For a Mac/Linux system, you can use "sudo route -n add/delete 172.25.0.0/16" with a gateway of **172.25.254.254** to route traffic across multiple interfaces.



The **Foundation0** system IP address **172.25.254.250** is the instructor system.



Grading scripts get downloaded locally to **/usr/local/lib** and several executables for the environment also live in **/usr/local/bin/lab**



To preserve system resources, the **Satellite** and **Tower** VMs can be turned except when they are needed to be used in Chapter 8.

1. Managing Security and Risk

1.1. Managing Security and Risk

1.1.1. Risk Management

Risk is any event presenting the possibility of loss (delays to services, financial, or productivity). Continuous risk management is a process for taking proactive approaches to discovering potential risks, collecting facts, and taking action to resolve risks.



Figure 2. Continuous Risk Management Lifecycle

1.1.2. Managing Security

Managing security requires continuous activities and represent multiple phases in the security lifecycle as shown in the diagram below.

Security and Risk Management Lifecycle

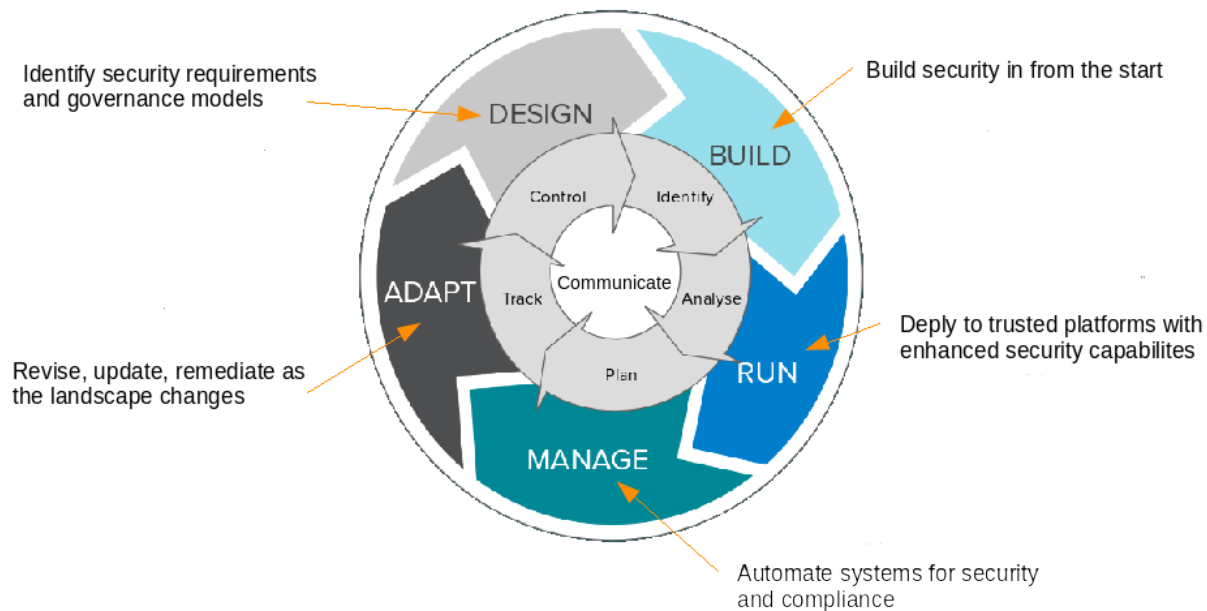


Figure 3. Continuous Security and Risk Management Lifecycle

Continuous Security

Security should be proactive and reactive. Each stage in the diagram above must be considered as part of the lifecycle.

- **Design**: Develop security requirements, processes and procedures, and communicate to all team members. This is when security policies will be developed for compliance.
- **Build**: Build features into apps and infrastructure
 - Automate and integrate security testing into build and deployment processes.
 - Define security profiles and automate configuration of profiles
- **Run**: Run on trusted platforms using built-in security features (SELinux)
- **Manage**: Maintain up-to-date listing of assets monitoring access and usage. Use a management and monitoring solution.
- **Adapt**: Use analytics and automation to adapt/revise/updates/remediate any changes.

1.1.3. How Red Hat Can Help Manage Security

Securing operating systems has changed, but one of the key components is keeping the operating system updated with the latest security patches. While this isn’t the only task anymore, it is still a key component to being proactive. Red Hat has a security team dedicated to analyze threats and vulnerabilities and provides regular updates, patches, and publishes bug advisories minimizing time from vulnerability discovery through patch deployment.

1.1.4. Red Hat Security Reporting

Red Hat Security Response

Red Hat has a Security Response team that is the point of contact for all customers regarding security issues for Red Hat products. Any security related issues should be sent to **secalert@redhat.com**. Only members of the Red Hat Product Security team has access to this e-mail mailing list.

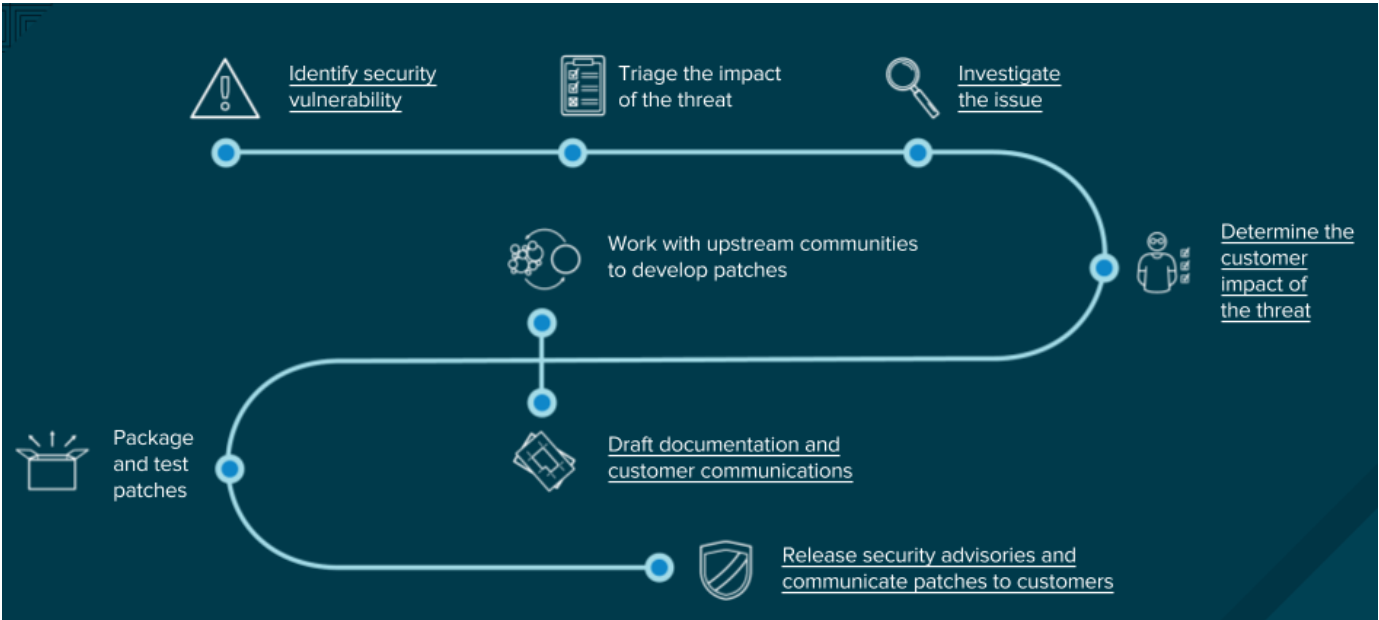


Figure 4. Security Risk Awareness Workflow

The image above depicts the workflow Red Hat uses to notify and communicate information to end-users regarding vulnerabilities that will affect Red Hat products.

1.1.4.1. Red Hat Security Severity Ratings

Red Hat uses a four-point scale (Low, Moderate, Important, and Critical), as well as *Common Vulnerability Scoring System (CVSS)* base scores. Using the four-point scale as a rating system, a prioritized risk assessment can be made so system upgrades can be scheduled. The scale allows system administrators to judge the severity of the issues and the potential risk associated with remaining unpatched.

Table 2. Issue Severity Classification

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

A can contain fixes for more than one vulnerability and for packages for more than a single product. Each issue in an advisory has impact ratings for the affected products. The overall severity rating is

based on the highest severity level for an individual issue.

1.1.5. Backporting Security Fixes

Backporting of fixes is a tradeoff between fixing security vulnerabilities and not breaking functionality by forcing updates to a newer version of software with different features and APIs. Red Hat often backports patches and security related issues without changing the major/minor version of the software.

Backporting Security Fix Procedure

- Identify fixes and isolate from other changes
- Make sure fixes don't introduce new or unwanted side effects
- Apply fixes to previously released versions

Backporting can create confusion to end users when only looking at versioning numbers. It is important to not just look at version numbers of the software packages, especially when using automated security scanning/auditing tools.

Red Hat CVEs and Errata

Red Hat works with the Common Vulnerabilities and Exposures (CVE) project to report and track security-related issues. Each CVE consists of an ID number with descriptions and issue summary. Red Hat also supplies OVAL definitions to third-party tools that can be used to determine vulnerability status and eliminate confusion resulting from backporting. After releasing a security fix, bug fix, or feature enhancement Red Hat will publish errata announcements.

Table 3. Red Hat Errata Announcements

Red Hat Announcement Type	Description
Red Hat Security Advisory (RHSA)	Packages to fix a security-related issue.
Red Hat Bug Fix Advisory (RHBA)	Packages to fix non-security related issues.
Red Hat Enhancement Advisory (RHEA)	Packages to add additional enhancements and new features.

Example 1. Using YUM to Manage Security Errata

Yum can be used to manage security related features and can be used to search, list, display, and install security errata.

Listing 1. Identifying Security Issues with YUM

1.2. Reviewing Recommended Security Practices

1.2.1. Baseline Standard Operating Environment

1.2.2. Securing Services

1.2.3. Configuring SSH Key-Based authentication

1.2.4. Customizing Your SSH Service Configuration

1.2.5. Escalating User Privileges

2. Automating Configuration and Remediation with Ansible

2.1. Configuring Ansible for Security Automation

Ansible can be used to manage security and configurations across a large number of systems to ensure consistent configuration and patching. Ansible provides a scalable solution ensuring all machines are in compliance.

Reasons for Ansible

1. Simple to use
2. Human readable Automation
3. Security tools with Ansible Playbooks/modules are provided



Ansible Engine will be used on a control node and run playbooks across systems to configure and update systems.

2.1.1. Ansible Concepts and Architecture

Ansible uses high-level **plays** which specify a desired state of a system. Ansible **playbooks** are files that contain one or more plays and are stored in the YAML format. Plays specify hosts and tasks to be performed across the hosts. The tasks are executed in sequential order and generally runs a small piece of code called a **module**. Tasks, plays, and playbooks are *idempotent* which means a playbook can be run multiple times on a host without making changes when the system is already in the correct state.

There are two (2) components to the Ansible architecture:

- **Control Node:** Host that has Ansible installed and where Ansible playbooks are initially executed.
- **Managed Host:** Systems that Ansible controls

Ansible uses an **inventory** to manage hosts and host groups. Inventories can be:

- Static Text files
- Dynamically determined by scripts



Hosts or host groups that Ansible playbooks run on must be defined in the current Ansible inventory.

2.1.2. Installing Ansible

2.1.3. Managing a Host Inventory

2.1.4. Configuring Ansible Operation

Table 4. Ansible Privilege Directives

Directive	Description
inventory	Specifies the path to the inventory file.
remote_user	The name of the user to log in as on the managed hosts; if not specified, current username is used.
ask_pass	Whether or not to prompt for an SSH password. Can be false if using SSH public key authentication.
become	Whether to automatically switch user on the managed host (typically to root) after connecting. This can also be specified by a play.
become_method	How to switch user (typically sudo, which is the default, but su is an option).
become_user	The user to switch to on the managed host (typically root, which is the default).
become_ask_pass	Whether to prompt for a password for your become_method. Defaults to false.

2.1.5. Testing Ansible with Ad Hoc Commands



Need to use **man ansible**, **man ansible-inventory**, and **man ansible-config**, as additional references.

2.2. Remediating Issues with Ansible Playbooks

2.2.1. Reading Ansible playbooks

Listing 2. Simple Ansible Playbook (YAML File Content)

```
---
- name: Name for the playbook
  hosts: server_groups
  tasks:
    - Task 1
    - Task 2
    - Task 3
```

Listing 3. Creating User Demo_User

```
---
- name: Configure Demo User for Systems
  hosts: server1.example.com
  tasks:
    - name: Create Demo User with UID 4000
      user:
        name: demo_user
        uid: 4000
        state: present
```

2.3. Managing Playbooks with Red Hat Ansible Tower

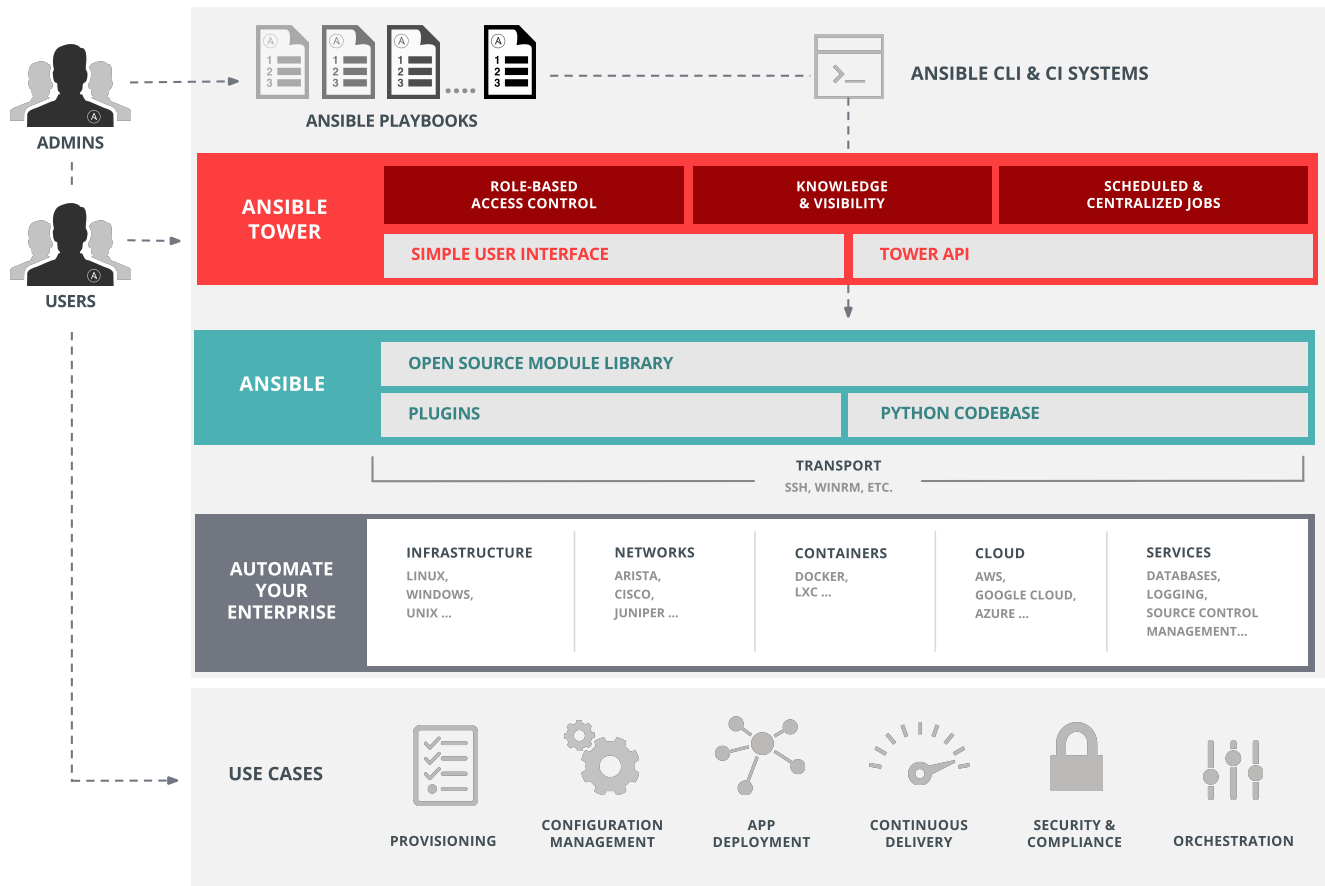


Figure 5. Ansible Tower Architecture

Ansible Tower Objects

Ansible Tower uses **Organization** to group items such as **Inventories**, **Teams**, and **Projects**. The Inventory defines and groups managed hosts.

High Level Steps for Running Ansible Playbook with Tower

1. Login as **admin** to the Tower Dashboard
2. Create a user and add it to appropriate Organization
3. Create a credential for newly created user. Specify privilege escalation settings and grant **use** permission.
4. Create a project and add user with the **Use** Privilege
5. Create an inventory and grant **Ad Hoc** and **Use** access to the intended user
6. Add a host group to the inventory defined with the managed Hosts
7. Login as the new user added in **Step 1**
8. Create a new job template specifying appropriate **Project, Inventory, Playbook, and Credential**. Check the **Enable Privilege Escalation** box.
9. Launch the new job template
10. Review the logs of the Ansible play in the dashboard. :imagesdir: images/

3. Protecting Data with LUKS and NBDE

3.1. Managing File System Encryption with LUKS

3.1.1. Encrypting Storage with Linux Unified Key Setup (LUKS)

Red Hat Enterprise Linux (RHEL) supports block device encryption with Linux Unified Key Setup (LUKS). Encrypting data mitigates risk in the event that the device or disk is stolen. Encryption of drives is easiest during installation time, but LUKS can be configured post installation.



Setting up LUKS encryption requires that the file systems on the device be reformatted.

3.1.2. Creation of Encrypted Devices at Installation

When performing interactive installations of RHEL, the **Encrypt** box should be checked during the partition creation to encrypt the partition. In this method, the passphrase must be manually entered every time the system boots.



Automated installations using kickstart

3.1.3. Encrypting with LUKS after Installation

3.2. Controlling File System Decryption with NBDE

3.2.1. Introducing Network-Bound Disk Encryption

Network-bound Disk Encryption (NBDE) automates the decryption of encrypted disks without manually entering a passphrase at boot time in a secure way. The usage of NBDE can automatically decrypt LUKS devices containing both root and non-root filesystems.

3.2.2. Persistently Mounting LUKS File Systems

3.2.3. Unattended Device Decryption at Boot Time

3.2.4. Configuring Clevis and Tang

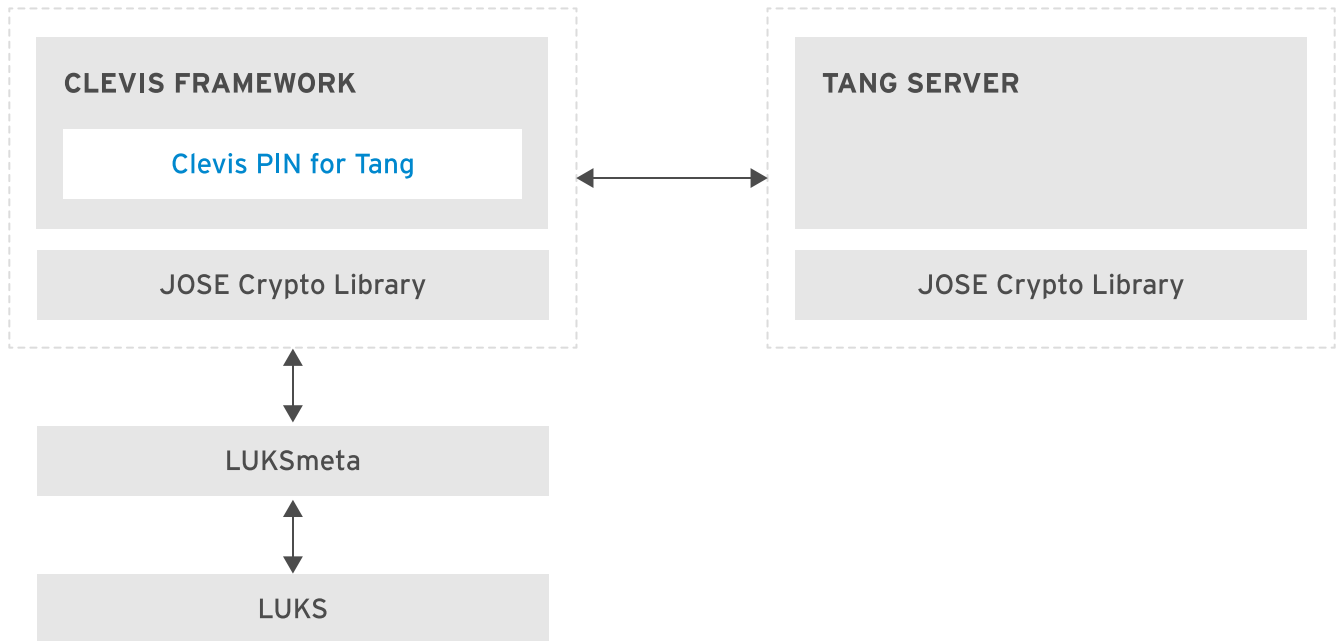


Figure 6. Clevis and Tang



A Tang server can be tested and see if it is accessible, using **curl -f http://tangserver_address/adv** command.

Clevis Decryption



If you are configuring Clevis to decrypt a root file system, you must recreate the **initramfs** with the **dracut -f** command. For any other block devices, path based execution should be enabled by **clevis-luks-askpass.path**.

```
# systemctl enable clevis-luks-askpass.path
```

4. Restricting USB Device Access

4.1. Controlling USB Access with USBGuard

4.1.1. Introduction to USBGuard

USBGuard is a software framework protecting systems against rogue USB devices by implementing whitelisting and blacklisting capabilities based on attributes of a USB device. The Linux kernel USB device authorization system is used to enforce settings configured with USBGuard allowing you to control access to USB devices.

USBGuard framework provides:

- Daemon component, which has IPC interface for dynamic interaction and policy enforcement
- The command-line interface (CLI) used to interact with running USBGuard instances
- The rule language, used to write USB device authorization policies
- The C++ API used to interact with the daemon component implemented by a shared library.

4.1.1.1. Installing USBGuard

The **usbguard** package provides the *usbguard-daemon* which works with the kernel and the **usbguard** command to manage USBGuard policies and devices.

Other Optional Packages for USBGuard include:

- **usbutils**: Provides **lsusb** for displaying information about USB buses on the system and devices connected to them.
- **udisks2**: Interface to enumerate and perform operations on disks and storage devices. This specifically will provide access to **udiskctl** CLI tool which interacts with the **udisksd** daemon. When **udiskctl** is used with the **status** command, it provides high-level information about disk drives and block devices.

Listing 4. Installing USBGuard

```
# yum install usbguard
```

Listing 5. Installing USB Supporting Packages

```
# yum install usbutils udisks2
```

4.1.2. Using USBGuard

Rule Targets

The target of a rule specifies whether the device will be authorized. There are three types of recognized targets.

Table 5. Target Rule Types

Rule Type	Description
allow	Authorize the device. The device and its interfaces will be allowed to communicate with the system.
block	Do not authorize the device. The device is visible to the system but will remain in a blocked state until it is authorized.
reject	Deauthorize and remove the device from the system. The device will have to be re-inserted to become visible to the system again.

4.1.2.1. Using the USBGuard Command-Line Interface (CLI)

The **usbguard** command provides a CLI to **usbguard** daemon.

Table 6. Common *usbguard* Subcommands

Subcommands	Description
list-devices	List all USB devices recognized by the USBGuard daemon.
allow-device id	Authorize a device identified by the device id to interact with the system.
block-device id	Deauthorize a device identified by the device id.
reject-device id	Deauthorize and remove a device identified by the device id.
list-rules	List the rule set (policy) used by the USBGuard daemon.
append-rule rule	Append the new rule after a rule with the specified rule id.
remove-rule id	Remove a rule identified by the rule id from the rule set.
generate-policy	Generate a rule set (policy) which authorizes the currently connected USB devices.

4.1.2.2. Creating an Initial Rule Set

Persistent USBGuard rules are stored in `/etc/usbguard/rules.conf`. The **usbguard generate-policy** command can be used to generate an initial set of rules authorizing currently connected devices. After generating policies, the USBGuard daemon must be restarted.

Listing 6. Generating Initial USBGuard Policy

```
# usbguard generate-policy > /etc/usbguard/rules.conf
# systemctl restart usbguard
```

Listing 7. Listing USBGuard Rules

```
# usbguard list-rules 1: allow id 1d6b:0002 serial "0000:00:14.0" name "xHCI Host
Controller" hash "jEP/6WzviqdJ5VSeTUY8PatCNBKeaREvo20qdpLND/o=" parent-hash
"G1ehGQdrL3dJ9HvW9w2HdC//pk87pKzFE1WY25bq8k4=" with-interface 09:00:00
2: allow id 1d6b:0003 serial "0000:00:14.0" name "xHCI Host Controller" hash
"3Wo3XWDgen1hD5xM3PSN13P98kLp1RUTgGQ5HSxtf8k=" parent-hash
"G1ehGQdrL3dJ9HvW9w2HdC//pk87pKzFE1WY25bq8k4=" with-interface 09:00:00
3: allow id 1050:0403 serial "" name "Yubikey 4 OTP+U2F" hash
"f0LUVUkaTk2LxG11CILS+oGoXlzsJrhTDQzmV9GHSRQ=" parent-hash
"jEP/6WzviqdJ5VSeTUY8PatCNBKeaREvo20qdpLND/o=" via-port "1-6" with-interface { 03:01:01
03:00:00 }
4: allow id 04f2:b596 serial "" name "Integrated Camera" hash
"iRuCmPLLZ2ZxVmtXoKqTSMRh5edyazRVJzf0DynAzkY=" parent-hash
"jEP/6WzviqdJ5VSeTUY8PatCNBKeaREvo20qdpLND/o=" via-port "1-8" with-interface { 0e:01:00
0e:02:00 0e:02:00 0e:02:00 0e:02:00 0e:02:00 0e:02:00 0e:02:00 0e:02:00 0e:02:00
0e:02:00 0e:02:00 }
5: allow id 138a:0090 serial "886a88ed14f9" name "" hash
"HH2ou1EU3eoA8eE58HBzpb8VXZyLvV10LKfS/7cN/30=" parent-hash
"jEP/6WzviqdJ5VSeTUY8PatCNBKeaREvo20qdpLND/o=" with-interface ff:00:00
6: allow id 8087:0a2b serial "" name "" hash
"AKESx3Z6zyDfDo9JPXqCbKiTQ1jmuAgbRc5qGDUwOKc=" parent-hash
"jEP/6WzviqdJ5VSeTUY8PatCNBKeaREvo20qdpLND/o=" via-port "1-14" with-interface { e0:01:01
e0:01:01 e0:01:01 e0:01:01 e0:01:01 e0:01:01 e0:01:01 }
```

4.1.2.3. Dynamically and Persistently Allowing/Blocking Devices with USBGuard

Devices can be added dynamically from the **usbguard** CLI utilities. These devices can be added/blocked/rejected either for the session (non-persistent) or permanently by adding it to the USBGuard rules file.

Steps to Modify USBGuard Rules

1. List Devices with **usbguard list-devices**
2. Locate the device you want to modify

3. Use **usbguard** <**allow-device** / **block-device** / **reject-device**> <**Device#**> to allow, block, or reject the device.
4. If you want the rule to be persistent, use the **-p** option on the command line to make the change persistent and add it to the **/etc/usbguard/rules.conf** file.
5. Check to ensure the policy was changed and the device is allowed.

USBGuard Device Management

USBGuard can be used to allow, block, and reject USB devices based on a variety of parameters. The easiest thing to do is to attach the device to the system and build the rules from the attached devices. To make the change permanent, add the **-p** option to the command line. Use the USBGuard commands for **allow**, **block**, **reject** along with the DEVICE ID number to perform the desired task.

Listing 8. Rejecting a Device



```
# usbguard reject-device ID
```

Listing 9. Blocking a Device

```
# usbguard block-device ID
```

Listing 10. Allowing a Device Permanently

```
# usbguard allow-device -p ID
```

Listing 11. Using the USBGuard CLI to Modify Rules

```
# usbguard list-devices
7: allow id 1d6b:0002 serial "0000:00:14.0" name "xHCI Host Controller" hash
"jEP/6WzviqdJ5VSeTUY8PatCNBKeaREvo20qdp1ND/o=" parent-hash
"G1ehGQdr13dJ9HvW9w2HdC//pk87pKzFE1WY25bq8k4=" via-port "usb1" with-interface 09:00:00
8: allow id 1d6b:0003 serial "0000:00:14.0" name "xHCI Host Controller" hash
"3Wo3XWDgen1hD5xM3PSN13P98kLp1RUTgGQ5HSxtf8k=" parent-hash
"G1ehGQdr13dJ9HvW9w2HdC//pk87pKzFE1WY25bq8k4=" via-port "usb2" with-interface 09:00:00
9: allow id 1050:0403 serial "" name "Yubikey 4 OTP+U2F" hash
"f0LUVUkaTk21xG11C1LS+oGoXlzsJrhTDQzmV9GHSRQ=" parent-hash
"jEP/6WzviqdJ5VSeTUY8PatCNBKeaREvo20qdp1ND/o=" via-port "1-6" with-interface { 03:01:01
03:00:00 }
10: allow id 04f2:b596 serial "" name "Integrated Camera" hash
"iRuCmPLL2ZxVmtXoKqTSMRh5edyazRVJzf0DynAzkY=" parent-hash
"jEP/6WzviqdJ5VSeTUY8PatCNBKeaREvo20qdp1ND/o=" via-port "1-8" with-interface { 0e:01:00
0e:02:00 0e:02:00 0e:02:00 0e:02:00 0e:02:00 0e:02:00 0e:02:00 0e:02:00 0e:02:00 0e:02:00
0e:02:00 0e:02:00 }
11: allow id 138a:0090 serial "886a88ed14f9" name "" hash
```

```

"HH2ou1EU3eoA8eE58HBzpb8VXZyLvV10LKfS/7cN/30=" parent-hash
"jEP/6WzviqdJ5VSeTUY8PatCNBKeaREvo20qdp1ND/o=" via-port "1-9" with-interface ff:00:00
12: allow id 8087:0a2b serial "" name "" hash
"AKEsx3Z6zyDfDo9JPXqCbKiTQ1jmuAgbRc5qGDUwOKc=" parent-hash
"jEP/6WzviqdJ5VSeTUY8PatCNBKeaREvo20qdp1ND/o=" via-port "1-14" with-interface { e0:01:01
e0:01:01 e0:01:01 e0:01:01 e0:01:01 e0:01:01 e0:01:01 }
16: block id 154b:0095 serial "20814765" name "USB 3.0 FD" hash
"pPa+c6J01Gy4ATG+6L5HbFowlVK292sSkDZ4iHsYpCA=" parent-hash
"3Wo3XWDgen1hD5xM3PSN13P98kLp1RUTgGQ5HSxtf8k=" via-port "2-5" with-interface 08:06:50

# usbguard allow-device 16 -p

# usbguard list-devices
7: allow id 1d6b:0002 serial "0000:00:14.0" name "xHCI Host Controller" hash
"jEP/6WzviqdJ5VSeTUY8PatCNBKeaREvo20qdp1ND/o=" parent-hash
"G1ehGQdr13dJ9HvW9w2HdC//pk87pKzFE1WY25bq8k4=" via-port "usb1" with-interface 09:00:00
8: allow id 1d6b:0003 serial "0000:00:14.0" name "xHCI Host Controller" hash
"3Wo3XWDgen1hD5xM3PSN13P98kLp1RUTgGQ5HSxtf8k=" parent-hash
"G1ehGQdr13dJ9HvW9w2HdC//pk87pKzFE1WY25bq8k4=" via-port "usb2" with-interface 09:00:00
9: allow id 1050:0403 serial "" name "Yubikey 4 OTP+U2F" hash
"f0LUVUkaTk2lxG11CILs+oGoXlzsjrhtDQzmV9GHSRQ=" parent-hash
"jEP/6WzviqdJ5VSeTUY8PatCNBKeaREvo20qdp1ND/o=" via-port "1-6" with-interface { 03:01:01
03:00:00 }
10: allow id 04f2:b596 serial "" name "Integrated Camera" hash
"iRuCmPLL2Z2xVmtXoKqTSMRh5edyazRVJzf0DynAzkY=" parent-hash
"jEP/6WzviqdJ5VSeTUY8PatCNBKeaREvo20qdp1ND/o=" via-port "1-8" with-interface { 0e:01:00
0e:02:00 0e:02:00 0e:02:00 0e:02:00 0e:02:00 0e:02:00 0e:02:00 0e:02:00 0e:02:00 0e:02:00
0e:02:00 0e:02:00 }
11: allow id 138a:0090 serial "886a88ed14f9" name "" hash
"HH2ou1EU3eoA8eE58HBzpb8VXZyLvV10LKfS/7cN/30=" parent-hash
"jEP/6WzviqdJ5VSeTUY8PatCNBKeaREvo20qdp1ND/o=" via-port "1-9" with-interface ff:00:00
12: allow id 8087:0a2b serial "" name "" hash
"AKEsx3Z6zyDfDo9JPXqCbKiTQ1jmuAgbRc5qGDUwOKc=" parent-hash
"jEP/6WzviqdJ5VSeTUY8PatCNBKeaREvo20qdp1ND/o=" via-port "1-14" with-interface { e0:01:01
e0:01:01 e0:01:01 e0:01:01 e0:01:01 e0:01:01 e0:01:01 }
16: allow id 154b:0095 serial "20814765" name "USB 3.0 FD" hash
"pPa+c6J01Gy4ATG+6L5HbFowlVK292sSkDZ4iHsYpCA=" parent-hash
"3Wo3XWDgen1hD5xM3PSN13P98kLp1RUTgGQ5HSxtf8k=" via-port "2-5" with-interface 08:06:50

```

4.1.3. Whitelisting and Blacklisting Devices

The `usbguard` daemon loads configuration from `usbguard-daemon.conf` file which is located at `/etc/usbguard/usbguard-daemon.conf` which stores all runtime parameters for USBGuard. The `usbguard -c` command/option combo can be used to modify the policy and authorization state of devices during runtime.

