
Fedora 13

SELinux FAQ

Frequently-asked questions about Security Enhanced Linux



Karsten Wade
Paul W. Fields
Scott Radvan

Copyright © 2010 Red Hat, Inc..

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at <http://creativecommons.org/licenses/by-sa/3.0/>. The original authors of this document, and Red Hat, designate the Fedora Project as the "Attribution Party" for purposes of CC-BY-SA. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, JBoss, MetaMatrix, Fedora, the Infinity Logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

For guidelines on the permitted uses of the Fedora trademarks, refer to https://fedoraproject.org/wiki/Legal:Trademark_guidelines.

Linux® is the registered trademark of Linus Torvalds in the United States and other countries.

Java® is a registered trademark of Oracle and/or its affiliates.

XFS® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

All other trademarks are the property of their respective owners.

Abstract

This FAQ answers many questions about Security-Enhanced Linux. The information in this FAQ is valuable for those who are new to SELinux. This FAQ is, however, not designed to provide full coverage of SELinux. For detailed guides and instructions on understanding and using SELinux,

please read the SELinux User Guide and Managing Confined Services Guide first. They are available at <http://docs.fedoraproject.org>

1. SELinux 2

External Resources

- NSA SELinux main website — <http://www.nsa.gov/research/selinux/index.shtml>
- NSA SELinux FAQ — <http://www.nsa.gov/research/selinux/faqs.shtml>
- SELinux project wiki — <http://selinuxproject.org/>
- SELinux Wikipedia page — http://en.wikipedia.org/wiki/Security-Enhanced_Linux
- SELinux community page — <http://selinux.sourceforge.net>
- Unofficial SELinux FAQ — <http://www.crypt.gen.nz/selinux/faq.html>
- Reference Policy — <http://oss.tresys.com/>
- SELinux policy development training course — <https://www.redhat.com/training/security/courses/rhs429.html>
- Getting Started with SE Linux — http://www.linuxtopia.org/online_books/getting_started_with_SELinux/index.html
- List of SELinux object classes and permissions — http://tresys.com/selinux/obj_perms_help.shtml
- Fedora mailing list — <mailto:fedora-selinux-list@redhat.com>; read the archives or subscribe at <https://admin.fedoraproject.org/mailman/listinfo/selinux>
- The SELinux User Guide and the SELinux Managing Confined Services Guide at <http://docs.fedoraproject.org>
- On IRC - [irc.freenode.net](irc://freenode.net), #fedora-selinux and #selinux

1. SELinux

Q: What is SELinux?

A: SELinux (*Security-Enhanced Linux*) in Fedora is an implementation of *mandatory access control* in the Linux kernel using the *Linux Security Modules* (LSM) framework. Standard Linux security is a *discretionary access control* model.

Discretionary access control (DAC)

DAC is standard Linux security, and it provides minimal protection from broken software or malware running as a normal user or root. Users can grant risky levels of access to files they own.

Mandatory access control (MAC)

MAC provides full control over all interactions of software. Administratively defined policy closely controls user and process interactions with the system, and can provide protection from broken software or malware running as any user.

In a DAC model, file and resource decisions are based solely on user identity and ownership of the objects. Each user and program run by that user has complete discretion over the user's objects. Malicious or flawed software can do anything with the files and resources it controls

through the user that started the process. If the user is the super-user or the application is **setuid** or **setgid** to root, the process can have root level control over the entire file system.

A MAC system does not suffer from these problems. First, you can administratively define a security policy over all processes and objects. Second, you control all processes and objects, in the case of SELinux through the kernel. Third, decisions are based on all the security relevant information available, and not just authenticated user identity.

MAC under SELinux allows you to provide granular permissions for all *subjects* (users, programs, processes) and *objects* (files, devices). In practice, think of subjects as processes, and objects as the target of a process operation. You can safely grant a process only the permissions it needs to perform its function, and no more.

The SELinux implementation uses *role-based access control* (RBAC), which provides abstracted user-level control based on roles, and *Type Enforcement*[®] (TE). TE uses a table, or *matrix* to handle access controls, enforcing policy rules based on the types of processes and objects. Process types are called *domains*, and a cross-reference on the matrix of the process's domain and the object's type defines their interaction. This system provides a high level of granularity in a Linux system.

Q: What is SELinux policy?

A: The SELinux policy describes the access permissions for all subjects and objects, that is, the entire system of users, programs, and processes and the files and devices they act upon. Fedora policy is delivered in a package, with an associated source package. Current shipping policy packages are:

selinux-policy-<version>.noarch.rpm

This package is common to all types of policy and contains config files/man pages. This includes the interface files for the development environment. This replaces the -sources package from the past. This package contains the interface files used in Reference Policy along with a Makefile and a small tool called **sepolgen** used to generate a policy template file. The interface files reside in **/usr/share/selinux/devel/include** directory. If you want to see all of the policy files used to build the Reference Policy you need to install the src.rpm.

Q: What happened to the strict policy?

A: Strict policy since Red Hat Enterprise Linux 5 and Fedora Core 5 has been the equivalent of targeted policy with the unconfined domains removed. This means all users had to have a type defined for them like **staff_t** or **user_t**. Also, all processes started by init would need to have policy written for them. As of Fedora Core 9, the strict policy was removed and merged into targeted policy.

Q: What programs are protected by the SELinux policy?

A: The number of programs that have SELinux policy defined for them is constantly changing and evolving. Different versions of policy have more or less executables covered. By convention all confined executables have a label type that ends with **exec_t**. The SELinux tools package (setools) includes the **seinfo** application which allows you to examine the installed policy.

```
# seinfo -t | grep exec_t | wc -l
620
```

Q: What is the Reference Policy?

A: The *Reference Policy* is a project maintained by Tresys Technology (<http://www.tresys.com/>) designed to rewrite the entire SELinux policy in a way that is easier to use and understand. To do this, it uses the concepts of modularity, abstraction, and well-defined interfaces. Refer to <http://oss.tresys.com/> for more information on the Reference Policy.

Note that Reference Policy is not a new type of policy. Rather, it is a new base that policies can be built from.

Q: What are file contexts?

A: *File contexts* are used by the **setfiles** command to generate persistent labels which describe the security context for a file or directory.

Fedora ships with the **fixfiles** script, which supports four options: **check**, **relabel**, **relabel** and **verify**. This script allows users to relabel the file system without having the **selinux-policy-targeted-sources** package installed. The command line usage is more friendly than the standard **setfiles** command.

Q: How do I view the security context of a file, user, or process?

A: The new option **-Z** is the short method for displaying the context of a subject or object:

```
ls -Z file.foo
id -Z
ps -eZ
```

Q: What is the difference between a *domain* and a *type*?

A: There is no difference between a domain and a type, although domain is sometimes used to refer to the type of a process. The use of domain in this way stems from Domain and Type Enforcement (DTE) models, where domains and types are separate.

Q: What are policy modules?

A: SELinux policies are modular, meaning making a change does not require getting entire policy source, modifying it, compiling it, and replacing the current policy with it. This means that third party developers can ship policy modules with their applications, and then they can be added to the policy without having to switch out the entire policy. The new module is then added to the module store, which results in a new policy binary that is a combination of the previous policy and the new module.

This actually works by separating out compile and link steps in the policy build procedure. Policy modules are compiled from source, and linked when installed into the module store (see [Managed Policy](#)). This linked policy is then loaded into the kernel for enforcement.

The primary command for dealing with modules is **semodule**, which lets you perform basic functions such as installing, upgrading, or removing modules. Other useful commands include **checkmodule**, which is the module compiler and is installed with the `checkpolicy` rpm, as well as **semodule_package**, which creates a policy package file (.pp) from a compiled policy module.

Modules are usually stored as policy package file (.pp extension) in `/usr/share/selinux/policyname/`. There you should at least find the `base.pp`, which is the base module.

To see how to write a simple policy module, check out [Local Policy Customizations](#).

Q: What is managed policy?

A: A library, **libsemanage**, exists to provide userspace tools an interface to making policy management easier. All policy management should use this library to access the policy store. The policy store holds all the policy information, and is found at `/etc/selinux/policyname/modules/`.

You should never have to edit the store directly. Instead, you should use tools that link against `libsemanage`. One example tool is **semanage**, which is a command line tool for managing much of the policy such as SELinux user mappings, SELinux port mappings, and file contexts entries. Other examples of tools that use `libsemanage` include **semodule** which uses it to manage the SELinux policy modules installed to the policy store and **setsebool** which uses it manage SELinux policy booleans. Additionally, graphical tools are currently being developed to utilize the functionality provided by `libsemanage`.

1.2. Controlling SELinux

Q: How do I install/not install SELinux?

A: The installer follows the choice you make in the **Firewall Configuration** screen. The default running policy is the targeted policy, and it is on by default.

Q: As an administrator, what do I need to do to configure SELinux for my system?

A: The answer might be nothing! There are many Fedora users that don't even realize that they are using SELinux. SELinux provides protection for their systems with an out-of-the-box configuration. That said, there are a couple of things an administrator might want to do to configure their system. These include:

booleans

Booleans are runtime settings that can be toggled to alter SELinux policy behavior without having to write new policy. There are many Booleans that can be set in Fedora, and they allow an administrator to configure SELinux in very fine detail depending on requirements. To view the available booleans and modify their settings, use **system-config-selinux** or the command line tools **getsebool** and **setsebool**.

setting customizable file contexts

Files on an SELinux system have a security context which is stored in the extended attribute of the file (behavior can vary from filesystem to filesystem, but this is how ext3 works). These are set by **rpm** automatically, but sometimes a user might want to set a particular context on a file. An example would be setting the context on a **public_html** directory so that **apache** can access it, as illustrated in [How do I make a user public_html directory work under SELinux](#).

For a list of types that you might want to assign to files, see `/etc/selinux/targeted/contexts/customizable_types`. These are types commonly assigned to files by users and administrators. To set these, use the **chcon** command. Note that the types in **customizable_types** are also preserved after a relabel, so relabeling the system will not undo this.

making badly behaving libraries work

There are many libraries around that behave badly and try to break the memory protections SELinux provides. These libraries should really be fixed, so please file a bug with the library maintainer. That said, they can be made to work. More information and solutions to make the libraries work can be found in [I have a process running as unconfined_t, and SELinux is still preventing my application from running](#).

Q: How do I enable/disable SELinux protection on specific daemons under the targeted policy?

A: Use **system-config-selinux**, also known as the **SELinux Administration** graphical tool, to control the Boolean values of specific daemons. For example, if you need to disable SELinux for Apache to run correctly in your environment, you can disable the value in **system-config-selinux**. This change disables the transition to the policy defined in **apache.te**, allowing **httpd** to remain under regular Linux DAC security.

The **getsebool** and **setsebool** commands can also be used, including on systems that do not have the **system-config-selinux** tool. Please refer to the manual pages for these commands: **getsebool(8)** and **setsebool(8)** for further details on their operation.

Q: In the past I have written local.te file in policy sources for my own local customization to policy, how do I do this now?

A: Since Fedora Core 5, a modular policy is used, so you don't have to have the complete policy source any more. Now, you can just create a local policy module for your local policy customizations. To do this, follow these steps.

1. Create a temporary directory, and change into it.

```
$ mkdir foo
$ cd foo
```

2. Create empty te, if, and fc files.

```
$ touch local.te local.if local.fc
```

3. Edit the local.te file, adding appropriate content. For example:

```
policy_module(local, 1.0)

require {
    attribute httpdcontent;
    type smbd_t;
}

allow smbd_t httpdcontent:dir create_dir_perms;
allow smbd_t httpdcontent:{ file lnk_file } create_file_perms;
```

There are 3 parts to this file.

- The **policy_module** call inserts statements to make the module work, including declaring the module and requiring system roles, classes, and permissions. Make sure the name declared here (local in this case) matches the name you gave the file (local.te).

- The **require** block lists the symbols that this module uses that must be declared in other modules. In this case, we require the attribute **httpdcontent** and the type **smbd_t**. Note that all types and attributes you use in rules must be required here unless you are declaring them yourself below.
 - The rest of the file is the policy, in this case consisting only of a couple of allow rules. You could also place type declarations, dontaudit statements, interface calls, or most things that can go in a normal te file here.
4. Build the policy module.

```
$ make -f /usr/share/selinux/devel/Makefile
Compiling targeted local module
/usr/bin/checkmodule: loading policy configuration from tmp/local.tmp
/usr/bin/checkmodule: policy configuration loaded
/usr/bin/checkmodule: writing binary representation (version 5) to tmp/local.mod
Creating targeted local.pp policy package
rm tmp/local.mod.fc tmp/local.mod
```

Note that this uses **checkmodule**, which is part of the checkpolicy rpm. So, make sure you install this rpm before doing this.

5. Become root, and install the policy module with **semodule**.

```
$ su -
Password:
# semodule -i local.pp
```

Module are uniquely identified by name

This means that if you later insert another **local.pp**, it will replace the one you just loaded. So, you should keep this **local.te** around, and just add to it if you need to make later policy customizations. If you lose it, but want to keep your previous policy around, just call the new local policy module something else (say local2.te).

Q: I have some avc denials that I would like to allow, how do I do this?

A: If you have specific AVC messages you can use **audit2allow** to generate a Type Enforcement file that is ready to load as a policy module.

```
audit2allow -M local < /tmp/avcs
```

This creates a **local.pp** which you can then load into the kernel using **semodule -i local.pp**. You can also edit the **local.te** to make additional customizations. To create a module allowing all the denials since the last reboot that you can then customize, execute the following:

```
audit2allow -m local -l -i /var/log/messages > local.te
```

Note that the above assumes you are not using the audit daemon. If you were using the audit daemon, then you should use `/var/log/audit/audit.log` instead of `/var/log/messages` as your log file. This generates a `local.te` file, that looks similar to the following:

```
module local 1.0;

require {
    class file { append execute execute_no_trans getattr ioctl read write };
    type httpd_t;
    type httpd_w3c_script_exec_t;
};

allow httpd_t httpd_w3c_script_exec_t:file { execute execute_no_trans getattr ioctl
read };
```

You can hand edit this file, removing allow statements that you don't want to allow, and then recompile and reload it using

- **checkmodule -M -m -o local.mod local.te** to compile the te file. Note that **checkmodule** is part of the checkpolicy rpm, so you need to have it installed.
- **semodule_package -o local.pp -m local.mod** to create a policy package.
- **semodule -i local.pp** to add it to the current machine's running policy. This installs a new module called local with these rules into the module store.

Important

In order to load this newly created policy package into the kernel, you are required to execute **semodule -i local.pp**

Note that if you later install another module called local, it will replace this module. If you want to keep these rules around, then you either need to append future customizations to this local.te, or give future customizations a different name.

Q: How can I help write policy?

A: Your help is definitely appreciated.

- You can start by joining the Fedora SELinux mailing list. You can subscribe and read the archives at <https://admin.fedoraproject.org/mailman/listinfo/selinux>.
- The Unofficial FAQ has some generic policy writing HOWTO information. Refer to <http://www.crypt.gen.nz/selinux/faq.html> for more information.
- Another resource is the Writing SE Linux policy HOWTO, located online at <http://www.lurking-grue.org/writingselinuxpolicyHOWTO.html>.

Also, since the Fedora policy is based on the [Reference Policy](#), you should look at the documentation on its project page. Another excellent source of information is the example policy files in `/usr/share/selinux/devel`.

If you want to create a new policy domain, you can look at the interface files in the `/usr/share/selinux/devel` sub-directories.

Making things easier with `sepolgen`

The tool `sepolgen` is an easy way to create SELinux policy. The following procedure is an example on how to use `sepolgen` to create the required policy for a daemon called `mydaemon`:

```
sepolgen /usr/sbin/mydaemon
```

`sepolgen` then performs the following:

1. Scans for appropriate templates in paths like `/var/lib`, `/var/run`, `/etc/init.d/rc.d/mydaemon`:

```
rpm -qlf /usr/sbin/mydaemon
```

2. Scans for `syslog`, `setuid`, `setgid`, etc. and adds the appropriate access:

```
nm -D /usr/sbin/mydaemon
```

Four files are then generated:

```
mydaemon.te - Contains all types and allow rules discovered for this daemon.  
mydaemon.if - Contains interfaces to be used with the types generated for this daemon.  
mydaemon.fc - Contains file context mapping between types and paths on disk.  
mydaemon.sh - Is a helper shell script used to compile/install policy and label the paths correctly.
```

The policy writer then only needs to execute `mydaemon.sh` and the policy will be compiled and installed - the daemon will then be ready to start testing.

The following procedure may help you to understand the testing process:

```
begin:  
  service mydaemon start  
  run tests against mydaemon  
  check for AVC messages  
  if None  
    Break;  
  audit2allow -R >> mydaemon.te  
  Verify the policy is good or fix it.  
  ./mydaemon.sh  
  goto begin
```

Q: How do I switch the policy I am currently using?

A:

**Use caution when switching policy**

Other than trying out a new policy on a test machine for research purposes, you should seriously consider your situation before switching to a different policy on a production system. The act of switching is straightforward. This method is fairly safe, but you should try it first on a test system.

To use the automated method, run the **Security Level Configuration** tool. From the GUI Main Menu, select **Desktop** → **System Settings** → **Security level**, or from a terminal, run **system-config-selinux**. Change the policy as desired and ensure that the **Relabel on next reboot** option is enabled.

You can also perform these steps manually with the following procedure:

1. Edit **/etc/selinux/config** and change the type and the mode of policy:

```
SELINUXTYPE=policyname SELINUX=permissive
```

This step ensures are not locked out after rebooting. SELinux runs under the correct policy, but does allow you to login if there is a problem such as incorrect file context labeling.

2. Set the system to relabel the file system on reboot:

```
touch /.autorelabel
```

3. Reboot the system. A clean restart under the new policy allows all system processes to be started in the proper context, and reveals any problems in the policy change.
4. Confirm your changes took effect with the following command:

```
sestatus -v
```

With the new system running in **permissive** mode, check **/var/log/messages** for **avc: denied** messages. These may indicate a problem that needs to be solved for the system to run without trouble under the new policy.

5. When you are satisfied that the system runs stable under the new policy, enable enforcing by changing **SELINUX=enforcing**. You can either reboot or run **setenforce 1** to turn enforcing on in real time.

Q: How can I back up files from an SELinux file system?

A: You can now use the **tar** command as normal, you no longer need to use **star**. Also, the [Bacula](#)¹ program supports the xattr extensions when using SELinux, and can properly back up SELinux file systems.

Q: How do I make a user **public_html** directory work under SELinux?

A: This process presumes that you have enabled user public HTML directories in your Apache configuration file, `/etc/httpd/conf/httpd.conf`. This process only covers serving static Web content. For more information about Apache; and SELinux, refer to the SELinux Managing Confined Services Guide at <http://docs.fedoraproject.org>.

1. If you do not already have a `~/public_html` directory, create it and populate it with the files and folders to be served.

```
cd ~
mkdir public_html
cp /path/to/content ~/public_html
```

2. At this point, `httpd` is configured to serve the contents, but you still receive a **403 forbidden** error. This is because `httpd` is not allowed to read the security type for the directory and files as they are created in the user's home directory. Change the security context of the folder and its contents recursively using the `-R` option:

```
ls -Z -d public_html/
drwxrwxr-x auser auser user_u:object_r:user_home_t public_html
chcon -R -t httpd_user_content_t public_html/ ls -Z -d public_html/
drwxrwxr-x auser auser user_u:object_r:httpd_user_content_t public_html/
ls -Z public_html/
-rw-rw-r-- auser auser user_u:object_r:httpd_user_content_t bar.html
-rw-rw-r-- auser auser user_u:object_r:httpd_user_content_t baz.html
-rw-rw-r-- auser auser user_u:object_r:httpd_user_content_t foo.html
```

You may notice at a later date that the user field, set here to `user_u`, is changed to `system_u`. This does not affect how the targeted policy works. The field that matters is the type field.

3. Your static webpages should now be served correctly. If you continue to have errors, ensure that the Boolean which enables user home directories is enabled. You can set it using `system-config-selinux`. Select the **SELinux** tab, and then select the **Modify SELinux Policy** area. Select **Allow HTTPD to read home directories**. The changes take effect immediately.

Q: How do I turn SELinux off at boot?

A: Set `SELINUX=disabled` in `/etc/selinux/config`.

Alternatively, you can add `selinux=0` to your kernel boot parameters. However, this option is not recommended.



Be careful when disabling SELinux

If you boot with `selinux=0`, any files you create while SELinux is disabled do not have SELinux context information. The file system is marked for relabeling at the next boot. If an unforeseen problem prevents you from rebooting normally, you may need to boot in single-user mode for recovery.

Q: How do I turn enforcing on/off at boot?

A: You can specify the SELinux mode using the configuration file `/etc/sysconfig/selinux`.

```
# 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 these two values:
#     targeted - Only targeted network daemons are protected.
#     mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

Setting the value to **enforcing** is the same as adding **enforcing=1** to the kernel boot parameters. Setting the value to **permissive** is the same as adding **enforcing=0** to the kernel boot parameters.

However, setting the value to **disabled** is not the same as the **selinux=0** kernel boot parameter. Rather than fully disabling SELinux in the kernel, the **disabled** setting instead turns enforcing off and skips loading a policy.



SELinux Configuration Precedence

The command line kernel parameter overrides the configuration file.

Q: How do I temporarily turn off enforcing mode without having to reboot?

A: Occasionally you may need to perform an action that is normally prevented by policy. Run the command **setenforce 0** to turn off enforcing mode in real time. When you are finished, run **setenforce 1** to turn enforcing back on.

Q: How do I turn system call auditing on/off at boot?

A: Add **audit=1** to your kernel command line to turn system call auditing on. Add **audit=0** to your kernel command line to turn system call auditing off.

System-call auditing is *on* by default. When on, it provides information about the system call that was executing when SELinux generated a **denied** message. The error message is helpful when debugging policy.

Q: How do I temporarily turn off system-call auditing without having to reboot?

A: Run **auditctl -e 0**. Note that this command does not affect auditing of SELinux AVC denials.

Q: How do I get status info about my SELinux installation?

A: As root, execute the command `/usr/sbin/sestatus -v`. For more information, refer to the **sestatus(8)** manual page.

Q: How do I write policy to allow a domain to use pam_unix.so?

- A:** Very few domains in the SELinux world are allowed to read the `/etc/shadow` file. There are constraint rules that prevent policy writers from writing code like

```
allow mydomain_t shadow_t:file read;
```

In RHEL4 you can setup your domain to use the `unix_chkpwd` command. The easiest way is to use the `unix_chkpwd` attribute. So if you were writing policy for an ftpd daemon you would write something like

```
daemon_domain(vsftpd, `auth_chkpwd`)
```

This would create a context where `vsftpd_t` -> `chkpwd_exec_t` -> `system_chkpwd_t` which can read `/etc/shadow`, while `vsftpd_t` is not able to read it.

In Fedora and Red Hat Enterprise Linux 5 and above, add the rule

```
auth_domtrans_chk_passwd(vsftpd_t)
```

- Q:** I created a new Policy Package, where do I put it to make sure that it gets loaded into the kernel?

- A:** You need to execute the command `semodule -i myapp.pp`. This modifies the policy that is stored on the machine. Your policy module now is loaded with the rest of the policy. You can even remove the pp file from the system.

`semodule -l` lists the currently loaded modules.

```
#semodule -i myapp 1.2.1
```

If you later would like to remove the policy package, you can execute `semodule -r myapp`.

1.3. Resolving Problems

- Q:** Where are SELinux AVC messages (denial logs, etc.) stored?

- A:** Messages can be found in `/var/log/messages` unless you choose to install and enable the `audit` daemon, in which case AVC messages will be in `/var/log/audit/audit.log`.

- Q:** My application isn't working as expected and I am seeing `avc: denied` messages. How do I fix this?

- A:** This message means that the current SELinux policy is not allowing the application to do something. There are a number of reasons this could happen.

First, one of the files the application is trying to access could be mislabeled. If the AVC message refers to a specific file, inspect its current label with `ls -alZ /path/to/file`. If it seems wrong, use the command `restorecon -v /path/to/file` to restore the file's default context. If you have a large number of denials related to files, you may want to use `fixfiles relabel`, or run `restorecon -R /path` to recursively relabel a directory path.

Denials are sometimes due to a configuration change in the program that triggered the denial message. For example, if you change Apache to also listen on port 8800, you must also change the security policy, **apache.te**.

If you are having trouble getting a specific application like Apache to work, refer to [How to use system-config-selinux](#) for information on disabling enforcement just for that application.

Q: Why doesn't SELinux give me the full path in an error message?

A: To answer this, let's first examine the structure of a typical SELinux AVC error message. