Exam Objectives

General nodes for the exams process:

- Update the **man page** database

man *Database Update*

```
# mandb
```

- Update the **locate** database

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

NOTE

YUM Usage

Getting Security Info from yum

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

Getting Security Update Information

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

Listing Updates and Searching for Critical

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

Use Red Hat Ansible Engine

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

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

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

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

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 1. Creating an Ansible Inventory

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

inventory *File Contents*

```
[LOCAL]
workstation

[SERVERS]
servera
serverb

[EVERYONE:children]
LOCAL
SERVERS
```

3. Create an Ansible Config File

Example 2. Creating an Ansible Configuration File

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

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
```

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

TIP 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.

NOTE 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.

Configure managed nodes

NOTE 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.

NOTE

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.

Generate SSH Keys

```
# ssh-keygen
```

Copy SSH Keys to Managed Nodes

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

Create a SUDOERS File for Ansible User

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

Copy SUDOERS File for Ansible User to Managed Nodes

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

Configure simple inventories

Example 3. Creating an Ansible Inventory

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

inventory *File Contents*

```
[LOCAL]
workstation

[SERVERS]
servera
serverb

[EVERYONE:children]
LOCAL
SERVERS
```

Perform basic management of systems

Run a provided playbook against specified nodes

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

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]$
```

WARNING

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:

Ansible Playbook Limiting to Specified Server

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

NOTE

System Documentation and Man Pages

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

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
```

TIP

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

Configure intrusion detection

Installation and Configuration of AIDE.

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

Install AIDE

Installing AIDE

```
# yum install aide
```

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

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.

Configuring AIDE

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

Initializing AIDE

```
# aide --init
```

Configure encrypted storage

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

Encrypt and decrypt block devices using LUKS

1. Configure the partition

Example 4. Using Parted to Create/Lookup Partition

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:
```

Parted to create partitions

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

TIP

parted *options*

Use **man parted** to get the syntax

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

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

2. Create the LUKS partition

Using Cryptsetup to apply LUKS to partition

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

3. Name and Open the LUKS partition

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

Preparting 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

Closing LUKS partition

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

Configure encrypted storage persistence using NBDE

TIP 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 5. Installing TANG on Servers

1. Install TANG

Installing TANG packages

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

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

Enabling TANG

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

3. Configure port 80 on the firewall and reload rules

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 6. Installing Clevis Packages

1. Install the clevis packages

Installation of Clevis

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

2. Configure Clevis to Bind to TANG servers

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

CAUTION The "URL" must be in quotes.

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

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

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 Cryptab and FStab files for the encrypted volume

Edit **cryptab**

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

Edit **fstab**

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

Change encrypted storage passphrases

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

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

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

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
```

Restrict USB devices

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

Install USBGuard

Installing USBGuard

```
# yum install -y usbguard usbutils udisks2
```

NOTE

Getting Help

USBGuard man page

```
# man usbguard
```

USBGuard-Rules Config man page

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

USBGuard-Daemon man page

```
# man usbguard-daemon
```

USBGuard-Daemon Config man page

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

Write device policy rules with specific criteria to manage devices

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

Generating a USB Policy

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

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**.

Enabling USB Guard and Listing devices

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

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.

WARNING

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.

Using **authconfig** *to create a backup*

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

TIP

Getting Help with man and config files

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

PAM *Modules*

```
# man -k pam_
```

PAM *Configuration File*

```
# man pam.conf
```

The **authconfig** *tool*

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

The **pam_faillock** *Module*

```
# man pam_faillock
```

PAM *Login Definitions*

```
# man login.defs
```

The **pam_time** Module

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

The **pam_access** Module

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

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

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

Configure password quality requirements

Modifying the Password Quality File

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

NOTE

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

Modifying Password Quality with **authconfig**

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

Configure failed login policy

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.

Using authconfig to modify PAM for faillock

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

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.

Including the AC files in the local file

```
# 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
```

CAUTION

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

Configure system auditing

Write rules to log auditable events

Enable pre-packaged rules

Produce audit reports

Configure SELinux

Enable SELinux on a host running a simple application

Interpret SELinux violations and determine remedial action

Restrict user activity with SELinux user mappings

Analyze and correct existing SELinux configurations

Enforce security compliance

Install OpenSCAP and Workbench

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

Use OpenSCAP Workbench to tailor policy

Use OpenSCAP Workbench to scan an individual host for security compliance

Use Red Hat Satellite server to implement an OpenSCAP policy

Apply OpenSCAP remediation scripts to hosts