USBGuard Configuration Options `usbguard-daemon.conf`

There are several configuration options for USBGuard which are specified in the `/etc/usbguard/usbguard-daemon.conf` file.

Table 7. USBGuard Configuration Options

USBGuard Option	Description
RuleFile=full path to file	The usbguard daemon loads policy rule set from the file and writes new rules received from the IPC to this file.
IPCAllowedUsers= usernames	A space-delimited list of user names that the daemon will accept IPC connections from.
IPCAllowedGroups= groupnames	A space-delimited list of group names that the daemon will accept IPC connections from.
IPCAccessControlFiles=full path to directory	Path to the directory holding the IPC access control files
ImplicitPolicyTarget=single value of allow,block, or reject	How to treat devices that do not match any rule in the policy. Target value is one of: allow , reject , or block .
PresentDevicePolicy=policy	<p>How to treat devices that are already connected to the system when the daemon starts.</p> <p>allow: authorize every connected device</p> <p>block: deauthorize every connected device</p> <p>reject: remove every connected device</p> <p>keep: synchronize the internal state and keep whatever state the device is currently in</p> <p>apply-policy: evaluate the rule set for every connected device</p>

USBGuard Option	Description
PresentControllerPolicy=<i>policy</i>	<p>How to treat USB controllers that are already connected when the daemon starts</p> <p>allow: authorize every connected device</p> <p>block: deauthorize every connected device</p> <p>reject: remove every connected device</p> <p>keep: synchronize the internal state and keep whatever state the device is currently in</p> <p>apply-policy: evaluate the rule set for every connected device</p>

4.1.3.1. Securing Access to the USBGuard IPC

The configuration file in the example below, instructs the **usbguard** daemon to load rules from the **/etc/usbguard/rules.conf** file and only allow users from the **usbguard** group to use the IPC interface.

Listing 12. Example Config

```
RuleFile=/etc/usbguard/rules.conf
IPCAccessControlFiles=/etc/usbguard/IPCAccessControl.d/
```

Adding and Removing Users and Permissions for the IPC Access Control List

The **usbguard add-user** and **usbguard remove-user** commands can be used to modify the IPC ACLs. In the example below, we will allow users from the USBGuard group to modify the USB device authorization state, list USB devices, and listen to events as well as list the USB authorization policy.

Listing 13. Adding IPC Permissions to USBGuard group

```
# usbguard add-user -g usbguard --devices=modify,list,listen --policy=list
--exceptions=listen
```



The **IPCAccessControlFiles** option or the **IPCAccessControlFiles** and **IPCAccessControlFiles** options should be used to limit access to the IPC interface. Otherwise, it exposes the interface to all local users.

4.1.4. Applying Rules to Specific Devices and Classes of Device

USBGuard allows blocking/rejecting/allowing of specific devices which are examined and accurately

identified. However, it also provides the ability to add more flexible rules based on device characteristics. The device has is the most specific way to create rules, but you can also use Name, and Serial attributes without using the hash.

Devices can be managed based on Interface device characteristics. These are formatted as three 8-bit colon-delimited hexadecimal numbers.

Example 3. Interface Type Rules

```
{ interface class:subclass:protocol }
```

In the example below, the policy will allow only USB Mass Storage Devices (USB Flash)

Listing 14. Allowing USB Storage

```
allow with-interface equals { 08:*:* }
```

There are times where some USB devices contain multiple interfaces (storage, keyboard, network) that may need to be blocked. The following rule allows USB flash disks and rejects devices with additional/suspicious interfaces.

Listing 15. Allowing USB Storage but rejecting multi-interface storage devices

```
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:*:* }
```

Additional Information

There are lots of resources for USBGuard. The most information can be found:



- **USBGuard Github:** <https://usbguard.github.io/>
- **IPC Access Control:** <https://usbguard.github.io/blog/2017/IPC-Access-Control>
- **Configuration:** <https://usbguard.github.io/documentation/configuration.html>
- **Rule Language:** <https://usbguard.github.io/documentation/rule-language.html>

5. Controlling Authentication with PAM

5.1. Auditing the PAM Configuration

5.1.1. Describing PAM

The *Pluggable Authentication Modules* system (PAM) provides a generic way for applications to implement support for authentication and authorization, preventing applications from needing custom modules. By using PAM, applications don't need to be re-written each time a new authentication method is created. A PAM-enabled application calls the PAM library **libpam** to perform all authentication tasks on its behalf and returns pass/fail.



Administrators can use PAM configuration files to select modules to use for each application.

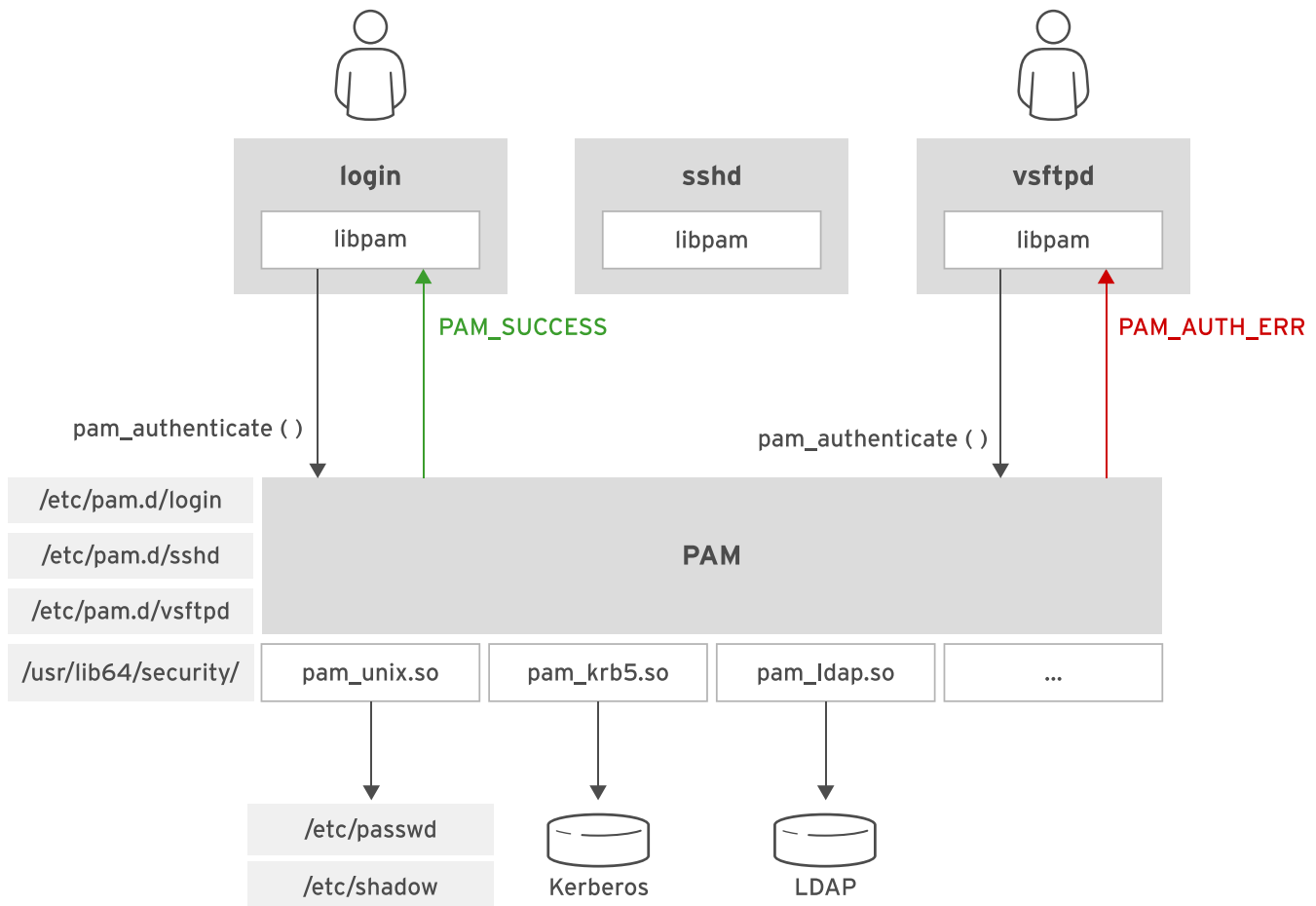


Figure 7. Ansible Tower Architecture

In the above figure, **login** contacts PAM for authentication. PAM reads the **/etc/pam.d/login** configuration file to retrieve list of modules needed for authentication. PAM calls modules stored in **/usr/lib64/security** and the modules are used to perform authentication. In this figure, the user was

able to authenticate with the **login** application, but failed the login for **vsftpd**.

5.1.2. Configuring PAM

Table 8. 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 <code>pam_unix</code> module. This module verifies the provided user name and password against the <code>/etc/shadow</code> file.
account	The application uses the account management group to verify that an account is valid at this time, and that passwords have not expired. In the <code>/etc/pam.d/system-auth</code> file above, a rule also calls the <code>pam_unix</code> module in that management group. In this group, the <code>pam_unix</code> module uses the expiration information from the <code>/etc/shadow</code> file to determine if the password is still valid.
password	Modules in the password management group control password changes. The rules in this management group have nothing to do with authentication or authorization. If the application provides a feature for users to change their password, PAM calls these rules when a user is attempting to change their password through the application. In the <code>/etc/pam.d/system-auth</code> file above, the first module in this group is <code>pam_pwquality</code> . This module verifies the quality of the new password provided by the user. The <code>pam_pwquality</code> module is described in more detail in the following section. The <code>pam_unix</code> module is called next. In that password management group, <code>pam_unix</code> stores the new password in <code>/etc/shadow</code> .
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, device or console ownership.

Table 9. Common PAM Controls

Control	Description
required	The associated module must succeed. If it fails, PAM sets the management group overall result to fail. The other rules are still tested to disguise why the failure happened from a potential attacker.
requisite	This control is similar to required but stops testing on error. PAM directly gives back the control to the application or to the calling file. The substack control bellow describes how to perform such a call.
sufficient	Returns success immediately to the application or the calling file if the associated module succeeds and no previous module has failed. If the module fails, PAM ignores the test and continues checking.
optional	PAM ignores the result of the test, even if it fails.
include	Include the rules from the provided PAM configuration file as if you directly entered them at that point.
substack	This control works like include except that a failed test in the called file gives back the control to the current file instead of giving back the control to the application.

5.2. Modifying the PAM Configuration

5.2.1. Preparing for Configuration update

Errors in PAM configurations can have severe consequences and can possibly even lock out the root account from the system. To mitigate issues, the following items should be done:

- Use **authconfig --savebackup=/root/pambackup** command to backup PAM configuration

```
# authconfig --savebackup=/root/pambackup
```



The backup directory should be created prior to running the **authconfig --savebackup** command



To restore configuration, use the **--restorebackup=backupdir** option.

- Open a second root shell/terminal screen and leave it open when testing connections, that way if

the configuration causes the user to be locked out, a shell is still there do fix the problem. DO NOT reboot the system before fixing issues.

- Use provide tools to configure PAM rather than manually modifying the files. The **authconfig** command has options to perform the most common configurations. Use **--help** option to review the tool's capabilities.

5.2.2. Using authconfig to Configure pam

5.2.3. Manually Configuring PAM

In most situations, users should use the **authconfig** command to modify configurations. For specific configurations, it may be needed to edit the PAM configuration files manually.



When making changes to the PAM configuration file, make sure to make all changes needed before saving the file. PAM will immediately apply modifications, so partial configurations could result in the system being inaccessible.



During the Guided Exercise, in Step 4, if the PAM doesn't appear to be locking people out, ensure that you ran the **authconfig --enablepamaccess --update**.

5.3. Configuring Password Quality Requirements

5.3.1. Describing the pam_pwquality Module

Red Hat Enterprise Linux (RHEL) can enforce password policy/complexity rules using the PAM module **pam_pwquality**. PAM uses rules in the **password** management group when users attempt to change their password through an application session.



PAM rules are parsed and executed top to bottom, so the first rule PAM calls is the **requisite pam_pwquality.so** entry.

5.4. Limiting Access After Failed Logins

5.4.1. Locking Accounts with Multiple Failed Logins

6. Recording System Events with Audit

6.1. Configuring Audit to Record System Events

6.1.1. The Linux Audit System

The Linux Audit system is a mechanism in the kernel providing a way to track security-related events and information on your systems. A set of rules can be defined and loaded into the kernel specifying which events it should record in the audit log. The **auditd** system daemon writes the logged events to the local disk or forwards them to remote log servers.

Audit can be configured to include or exclude events based on user identity, security contexts, or other labels. The Audit system can collect events including:

- Data and time, type, and outcome of an event
- Sensitivity labels of subjects and objects
- Association of an event with the identity of the user triggering the event
- All uses of authentication mechanisms, such as SSH, Kerberos, and others
- Attempts to import or export information into or out of the system

The Audit system may be needed in order to meet compliance requirements within security specifications.

6.1.2. Auditing Your System with auditd

auditd is the user-space component of the Linux auditing subsystem. The **systemd** unit file starts **auditd** normally loading persistent audit rules when the daemon is started. After audit rules are loaded into the kernel, the kernel uses them to record events of interest. The kernel sends information about those events to **auditd**. The main role of **auditd** is to collect audit event messages and save them to a file.

When **auditd** is running, logs are typically configured to go to **/var/log/audit/audit.log** collecting audit messages sent by the kernel. If **auditd** is not running, **rsyslog** receives the kernel audit messages.



Administrators can use **auditctl** command to configure/create **audit** rules that control the audit system, watch files, or record information about any system call.

6.1.3. Configuring auditd

Table 10. Configuring auditd

Filename	Description
/etc/audit/auditd.conf	Main auditd configuration file.

Filename	Description
<code>/etc/audit/audit.rules</code>	The audit rules loaded by auditd . Do not edit this file. It is generated from the files in <code>/etc/audit/rules.d</code> when auditd starts.
<code>/etc/audit/rules.d</code>	Directory containing manually configured audit rules. All files ending in .rules are combined into <code>/etc/audit/audit.rules</code> and loaded into the kernel when the auditd service starts.
<code>/etc/audit/rules.d/audit.rules</code>	Example name of a manually managed rule file.

The `/etc/audit/audit.conf` file configures the behavior of the **auditd** daemon. Administrators can determine if **auditd** service is running using the **systemctl status auditd** and **systemctl is-enabled auditd** commands.



When **auditd** is started, **systemd** unit file automatically rebuilds the audit rule set from the rules files in the `/etc/audit/rules.d` and attempts to load them into the kernel. If the service is reloaded, it attempts to rebuild rules file and load new rules using the **augenrules** command.

6.1.3.1. Adjusting auditd Settings to Manage Storage

Depending on the loaded audit rules, it is possible for audit log to grow quickly. It is a good idea for `/var/log/audit` to be a mount point on its own filesystem.



The **auditd** daemon has ability to warn administrators in the event of low disk space.

To configure **auditd** to comply with security policies, variables can be adjusted in `/etc/audit/auditd.conf`.

6.1.3.2. Adjusting auditd Settings to Tune Performance

Performance can be improved with **auditd** service with the following settings in `/etc/audit/audit.conf` configuration file:

- Set the value for **flush** parameter to **INCREMENTAL_ASYNC** to enable asynchronous flushing of records to storage.
- Set the **freq** parameter to **50** to flush the Audit log after every fifty records.

6.1.4. Remote Logging with auditd

Audit messages can be sent to remote systems in two ways:

- **rsyslog**: rsyslog can be configured to forward logs to a remote server.
- **auditd**: auditd can be configured to send logs to a remote **auditd** service.

Configuring Clients Auditd clients must be configured by making changes to **/etc/audit/auditd.conf**

- Set **log_format = ENRICHED** to resolve UID, GID, system call number, and socket address information to names before transmitting the event. That ensures remote system log information is correct as the remote system might have different mappings.
- Set **name_format = HOSTNAME** to include machine's hostname in each message.

Configuring a Server

If configuring **rsyslog** to collect remote audit messages, need to configure remote logging in **/etc/rsyslog.conf**. Remember to ensure that firewall port is open for the service (514/UDP).

If configuring **auditd** to listen for remote audit messages, edit the **/etc/audit/auditd.conf** file by uncommenting **tcp_listen_port = 60**. This configures **auditd** to listen on port 60/TCP so the firewall will need to be updated to allow connectivity on this port.



After making changes to the system services, it is necessary to restart the modified services in order to pickup changes from the configuration file.

Audit Messages and Encryption

Both methods of sending audit messages to a remote server use clear text protocols without encryption by default leaving message contents subject to interception and tampering.



If using **rsyslog** TLS authentication and encryption should be used to protect audit log traffic.

If using **auditd** transport, you should configure it to use Kerberos authentication and encryption in **/etc/audit/auditd.conf** file on the client and **/etc/audit/auditd.conf** on the server. Consult the **audisp-remote.conf** and **auditd.conf** man pages.



The following man pages can be used as references: **auditd**, **auditd.conf**, **audispd.conf**, **audisp-remote.conf**, **audit.rules**.

6.2. Inspecting Audit Logs

6.2.1. Interpreting Audit Messages

Audit events stored in `/var/audit/audit.log` include a ton of information in condensed formats. A single event could log multiple *audit records* of different types in the log as separate messages.

6.2.2. Searching for Events

The auditing system in RHEL ships with the **ausearch** command, which is a tool allowing the audit logs to be searched. **ausearch** can be used to search for and filter various event types. The table below shows common **ausearch** options.

Table 11. Common **ausearch** Options

Option	Description
-i	Interpret the log records, translate numeric values into names. This is very useful when you have raw log files.
--raw	Print raw log entries, do not even put separators between events. This is useful if you have other tools that can parse the raw log format. The short option -r is equivalent.
-a <EVENT-ID>	Show all records for the event that has <EVENT-ID> as its event ID.
-m <MESSAGE-TYPE>	Show all events that include a record with <MESSAGE-TYPE> as its message type. The long option --message is equivalent.
-f <FILENAME>	Search for all events related to a specific filename. The long option --file is equivalent.
-k <KEY>	Search for all events labeled with the <KEY> key.

Option	Description
--start [start-date] [start-time]	<p>Only search for events after start-date and start-time. If you do not specify a starting time, the search assumes midnight. If you omit the starting date, the search assumes today.</p> <p>The time format depends on your current locale. Other values that you can use include recent (past ten minutes), this-week, this-month, and this-year.</p> <p>--end can be used to search for events that occurred before a specific date and time, and uses the same syntax.</p>



For a complete **ausearch** options listing, refer to **#man ausearch** man pages.

6.2.3. Reporting on Audit Messages

The **aureport** command can be used instead of searching/reading individual audit messages. **aureport** can provide a quick overview of audit messages or detailed reports on specific event types.

When running **aureport** without options, it shows an overview of how many different types of events are present in the logs. Most of the options in **ausearch** can be specified and used with **aureport** to show events matching a list of the search criteria. Reports can be created for specific types of events by providing the appropriate options. When using **ausearch --raw** command to search for specific events, the unformatted search results can be provided as input to **aureport** to generate formatted reports.



Two specialized tools exist: **aulast** and **aulastlog**. They replace **last** and **lastlog** but they parse audit logs instead of **/var/log/wtmp** and **/var/log/btmp**.

6.2.4. Tracing a Program

In order to investigate system calls performed by a process, you can run the **autrace** command. Running **autrace** removes all custom auditing rules, replacing them with rules specifically for tracing the specified program. When execution of the trace finishes, the **autrace** command clears the rules and provides an example **ausearch** command.



autrace Requirements

The **autrace** command removes any active audit rules or requires active rules to be removed before it is run. This could cause missed events from other processes that the existing rules would normally record.

If audit rules are locked, the **autrace** will not be able to unload existing rules and the **autrace** process will fail to work.

6.3. Writing Custom Audit Rules

6.3.1. Adding Rules

The **auditctl** command can be used to add audit rules by command line. By default, rules are added to the bottom of the current list, but that can be inserted at the top.



Audit Rule Usage

Audit rules work on a first-match-wins basis.

Example: Two rules with separate keys, one logs all access to **/etc** and the other logs access to **/etc/sysconfig** directory in that order. The access rule for **/etc/sysconfig** will not be triggered because it is the second rule.

If the order is reversed, only the **/etc/sysconfig** rule will log access and any operations to **/etc** will also be matched noting that some **/etc/sysconfig** access might also be triggered as a second event for **/etc**.

There are three audit rule types: Filesystem rules, System calls, and Control rules.

Table 12. Audit rule types

Rule Types	Description
Filesystem	Audits access to files and directories
System Call	Audits execution of system calls made by processes communicating with the kernel to access system resources
Control	Config the audit system itself

6.3.1.1. Setting File System Rules (Watches)

Adding rules/watches can be done on directories/files and can be triggered on specific types of access.


Listing 16. Basic Syntax of Filesystem Rule/Watch

```
# auditctl -w file -p permissions -k key
```

The example above uses the **-w**, **-p**, and **-k** options with directives to get a desired result.

Table 13. Filesystem Watches

Option	Description and Option Parameters
-w	This option takes a file or directory as the option
-p	<p>This option takes a list of accesses to monitor by permission type.</p> <p>- r for read access</p> <p>- w for write access</p> <p>- x for execute access</p> <p>- a for changes to attributes</p>
-k	Takes a key to set on the audit record to make it easier to find with specific ausearch queries.



A watch will not cross file system boundaries. If a watch is set on / and **/tmp** is a mount point that has a separate filesystem mounted on it, rule will not apply to contents of **/tmp**

6.3.1.2. Setting System Call Rules

System call rules are more complex to author than Filesystem Rules (Watches)

Listing 17. System Call Rules syntax

```
# auditctl -a <list>,<action> \  
> [-F <filter-expression>]... \  
> [-C <compare-expression>]... \  
> [-S <system-call>]...
```

Rules for system calls are set on one of four lists:

- **exit**: Evaluates all system calls when they exit (most commonly used rule set)
- **user**: Evaluates events originating in user space
- **task**: Rarely used and checked only during **fork(2)** and **clone(2)** system calls.

- **exclude:** Sometimes used to completely filter events from being logged.



If you are using **auditctl -a** rules are placed at the bottom of the list. The **auditctl -A** will insert the specified rule at the top of the list instead of the bottom.



In most cases **<list>,<action>** you will use **exit,always** to always record the event when the system call exits.



System call rules are checked for every system call issued on the system - possibly reducing system performance. You should try to keep the list of system call rules short and general.

6.3.1.3. Setting Control Rules

Control rules are the final class of audit rule. These modify kernel configuration of Linux Audit. These are usually set at the top of the rules list and are kept short and simple, with the exception of **-e 2** rule which prevents further changes to the audit rule set.

6.3.2. Removing Rules

Filesystem watch rules can be removed with the **-W** option. You need to 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 being removed.



To remove all rules, you can use **auditctl -D**.

6.3.3. Inspecting Rules

Audit rules can be inspected with **auditctl -l**. To review the current status of Audit, the **auditctl -s** command can be used. This command displays information on the internal buffer sizes to store audit messages when storage cannot follow.

6.3.4. Making Rules Immutable

6.3.5. Persistent Rules

6.4. Enabling Prepackaged Audit Rulesets

6.4.1. Prepackaged Audit Rulesets

There are several pre-packaged rulesets that ship with the *audit* package. The rules are available in **/usr/share/doc/audit-*/rules** directory with a suffix of **.rules**.

6.4.2. Full Terminal Keystroke Logging

Some auditing policies require keystroke logging. Audit provides this keylogging functionality in conjunction with **pam_tty_audit** PAM module. Every keystroke is recorded in the audit log file (**/var/log/audit/audit.log**).

To enable keystroke logging, add the **pam_tty_audit** module to **/etc/pam.d/system-auth** and **/etc/pam.d/password-auth** files so all system daemons will implement the terminal functionality and begin logging keystrokes.



The **pam_tty_audit.so** module only implements **session** functionality. Adding this module to any other section in PAM prevents any user from logging into the system.



The **pam_tty_audit** module takes either **enable** or **disable** as arguments.



To convert data logged in Audit system logs to more readable format, you can use **aureport --tty** command. :imagesdir: images/

7. Monitoring File System Changes

7.1. Detecting File System Changes with AIDE

7.1.1. Analyzing File System Changes with AIDE

Normal operations on servers require files to be added/removed/modified on filesystems. However, some changes are unexpected to configuration files, executable files, and other files could indicate unauthorized modifications or other security issues.

RHEL offers a utility to monitor the system for file system changes **Advanced Intrusion Detection Environment (AIDE)**. AIDE is configured to monitor files for changes including: permissions, contents, and other characteristics.

7.1.1.1. Installing AIDE

The AIDE package is not installed by default. It can easily be installed using YUM.

Listing 18. Installing AIDE

```
# yum install aide
```

7.1.1.2. Configuring AIDE

The AIDE configuration file is **/etc/aide.conf**. The **aide.conf** file controls which files AIDE monitors for changes and what is monitored for each file. By default, AIDE monitors files in **/etc** for permission changes only.



AIDE ships with a default config file **/etc/aide.conf** that can be used to build an initial AIDE database. This file should be modified or tuned prior to building or updating the AIDE database.

Editing the **/etc/aide.conf** file tunes the operations of AIDE. Each line in the file is a directive. The **aide.conf** file contains three types of lines: configuration lines, selection lines and macro lines.

- **Configuration Lines:** Used to adjust parameters in AIDE. Can tune AIDE globally or set a *group definition*.
 - Syntax of configuration line: **parameter = value**

Listing 19. Example of a Permission Group Definition

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

The group definition creates a group called **PERMS**. If a selection line uses this group definition, files

selected by that line will be monitored for changes to permissions (**p**), user (**u**), group (**g**), Access Control List permissions (**acl**), SELinux context (**selinux**) and file system Extended Attributes (**xattrs**).

- **Selection Lines:** Specify the files and directories that AIDE monitors. Selection lines can be **regular**, **equals**, or **negative**.
- **Macro Lines:** Set or unset variables that refer to lengthy URLs or file-system paths in multiple occurrences through the AIDE configuration file.

7.1.1.3. Initializing the AIDE Database

Once AIDE is installed, it is necessary to ensure AIDE is aware of the current filesystem. AIDE uses this as a reference point. This process is referred to as initializing the AIDE database.



In production, AIDE checks should be run periodically, most likely as a CRON job.

Listing 20. Initializing AIDE

```
# aide --init
```

7.1.1.4. Verifying Integrity with AIDE

Once AIDE has been configured and the database has been initialized, it can detect file-system changes comparing against the AIDE database.

Listing 21. Manually Using AIDE for Integrity Checks

```
# aide --check
```

AIDE integrity checks compare current system status with the status in the AIDE database. An AIDE report is available in `/var/log/aide/aide.log`.

7.1.1.5. Updating the AIDE Database

It is important to update the AIDE database after known/expected changes are made to a system. This will help avoid false positives and AIDE reporting known changes. It is best to perform an AIDE database update after system upgrades or configuration changes.

Listing 22. Updating the AIDE Database for Changes

```
# aide --update
```



AIDE Database Maintenance

Don't forget to replace the old database file with the updated database file created by the **aide --update** command. Otherwise, AIDE continues to use the old database for baseline checks.

The locations of files are specified in **/etc/aide.conf**. The database used for checks defaults to **/var/lib/aide/aide.db.gz**. By default **--update** writes an updated database to **/var/lib/aide/aide.db.new.gz**.

7.2. Investigating File System Changes with AIDE

7.2.1. Combining AIDE and Audit

AIDE is useful to detect changes on computer's filesystem, but doesn't tell what caused the changes. AIDE can be configured to work with Linux Audit system to monitor for activity that could cause a change and log useful information about the time of the activity or user/process that made the change.

By configuring AIDE and Audit to monitor key files/directories, the AIDE report can be used to identify unexpected changes and then the audit log can be used to see what has changed since the last AIDE report.



One limitation of AIDE is that it does not notify in real time when a change has occurred. It only detects changes when an AIDE check is run. AIDE checks can be expensive in terms of system resources, so if the frequency of AIDE checks is high additional load on system CPU and disk I/O is generated. Additionally, large amounts of audit logging can also impact system performance.

7.2.2. Configuring AIDE and Audit

7.2.3. Investigating File System Changes

7.2.3.1. Interpreting Audit Events

Table 14. Common Audit Record Types

Record Type	Description
CWD	The current working directory relevant to this event.

Record Type	Description
PATH	Information about the path to a file involved in this event. Watch in particular for its objtype field, which records the reason why this record is involved in an event including a SYSCALL record. There may be multiple PATH records in the same event, for files that are involved for different reasons.
PROCTILE	The full command line that triggered this event.
SYSCALL	The system call made to the kernel that triggered this event. Particular system calls are likely to be involved in a file change event, such as open() , fchmod() , and setxattr() . This record is also the one that will likely have the most information about which user and process is involved in the operation.

Table 15. 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 <code>x86_64</code> because it will record the access mode being used to open the file. <code>O_WRONLY</code> or <code>O_RDWR</code> mode indicates that the system call is trying to open the file in a writable mode.
a2	Third argument of the system call.
a3	Fourth argument of the system call.
audit	Audit UID. This is the user ID that was used to log into the system initially. It will stay the same even after the user uses <code>su</code> or some other command to change their real UID.
cwd	Current working directory relevant to this event.
euid	Effective UID. This is the user ID that the process has for permission checks. For example, an executable that is set-UID root and run by a regular user will have an effective UID of root unless it drops privileges.

Field	Description
egid	Effective GID. This is the group ID that the process has for permission checks.
proctitle	Process title. It shows the full command line along with the arguments of the executed command. This is encoded in hexadecimal notation. Use -i option of ausearch to decode it.
syscall	The system call sent to the kernel.
success	Shows whether the system call is successful.
tty	The terminal or pseudoterminal controlling the process. If the command was run as part of an interactive session and the user had several in progress, this can be used to help identify which session was used to run the command.
uid	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 su to switch from one user to another, changing their real UID. Likewise, if the process was started by a set-UID or set-GID executable, the effective UID or GID of the process may be different.

8. Mitigating Risk with SELinux

8.1. Enabling SELinux from Disabled State

8.1.1. Reviewing Basic SELinux Concepts

Security Enhanced Linux (SELinux) is an additional layer of system security. SELinux has a primary goal to protect user data from compromised system services. SELinux confines programs (mainly system services) to a least privilege model required for doing jobs successfully. A set of security rules determine which processes can access which files, directories, and ports. If there is no explicit rule for access, SELinux denies the operation.



Every file, process, director, and port has a security label which is the **SELinux context**. A context is a name that SELinux rules use to control access.



SELinux Contexts

- For files and directories, SELinux stores contexts as filesystem extended attributes.
- For processes and ports, the linux kernel maintains the contexts in memory.



Contexts of processes and files can be viewed with adding the **-Z** option with the **ps** and **ls** commands.



Contexts of ports can be seen with the **semanage port -l** command.

Example 4. SELinux Contexts

```
user:role:type:level
```

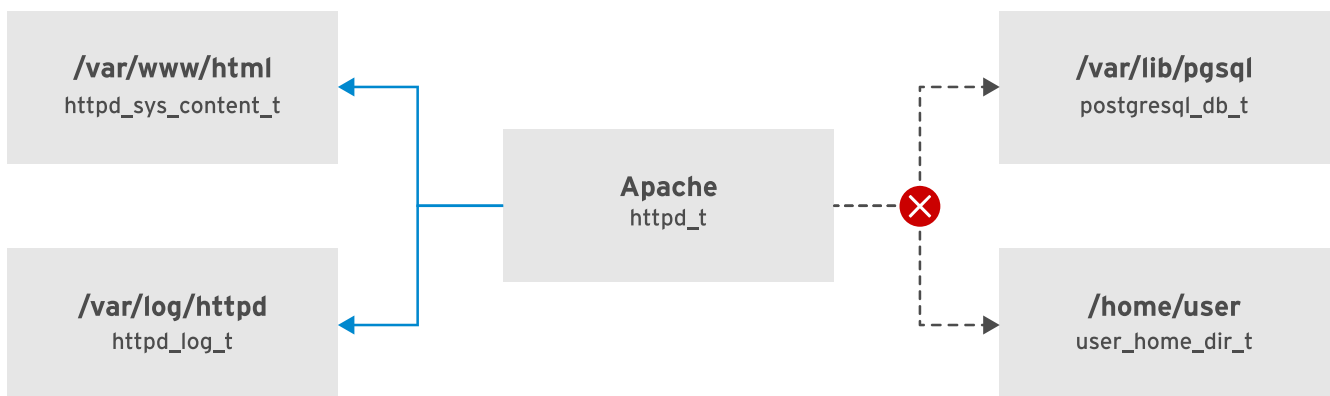


Figure 8. SELinux protection on Apache Service

8.1.1.1. Changing SELinux Contexts for Files and Directories

8.1.1.2. Defining SELinux Default File Context Rules

8.1.1.3. Labeling SELinux Ports

8.1.1.4. Using SELinux Booleans

8.1.1.5. Accessing the Documentation

8.1.2. Configuring SELinux Modes

Table 16. SELinux Modes

Record Type	Description
enforcing	In this mode, SELinux actively controls access. SELinux both logs and protects. The location of the log file and its content is explained in more detail below.
permissive	The permissive mode is often used to troubleshoot issues. In this mode, SELinux does not block any access, but still logs what should have been denied. You do not need to reboot to switch from enforcing to permissive or back again.
disabled	This mode completely disables SELinux. You need to reboot your system to disable SELinux, or to get from the disabled mode to the enforcing or permissive mode.

8.1.3. Enabling SELinux from Disabled Mode

8.1.3.1. Reviewing SELinux Access Violation Audit Events

8.1.3.2. Using Permissive Domains

8.1.3.3. Using Ansible for Remediation

8.2. Controlling Access with Confined Users

8.2.1. Defining SELinux Users

The SELinux policy defines its own SELinux users which are distinct from Linux users.



SELinux users can be listed with **semanage user -l** command.

Example 5. SELinux Users

--

In the example above, each SELinux user has access to a set of SELinux roles. The roles define which programs an SELinux user can run. The **system_r** role allows the usage of **su** and **sudo** commands.

8.2.1.1. Mapping Linux Users to SELinux Users

During login, SELinux maps Linux users to SELinux users. Linux users inherit restrictions assigned to their associated SELinux users.



semanage login -l command displays the table SELinux uses for user mapping.

Example 6. SELinux User Mapping for Login

--

In the example above, SELinux maps the Linux root user to the **unconfined_u** SELinux user. The **_default_** entry instructs SELinux to map all Linux users not explicitly listed to the **unconfined_u** SELinux user.



Linux users mapped to the **unconfined_u** are not confined by SELinux and only the traditional Linux rights limit programs they can run. By default on RHEL, Linux users are not confined.



SELinux uses the **system_u** user for the system services. DO NOT use it for your Linux users.

8.2.1.2. Comparing the SELinux Users

8.2.1.3. SELinux User Booleans

8.2.2. Confining User Accounts

8.2.2.1. Confining Different User Accounts

8.2.2.2. Confining System Administrators

8.2.2.3. Confining Staff Users



man semanage-login can be used to gain information on SELinux user contexts.

8.3. Auditing the SELinux Policy

8.3.1. Introducing the SELinux Policy

A policy is a set of rules that guide the SELinux security engine. It defines SELinux users, booleans, types for files and processes.



seinfo command can be used to list all objects in the policy. This command is part of the *setools-console* package.

Example 7. seinfo Command to List Policy Objects

The policy also includes the rules that determine how each domain can access each type; keep in mind a domain is what a type is called when it applies to a process.



SELinux associates *attributes* with types for categorization and organization. Attributes are a way to group types.

Attributes can help reduce and simplify access rules. All attributes can be listed with the **seinfo --attribute** command and the types in an attribute can be listed with the **seinfo --attribute=<attribute> -x** command.

Example 8. seinfo Command to List Attributes and Types

8.3.2. Analyzing the Targeted Policy

8.3.2.1. Interpreting the Allow Rules

Table 17. sestatus -A options

Option	Description
-s name	Source domain type or attribute
-t name	Target type or attribute
-c name	Class of the target object. Use seinfo -c to get the full list.
-p perm, perm,...	List of permissions

Option	Description
-C	Print the name of the SELinux boolean that enables the rule.

8.3.2.2. Disabling and Enabling the "dontaudit" Rules

8.3.2.3. Creating Custom Policy Modules with audit2allow

8.3.2.4. Analyzing Domain Transitions

8.3.2.5. Analyzing File Transitions

9. Managing Compliance with OpenSCAP

9.1. Installing OpenSCAP

9.1.1. OpenSCAP and Security Compliance in Red Hat Enterprise Linux

To ensure desired security baseline settings NIST along with others developed a standard compliance rule set called Security Content Automation Protocol (SCAP). SCAP is a framework of security specification that supports automated configuration, vulnerability and patch checking, technical control compliance activities, and security measures. OpenSCAP is a project developing tools for implementing and enforcing the SCAP standard.

9.1.1.1. Security Compliance Tools

OpenSCAP

The **oscap** command-line utility is used to perform configuration and vulnerability scans and to generate reports and guides based on scan results.

SCAP Security Guide (SSG)

The SSG is a predefined collection of security policies for Linux systems. It provides a catalog of hardening advice, linked to various government requirements to help define and customize security policies.

Script Check Engine (SCE)

The **openscap-engine-sce** package provides SCE extension that allows to write security content using Bash, Python, or Ruby.

SCAP Workbench

Graphical utility to perform scans on a single local or remote system and to generate security reports based on the scans.

9.1.2. SCAP Security Guide

SCAP Security Guide is a collection of security policies for Linux systems in the form of SCAP documents. SCAP Security Guide can be used with OpenSCAP tools to automate the auditing of a Linux system.

SCAP Security Guide transforms security guidelines into a machine-readable format which can be used by OpenSCAP to audit a system. The available security policies in SSG are broken down into profiles.



SCAP Security Guide can be installed with **yum install -y scap-security-guide**.

The installation of **scap-security-guide** automatically installs the *openscap-scanner* package as a dependency.

The *scap-security-guide* package installs predefined profiles in the `/usr/share/xml/scap/ssg/content` directory. Each file is an XCCDF file and like everything in SCAP it is based on XML. The *Extensible Configuration Checklist Description Format (XCCDF)* is a language to describe the security checklists. XCCDF supports document generation, information exchange, automation of compliance testing, compliance scoring, and tailoring.



The **oscap** command can be used to generate a user-friendly HTML version of the security guide for a specific profile.

In order to generate HTML security guides for a specific profile using **oscap** you must provide the profile's unique **id** attribute. The **oscap info** can parse the XCCDF XML file and display profiles along with the **id** attribute.

Example 9. Generating HTML Security Guide

```
# oscap xccdf generate guide \  
> --profile xccdf_org.ssgproject.content_profile_stig-firefox-upstream \  
> /usr/share/xml/scap/ssg/content/ssg-firefox-ds.xml > guide.html
```

9.1.3. SCAP Workbench

SCAP Workbench (scap-workbench) is a graphical tool allowing users to perform configuration scans, perform remediation of the system, and generate reports based on evaluations. The tool can be executed on a single local system or a remote system using SSH.

Example 10. Installing SCAP Workbench

```
yum install scap-workbench
```

9.1.3.1. Local System OpenSCAP Scan

To prepare systems for OpenSCAP scans, it requires the *openscap-scanner-package* and the *scap-security-guide* packages so that the **oscap** command-line utility is installed and the content from the SCAP Security Guide.

Listing 23. Installing the SCAP Tools

```
yum install openscap-scanner
```

Listing 24. Installing the SCAP Content

```
yum install scap-security-guide
```



If you install the **scap-security-guide** package, then the **openscap-scanner** is installed as a dependency.

9.2. Scanning and Analyzing Compliance

9.2.1. Introducing the oscap Command

The **oscap** command-line tool is a front end to the OpenSCAP libraries. This tool can scan the system, generate remediation scripts, and create reports and guides. The **oscap** command needs security content to work.



The SCAP Security Guide **scap-security-guide** package installs files in the **/usr/share/xml/scap/ssg/content/**.



Available SCAP profiles can be listed using the **oscap info** command.

Listing 25. Installing the SCAP Tools

```
# oscap info ./ssg-rhel7-ds.xml
Document type: Source Data Stream
Imported: 2018-04-27T15:03:29

Stream: scap_org.open-scap_datastream_from_xccdf_ssg-rhel7-xccdf-1.2.xml
Generated: (null)
Version: 1.2
Checklists:
  Ref-Id: scap_org.open-scap_cref_ssg-rhel7-xccdf-1.2.xml
  Status: draft
  Generated: 2018-04-27
  Resolved: true
  Profiles:
    Title: Standard System Security Profile
    Id: xccdf_org.ssgproject.content_profile_standard
    Title: PCI-DSS v3 Control Baseline for Red Hat Enterprise Linux 7

.... Output Ommitted ....
```

9.2.2. Scanning a System for Compliance

In order to scan the system for compliance, run the **oscap xccdf eval** command as the root user. The data stream file must be provided as an argument to the command and the identifier of the profile to use with **--profile**. The results can be saved in an XML file with the **--results** option.

Listing 26. Using OSCP to Scan a System

```
# oscap xccdf eval \  
> --profile xccdf_org.ssgproject.content_profile_pci-dss \  
> --results /root/results.xml \  
> /usr/share/xml/scap/ssg/content/ssg-rhel7-ds.xml  
WARNING: This content points out to the remote resources. Use '--fetch-remote-  
resources' option to download them.  
WARNING: Skipping https://www.redhat.com/security/data/oval/com.redhat.rhsa-  
RHEL7.xml.bz2 file which is referenced from XCCDF content  
Title    Ensure Red Hat GPG Key Installed  
Rule     xccdf_org.ssgproject.content_rule_ensure_redhat_gpgkey_installed  
Ident    CCE-26957-1  
Result   pass  
  
.... Output Ommitted ....
```

9.2.2.1. Generating the HTML Report

The **oscap xccdf generate report** can be used to generate a complete HTML report.

Listing 27. Using OSCP to Generate an HTML Report

```
# oscap xccdf generate report results.xml > results.html
```

9.3. Customizing OpenSCAP Policy

9.3.1. Customizing a SCAP Security Guide Profile

The SCAP Security Guide provides profiles for verifying systems compliance against established standards. The SCAP Workbench utility can create custom profiles.

9.3.1.1. Creating a Tailoring File

Run the **scap-workbench** command to start the utility to create profile customization. The tailoring file that is created can be used on the local workstation or copied to another system for scanning.

9.3.2. Scanning a System Using a Profile Customized with a Tailoring File

To prepare a system for scanning with the tailoring file, the following items must be available:

- XML tailoring file

- Custom profile identified that was defined when tailoring file was created. The **oscap info** command can be used to retrieve that identifier.
- The **openscap-scanner** and **scap-security-guide** packages

Listing 28. Scanning for Compliance Using Custom Tailoring File

```
# oscap xccdf eval \
> --profile custom_profile_ID \
> --tailoring-file tailoring_file.xml \
> --results result_file.xml \
> /usr/share/xml/scap/ssg/content/ssg-rhel7-ds.xml
```

9.4. Remediating OpenSCAP Issues with Ansible

9.4.1. Generating a Remediation Ansible Playbook

The XCCDF files in the SCAP Security Guide include remediation scripts to fix the non-compliant checks. The script take the form of shell scripts, Ansible snippets, Puppet snippets, or kickstart commands.



The **oscap** command can generate an Ansible Playbook from the SCAP Security Guide profile or from a scan result XML file.

9.4.1.1. Creating an Ansible Playbook for a Profile

The **oscap xccdf generate fix** command can generate an Ansible Playbook with taks for remediation.

Example 14. Scanning for Compliance and Generating Ansible Remediation Playbooks

```
# oscap xccdf generate fix \
> --profile xccdf_org.ssgproject.content_profile_pci-dss \
> --fix-type ansible \
> /usr/share/xml/scap/ssg/content/ssg-rhel7-ds.xml > pci-dss.yml
```



Since Ansible playbooks are idempotent, the remediation playbook can be run regularly on the system to ensure compliance and ensure there is no impact on items that are already compliant.



Not all SCAP rules have Ansible remediation snippets. For those rules, custom playbook tasks will need to be developed.

9.4.1.2. Creating an Ansible Playbook from a Result XML File

The **oscap xccdf generate fix** command can also generate Ansible Playbooks from an XML results file to fix non-compliant checks. .Generating Ansible Remediation Playbooks from XML Results File

```
# oscap xccdf eval \  
> --profile xccdf_org.ssgproject.content_profile_pci-dss \  
> --results /root/results.xml \  
> /usr/share/xml/scap/ssg/content/ssg-rhel7-ds.xml  
...output omitted...  
  
# oscap xccdf generate fix \  
> --profile xccdf_org.ssgproject.content_profile_pci-dss \  
> --fix-type ansible \  
> --result-id "" \  
> /root/results.xml> remediation-playbook.yml
```

9.4.1.3. Adjusting Variables in the Remediation Ansible Playbook

Ansible Playbooks that are generated can be edited and some of the variables changed.

Listing 29. Example Remediation 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...
```

9.4.2. Running a Remediation Ansible Playbook

The **oscap xccdf generate fix** command creates Ansible Playbooks with **hosts** set to **all**.



Custom inventory files can be specified with **-i inventory_file** option of the **ansible-playbook**.

Listing 30. Specifying Custom Inventory to Run a Playbook

```
# ansible-playbook -i ./myinventory pci-dss.yml
```

9.4.2.1. Filtering Tasks

Ansible remediation playbooks can be large. For that reason, tags are assigned in Ansible Playbooks attached to remediation tasks. The tags come from SCAP Security Guide XCCDF files.



It is possible to run commands from a playbook based on specified tasks.

Listing 31. Running Playbooks Based on Tags

```
# ansible-playbook --tags=high_severity pci-dss.yml
```

9.4.3. Checking Compliance During Installation

The OpenSCAP installer add-on can be installed from a kickstart file to check compliance during installation.

Example 15. Scanning for Compliance During Kickstart

Listing 32. Kickstart Snippet

```
%addon org_fedora_oscaps
content-type = scap-security-guide
profile = pci-dss
%end
```

10. Automating Compliance with Red Hat Satellite

10.1. Configuring Red Hat Satellite for OpenSCAP

10.1.1. Security Compliance and Management with Red Hat Satellite

Red Hat Satellite 6 uses **Security Content Automation Protocol (SCAP)** to define security compliance policies. All registered systems can be reported to Satellite server for compliance auditing and reporting, allowing security administrators to manage, monitor, and remediate hosts based on compliance from a single interface.

10.1.2. Integrating OpenSCAP with Red Hat Satellite

Red Hat Satellite server provides the default SCAP content based on the version RHEL. The Satellite administrator can either create SCAP content or upload the SCAP content from external sources. Satellite has the ability to schedule an audit known as *compliance policy* against an XCCDF profile. Upon completion of the compliance scan, an *Asset Reporting File (ARF)* is generated in XML format and uploaded to a Satellite server. The reports can be viewed from the compliance policy dashboard.

10.1.2.1. Installing Satellite Server with the OpenSCAP Plugin

Red Hat Satellite installation must enable the OpenSCAP plugin to allow integration of Satellite and OpenSCAP. The OpenSCAP plugin allows OpenSCAP controls from the Satellite WebUI under the **Compliance** section in **Hosts**.

Listing 33. Enabling OpenSCAP Plugin on Satellite

```
# satellite-installer --enable-foreman-plugin-openscap
```

10.1.2.2. Uploading OpenSCAP Content to the Satellite Server

The default OpenSCAP content must be uploaded into Satellite server before a compliance policy can be created or applied to a host.

Listing 34. Uploading SCAP Content

```
foreman-rake foreman_openscap:bulk_upload:default
Saved /usr/share/xml/scap/ssg/content/ssg-firefox-ds.xml as Red Hat firefox default
content
Saved /usr/share/xml/scap/ssg/content/ssg-jre-ds.xml as Red Hat jre default content
Saved /usr/share/xml/scap/ssg/content/ssg-rhel6-ds.xml as Red Hat rhel6 default content
Saved /usr/share/xml/scap/ssg/content/ssg-rhel7-ds.xml as Red Hat rhel7 default content
```

The **hammer** command can be used to list SCAP contents from the command line, or it can be checked in the Satellite WebUI under **Hosts** ⇒ **SCAP Contents**.

Listing 35. Listing SCAP Content with Hammer

```
# hammer scap-content list
---|-----
ID | TITLE
---|-----
1  | Red Hat firefox default content
2  | Red Hat jre default content
3  | Red Hat rhel6 default content
4  | Red Hat rhel7 default content
---|-----
```

10.1.2.3. Importing OpenSCAP Puppet Module in Satellite Server

The OpenSCAP plugin provides Puppet classes required to setup hosts to perform OpenSCAP scans and creates cron jobs for automated compliance scanning. Each puppet environment associated with hosts/host groups to be audited using OpenSCAP needs to import the **foreman_scap_client** Puppet module. The **foreman_scap_client** Puppet module provides a client script that runs OpenSCAP scans and can upload results to the Satellite server. The module is executed by **cron** jobs for automated compliance scanning.

Example 16. Importing the Puppet Module in Satellite

1. Satellite WebUI, navigate to Configure → Environments.
2. Click Import, then Import environments from satellite.lab.example.com.
3. Navigate to Configure → Host groups. Select the hostgroups to edit. Ensure that the following fields are correctly set based on the hosts to be audited using OpenSCAP:
 - Puppet Environment
 - Puppet Master
 - Puppet CA
 - OpenSCAP Capsule
4. In the Puppet Classes tab, click the + to add the foreman_scap_client Puppet class listed under the foreman_scap_client Puppet module.
5. Click Submit.

10.1.2.4. Initiating a Puppet Agent Run on a Host

The imported **foreman_scap_client** Puppet module configures the SCAP components on the host using the Puppet agent.

10.1.2.5. Initiating a Puppet Agent Run using Remote Execution

Example 17. Initiating Puppet Agents from Satellite Server Interface

1. Ensure the hosts on which the Puppet agent run will be performed uses a SSH key-based authentication between the Satellite server and the host.
2. Satellite WebUI, navigate to Hosts → All hosts. Select the hosts in the **Hosts** page on which the Puppet agent will be executed.
3. Choose Schedule Remote Job from the **Select Action** drop-down list.
4. In the **Job invocation** page, choose Puppet from the **Job category** drop-down list. Ensure that for **Schedule**, **Execute now** is selected. Click Submit.
5. In the **Overview** tab, wait till you see the green circle with the 100% Success message. Click the host in the Hosts tab to see the output of the Puppet agent run.

10.2. Scan OpenSCAP Compliance with Red Hat Satellite

10.2.1. Performing OpenSCAP Scans using Red Hat Satellite

Compliance scans on Red Hat Satellite WebUI requires the admin to create a user with specific compliance roles. Each role contains one or more permission filters.

Table 18. SELinux Modes

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 is role is required to manually run an OpenSCAP scan from Satellite server.

10.2.2. Managing Compliance Policies

A compliance policy comprises of a scheduled task that checks hosts for compliance against an XCCDF profile. After completion, an Asset Report File (ARF) format is uploaded to the Satellite server using the **foreman_scap_client** command.

10.2.2.1. Creating a Compliance Policy

Compliance policy is defined by user on the WebUI.

- SCAP Content
- XCCDF profile for a specific SCAP content
- Host groups that should comply with SCAP policy
- The scheduled interval in which audit shall occur

10.2.2.2. Creating a SCAP Policy

10.2.3. Running Compliance Scans

10.2.3.1. Running an OpenSCAP Scan Manually

10.2.3.2. Running an OpenSCAP Scan using `foreman_scap_client`

Running the `*foreman_scap_client *` command manually on a host will complete an OpenSCAP scan, archive the results, and upload the results to the Satellite server.

1. On the host as root, use the following command to run the Puppet agent to fetch the changes to a SCAP policy:

Listing 36. Using Puppet to Fetch Policy

```
[student@serverd ~]$ sudo -i
[root@serverd ~]# puppet agent --test --verbose
...output omitted...
```

2. Open the `/etc/foreman_scap_client/config.yaml` SCAP client configuration file and note down the compliance policy ID.
3. Execute the `foreman_scap_client` command with the compliance policy ID as an argument. In the below screen, 1 is the policy ID

Listing 37. Executing `foreman_scap_client`

```
[root@serverd ~]# foreman_scap_client 1
File
/var/lib/openscap/content/96c2a9d5278d5da905221bbb2dc61d0ace7ee3d97f021fccac994d26296d986d.xml is missing.
Downloading it from proxy.
Download SCAP content xml from:
https://satellite.lab.example.com:9090/compliance/policies/1/content/96c2a9d5278d5da905221bbb2dc61d0ace7ee3d97f021fccac994d26296d986d
DEBUG: running: oscap xccdf eval --profile
xccdf_org.ssgproject.content_profile_common
--results-arf /tmp/d20180727-2719-ois40f/results.xml /var/lib/openscap/
content/96c2a9d5278d5da905221bbb2dc61d0ace7ee3d97f021fccac994d26296d986d.xml
WARNING: This content points out to the remote resources. Use '--fetch-remote-resources' option to download them.
WARNING: Skipping https://www.redhat.com/security/data/oval/com.redhat.rhsa-RHEL7.xml.bz2 file which is referenced from XCCDF content
DEBUG: running: /usr/bin/bzip2 /tmp/d20180727-2719-ois40f/results.xml
Uploading results to https://satellite.lab.example.com:9090/compliance/arf/1
```

10.2.4. Reviewing OpenSCAP Scan Results in Satellite Server

10.2.4.1. Viewing Compliance Policy Dashboard

10.2.5. Evaluating OpenSCAP Scan Report

10.2.5.1. Viewing Compliance Report

10.2.5.1.1. Viewing Compliance Report in Satellite Server

10.3. Customize the OpenSCAP Policy in Red Hat Satellite

10.3.1. Customizing SCAP Policy in Red Hat Satellite

10.3.1.1. Uploading a Tailoring File

10.3.1.2. Uploading a Tailoring File Using Satellite WebUI

10.3.1.3. Assigning a Tailoring File to a Policy

10.3.1.4. Assigning a Tailoring File using the Satellite WebUI

10.3.2. Executing a Compliance Scan Using a Customized Policy

11. Analyzing and Remediating Issues with Red Hat Insights

11.1. Registering Systems with Red Hat Insights

11.2. Reviewing Red Hat Insights Reports

11.3. Automating Issue Remediation

Appendix A: Exam Objectives

Exam Time: 4 Hours

General nodes for the exams process:

- Update the **man page** database

Listing 38. man Database Update

```
# mandb
```

- Update the **locate** database

Listing 39. locate Database Update

```
# updatedb
```

General Information for the Course

- **Chapter 2:** Ansible
- **Chapter 3:** LUKS and NBDE
- **Chapter 4:** USB Access Restriction
- **Chapter 5:** PAM and PAM Security Modules
- **Chapter 6:** Audit Rules
- **Chapter 7:** AIDE
- **Chapter 8:** SELinux
- **Chapter 9:** OpenSCAP
- **Chapter 12:** Comprehensive Review

YUM Usage

Listing 40. Getting Security Info from yum

```
# yum updateinfo --security
```



Listing 41. Getting Security Update Information

```
# yum --security list updates
```

Listing Updates and Searching for Critical

```
# yum updateinfo list updates | grep Critical
```

A.1. Use Red Hat Ansible Engine

The contents for these objectives are located in Chapter 2. Comprehensive Review 1 Covers the objectives



An example **ansible.cfg** file can be found in */etc/ansible/ansible.cfg*

A.1.1. Install Red Hat Ansible Engine on a control node

Ansible Engine needs to be installed on a control node which will perform operations on all managed nodes.

Guided Exercise: Configuring Ansible for Security Automation

Basic Steps for Installing and Configuring Ansible on a Control Node:

1. Install the Ansible application

Listing 42. Installing Ansible on Control Node

```
student@workstation ~]$ sudo yum install ansible ansible-doc
[sudo] password for student:
Loaded plugins: langpacks, search-disabled-repos
Resolving Dependencies
--> Running transaction check
---> Package ansible.noarch 0:2.5.5-1.el7ae will be installed
--> Finished Dependency Resolution

... Output Ommitted ...

Installed:
  ansible.noarch 0:2.5.5-1.el7ae

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

2. Create a directory for Ansible Configurations and Ansible Inventories

Listing 43. Configuring Ansible on Control Node

```
[ansible-testuser@workstation ~]$ mkdir security-ansible
[ansible-testuser@workstation ~]$ cd security-ansible
[ansible-testuser@workstation security-ansible]$
```

3. Create an Inventory File

Example 19. Creating an Ansible Inventory

```
[ansible-testuser@workstation security-ansible]$ vim inventory
```

Listing 44. **inventory** File Contents

```
[LOCAL]
workstation

[SERVERS]
servera
serverb

[EVERYONE:children]
LOCAL
SERVERS
```

3. Create an Ansible Config File

Example 20. Creating an Ansible Configuration File

```
[ansible-testuser@workstation security-ansible]$ vim ansible.cfg
```

Listing 45. **ansible.cfg** File Contents

```
[defaults]
inventory      = ./inventory
remote_user    = ansible-testuser
ask_pass       = True

[privilege_escalation]
become=True
become_method=sudo
become_user=root
become_ask_pass=True

[ssh_connection]
ssh_args = -o StrictHostKeyChecking=no
```



The **ssh_args = -o StrictHostKeyChecking=no** is not in the initial Ansible config file in **/etc/ansible/ansible.cfg**



The **ask_pass=True** option in the defaults section requires Ansible to prompt for the SSH password even if the key exists on the managed nodes.



It is easy to copy the **/etc/ansible/ansible.cfg** to a local **ansible.cfg** and edit the top portion. Be sure to change **sudo_user** to **remote_user**. Also, search for the privileges section and uncomment those portions to set the configs.

A.1.2. Configure managed nodes



Typically it is necessary to configure **sudo** privileges for the **ansible user** on all managed hosts.

It is often helpful to copy the SSH Keys from the control node to all managed hosts.

Configuring Ansible with SUDO and SSH-Keys

You will need to generate and copy SSH keys from the Ansible control node to all the managed nodes.

Listing 46. Generate SSH Keys

```
# ssh-keygen
```

Listing 47. Copy SSH Keys to Managed Nodes



```
# ssh-copy-id <ansible_user>@<ansible_managed_node>
```

Listing 48. Create a SUDOERS File for Ansible User

```
# echo "username  ALL=(ALL) NOPASSWD:ALL" > username
```

Listing 49. Copy SUDOERS File for Ansible User to Managed Nodes

```
# scp username root@<managed_node>:/etc/sudoers.d/
```

A.1.3. Configure simple inventories

```
[ansible-testuser@workstation security-ansible]$ vim inventory
```

Listing 50. **inventory** File Contents

```
[LOCAL]
workstation

[SERVERS]
servera
serverb

[EVERYONE:children]
LOCAL
SERVERS
```

A.1.4. Perform basic management of systems

A.1.5. Run a provided playbook against specified nodes

Ansible playbooks are run using the **ansible-playbook** command.

Listing 51. Running an Ansible Playbook

```
[ansible-testuser@workstation ansible-remediate]$ ansible-playbook webservers.yml

SSH password:

PLAY [installs, configures and starts apache]
*****

TASK [Gathering Facts]
*****
ok: [servera]
ok: [serverb]

... Output Ommitted ...

PLAY RECAP
*****
servera          : ok=7    changed=6    unreachable=0    failed=0
serverb          : ok=7    changed=6    unreachable=0    failed=0

[ansible-testuser@workstation ansible-remediate]$
```



Running a playbook on a single host

The **ansible-playbook** command will run the playbook on all hosts in the inventory file. It can be limited by using:

Listing 52. Ansible Playbook Limiting to Specified Server

```
# ansible-playbook -l <ServerName> playbook.yml
```

System Documentation and Man Pages

The Man pages that should be looked at are **ansible**, **ansible-playbook** and **ansible-doc**.

Listing 53. Ansible Man Page

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

ANSIBLE(1)                      System administration commands
ANSIBLE(1)

NAME
    ansible - Define and run a single task 'playbook' against a set of
    hosts

SYNOPSIS
    ansible <host-pattern> [options]

.... Output Ommitted ....

SEE ALSO
    ansible-config(1), ansible-console(1), ansible-doc(1), ansible-
    galaxy(1),
    ansible-inventory(1), ansible-playbook(1), ansible-pull(1),
    ansible-vault(1)

    Extensive documentation is available in the documentation site:
    http://docs.ansible.com. IRC and mailing list info can be found in
    file
    CONTRIBUTING.md, available in: https://github.com/ansible/ansible
```



The **ansible-doc** command gives the information about all the Ansible plugins and modules as well as the corresponding syntax.

A.2. Configure intrusion detection

Installation and Configuration of AIDE.

Chapter 7: GE1/GE2/Lab Comp Review: CR4

A.2.1. Install AIDE

Listing 54. Installing AIDE

```
# yum install aide
```



To find out some information on using AIDE, you can use **#man aide** and **#man aide.conf**

A.2.2. Configure AIDE to monitor critical system files

The AIDE config file needs to be changed in the **/etc/aide.conf** location and then it will need to be initialized.

Listing 55. Configuring AIDE

```
# vim /etc/aide.conf
```

Listing 56. Initializing AIDE

```
# aide --init  
# mv /var/lib/aide/aide.db.new.gz /var/lib/aide/aide.db.gz
```



It is extremely important that after the AIDE database has been initialized that the new AIDE database is copied to the correct location. The default location is: **/var/lib/aide/aide.db.gz**.

A.3. Configure encrypted storage

Chapter 3: Guided Exercises 1 and 2 Comp Review Exercise 2

A.3.1. Encrypt and decrypt block devices using LUKS

1. Configure the partition

Listing 57. Parted to look for disks/partions

```
[root@servera ~]# parted -l
Model: Virtio Block Device (virtblk)
Disk /dev/vda: 10.7GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start   End     Size    Type    File system  Flags
  1      1049kB  10.7GB  10.7GB  primary xfs          boot

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:
```

Listing 58. Parted to create partitions

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

parted options

Use **man parted** to get the syntax

Listing 59. pinfo for parted

```
# pinfo Parted

ile: parted.info,  Node: Top,  Next: Introduction,  Up: (dir)

GNU Parted User Manual
*****

This file documents the use of GNU Parted, a program for creating and
manipulating partition tables.
```

```
    This document applies roughly to version *3.1* of GNU Parted.
```

```
... output omitted ...
```

In command line mode, this is followed by one or more commands. For example:

```
# parted /dev/sda mklabel gpt mkpart P1 ext3 1MiB 8MiB
```

Options (like '--help') can only be specified on the command line.

In interactive mode, commands are entered one at a time at a prompt, and modify the disk immediately. For example:

```
(parted) mklabel gpt
(parted) mkpart P1 ext3 1MiB 8MiB
```

```
parted <device> mklabel <label_name> mkpart primary <fstype> <start -
Typically 1M> <end - Set to 1G for a 1GB size>
```



Use **man cryptsetup** to get the LUKS commands and syntax that will be used.

2. Create the LUKS partition

Listing 60. Using Cryptsetup to apply LUKS to partition

```
# cryptsetup luksFormat /dev/vdb1
```

3. Name and Open the LUKS partition

Listing 61. Naming LUKS storage

