Linux Security:
Access Control Framework

This material is based on work supported by the National Science Foundation under Grant No. 0802551

National Science Foundation
WHERE DISCOVERIES BEGIN

Lesson Overview

As a Linux administrator, you will be responsible for securing desktops, laptops, mobile devices, and servers from unwanted changes and access. If you are working in a regulated organization, you must ensure that users do not have access to data that is not required for them to do their jobs.

Laws like the Health Insurance Privacy Act, for those in the medical and insurance industries, or the Federal Educational Records and Privacy Act, for those in the education industry, and the financial privacy acts, for those in the financial industries, require data be kept confidential. Thus, it is important that computer users only have access to the information they need to do their jobs and nothing else. This system of data access rules also minimizes the risk of unwanted changes to the operating system.

In this lesson, we will explore processes, tools, and modules that make it possible to limit access to sensitive data on a Linux-based system. Understanding this topic is important for any system administrator configuring systems in the workplace.

Objective

You should know what will be expected of you when you complete this lesson. These expectations are presented as objectives. Objectives are short statements of expectations that tell you what you must be able to do, perform, learn, or adjust after reviewing the lesson.

Lesson Objective:

Given the need to provide enhance security to Internet facing servers, the student will be able to propose and implement access control tools, including GrSecurity, AppArmor, or SELinux tools as required by industry standards.

Lesson Outline

In this lesson, you will explore:

- Access Control Frameworks
- DAC
- MAC
- RBAC
- RSBAC
- GrSecurity
- AppArmor
- SELinux

Resources and Notes

This lesson uses Fedora Linux for demonstration. To complete this lesson successfully, you must have access to:

- ❖ Fedora Linux on bare metal or as a virtual install
- ❖ 10 Gb of hard drive space dedicated to the operating system's use
- ❖ A command shell
- ❖ Internet for research
- ❖ Word processor

Use the resources on the right to configure your system for Fedora.

Resources:

- Download Virtualbox
- Virtualbox for Linux Hosts
- Install Fedora to Virtualbox
- Virtualbox manual
- Using Virtualbox with Ubuntu (process similar to Fedora)

Access Control Frameworks

In this lesson, we will begin with what we know about Linux system security. Then we will explore Discretionary Access Control (DAC) and Mandatory Access Control (MAC). We will then explore and work with different packages that provide these levels of access control. Finally, you will configure a system using these new skills to meet a set of industry specific standards.

Before we go much further, we need to define what we mean by "access." Access is the ability to do something (execute an action) within the computer. This action could be reading a file, writing a file, executing a program, or accessing a resource.

Access control is the means by which the ability to perform an action on the computer is explicitly enabled or restricted in some way. This restriction can be either physical or system-based. Computer-based access controls can prescribe what user has access or what processes may have access to a specific system resource, and also what type of access is permitted .

Recommended Reading

- Access Control

Evolution of Access Control

Access control for the Linux operating system evolved in the same way, or perhaps we should say, “started with” the access control process found on the Unix operating system. Unix began with the file level controls that the user can adjust by using *chmod* and *chown* commands.

The first access control was based on file rights that were used to verify whether a user had the right to read, write, or execute a file. In this system, every user had a user ID (login) and a set of files (directory tree) assigned to that user. If a user did not own a directory tree, he or she would have rights on someone else’s tree to read, write, or change.

The security was simple. If *User A* wanted *User X* to be able to write to a file, *User A* would ensure *User B* was in the appropriate user group, and then *User A* would use the *chown* and *chmod* command to grant that group of users access to the file. Conversely, if *User A* wanted to take away the rights to that file, he would use the *chown* and *chmod* commands to accomplish the same goal.

The process above is an example of Discretionary Access Control (DAC).

Recommended Reading

- Rule Based Access Control

Introduction to DAC

DAC is short for *Discretionary Access Control*—an access control system in which users have the ability to make decisions regarding access to files, directories, and devices. Examples of DAC include the type of permissions on the files in a user's directory and the *chmod* and *chown* commands.

The advantage of DAC is that users can make decisions on their own without administrator help. The disadvantage is that users can make access control decisions on their own without administrator help! A user could accidently make their entire directory tree (regardless of the sensitivity of the information) available for anyone to browse, read, or change.

For example, users could issue a **chmod –R a+wrx *** command, which would make all files in their directory tree executable, readable, and writeable by all users without restriction. Files that are writeable by default also means that they can be deleted.

While DAC is helpful, the risk of exposing sensitive data makes it unsuitable for organization with privacy concerns.

There is another type of access control that eliminates (or at least, mitigates) this risk.

Required Reading
- DAC Access Control

Introduction to MAC

One of the risks of DAC was that users had the option of participating or not participating in access control, so Mandatory Access Control (MAC) was developed to require participation from every user.

With MAC, users do not have the option of opting in or out of access control. It is mandatory. In a MAC environment, the operating system constrains the ability of the user to perform an operation or access a program unless the user has specific permission to perform the activity.

MAC is usually accomplished through the use of a kernel patch or modules along with a set of utilities for configuration. SELinux, which is one type of MAC, will be discussed in much greater detail later in this lesson.

Mandatory access control has both positive and negative attributes. MAC forces all users to adhere to pre-set access control policies, but the process of setting up control policies is time-consuming for a system administrator.

In both DAC and MAC, there are two available configurations of access control. First, there is Role Based Access Control (RBAC) and then there is Rule Set Based Access Control (RSBAC). Both will be discussed.

Recommended Reading

- Mandatory Access Control

Role Based Access Control

Role-based access control, also known as RBAC, extends the security of Linux operating systems through the use of access control lists. RBAC uses kernel patches to add to the inherent security of the Linux kernel. These kernel modules also add to the auditing capability of the Linux kernel.

In Role Based Access Control, the decisions on which users are allowed access to which tasks are based on the role or job function of the user.

For example, a doctor, would have access to operations having to do with diagnosis, prescriptions, and the ordering of tests. The role of a researcher can be limited to gathering anonymous clinical information for studies. The role of a receptionist can be limited to taking phone calls and scheduling appointments. The role of the billing clerks can be limited to printing bills and posting payments. As you can see, role based access control is one way that personal information can be protected. If a user does not have a need to access information in his or her job function or level of responsibility, then he or she will not have access.

Continued on next slide . . .

Recommended Reading
- Securing Linux

Role Based Access Control (Contd)

Based on role definitions, organizations can establish the rules for the association of operations with roles.

Operations can be specified in a manner that can be used in the demonstration and enforcement of laws or regulations.

Two such laws (or regulations) that Linux Administrators need to be aware of are HIPAA (Health Insurance Privacy Act) and FERPA (Federal Education Records Privacy Act). Both of these rules require role-based access control to be used on all systems.

Recommended Reading

- Better Security

Advantages of RBAC

Role Based Access Control (RBAC) has several advantages. Once the roles are clearly defined, the system administrator can control access at a level that is natural to the way the organization conducts business.

Additionally, where more than one level of access is needed, the system administrator can overlap the role assignments to grant additional access.

For the system administrator, RBAC becomes an exercise in adding or removing users from a role rather than changing all rights on a set of directories or file structures.

Another form of access control is the Rule Set Based Access Control. We will explore this option next.

Recommended Reading

- Role Based Access Control

Rule Set Based Access Control

SELinux

Rule Set Based Access Control

Rule Set Based Access Control (RSBAC) is an open source security extension to current Linux kernels, which has been continuously developed for several years. Version 1.1.2 was released on August 27, 2001.

RSBAC was designed according to the Generalized Framework for Access Control (GFAC) to overcome the deficiencies of access control in the standard Linux systems and to make a flexible combination of security models as well as proper access logging as possible.

One of the more frequently used Rule Set Based Access Control applications is SELinux.

Required Reading:

- Key Features of RSBAC

SELinux

SELinux stands for *Security Enhanced Linux* and is based on the United States Department of Defense mandatory access controls. It makes use of the Linux Security Modules (LSM) in the Linux Kernel.

SELinux is not a different distribution of Linux, rather it is a set of kernel enhancements and applications that can be added to various Linux distributions.

The SELinux architecture strives to control and streamline the volume of rules software administrators use to enforce security policies. SELinux can be installed on most distributions and Fedora and Ubuntu are no exceptions.

Much of the work to implement SELinux and other kernel-based security packages has been a joint effort between the NSA, RedHat, and the community of SELinux developers.

For Review
- SELinux Project
- Security Enhanced

Required Reading
- NSA and SELinux

What does SELinux do?

SELinux makes decisions based on policies and roles to which users and objects are assigned. It is easy to think of SELinux as a gatekeeper.

If the user or service tries to run an application, SELinux compares the request to the default system policies.

If the request matches the approval policies, then SELinux reviews the user or role polices.

If the request passes these additional policies, then the object is accessed. If the service does not pass these policies then the security server generates a "policy denied" message.

Resources
- Fedora SELinux
- SELinux FAQs

SELinux Decision Process

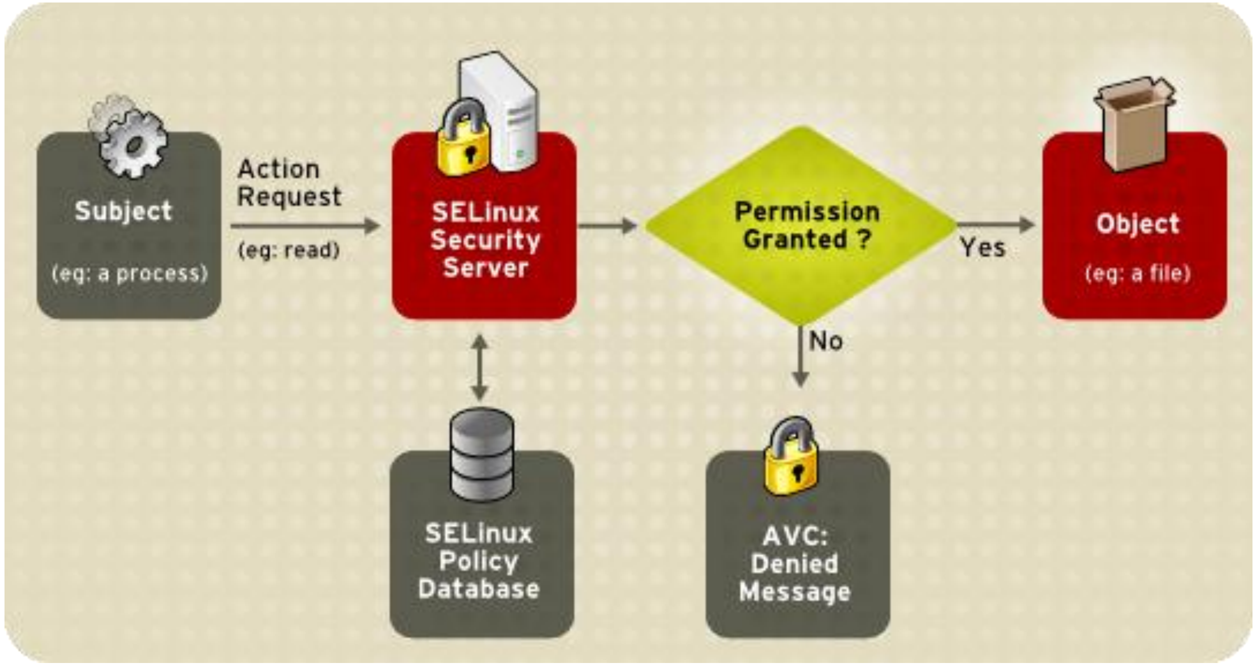


Image source: centos.org

SELinux Decision Making Flow Chart

Resources

- Introduction to SELinux

How to Install SELinux in Fedora

Installing SELinux is done through the software management tools for your distribution. If you have installed Fedora, it is most likely SELinux is already installed, though you will want to verify this.

If you have installed Ubuntu you will need to install SELinux through the APT software management tools. On Fedora, follow these steps:

Open a terminal session.

1. Type: **su**
2. Enter your root password when prompted.
3. Update the system first by typing: **yum upgrade**
4. Type: **yum install setools policy-coreutils-newrole selinux-policy-devel**
5. REBOOT once completed

Fedora 13 and above includes SELinux by default in the installation process.

Select **PLAY** below for a video on installing prerequisites.

View Video VideoLesson2InstallSELinux(C5L2S18).mp4

How to Install SELinux in Ubuntu

Installing SELinux on Ubuntu is done through the software management tools for your distribution. If you have installed Ubuntu it is most likely not installed, though this may change in a future release. If you have installed Ubuntu please follow these steps:

Open a terminal session.

1. Type: **sudo apt-get update**
2. Enter your password when prompted.
3. Type: **sudo apt-get upgrade**
4. Your system is now upgraded.
5. Type: **apt-get install setools selinux-policy-default selnux-policy-doc checkpolicy selinux**
6. REBOOT once completed.

Once the system is rebooted, you now have a system protected by SELinux.

Next, we need to explore the configuration files and the levels of protection they offer.

Select **PLAY** below for a video on installing prerequisites.

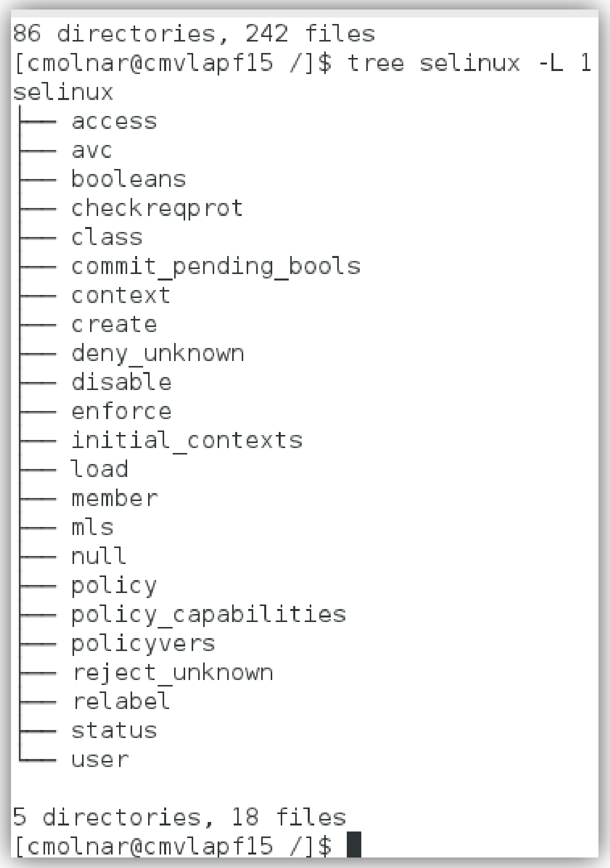
View Video VideoLesson2SeLinuxToUbuntu(C5L2S19).mp4

SELinux File System

When SELinux is operational on a machine, it creates an pseudo-file system that contains commands mostly used by the kernel subsystems. This type of file system is similar to the */proc/ pseudo-file* system.

The pseudo filesystem is usually self-sufficient and is not usually manipulated by administrators. The image to the right is an example of the */selinux/directory* structure.

SELinux is configured through configuration files in */etc/selinux* as well as several applications in the *Setools* package.



```
86 directories, 242 files
[cmolnar@cmvlapf15 /]$ tree selinux -L 1
selinux
├── access
├── avc
├── booleans
├── checkreqprot
├── class
├── commit_pending_bools
├── context
├── create
├── deny_unknown
├── disable
├── enforce
├── initial_contexts
├── load
├── member
├── mls
├── null
├── policy
├── policy_capabilities
├── policyvers
├── reject_unknown
├── relabel
├── status
└── user

5 directories, 18 files
[cmolnar@cmvlapf15 /]$
```

/selinux directory structure

The /etc/selinux/config File

SELinux is normally not used in its strictest form on machines without Internet access. For example, if a user's machine is behind a network firewall with NAT (Address Translation), SELinux does not have to enforce all its policies.

SELinux can run in three modes: *Enforcing*, *Permissive*, and *Disabled*.

- By setting SELinux to the *Enforcing* mode, it will block all services, files, and network activity contained in the rule set.
- By setting it to *Permissive* mode, SELinux will allow activities that may violate rules but will display a warning message.
- By setting the mode to *Disabled* SELinux will not operate at all, no warnings will be issued and no activities will be prevented.

It is recommended that the Linux administrator run the "permissive" mode prior to running the "enforcing" mode. This allows for troubleshooting and configuration prior to locking down a system.

Continued on next slide . . .

Select **PLAY** below for a video on updating your system.

View Video VideoLesson2SelinuxConfig(C5L2S21).mp4

Required Reading

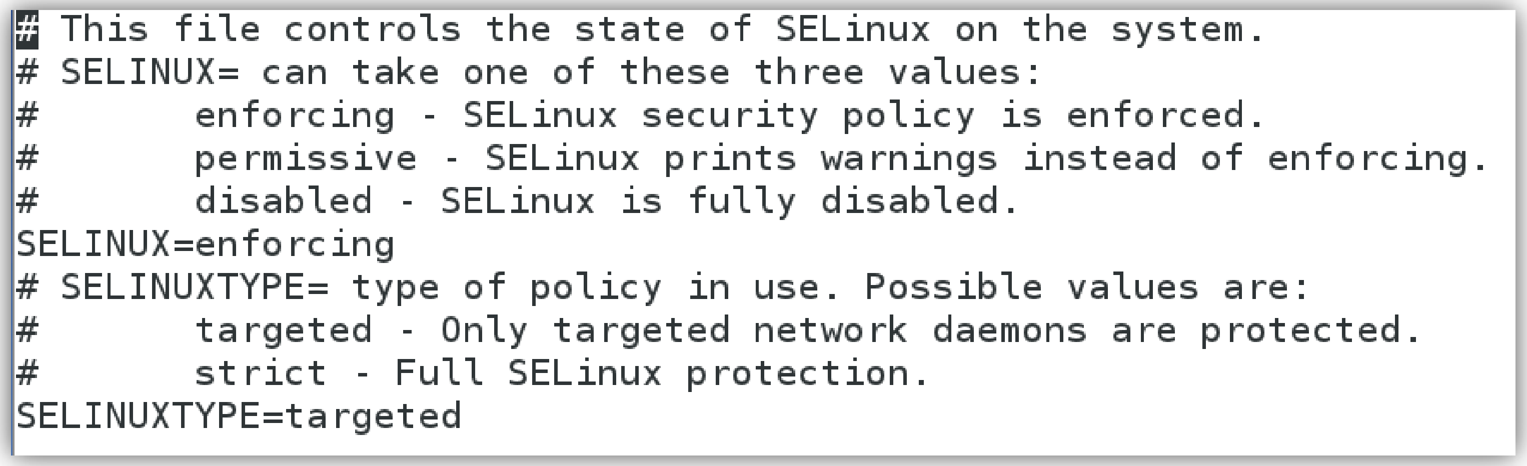
- SELinux Configuration File

The SELinux Configuration File (Contd)

In addition to the enforcing mode, the Linux administrator is able to set the control access policy. There are two options:

- ❖ Strict means that the entire Linux system is protected, and
- ❖ Targeted means that it will only enforce the policies against a set of pre-defined network services

These and other options are set in the SELinux configuration file located at: */etc/selinux/config*. The SELinux configuration is shown in the image below.



```
# 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 - SELinux is fully disabled.
SELINUX=enforcing
# SELINUXTYPE= type of policy in use. Possible values are:
#       targeted - Only targeted network daemons are protected.
#       strict - Full SELinux protection.
SELINUXTYPE=targeted
```

Additional Files Under /etc/selinux

As a root user, perform a directory tree listing for
/etc/selinux

tree –L2 /etc/selinux

You will notice many levels of additional files.

Do not alter these files manually. Many of these files are only altered using the utilities found in the Setools package as well as several GUI applications available. The SELinux system is one of the configurations in which many administrators make use of the GUI tools when available.

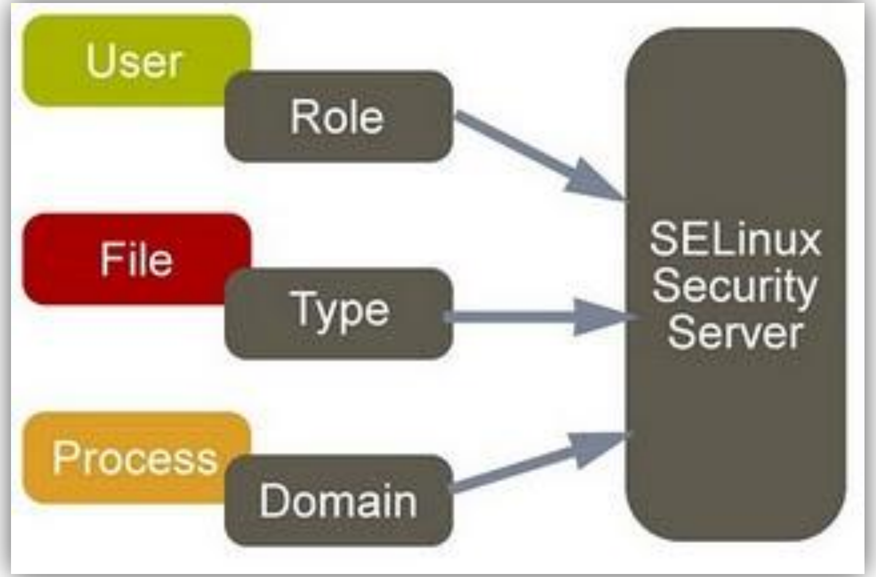
In the next part of this lesson, we are going to explore some SELinux tools associated with roles, types, and policies.

SELinux Policy

SELinux policies are a set of rules that guide the SELinux security engine. The policy defines *types* for file objects and *domains* for their processes.

The policies use *roles* to limit the domains that can be entered and *user identities* to specify the roles that can be attained.

In essence, types and domains are equivalent. The difference between the two is that types apply to objects, while domains apply to processes.



SELinux Type

An SELinux *type* is a way of grouping items based on their similarity from a security perspective. This grouping is not necessarily related to the unique purpose of an application or the content of a document.

For instance, a text file that resides in the user's home directory will be grouped with other files that reside in the user's home directory regardless of content and will be given the security type *user_home_t.*

Similarly, a database file that resides in the user's directory will be given a *user_home_t* security type because the database lives in the user's home directory.

SELinux system does not care about the purpose of the file, just the location and owner.

Required Reading

- SELinux Policies

Access Control Framework

SELinux Tools and Utilities

Setenforce Options

Most configuration files for SELinux operate in binary mode, which means decisions are made based on “yes” or “no” (1=yes; 2=no) choices. The configuration utilities understand this nomenclature and adjust settings and run-time (system generated) files accordingly.

One of the most frequently used applications is the *setenforce* application. Setenforce has two options:

setenforce 1 - forces SELinux to operate in enforcing mode

setenforce 2 –forces SELinux to operate in permissive mode

These settings are temporary until the next system boot. If you wish to alter these settings on a permanent basis, you must change the contents of /etc/selinux/config.

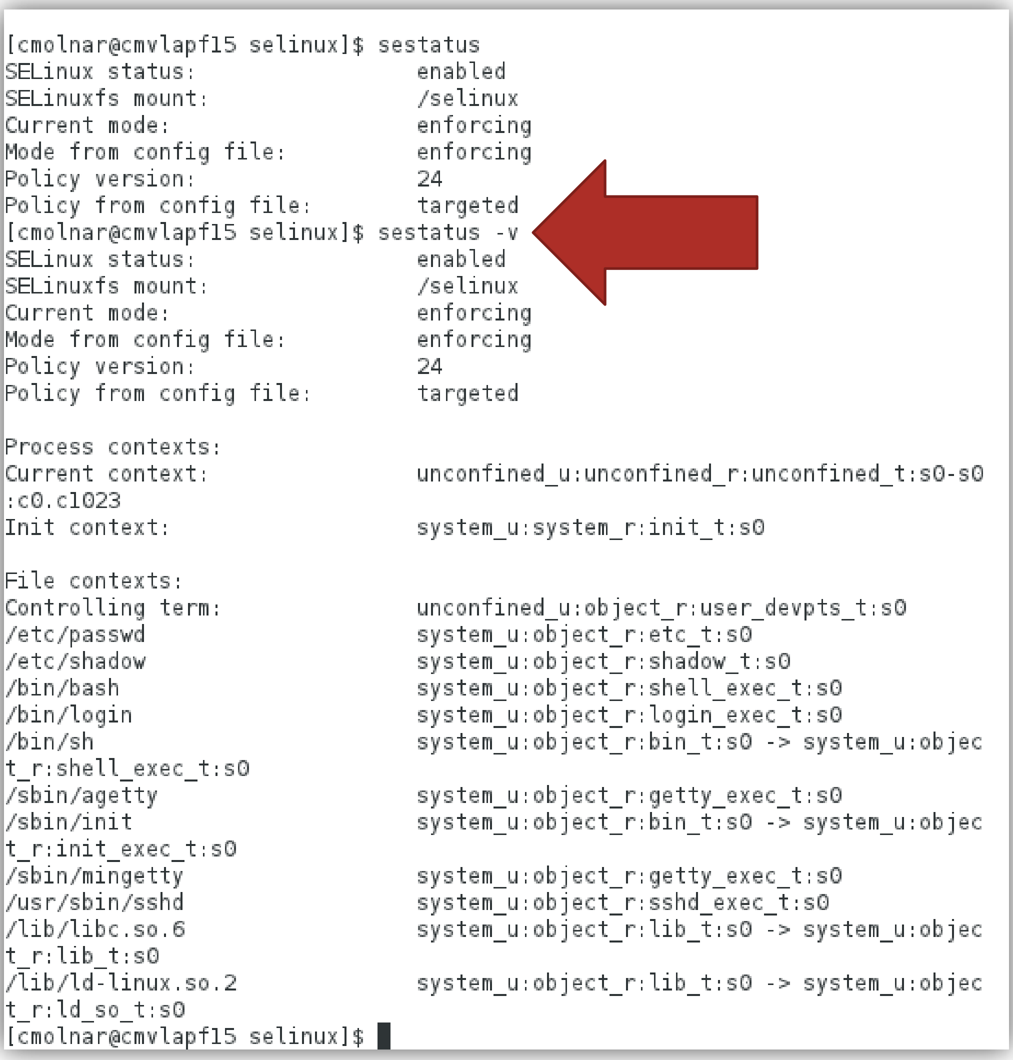
It is recommended that you operate in permissive mode initially and then move to enforcing mode when everything is working correctly. Remember for any command on the Linux system, you may type a **man** [**command_name**] and retrieve the manual page for that command.

Sestatus

To get a quick read on the operation of SELinux on an operating system, a Linux Administrator may use the **sestatus** utility. print a system status of the operation of SELinux.

For additional information, add the **-v** for verbose output onto the end of the command.

sestatus -v



```
[cmolnar@cmvlapf15 selinux]$ sestatus
SELinux status:                 enabled
SELinuxfs mount:                /selinux
Current mode:                   enforcing
Mode from config file:          enforcing
Policy version:                 24
Policy from config file:        targeted
[cmolnar@cmvlapf15 selinux]$ sestatus -v
SELinux status:                 enabled
SELinuxfs mount:                /selinux
Current mode:                   enforcing
Mode from config file:          enforcing
Policy version:                 24
Policy from config file:        targeted

Process contexts:
Current context:                unconfined_u:unconfined_r:unconfined_t:s0-s0
:c0.c1023
Init context:                   system_u:system_r:init_t:s0

File contexts:
Controlling term:               unconfined_u:object_r:user_devpts_t:s0
/etc/passwd                     system_u:object_r:etc_t:s0
/etc/shadow                     system_u:object_r:shadow_t:s0
/bin/bash                       system_u:object_r:shell_exec_t:s0
/bin/login                      system_u:object_r:login_exec_t:s0
/bin/sh                         system_u:object_r:bin_t:s0 -> system_u:objec
t_r:shell_exec_t:s0
/sbin/agetty                    system_u:object_r:getty_exec_t:s0
/sbin/init                      system_u:object_r:bin_t:s0 -> system_u:objec
t_r:init_exec_t:s0
/sbin/mingetty                  system_u:object_r:getty_exec_t:s0
/usr/sbin/sshd                  system_u:object_r:sshd_exec_t:s0
/lib/libc.so.6                  system_u:object_r:lib_t:s0 -> system_u:objec
t_r:lib_t:s0
/lib/ld-linux.so.2              system_u:object_r:lib_t:s0 -> system_u:objec
t_r:ld_so_t:s0
[cmolnar@cmvlapf15 selinux]$
```

Repair Permissions

As you may remember from an earlier slide and from your reading, SELinux stores permission and role information along with the files in a set of extended attributes. Occasionally these attributes can become damaged and may need to be repaired. The SELinux toolset comes with two utilities that can help restore and repair these attributes—*fixfiles* and *restorecon*.

The *fixfiles* command will check and/or correct the security settings (context) for all files on a file system. All changes that have been made will be restored to the most recent information stored in Selinux's binary databases.

The *restorecon* command will restore the security context of one or more files by replacing the files' extended attributes with the last known security attributes.

SELinux uses these security attributes by comparing them against the policies.

Select **PLAY** below for a video on repairing permissions.

View Video VideoLesson2Permissions(C5L2S29).mp4

Access Control Framework

Boot Process

Step 1 – System Boot

After the kernel has been loaded during the boot cycle, the initialization process is assigned the predefined initial SELinux ID (initial SID) kernel.

