

[An Introduction to SELinux on CentOS 7](#) >[An Introduction to SELinux on Cent...](#) ▼ Subscribe Share Contents ▼

# An Introduction to SELinux on CentOS 7 – Part 1: Basic Concepts

  
53Posted September 5, 2014  298.6k

SECURITY

CENTOS

By: Sadequl Hussain

## Introduction

Security Enhanced Linux or SELinux is an advanced access control mechanism built into most modern Linux distributions. It was initially developed by the US National Security Agency to protect computer systems from malicious intrusion and tampering. Over time, SELinux was released in the public domain and various distributions have since incorporated it in their code.

Many system administrators find SELinux a somewhat uncharted territory. The topic can seem daunting and at times quite confusing. However, a properly configured SELinux system can greatly reduce security risks, and knowing a bit about it can help you troubleshoot access-related error messages. In this tutorial we will learn about the concepts behind SELinux – its packages, commands, and configuration files – and the error messages it logs when access is denied. We will also see a few practical instances of putting SELinux in action.

### Note

The commands, packages, and files shown in this tutorial were tested on CentOS 7. The concepts remain the same for other distributions.

In this tutorial, we will be running the commands as the root user unless otherwise stated. If you don't have access to the root account and use another account with sudo privileges, you need to

precede the commands with the `sudo` keyword.

## Why SELinux

Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics.



Sign Up

SELinux implements what's known as **MAC** (Mandatory Access Control). This is implemented on top of what's already present in every Linux distribution, the **DAC** (Discretionary Access Control).

To understand DAC, let's first consider how traditional Linux file security works.

In a traditional security model, we have three entities: User, Group, and Other (u,g,o) who can have a combination of Read, Write, and Execute (r,w,x) permissions on a file or directory. If a user `jo` creates a file in their home directory, that user will have read/write access to it, and so will the `jo` group. The "other" entity will possibly have no access to it. In the following code block, we can consider the hypothetical contents of `jo`'s home directory.

You don't need to set up this `jo` user - we'll be setting up plenty of users later in the tutorial.

Running a command like this:

```
ls -l /home/jo/
```

can show output like the following:

```
total 4
-rwxrw-r--. 1 jo jo 41 Aug  6 22:45 myscript.sh
```

Now `jo` can change this access. `jo` can grant (and restrict) access to this file to other users and groups or change the owner of the file. These actions can leave critical files exposed to accounts who don't need this access. `jo` can also restrict to be more secure, but that's discretionary: there's no way for the system administrator to enforce it for every single file in the system.

Consider another case: when a Linux process runs, it may run as the root user or another account with superuser privileges. That means if a black-hat hacker takes control of the application, they can use that application to get access to whatever resource the user account has access to. For processes running as the root user, basically this means everything in the Linux server.

Think about a scenario where you want to restrict users from executing shell scripts from their home directories. This can happen when you have developers working on a production system. You would like them to view log files, but you don't want them to use `su` or `sudo` commands, and you don't want them to run any scripts from their home directories. How do you do that?

Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics. ✕

Sign Up

can define what a user or process can do. It confines every process to its own domain so the process can interact with only certain types of files and other processes from allowed domains. This prevents a hacker from hijacking any process to gain system-wide access.

## Setting Up a Test System

To help us learn the concepts, we will build a test server running both a web and an SFTP server. We will start with a bare installation of CentOS 7 with minimal packages installed and install the Apache and vsftpd daemons on that server. However, we will not configure either of these applications.

We will also create a few test user accounts in our cloud server. We will use these accounts in different places throughout the lesson.

Finally, we will install needed SELinux-related packages. This is to ensure we can work with the latest SELinux commands.

## Installing Apache and SFTP Services

First, let's log in to the server as the **root** user and run the following command to install Apache:

```
yum install httpd
```

The output will show the package being downloaded and ask you for permission to install:

```
Loaded plugins: fastestmirror, langpacks
...
...
=====
Package           Arch             Version          Repository        Size
=====
Installing:
httpd              x86_64           2.4.6-18.el7.centos updates           2.7 M

Transaction Summary
=====
```

Install 1 Package

Total download size: 2.7 M

Installed size: 9.3 M

Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics.



Sign Up

Pressing **y** will install the Apache web server daemon.

Downloading packages:

httpd-2.4.6-18.el7.centos.x86\_64.rpm | 2.7 MB 00:01

Running transaction check

Running transaction test

Transaction test succeeded

Running transaction

Installing : httpd-2.4.6-18.el7.centos.x86\_64 1/1

Verifying : httpd-2.4.6-18.el7.centos.x86\_64 1/1

Installed:

httpd.x86\_64 0:2.4.6-18.el7.centos

Complete!

Start the daemon manually:

```
service httpd start
```

Running the `service httpd status` command will show the service is now running:

```
Redirecting to /bin/systemctl status httpd.service
```

```
httpd.service - The Apache HTTP Server
```

```
Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled)
```

```
Active: active (running) since Tue 2014-08-19 13:39:48 EST; 1min 40s ago
```

```
Main PID: 339 (httpd)
```

```
...
```

```
...
```

Next we will install vsftpd:

```
yum install vsftpd
```

The output should look similar to the following:

Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics. ×

Sign Up

```
...
...
=====
Package                        Arch                Version              Repos
=====
Installing:
vsftpd                        x86_64              3.0.2-9.el7          base

Transaction Summary
=====
Install 1 Package

Total download size: 165 k
Installed size: 343 k
Is this ok [y/d/N]:
```

Press **y** to install the package.

Next, we will use the `service vsftpd start` command to start the vsftpd daemon. The output should show something like the following:

```
Redirecting to /bin/systemctl status vsftpd.service
vsftpd.service - Vsftpd ftp daemon
   Loaded: loaded (/usr/lib/systemd/system/vsftpd.service; disabled)
   Active: active (running) since Tue 2014-08-19 13:48:57 EST; 4s ago
     Process: 599 ExecStart=/usr/sbin/vsftpd /etc/vsftpd/vsftpd.conf (code=exited, status=0/SUCCESS)
    Main PID: 600 (vsftpd)
   ...
   ...
```

## Installing SELinux Packages

A number of packages are used in SELinux. Some are installed by default. Here is a list for Red Hat-based distributions:

- *policycoreutils* (provides utilities for managing SELinux)
- *policycoreutils-python* (provides utilities for managing SELinux)
- *selinux-policy* (provides SELinux reference policy)

Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics.



Sign Up

- *libselinux-utils* (provides some tools for managing SELinux)
- *setroubleshoot-server* (provides tools for deciphering audit log messages)
- *setools* (provides tools for audit log monitoring, querying policy, and file context management)
- *setools-console* (provides tools for audit log monitoring, querying policy, and file context management)
- *mcstrans* (tools to translate different levels to easy-to-understand format)

Some of these are installed already. To check what SELinux packages are installed on your CentOS 7 system, you can run a few commands like the one below (with different search terms after `grep`) as the root user:

```
rpm -qa | grep selinux
```

The output should look something like this:

```
libselinux-utils-2.2.2-6.el7.x86_64  
libselinux-2.2.2-6.el7.x86_64  
selinux-policy-targeted-3.12.1-153.el7.noarch  
selinux-policy-3.12.1-153.el7.noarch  
libselinux-python-2.2.2-6.el7.x86_64
```

You can go ahead and install all the packages with the command below (yum will just update any you already have), or just the ones that you find missing from your system:

```
yum install policycoreutils policycoreutils-python selinux-policy selinux-policy-targeted
```

Now we should have a system that's loaded with all the SELinux packages. We also have Apache and SFTP servers running with their default configurations. We also have four regular user accounts ready for testing in addition to the **root** account.

# SELinux Modes

It's time to start playing around with SELinux, so let's begin with SELinux modes. At any one time, SELinux can be in any of three possible modes:

Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics.



Sign Up

- Permissive
- Disabled

In enforcing mode SELinux will *enforce* its policy on the Linux system and make sure any unauthorized access attempts by users and processes are denied. The access denials are also written to relevant log files. We will talk about SELinux policies and audit logs later.

Permissive mode is like a semi-enabled state. SELinux doesn't apply its policy in permissive mode, so no access is denied. However any policy violation is still logged in the audit logs. It's a great way to test SELinux before enforcing it.

The disabled mode is self-explanatory – the system won't be running with enhanced security.