```
[root@servera ~]# cryptsetup luksOpen /dev/vdb1 storage
Enter passphrase for /dev/vdb1:
```

4. Format the filesystem and mount to newly created directory

Listing 62. Preparing Device for use

```
[root@servera ~]# mkdir /storage
[root@servera ~]# mkfs.xfs /dev/mapper/storage
meta-data=/dev/mapper/storage  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
[root@servera ~]# mount /dev/mapper/storage /storage/
```

5. Unmount the filesystem and close the LUKS partition

Listing 63. Closing LUKS partition

```
[root@servera ~]# cryptsetup luksClose storage
```

A.3.2. Configure encrypted storage persistence using NBDE



Search man pages: **man tang** and **man clevis** and **man clevis-encrypt-sss**

1. Install TANG on the TANG servers for the encrypted clients to connect.

Example 23. Installing TANG on Servers

1. Install TANG

Listing 64. Installing TANG packages

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

2. Configure the **tangd.socket** to enabled

Listing 65. Enabling TANG

```
[root@serverb ~]# systemctl enable tangd.socket --now
```

3. Configure port 80 on the firewall and reload rules

Listing 66. Configure the Firewall

```
firewall-cmd --zone=public --add-port=80/tcp --permanent
```

```
[root@serverb ~]# firewall-cmd --reload  
success
```

2. Install Clevis packages and configure Clevis on the encrypted client nodes

Example 24. Installing Clevis Packages

1. Install the clevis packages

Listing 67. Installation of Clevis

```
[root@servera ~]# yum install clevis clevis-luks clevis-dracut
```

2. Configure Clevis to Bind to TANG servers



man clevis, man clevis-luks-bind and man clevis-encrypt-sss



The "URL" must be in quotes.

Listing 68. Clevis configuration - Setting the SSS

```
[root@servera ~]#  
cfg='{ "t":3, "pins":{"tang":[{"url":"http://serverb"}, {"url":"http://serverc"}, {"url":  
"http://serverd"}]}}'
```

3. Associate and Bind Clevis to LUKS

Listing 69. Clevis LUKS Binding

```
[root@servera ~]# clevis luks bind -d /dev/vdb1 sss "$cfg"  
The advertisement contains the following signing keys:
```

```
rP_G6voKt9Kr3w6TgZXcgHA0NCg
```

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

```
... Output Ommitted ...
```

```
A backup is advised before initialization is performed.
```

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

```
Enter existing LUKS password:
```

4. Enable the Clevis Service

Listing 70. Starting Clevis Service

```
[root@servera ~]# 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.
```

5. Modify the Crypttab and FStab files for the encrypted volume

Listing 71. Edit **crypttab**

```
[root@servera ~]# vi /etc/crypttab
storage          /dev/vdb1 none _netdev
```

Listing 72. Edit **fstab**

```
[root@servera ~]# vi /etc/fstab
/dev/mapper/storage /storage      xfs      _netdev      1 2
```

A.3.3. Change encrypted storage passphrases

This is performed on the TANG servers. The encrypted keys are in **/var/db/tang**.

Rotating TANG Keys

Use **man tang** and copy from the example.

Listing 73. Key Rotation based on man page



```
# DB=/var/db/tang

# jose jwk gen -i '{"alg":"ES512"}' -o $DB/signature.jwk

# jose jwk gen -i '{"alg":"ECMR"}' -o $DB/exchange.jwk
```



To change out LUKS keys on a volume, use **luksChangeKey**.

A.4. Restrict USB devices

The information can be found in Chapter 4, Section 12 of the RHEL7 Security Guide.

A.4.1. Install USBGuard

Comp Review

Listing 74. Installing USBGuard and Enable

```
# yum install -y usbguard usbutils udisks2
# systemctl enable usbguard --now
```

Getting Help

Listing 75. USBGuard man page

```
# man usbguard
```

Listing 76. USBGuard-Rules Config man page

```
# man usbguard-rules.conf
```

Listing 77. USBGuard-Daemon man page

```
# man usbguard-daemon
```

Listing 78. USBGuard-Daemon Config man page

```
# man usbguard-daemon.conf
```



A.4.2. Write device policy rules with specific criteria to manage devices

USB policies can be generated from current allowed rules and saved.

Listing 79. Generating a USB Policy

```
# usbguard generate-policy -X > /etc/usbguard/rules.conf
```

Listing 80. Generating a USB Policy to Reject Everything

```
# usbguard generate-policy -X -t reject > /etc/usbguard/rules.conf
```

USB Guard Additional Help

USBGuard has both man pages and built-in help files for commands.



Listing 81. USBGuard Help

```
# usbguard --help Usage: usbguard [OPTIONS] <command> [COMMAND OPTIONS]
...

Options:

Commands:
  get-parameter <name>          Get the value of a runtime parameter.
  set-parameter <name> <value>  Set the value of a runtime parameter.
  list-devices                  List all USB devices recognized by the
USBGuard daemon.
  allow-device <id>             Authorize a device to interact with the
system.
  block-device <id>             Deauthorize a device.
  reject-device <id>            Deauthorize and remove a device from the
system.

  list-rules                    List the rule set (policy) used by the
USBGuard daemon.
... output omitted ...

  read-descriptor               Read a USB descriptor from a file and
print it in human-readable form.

  add-user <name>               Add USBGuard IPC user/group (requires
root privileges)
  remove-user <name>            Remove USBGuard IPC user/group (requires
root privileges)
```

Listing 82. USBGuard Policy Help

```
# usbguard generate-policy --help
Usage: usbguard generate-policy [OPTIONS]

Options:
  -p, --with-ports  Generate port specific rules for all devices.
  -P, --no-ports-sn Don't generate port specific rule for devices
without an iSerial value.
  -t, --target <T> Generate an explicit "catch all" rule with the
specified target. Possible targets: allow, block,
reject.
  -X, --no-hashes   Don't generate a hash attribute for each device.
  -H, --hash-only    Generate a hash-only policy.
  -h, --help         Show this help.
```

A.4.3. Manage administrative policy and daemon configuration

Usage of USB Guard can be done multiple ways. The daemon needs to be enabled and the list of persistent rules is kept in `/etc/usbguard/rules.conf`.

Listing 83. Enabling USB Guard and Listing devices

```
# systemctl enable usbguard --now
# usbguard list-devices
# usbguard list-rules
```

Listing 84. Adding USB Guard Rules and Listing devices

```
# usbguard list-devices
# usbguard allow-device -p <device #>
```

Securing Access to the USBGuard IPC

The configuration file in the example below, instructs the **usbguard** daemon to load rules from the `/etc/usbguard/rules.conf` file and only allow users from the **usbguard** group to use the IPC interface.

Listing 85. Example Config

```
RuleFile=/etc/usbguard/rules.conf
IPCAccessControlFiles=/etc/usbguard/IPCAccessControl.d/
```

Adding and Removing Users and Permissions for the IPC Access Control List

The **usbguard add-user** and **usbguard remove-user** commands can be used to modify the IPC ACLs. In the example below, we will allow users from the USBGuard group to modify the USB device authorization state, list USB devices, and listen to events as well as list the USB authorization policy.

Listing 86. Adding IPC Permissions to a specific user

```
# usbguard add-user -u travis --devices=modify,list,listen --policy=list
--exceptions=listen
```

A.5. Manage system login security using pluggable authentication modules (PAM)

PAM is covered in Chapter 5 of the Student Guide.

PAM config files are located in `/etc/pam.d/`. Some can be modified by hand ONLY and others can be modified using the **authconfig** utility. Several configuration definition files for PAM modules are

located in `/etc/security` such as the **time.conf** and **pwquality.conf** files.



Backup of PAM Configuration

It is a good idea to always have a backup copy of the original configuration files as well as have a root terminal open on a system in the event of incorrect configurations.

*Listing 87. Using **authconfig** to create a backup*

```
# authconfig --savebackup=/root/authconfigbackup
```

Getting Help with man and config files

There are many man pages for PAM and PAM modules for getting help.

Listing 88. PAM Modules

```
# man -k pam_
```

Listing 89. PAM Configuration File

```
# man pam.conf
```

Listing 90. The authconfig tool

```
# man authconfig  
# authconfig --help
```

Listing 91. The pam_faillock Module

```
# man pam_faillock
```



Listing 92. PAM Login Definitions

```
# man login.defs
```

Listing 93. The pam_time Module

```
# man pam_time  
# man time.conf
```

Listing 94. The pam_access Module

```
# man pam_access  
# man access.conf
```

*Listing 95. The PAM Main Config Files that are linked to *-ac*

```
# man system-auth  
# man password-auth
```

Listing 96. Using PAM to Audit TTY Input

```
# man pam_tty_audit
```

A.5.1. Configure password quality requirements

Listing 97. Modifying the Password Quality File

```
# vim /etc/security/pwquality.conf
```



Password quality requirements can also be changed/modified with the **authconfig** utility.

*Listing 98. Modifying Password Quality with **authconfig***

```
# authconfig --passminlen=12 --update
```

A.5.2. Configure failed login policy (Chapter 5 Lab)

The PAM **faillock** module is used to configure failed login policies. In the example below, we are wanting to lockout accounts for 3 incorrect attempts for 10 minutes and a reset/unlock time of 10 minutes. We want to even deny the root account for incorrect login attempts.

*Listing 99. Using **authconfig** to modify PAM for **faillock***

```
# authconfig --enablefaillock --faillockargs="even_deny_root deny=3 fail_interval=600  
unlock_time=600" --update
```

A.5.3. Modify PAM configuration files and parameters

There are multiple ways to interact with PAM. When using the **authconfig** tool, changes are made to the **-ac** files and those are generally linked to the ***system-auth** and **password-auth** files.

If you are using both manual and **authconfig** methods to make changes it is a good idea to have the **system-auth-local** and **password-auth-local** linked symbolically to the main config files and that the **system-auth-ac** and **password-auth-ac** files are referenced within the *-local files.


```
# vim system-auth-local
auth            include      system-auth-ac
account         required     pam_time.so
account         include      system-auth-ac
password        include      system-auth-ac
session         include      system-auth-ac

# vim password-auth-local
auth            include      password-auth-ac
account         required     pam_time.so
account         include      password-auth-ac
password        include      password-auth-ac
session         include      password-auth-ac
```



When using the **authconfig** command, ALWAYS end the command with **--update** to write the update to the correct ***-ac** file.

A.6. Configure system auditing

Auditing is covered in Chapter 6 of the student lab guide.

The system auditing is performed with the **auditd** daemon/service. The configuration files are managed in **/etc/audit/auditd.conf**.

Useful Audit Rules and Articles



- **Auditing** **Including/Excluding** **Specific** **Users** **Watch** **Files:**
<https://access.redhat.com/solutions/2482221>
- **Auditing** **Including/Excluding** **Specific** **Users** **System** **Calls:**
<https://access.redhat.com/solutions/2477471>
- **Excluding Crond from Auditing:** <https://access.redhat.com/solutions/124213> - It should be noted that this requires SELinux to be running and contexts to be correctly labeled.

A.6.1. Write Rules to Log Auditable Events (GE: P222)

When rules are changed, the service will need to be restarted.

Getting Help with Auditing

Listing 101. auditctl usage

```
# man auditctl
```

Listing 102. audit.rules usage

```
# man audit.rules
```



Listing 103. Man auditd

```
# man auditd
```

Listing 104. auditd Configuration

```
# man auditd.conf
```

Listing 105. audispd.conf Configuration

```
# man audispd.conf
```

Listing 106. Restarting the Audit Service

```
# service auditd restart
```

There are some extra packages that can be installed for auditing

Listing 107. audispd-plugins

```
# yum install audispd-plugins
```

Example 25. Setting Audit Logs up to send Audit Logs to a Remote Logging Server

Need to install some packages and plugins for auditd.

Listing 108. audispd-plugins

```
# yum install audispd-plugins
```

Listing 109. Setup Forwarding to Remote Server (audisp-remote.conf)

```
# vim /etc/audisp/audisp-remote.conf
...output omitted...
remote_server = 172.25.250.11
port = 60
```

Listing 110. Setup Forwarding to Remote Server (auremote.conf)

```
# vim /etc/audisp/auremote.conf
active = yes
```

Example 26. Setting Audit Logs up to receive Audit Logs from a Remote Server

Need to install some packages and plugins for auditd.

Listing 111. audispd-plugins

```
# yum install audispd-plugins
```

Listing 112. Configure **Audit** to Listen

```
# vim /etc/audit/auditd.conf
tcp_listen_port = 60
```

Listing 113. Configure the Firewall

```
# firewall-cmd --add-port=60/tcp --permanent
# firewall-cmd --reload
```

A.6.2. Enable pre-packaged rules

Audit rules are kept in `/usr/share/doc/audit-*/rules/`

Listing 114. Copying the STIG Audit Rules

```
# cp /usr/share/doc/audit-*/rules/30-stig.rules /etc/audit/rules.d/
```

Listing 115. Enabling the STIG Audit Rules

```
# augenrules --load
```

A.6.3. Produce audit reports

There are multiple ways to get help with auditing. The Linux Security Guide document has many examples of producing audit reports in Chapter 6, Section 8.

Example 27. Audit Report Sample

Listing 116. Example Audit Report

```
# aureport -x --summary

Executable Summary Report
=====
total  file
=====
1625  /usr/sbin/sshd
540   /usr/sbin/crond
261   /usr/lib/systemd/systemd
123   /usr/sbin/xtables-multi
46    /usr/bin/sudo
30    /usr/bin/kmod
18    /usr/sbin/ebtables-restore
16    /usr/sbin/groupadd
12    /usr/sbin/useradd
7     /usr/bin/su
6     /usr/lib/systemd/systemd-update-utmp
4     /usr/bin/passwd
```

Getting Help

Listing 117. ausearch

```
# man ausearch
```



Listing 118. autrace

```
# man autrace
```

Listing 119. aureport

```
# man aureport
```

A.7. Configure SELinux (Chapter 8 of Student Guide)

SELinux is used to confine Linux users and processes. There are also special man pages that can be built and used for SELinux.

Regular Man Pages

There are many existing **man** pages for SELinux. These can be used from any system without needing to build new man pages.

Listing 120. setsebool

```
# man setsebool
```

Listing 121. restorecon

```
# man restorecon
```

Listing 122. seinfo

```
# man seinfo
```

Listing 123. sesearch

```
# man sesearch
```

Listing 124. semodule

```
# man semodule
```

Listing 125. audit2allow

```
# man audit2allow
```

Extending SELinux Tools and Man Pages

There are utilities that can be installed in order to work with SELinux on the system as well as additional **man** page sources.

Listing 126. Installing SELinux Policy Utilities

```
# yum install policycoreutils-devel
```



Listing 127. Installing SELinux Policy Utilities

```
# sepolicy manpage -a
/tmp/NetworkManager_selinux.8
/tmp/abrt_selinux.8
/tmp/abrt_dump_oops_selinux.8
/tmp/abrt_handle_event_selinux.8
/tmp/abrt_helper_selinux.8
/tmp/abrt_retrace_coredump_selinux.8
/tmp/abrt_retrace_worker_selinux.8
/tmp/abrt_upload_watch_selinux.8

... output omitted ...

/tmp/zoneminder_selinux.8
/tmp/zoneminder_script_selinux.8
/tmp/zos_remote_selinux.8
```

Using the additional SELinux Man pages

Listing 128. Additional SELinux Man Pages

```
# man /tmp/user_ssh_agent_selinux.8
ser_ssh_agent_selinux(8)SELinux Policy
user_ssh_agenuser_ssh_agent_selinux(8)

NAME
    user_ssh_agent_selinux - Security Enhanced Linux Policy for
    the
    user_ssh_agent processes

DESCRIPTION
    Security-Enhanced Linux secures the user_ssh_agent processes via
    flexi
    ble mandatory access control.

    The user_ssh_agent processes execute with the user_ssh_agent_t
    SELinux
    type. You can check if you have these processes running by
    executing
    the ps command with the -Z qualifier.

    For example:

    ps -eZ | grep user_ssh_agent_t

... output omitted ...

SEE ALSO
    selinux(8), user_ssh_agent(8), semanage(8), restorecon(8),
    chcon(1),
    sepolicy(8) , setsebool(8)

user_ssh_agent                                18-10-05
user_ssh_agent_selinux(8)
```

A.7.1. Enable SELinux on a host running a simple application

SELinux needs to be enabled in GRUB, in the Config file and can be set from the command line.

Listing 129. semanage Man Pages

```
# man -k semanage
semanage (8) - SELinux Policy Management tool
semanage-boolean (8) - SELinux Policy Management boolean tool
semanage-dontaudit (8) - SELinux Policy Management dontaudit tool
semanage-export (8) - SELinux Policy Management import tool
semanage-fcontext (8) - SELinux Policy Management file context tool
semanage-ibendport (8) - SELinux Policy Management ibendport mapping tool
semanage-ibpkey (8) - SELinux Policy Management ibpkey mapping tool
semanage-import (8) - SELinux Policy Management import tool
semanage-interface (8) - SELinux Policy Management network interface tool
semanage-login (8) - SELinux Policy Management linux user to SELinux User m...
semanage-module (8) - SELinux Policy Management module mapping tool
semanage-node (8) - SELinux Policy Management node mapping tool
semanage-permissive (8) - SELinux Policy Management permissive mapping tool
semanage-port (8) - SELinux Policy Management port mapping tool
semanage-user (8) - SELinux Policy Management SELinux User mapping tool
semanage.conf (5) - global configuration file for the SELinux Management l...
```

1. Determining if SELinux is Enabled

Listing 130. Using getenforce

```
# getenforce
Enforcing
```

2. Modifying the SELinux Config File

Listing 131. Editing the Config File

```
# vim /etc/selinux/config
# 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.
SELINUX=enforcing
# SELINUXTYPE= can take one of three two values:
#   targeted - Targeted processes are protected,
#   minimum - Modification of targeted policy. Only selected processes are protected.
#   mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

3. Editing GRUB for SELinux

The following should be checked for SELinux at boot time.



The GRUB boot loader can override what is specified in the `/etc/sysconfig/selinux/config` file.

Listing 132. Ensuring SELinux is Enabled for Grub

```
# cat /etc/grub.conf
.....
    root (hd0,0)
    kernel /vmlinuz-2.6.32-279.el6.x86_64 root=/dev/md3 selinux=1 enforcing=1
    initrd /initramfs-2.6.32-279.el6.x86_64.img
.....
```

4. Labeling SELinux Contexts on the Filesystem

Listing 133. Setting System to Relabel on Boot

```
# touch /.autorelabel
# reboot
```

A.7.2. Interpret SELinux violations and determine remedial action

A.7.3. Restrict user activity with SELinux user mappings (Guided Exercise - P301)

The SEManage command with the login feature can be used to set/maintain user mappings. The student guide contains some information in Chapter 8, page 296.

Getting Help with SELinux User Mappings



Listing 134. The SEManage Login

```
# man semanage-login
semanage-login(8)
semanage-login(8)

NAME
    semanage-login - SELinux Policy Management linux user to SELinux
    User mapping tool

SYNOPSIS
    semanage login [-h] [-n] [-N] [-S STORE] [ --add -s SEUSER -r
    RANGE LOGIN | --delete LOGIN | --deleteall | --extract | --list [-C] |
    --modify -s
    SEUSER -r RANGE LOGIN ]

... output omitted ...

EXAMPLE
    Modify the default user on the system to the guest_u user
    # semanage login -m -s guest_u __default__
    Assign gijoe user on an MLS machine a range and to the staff_u
    user
    # semanage login -a -s staff_u -rSystemLow-Secret gijoe
    Assign all users in the engineering group to the staff_u user
    # semanage login -a -s staff_u %engineering

SEE ALSO
    selinux (8), semanage (8), semanage-user (8)

AUTHOR
    This man page was written by Daniel Walsh <dwalsh@redhat.com>
```

Listing 135. The SEManage User

```
# man semanage-user
semanage-user(8)
semanage-user(8)

NAME
    semanage-user - SELinux Policy Management SELinux User mapping tool

SYNOPSIS
    semanage user [-h] [-n] [-N] [-S STORE] [ --add ( -L LEVEL -R
ROLES -r RANGE -s SEUSER selinux_name ) | --delete selinux_name |
--deleteall |
    --extract | --list [-C] | --modify ( -L LEVEL -R ROLES -r RANGE -s
SEUSER selinux_name ) ]

DESCRIPTION
    semanage is used to configure certain elements of SELinux policy
without requiring modification to or recompilation from policy sources.
seman
    age user controls the mapping between an SELinux User and the roles
and MLS/MCS levels.

... output omitted ...

EXAMPLE
    List SELinux users
    # semanage user -l
    Modify groups for staff_u user
    # semanage user -m -R "system_r unconfined_r staff_r" staff_u
    Add level for TopSecret Users
    # semanage user -a -R "staff_r" -rs0-TopSecret topsecret_u
```

SELinux login contexts can be viewed with SEManage.

Listing 136. Viewing SELinux user mappings

```
# 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		

1. Preparing to set SELinux User contexts

Listing 137. Viewing SELinux user mappings

```
# semanage login --help
usage: semanage login [-h] [-n] [-N] [-S STORE] [ --add -s SEUSER -r RANGE LOGIN |
--delete LOGIN | --deleteall | --extract | --list -C | --modify -s SEUSER -r RANGE
LOGIN ]

positional arguments:
  login                login_name | %groupname

optional arguments:
  -h, --help            show this help message and exit
  -C, --loccallist      List login local customizations
  -n, --noheading       Do not print heading when listing login object types
  -N, --noreload       Do not reload policy after commit
  -S STORE, --store STORE
                        Select an alternate SELinux Policy Store to manage
  -r RANGE, --range RANGE
                        MLS/MCS Security Range (MLS/MCS Systems only) SELinux
                        Range for SELinux login mapping defaults to the
                        SELinux user record range.
  -a, --add            Add a record of the login object type
  -d, --delete         Delete a record of the login object type
  -m, --modify         Modify a record of the login object type
  -l, --list           List records of the login object type
  -E, --extract        Extract customizable commands, for use within a
                        transaction
  -D, --deleteall      Remove all login objects local customizations
  -s SEUSER, --seuser SEUSER
                        SELinux user name
```

2. Adding new user to group and SELinux Conext

The following will add the **operator2** users to the **sysadm_u** SELinux context and add it to the wheel group.

Listing 138. Using the USERADD command with SELinux Mapping

```
# useradd -G wheel -Z sysadm_u operator2
```

3. Setting the SEBool parameters

When changing user context mapping it is important to ensure that the contexts have the correct permissions and mappings.

*Listing 139. Allowing SSH for the **sysadm_n** User Context*

```
# setsebool -P ssh_sysadm_login on
```

A.7.4. Analyze and correct existing SELinux configurations

There are some additional tools for SELinux policy editing and troubleshooting.

Listing 140. SELinux Troubleshooting

```
# yum install policycoreutils-devel setools-console
```

Sometimes, filesystem SELinux labels and contexts are not correct and need to be set.



SEBoolean Values

SELinux users can be prevented from executing files in the user's home directory and the /tmp directory.

Listing 141. SELinux Boolean to set user_exec_content bit

```
# setsebool -p user_exec_content off
```

SELinux users can be prevented from logging in with SSH when confined.

Listing 142. SELinux Boolean to allow sysadm SELinux users SSH Login

```
# setsebool -p ssh_sysadm_login on
```

Getting help for SELinux Contexts

Listing 143. SELinux Contexts



```
# man semanage-fcontext
semanage-fcontext(8)
semanage-fcontext(8)
```

NAME

semanage-fcontext - SELinux Policy Management file context tool

SYNOPSIS

```
semanage fcontext [-h] [-n] [-N] [-S STORE] [ --add ( -t TYPE -f
FTYPE -r RANGE -s SEUSER | -e EQUAL ) FILE_SPEC ) | --delete ( -t TYPE -f
FTYPE
| -e EQUAL ) FILE_SPEC ) | --deleteall | --extract | --list [-C] |
--modify ( -t TYPE -f FTYPE -r RANGE -s SEUSER | -e EQUAL ) FILE_SPEC ) ]
```

DESCRIPTION

semanage is used to configure certain elements of SELinux policy without requiring modification to or recompilation from policy sources.

semanage

semanage fcontext is used to manage the default file system labeling on an SELinux system. This [command](#) maps file paths using regular expressions to

SELinux labels.

... output omitted ...

EXAMPLE

remember to run restorecon after you [set](#) the file context

Add file-context [for](#) everything under /web

```
# semanage fcontext -a -t httpd_sys_content_t "/web(/.*)?"  
# restorecon -R -v /web
```

Substitute /home1 with /home when setting file context

```
# semanage fcontext -a -e /home /home1  
# restorecon -R -v /home1
```

For home directories under top level directory, [for](#) example /disk6/home,

execute the following commands.

```
# semanage fcontext -a -t home_root_t "/disk6"  
# semanage fcontext -a -e /home /disk6/home  
# restorecon -R -v /disk6
```

SEE ALSO

selinux [\(8\)](#), semanage [\(8\)](#)

AUTHOR

This man page was written by Daniel Walsh <dwalsh@redhat.com>

SELinux Boolean Values

SELinux has several boolean values and settings. It is helpful to be able to get a list of the values that can be used with the **setsebool** command.

Listing 144. Listing SEBool Values



```
# getsebool -a
abrt_anon_write --> off
abrt_handle_event --> off
abrt_upload_watch_anon_write --> on
antivirus_can_scan_system --> off
antivirus_use_jit --> off
auditadm_exec_content --> on
authlogin_nsswitch_use_ldap --> off

... output omitted ...

zarafe_setrlimit --> off
zebra_write_config --> off
zoneminder_anon_write --> off
zoneminder_run_sudo --> off
```



The SELinux **chcon** command can be used to change SELinux contexts, but it is not a permanent fix or solution as a **restorecon** command will undo all changes.

Listing 145. The **chcon** command

```
# man chcon

# chcon --reference =/reference_path /New_File_path
```



SELinux Reference

There is a really good SELinux reference:

<https://www.digitalocean.com/community/tutorials/an-introduction-to-selinux-on-centos-7-part-3-users>

A.8. Enforce security compliance (Chapter 9)

Security compliance can be enforced and checked using OpenSCAP.

A.8.1. Install OpenSCAP and Workbench

Listing 146. Installing SCAP Workbench

```
# yum install scap-workbench
```



SCAP utilities need to be installed on systems being scanned. The **scap-security-guide** (SSG) will install OpenSCAP and all other utilities needed including the SSG content.

Listing 147. Installing OpenSCAP Utilities

```
# yum install scap-security-guide
```

Getting SCAP Help

The SCAP Security Guide includes various pre-defined XCCDF rules.



Listing 148. Getting help from the SCAP Security Guide

```
# man scap-security-guide
scap-security-guide(8)                                System
Manager's Manual                                     scap-security-
guide(8)
```

NAME

SCAP Security Guide - Delivers security guidance, baselines, and associated validation mechanisms utilizing the Security Content Automation Protocol (SCAP).

DESCRIPTION

The project provides practical security hardening advice for Red Hat products, and also links it to compliance requirements in order to ease deployment activities, such as certification and accreditation. These include requirements in the U.S. government (Federal, Defense, and Intelligence Community) as well as of the financial services and health care industries. For example, high-level and widely-accepted policies such as NIST 800-53 provides prose stating that System Administrators must audit "privileged user actions," but do not define what "privileged actions" are. The SSG bridges the gap between generalized policy requirements and specific implementation guidance, in SCAP formats to support automation whenever possible.

The projects homepage is located at: <https://www.open-scap.org/security-policies/scap-security-guide>

EXAMPLES

To scan your system utilizing the OpenSCAP utility against the osppe-rhel7 profile:

```
oscap xccdf eval --profile osppe-rhel7 --results /tmp/
`hostname`-ssg-results.xml --report /tmp/`hostname`-ssg-results.html
--oval-results
/usr/share/xml/scap/ssg/content/ssg-rhel7-xccdf.xml
```

Listing 149. Getting help from the **oscap**

```
# man oscap
OSCAP(8)                                     System
Administration Utilities
OSCAP(8)

NAME
    oscap - OpenSCAP command line tool

SYNOPSIS
    oscap [general-options] module operation [operation-options-and-arguments]

DESCRIPTION
    oscap is Security Content Automation Protocol (SCAP) toolkit based on OpenSCAP library. It provides various functions for different SCAP specifications (modules).

    OpenSCAP tool claims to provide capabilities of Authenticated Configuration Scanner and Authenticated Vulnerability Scanner as defined by The National Institute of Standards and Technology.

... output omitted ...

EXAMPLES
    Evaluate XCCDF content using CPE dictionary and produce html report. In this case we use United States Government Configuration Baseline (USGCB) for Red Hat Enterprise Linux 5 Desktop.

    oscap xccdf eval --fetch-remote-resources --oval-results \
        --profile
united_states_government_configuration_baseline \
        --report usgcb-rhel5desktop.report.html \
        --results usgcb-rhel5desktop-xccdf.xml.result.xml \
        --cpe usgcb-rhel5desktop-cpe-dictionary.xml \
        usgcb-rhel5desktop-xccdf.xml
```

A.8.2. Use OpenSCAP and Red Hat Insights to scan hosts for security compliance

Skipping for now

A.8.3. Use OpenSCAP Workbench to tailor policy

Getting OSCAP Info from a File or Tailoring File

Sometimes when using a tailoring file or datastream file, it is difficult to know the name of the **profile** being used.

*Listing 150. Using **oscap** to get Profile Info*

```
# oscap info
ERROR:    SCAP file needs to be specified!

Usage:    oscap [options] info some-file.xml
Help:     oscap info -h
```

*Listing 151. Using **oscap** to get Profile Info from DS File*

```
# oscap info /usr/share/xml/scap/ssg/content/ssg-rhel7-ds.xml
Document type: Source Data Stream
Imported: 2018-01-08T08:03:07

Stream: scap_org.open-scap_datastream_from_xccdf_ssg-rhel7-xccdf-1.2.xml
Generated: (null)
Version: 1.2
Checklists:
  Ref-Id: scap_org.open-scap_cref_ssg-rhel7-xccdf-1.2.xml
  Status: draft
  Generated: 2018-01-08
  Resolved: true
  Profiles:
    Title: Standard System Security Profile
    Id: xccdf_org.ssgproject.content_profile_standard
    Title: PCI-DSS v3 Control Baseline for Red Hat Enterprise
Linux 7
    Id: xccdf_org.ssgproject.content_profile_pci-dss
    Title: C2S for Red Hat Enterprise Linux 7
    Id: xccdf_org.ssgproject.content_profile_C2S
    Title: Red Hat Corporate Profile for Certified Cloud Providers
(RH CCP)
    Id: xccdf_org.ssgproject.content_profile_rht-ccp
    Title: Common Profile for General-Purpose Systems
    Id: xccdf_org.ssgproject.content_profile_common
    Title: DISA STIG for Red Hat Enterprise Linux 7
    Id: xccdf_org.ssgproject.content_profile_stig-rhel7-disa

... output omitted ...
```



Listing 152. Getting the Profile ID from a Tailoring File

```
# oscap info lab-tailoring.xml
Document type: XCCDF Tailoring
Imported: 2018-10-05T21:26:04
Benchmark Hint: /usr/share/xml/scap/ssg/content/ssg-rhel7-ds.xml
Profiles:
  Title: Common Profile for General-Purpose Systems [CUSTOMIZED]
  Id: xccdf_com.example_profile_lab-rhel7
```

A.8.4. Use OpenSCAP Workbench to scan an individual host for security compliance



Need to ensure SCAP packages are loaded onto the systems being scanned from SCAP Workbench.

A.8.5. Use Red Hat Satellite server to implement an OpenSCAP policy

Skipping for now

A.8.6. Apply OpenSCAP remediation scripts to hosts

OpenSCAP can be used to scan systems and generate remediation scripts. It can also be used to create HTML compliance reports.

Listing 153. Generating an OpenSCAP Results File

```
oscap xccdf eval \  
> --profile xccdf_com.example_profile_lab-rhel7 \  
> --tailoring-file /home/student/lab-tailoring.xml \  
> --results /root/lab-results.xml \  
> /usr/share/xml/scap/ssg/content/ssg-rhel7-ds.xml
```

Listing 154. Generating an OpenSCAP Results Report

```
# oscap xccdf generate report \  
> lab-results.xml > lab-results.html
```

Listing 155. Generating an Ansible Playbook from Results File

```
# oscap xccdf generate fix \  
> --profile xccdf_com.example_profile_lab-rhel7 \  
> --tailoring-file lab-tailoring.xml \  
> --fix-type ansible \  
> --result-id "" \  
> lab-results.xml > /home/student/RH415/labs/oscap-review/fix.yml
```