This initial SID is used for bootstrapping (defined as a self-propagating process used to start the Linux system) before the remaining policies are loaded.

Required Reading

- Selinux at boot

Step 2 – Kernel Mounts Drives

Once the kernel has bootstrapped itself and can load files and directories from the boot partition, the kernel runs /sbin/init and then /sbin/init mounts the /proc/ partition and then searches for the selinuxfs file system type.

If selinuxfs type is present, the system knows that SELinux is installed, enabled in the kernel, and should be running.

Required Reading

- Configure SELinux

Step 3– SELinux or not SELinux

If the /sbin/init does not find SELinux in the kernel, because the /selinux psuedo-directory is missing, then the system will continue to boot as a non-SELinux system.

Additionally, SELinux may not start because the parameter **selinux=0** was passed to the kernel at the boot prompt.

Furthermore, setting the line in /etc/selinux/config to **SELINUX=disables** will allow the system to start as a non-SELinux system.

Required Reading

- Turning off SELinux
- Quick Tips SELinux

Step 4 – Mount /selinux/

If init finds that SELinux should be loaded, the system will mount /selinux

The kernel then checks /selinux/policyvers for the supported policy version. Additionally, /sbin/init inspects /etc/selinux/config to determine which policy is active, (the targeted policy) and then loads the associated file for that policy.

If for some reason the binary policy is not the one supported by the kernel, SELinux attempts to load the policy file from a prior version to ensure backwards compatibility.

Required Reading

- Administer SELinux

Step 5 – Fully Loaded Policy

At this point, the policy is fully loaded into the kernel. The kernel then maps the initial SIDs to the security contexts of the policy. The kernel is now able to retrieve the security information (contexts) dynamically from the in-kernel security server.

/sbin/init then re-executes itself so it can transition to a different domain if the policy that the kernel has loaded defines it. For a targeted policy, there is no transition defined and /sbin/init remains in the unconfirmed domain.

Once /sbin/init re-executes itself, the kernel will continue with the normal boot process working its way through the contents of /etc/init.d.

At this point, the system is secure according to all of the policies and roles defined in SELinux.

Required Reading

- Enable Disable SELinux

Access Control Framework

Other Access Control Packages

GrSecurity

GRSecurity is still maintained as an external patch set and is not part of the Linux Kernel proper.

GrSecurity sets the kernel stack and program execution space as read-only and provides extensive RBAC and chroot mechanisms. It is very easy to configure and use, though it is less popular than SELinux and AppArmor. For our purposes as Linux Administrators, we want to choose between AppArmor and SELinux.

As of this writing, the author of GrSecurity has concerns about the Linux Security Module framework, so it is highly unlikely that GrSecurity will be included in the production versions of the Linux kernel.

Suggested Reading
- GRSecurity

AppArmor

AppArmor is a security module for the Linux kernel released to open source. AppArmor was maintained by Novel as part of SUSE Linux from 2005 to 2007. The purpose of AppArmor is to associate a security profile with each program that restricts the capabilities of the program.

By default, Linux uses discretionary access (DA) to control system access, but by using the kernel space modules, Apparmor enhances the default DAC's with mandatory access controls.

AppArmor operates first in a learning mode that profiles all the programs on a system. Once the programs are profiled, AppArmor watches for changes or problems that could suggest that the program is not behaving properly.

For many users, AppArmor is considered one of the easiest security frameworks to setup and implement, but since Novel stopped developing the product in 2007, there has not been a consistent flow of upgrades, which is considered risky for a security application.

Required Reading

- SELinux vs AppArmor vs GRSecurity
- AppArmor for Geeks

Lesson Summary

In this lesson, we discussed the various types of Linux Access Control Frameworks. We began with the most simple, the DAC, and then moved to the more enhanced, the MAC (Or mandatory access control) required in most government, healthcare, and educational applications.

We discussed in more detail the various types of MAC's including RBACs and RSBAC's such as SELinux, AppArmor, and GrSecurity. All these products run in conjunction with the kernel at boot time. The goal is that the kernel sends all requests to the Access Control Framework and then only honors the request if it passes the security guidelines.

We spent additional time on SELinux as it is gaining popularity among all Linux distributions and will be found on the majority of the systems as Linux develops.

Additional Resources

- Hardening Linux
- Building SELinux Policies