## Checking SELinux Modes and Status

We can run the `getenforce` command to check the current SELinux mode.

```
getenforce
```

SELinux should currently be disabled, so the output will look like this:

```
Disabled
```

We can also run the `sestatus` command:

```
sestatus
```

When SELinux is disabled the output will show:

```
SELinux status:      disabled
```

# SELinux Configuration File

The main configuration file for SELinux is `/etc/selinux/config`. We can run the following command to view its contents:

Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics. ✕

Sign Up

```
cat /etc/selinux/config
```

The output will look something like this:

```
# 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=disabled
# SELINUXTYPE= can take one of these two values:
#     targeted - Targeted processes are protected,
#     minimum - Modification of targeted policy. Only selected processes are protected
#     mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

There are two directives in this file. The `SELINUX` directive dictates the SELinux mode and it can have three possible values as we discussed before.

The `SELINUXTYPE` directive determines the policy that will be used. The default value is `targeted`. With a targeted policy, SELinux allows you to customize and fine tune access control permissions. The other possible value is "MLS" (multilevel security), an advanced mode of protection. Also with MLS, you need to install an additional package.

## Enabling and Disabling SELinux

Enabling SELinux is fairly simple; but unlike disabling it, should be done in a two-step process. We assume that SELinux is currently disabled, and that you've installed all of the SELinux packages from the earlier section.

As a first step, we need to edit the `/etc/selinux/config` file to change the `SELINUX` directive to permissive mode.

```
vi /etc/sysconfig/selinux
```



...

SELINUX=permissive

Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics.



Enter your email address

Sign Up

Setting the status to **permissive** first is necessary because every file in the system needs to have its context labelled before SELinux can be enforced. Unless all files are properly labelled, processes running in confined domains may fail because they can't access files with the correct contexts. This can cause the boot process to fail or start with errors. We will introduce *contexts* and *domains* later in the tutorial.

Now issue a system reboot:

```
reboot
```

The reboot process will see all the files in the server labelled with an SELinux context. Since the system is running in permissive mode, SELinux errors and access denials will be reported but it won't stop anything.

Log in to your server again as **root**. Next, search for the string "SELinux is preventing" from the contents of the `/var/log/messages` file.

```
cat /var/log/messages | grep "SELinux is preventing"
```

If there are no errors reported, we can safely move to the next step. However, it would still be a good idea to search for text containing "SELinux" in `/var/log/messages` file. In our system, we ran the following command:

```
cat /var/log/messages | grep "SELinux"
```

This showed some error messages related to the GNOME Desktop that was running. This was happening when SELinux was either disabled or in permissive mode:

```
Aug 20 11:31:14 localhost kernel: SELinux: Initializing.
```

```
Aug 20 11:31:16 localhost kernel: SELinux: Disabled at runtime.
```

```
Aug 20 11:31:21 localhost journal: Unable to lookup SELinux process context: Invalid :
```

```
Aug 20 11:33:20 localhost gnome-session: SELinux Troubleshooter: Applet requires SELir
```

```
Aug 20 11:37:15 localhost kernel: SELinux: Initializing.
Aug 20 11:37:17 localhost kernel: SELinux: Disabled at runtime.
Aug 20 11:37:23 localhost journal: Unable to lookup SELinux process context: Invalid a
```

Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics.




Sign Up

requires SELir

```
Aug 20 11:39:44 localhost kernel: SELinux: Disabled at runtime.
Aug 20 11:39:50 localhost journal: Unable to lookup SELinux process context: Invalid a
```

These types of errors are fine.

In the second phase, we need to edit the config file to change the SELINUX directive from **permissive** to **enforcing** in the `/etc/sysconfig/selinux` file:

```
...
SELINUX=enforcing
...
```

Next, reboot the server again.

```
reboot
```

Once the server is back online, we can run the `sestatus` command to check the SELinux status. It should now show more details about the server:

```
SELinux status:                enabled
SELinuxfs mount:              /sys/fs/selinux
SELinux root directory:      /etc/selinux
Loaded policy name:           targeted
Current mode:                 permissive
Mode from config file:       error (Success)
Policy MLS status:           enabled
Policy deny_unknown status:   allowed
Max kernel policy version:    28
```

Check the `/var/log/messages` file:

```
cat /var/log/messages | grep "SELinux"
```

There should be no errors. The output should look something like this:

Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics. ×

Sign Up

```
Aug 20 11:42:09 localhost systemd[1]: Successfully loaded SELinux policy in 183.302ms.
```

```
Aug 20 11:44:25 localhost kernel: SELinux: Initializing.
```

```
Aug 20 11:44:28 localhost systemd[1]: Successfully loaded SELinux policy in 169.039ms.
```

## Checking SELinux Modes and Status (Again)

We can run the `getenforce` command to check the current SELinux mode.

```
getenforce
```

If our system is running in enforcing mode the output will look like this:

```
Enforcing
```

The output will be different if SELinux is disabled:

```
Disabled
```

We can also run the `sestatus` command to get a better picture.

```
sestatus
```

If SELinux isn't disabled, the output will show its current status, its current mode, the mode defined in the configuration file, and the policy type.

```
SELinux status:                enabled
SELinuxfs mount:               /sys/fs/selinux
SELinux root directory:        /etc/selinux
Loaded policy name:             targeted
```

```
Current mode:                enforcing
Mode from config file:       enforcing
Policy MLS status:           enabled
Policy deny_unknown status:  allowed
```

Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics.



Sign Up

When SELinux is disabled the output will show:

```
SELinux status:              disabled
```

We can also temporarily switch between enforcing and permissive modes using the `setenforce` command. (Note that we can't run `setenforce` when SELinux is disabled.)

First change the SELinux mode from enforcing to permissive in our CentOS 7 system:

```
setenforce permissive
```

Running the `sestatus` command now shows the current mode is different from the mode defined in config file:

```
SELinux status:                enabled
SELinuxfs mount:               /sys/fs/selinux
SELinux root directory:        /etc/selinux
Loaded policy name:             targeted
Current mode:                   permissive
Mode from config file:          enforcing
Policy MLS status:              enabled
Policy deny_unknown status:     allowed
Max kernel policy version:      28
```

Switch back to **enforcing**:

```
setenforce enforcing
```

## SELinux Policy

At the heart of SELinux' security engine is its *policy*. A policy is what the name implies: a set of rules that define the security and access rights for everything in the system. And when we say *everything*,

we mean users, roles, processes, and files. The policy defines how each of these entities are related to one another.

## Some Basic Terminology

Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics.



Sign Up

tails later, but

here is a brief introduction. An SELinux policy defines user access to roles, role access to domains, and domain access to types.

### Users

SELinux has a set of pre-built users. Every regular Linux user account is mapped to one or more SELinux users.

In Linux, a user runs a process. This can be as simple as the user **jo** opening a document in the **vi** editor (it will be **jo**'s account running the **vi** process) or a service account running the **httpd** daemon. In the SELinux world, a process (a daemon or a running program) is called a *subject*.

### Roles

A *role* is like a gateway that sits between a user and a process. A role defines which users can access that process. Roles are not like groups, but more like filters: a user may enter or assume a role at any time provided the role grants it. The definition of a role in SELinux policy defines which users have access to that role. It also defines what process domains the role itself has access to. Roles come into play because part of SELinux implements what's known as **Role Based Access Control** (RBAC).

### Subjects and Objects

A *subject* is a process and can potentially affect an *object*.

An *object* in SELinux is anything that can be acted upon. This can be a file, a directory, a port, a tcp socket, the cursor, or perhaps an X server. The actions that a subject can perform on an object are the subject's *permissions*.

### Domains are for Subjects

A *domain* is the context within which an SELinux subject (process) can run. That context is like a wrapper around the subject. It tells the process what it can and can't do. For example, the domain will define what files, directories, links, devices, or ports are accessible to the subject.

### Types are for Objects

A *type* is the context for a file's context that stipulates the file's purpose. For example, the context of a file may dictate that it's a web page, or that the file belongs to the `/etc` directory, or that the file's owner is a specific SELinux user. A file's context is called its *type* in SELinux lingo.

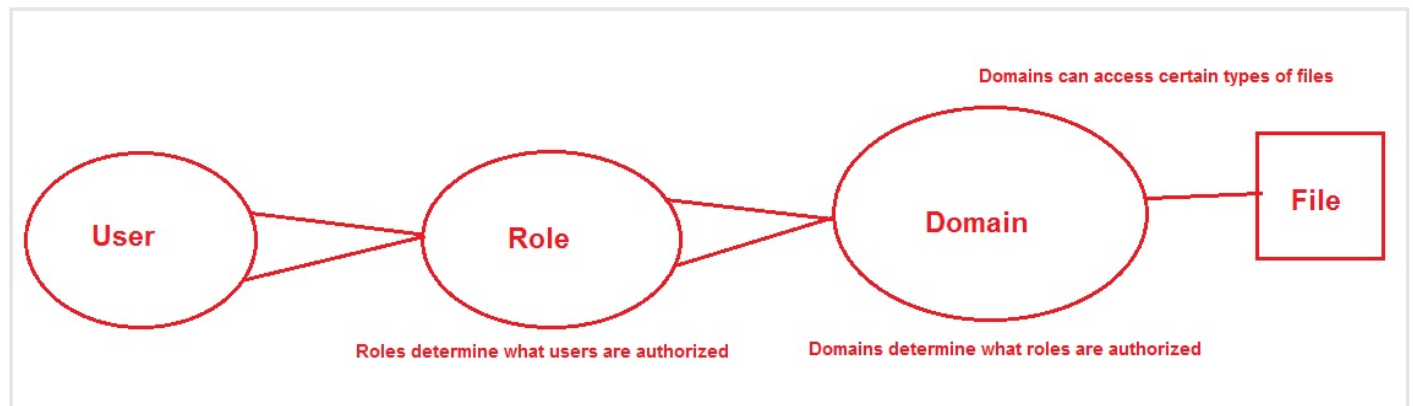
Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics.



Sign Up

SELinux policy defines user access to roles, role access to domains, and domain access to types. First the user has to be authorized to enter a role, and then the role has to be authorized to access the domain. The domain in turn is restricted to access only certain types of files.

The policy itself is a bunch of rules that say that so-and-so users can assume only so-and-so roles, and those roles will be authorized to access only so-and-so domains. The domains in turn can access only so-and-so file types. The following image shows the concept:



Terminology tip: The last bit, where a process running within a particular domain can perform only certain operations on certain types of objects, is called *Type Enforcement (TE)*.

Coming back to the topic of policies, SELinux policy implementations are also typically *targeted* by default. If you remember the SELinux config file that we saw before, the SELINUXTYPE directive is set to be `targeted`. What this means is that, by default, SELinux will restrict only certain processes in the system (i.e. only certain processes are targeted). The ones that are not targeted will run in unconfined domains.

The alternative is a deny-by-default model where every access is denied unless approved by the policy. It would be a very secure implementation, but this also means that developers have to anticipate every single possible permission every single process may need on every single possible object. The default behaviour sees SELinux concerned with only certain processes.

## SELinux Policy Behavior

SELinux policy is not something that replaces traditional DAC security. If a DAC rule prohibits a user access to a file, SELinux policy rules won't be evaluated because the first line of defense has

already blocked access. SELinux security decisions come into play *after* DAC security has been evaluated.

When an SELinux-enabled system starts, the policy is loaded into memory. SELinux policy comes in

Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics.

✕ Kernel modules,

Sign Up

are used by

SELinux keeps track of the modules that have been loaded. The `sestatus` command shows the policy store name. The `semodule -l` command lists the SELinux policy modules currently loaded into memory.

To get a feeling for this, let's run the `semodule` command:

```
semodule -l | less
```

The output will look something like this:

```
abrt      1.2.0
accountsd      1.0.6
acct       1.5.1
afs        1.8.2
aiccu      1.0.2
aide       1.6.1
ajaxterm     1.0.0
alsa       1.11.4
amanda     1.14.2
amtu       1.2.3
anaconda    1.6.1
antivirus    1.0.0
apache     2.4.0
...
...
```

`semodule` can be used for a number other tasks like installing, removing, reloading, upgrading, enabling and disabling SELinux policy modules.

By now you would probably be interested to know where the module files are located. Most modern distributions include binary versions of the modules as part of the SELinux packages. The policy files have a `.pp` extension. For CentOS 7, we can run the following command:

```
ls -l /etc/selinux/targeted/modules/active/modules/
```

The listing shows a number of files with the `.pp` extension. If you look closely, they will relate to different applications:

Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics. ×

Enter your email address

Sign Up

```
-rw-r--r--. 1 root root 10092 Aug 20 11:41 anaconda.pp
-rw-r--r--. 1 root root 11680 Aug 20 11:41 antivirus.pp
-rw-r--r--. 1 root root 24190 Aug 20 11:41 apache.pp
-rw-r--r--. 1 root root 11043 Aug 20 11:41 apcupsd.pp
...
```

The `.pp` files are not human readable though.

The way SELinux modularization works is that when the system boots, policy modules are combined into what's known as the *active policy*. This policy is then loaded into memory. The combined binary version of this loaded policy can be found under the `/etc/selinux/targeted/policy` directory.

```
ls -l /etc/selinux/targeted/policy/
```

will show the active policy.

```
total 3428
-rw-r--r--. 1 root root 3510001 Aug 20 11:41 policy.29
```

## Changing SELinux Boolean Settings

Although you can't read the policy module files, there's a simple way to tweak their settings. That's done through SELinux *booleans*.

To see how it works, let's run the `semanage boolean -l` command.

```
semanage boolean -l | less
```

This shows the different switches that can be turned on or off, what they do, and their current statuses:

```
ftp_home_dir                (off , off) Allow ftp to home dir
```



smartmon_3ware	(off , off)	Allow smartmon to 3ware
mpd_enable_homedirs	(off , off)	Allow mpd to enable homedirs
xdm_sysadm_login	(off , off)	Allow xdm to sysadm login
xen_use_nfs	(off , off)	Allow xen to use nfs

Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics.



tent  
medirs

Sign Up

...

...

We can see the first option allows the FTP daemon to access users' home directories. The setting is turned off at the moment.

To change any of the settings, we can use the `setsebool` command. As an example, let's consider the anonymous FTP write access:

```
getsebool ftpd_anon_write
```

This shows us the switch is off at the moment:

```
ftpd_anon_write --> off
```

Next we change the boolean to enable it:

```
setsebool ftpd_anon_write on
```

Checking the value again should show the change:

```
ftpd_anon_write --> on
```

Changed booleans are not permanent. They revert to their old values after a reboot. To make things permanent, we can use the `-P` switch with the `setsebool` command.

## Conclusion

In the first part of this tutorial we have tried to understand a few basic concepts around SELinux. We have seen how SELinux can secure a system, how we can enable it and what modes it can be

running in. We have also touched on the topic of SELinux policy. Next, we will learn how to use SELinux to restrict access to files and processes.

Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics.



Sign Up

or



Share



Editor:  
Sharon Campbell

## Tutorial Series

### An Introduction to SELinux on CentOS 7

SELinux is a Linux kernel security module that brings heightened security for Linux systems. This series introduces basic SELinux terms and concepts, demonstrating how to enable SELinux, change security settings, check logs, and resolve errors. After completing all three steps, you will have a working CentOS 7 system with SELinux enabled, with four users added with differing degrees of access.

Show Tutorials

## Introducing Projects on DigitalOcean

Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics.



Sign Up

READ MORE

### Related Tutorials

How To Protect Your Server Against the Meltdown and Spectre Vulnerabilities

How To Protect Your Linux Server Against the GHOST Vulnerability

How to Protect Your Server Against the Shellshock Bash Vulnerability


How to Protect Your Server Against the Heartbleed OpenSSL Vulnerability

How to Install TrueCrypt (CLI) on Linux

## 18 Comments

Leave a comment...

Log In to Comment

 [OD](#) September 9, 2014

1 I'd advise against the whole SELinux / grsecurity / MAC / RBAC setup -- unless you've deliberately added people to your system that you don't trust. Here's an email Nick Holland from the OpenBSD

Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics.



Sign Up

an old-style

multi-user system (i.e., lots of people have terminals on their desk, all logging into the Big Computer In Another Room), where most of the users are of very limited access rights, and you need to carefully manage what they are getting to, yes RBAC ("Role Based Access Control") is a great help. And maybe OpenBSD isn't your first choice.

However, OpenBSD systems are often deployed for web services or network services (or single-user systems like desktops). The only people with access to the OpenBSD command prompt are usually either moderately trusted or have administrative rights through sudo anyway. For this, RBAC is just extra baggage, something that's more likely to be exploited than to be useful.

OpenBSD's security model is more about -- as I phrase it -- keeping the bastards out, not controlling them (or hoping to control them) after they are in. Making life difficult for attackers once they get into your system is usually not going to be overly productive, and usually makes administration of the system much more difficult, which often creates NEW security problems of their own. While people like to talk about "Defense in depth" -- and it is not a bad idea -- your best goal is to keep the bastards on the outside of your systems, as once they are in, they can utilize anything you don't have perfectly bolted down to accomplish their goals (and yes, that statement puts me opposite a lot of people making a lot of money chasing down bad guys AFTER they infiltrate systems).

In the Real World: First thing most people do on an SELinux system is disable SELinux. At that point, all the RBAC "features" are now just pure glossy advertising -- worthless. For fear of breaking things, the Linux people have chosen to put a big on-off switch on SELinux...and so given a choice between fixing applications and turning off the switch...people just turn off the switch. ANY claimed benefits of SELinux are ONLY there if it is enabled and used properly.

As for GRSecurity...well, looking at their website, it is still a bunch of patches for Linux to be applied by the user; it still doesn't seem to be incorporated into any mainline Linux distros. I suspect this says far more about the Linux mindset than the merits of GRSecurity (even if the GRSecurity implementation sucked horribly...FIX IT and then incorporate it! Sheesh!)

What's different about OpenBSD is that the features like stack smash protection and W^X are in the base system, on all possible platforms (and a few that didn't seem possible at first!), always on, and there's no easy "off-switch", so crapplications HAVE to be improved in order to work. I can't prove this (and I doubt anyone could), but I suspect that OpenBSD has resulted in more improvements to programs commonly used on Linux than GRSecurity has.

A lot of people like to say "OpenBSD doesn't matter because few uses it", if that's true, then I think it is safe to say that "GRSecurity matters even less". I don't think either statement is fair, but I would like to see the code protection tools of GRSecurity in Redhat and Debian and other major Linux distros and without an SELinux-style off-switch.

Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics.

✕ has poor quality.

Enter your email address

Sign Up

ey were pretty

confident about the code in the base system. Auditing existing code is completely thankless work. You spend a lot of time, make a bunch of changes to the code base ... and no one notices a thing. You get far more praise making features like SELinux or GRSecurity that are not actually USED, and CAN'T be used until someone commits to making the code to be run suck a lot less.

REAL security is not a list of features, even if used.

The OS is just the tip of the security iceberg...or maybe more accurately, the base of it. You don't typically run an OS to run an OS. You run the OS to run applications, and if those applications are poorly written or poorly designed, there are limits to how much (if at all) the OS can help. The best OpenBSD can do is give you a good starting foundation.

Nick.

^ [cokesme](#) September 10, 2014



- 0 You should never advise someone against securing a system. Security should be a concern for any system you run including single-user access. Plus, this post is for instructional purposes. Once you have a general idea of SELinux works, you can automate the setup for policies. Security doesn't have to be a manual process once you have a foundation in the technology you are using. That is true for any system administration.

^ [OD](#) September 10, 2014



- 0 You should never advise someone against securing a system.

I will advise against securing a system if the security measure isn't worth it and/or creates more problems than it fixes.

^ [zegerman](#) December 2, 2014



- 0 You're of course welcome to advise anything you want, but "perimeter defense ONLY" really is terrible advice, and one that very few people should take to heart. Take the simple matter of a safe to store your valuables. There isn't a safe in the world that can withstand an attack of unlimited duration. Every material that it could be made out of can be cut, drilled, or otherwise

compromised. It's only by adding alarms, monitoring, a response force, and other layers of security (security in depth) that the safe is eventually successful -- deterring the theft "long enough" that the attack can be detected and thwarted.

Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics.



Enter your email address

Sign Up

MAC as a matter of regulatory compliance, and currently SELinux is the only way you get there on Linux. If someone chooses to disable SELinux on their personal webserver because it's more trouble than its worth to them, I won't disparage them. If your job depends on securing systems, however, you owe it to your employer to learn tools like SELinux in the same way you learn about monitoring, vulnerability scanning, configuration management, file integrity protection, immutable audit archives, and a bevy of other techniques.

OD December 25, 2014

o Hi! Thanks for your insight. I'd also like to share the latest reply from Nick Holland:

I (hopefully) never said "perimeter defense ONLY". In fact, I said defence in depth is "not a bad idea", and I probably should have made it more positive than that. I did say that you want to stop them at the outer edge if at all possible, and assuming your inner layers will do anything other than slow them down is not wise. Using the safe analogy (which is highly imperfect -- you don't "grind" your way through computer defenses -- they either stop the attacker cold or they don't), you don't want the bastards in your safe with your goods, for when an exit is found, the data moves quickly. The intrusion experts I know talk about "lateral movement" once an intruder is inside your network -- wandering around from system to system finding ANY weak spot, and almost always, there are weaknesses to be found (if nothing else, there's always the users). "Defense in depth" is good. But stopping them at the edge is still your best goal, and not doing so is a failure, and now you are on to damage control.

Unfortunately, I have an answer to the "how could it not?": bad code. Complexity leads to errors, errors lead to exploits. This bit the PaX people badly.

The problem with SELinux is not SELinux, but the "Off" switch, and the fact that most people don't understand how it works, so they flip the off switch. I was talking to someone recently who told me he (almost) never turned off SELinux...but when looking at several of the machines he administered, I found it WAS turned off. From what I've seen, SELinux is barely even "Unix" anymore -- which isn't necessarily bad, but it requires a new level of knowledge and training that just doesn't exist in the real world.

I'll sadly admit I know almost nothing about SELinux operation, and in fact, I appreciate you showing me this article so I can read up on it and learn more about it, and I hope other people do, too. If you toss any of my comments back to the author (I have no interest in registering for Yet Another Site to do so myself), PLEASE include this part too! I certainly wish no disrespect to the author of this article or to anyone who maintains,

documents or uses SELinux. My primary points are that there are many aspects to "security", and that security features that are not used are not really security features, and unfortunately, from everything I've seen and heard, SELinux is underused (I'd even go as far as saying "almost UNUSED", but since RH/CentOS has been shipping with it

Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics.




Sign Up

^ [snger](#) October 4, 2014



0 nice series

^ [mariajosetorres](#) January 5, 2015



0 Hi Sadequl. I hope you're fine. I'm María José de Cuenca-Ecuador. At my job we're interested in a training course about SELinux and advanced Linux security features. Our Institution (Universidad de Cuenca) will afford everything if you accept.

Thanks in advance.

^ [Rizin](#) January 19, 2015



0 [Refer you students to these Official site of Redhat Documentation](#)

^ [SadequlHussain](#) July 12, 2015



0 Hi mariajosetorres, sorry I am answering your post so long time after it has been posted. For some reason the notification from the message boards in DO did not redirect posts to my e-mail for a long time. I am now going through comments and answering them as best as I could. If you are still interested for me to help your students, let me know. However, just letting you know that I am based in Australia :)

^ [Binyamin](#) January 5, 2015



0 SELINUX=enforcing would disable sshd non-22 port access until:  
semanage port -a -t ssh\_port\_t -p tcp 12345

^ [Rizin](#) January 19, 2015



0

I appreciate your work Sadeequl Hussain, However you should mention that these resources is the photocopy of RedhatLinux Knowledgebase [Redhat Documentation](#) so that other users will get benefited from these resources too.

Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics.



Sign Up

- o Hi Rizin, sorry to say, the series is not a "photocopy" of RedHat documentation as you say. It took us quite a few hours and weeks to work out all the examples and test them for accuray and then it went through our editorial process which in itself is a rigorous process in DO. Of course anyone can use the RedHat documentation as well. I would recommend you check your facts first.

---

^ [SELinuxForum](#) February 10, 2015



o Hi,

support my forum on SELinux.

There are many unanswered questions.

SELinux Forum

Appreciate if you can help.

Thank you.

SELinux Forum

---

^ [SELinuxForum](#) February 10, 2015



o Also Sadequl Hussain, would you like to become a Moderator on my forum? Please contact me.

I really need some help here.

---

^ [SadequlHussain](#) July 12, 2015



o Hi SELinuxForum,

Again, sorry for the late reply, as I posted in reply to a message before, I probably should have manually checked the message threads so I could answer in time. Let me know if you still need help with your site's users

---

[LaxSlash](#) June 13, 2015



^ I have an issue here. /etc/selinux/config has it in the enforcing state by default. However, when I run  
0 sestatus, I only get one line of output saying disabled. If I change it to permissive and reboot, same thing. Does not work at all. The symlink is still intact. Help?

Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics.



Sign Up

0 Hi LaxSlash, yes it is quite weird. I would recommend you try to disable it first and then test to see if that's showing disabled. Then enforce it and then check output again. If they don't work, you need to look at the log files.

^ [Darr247](#) March 25, 2017

0 **service httpd start**

doesn't look like a valid CentOS 7 command.



This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.



Copyright © 2018 DigitalOcean™ Inc.

[Community](#) [Tutorials](#) [Questions](#) [Projects](#) [Tags](#) [Newsletter](#) [RSS](#)

[Distros & One-Click Apps](#) [Terms, Privacy, & Copyright](#) [Security](#) [Report a Bug](#) [Write for DOnations](#) [Shop](#)

Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics.



Sign Up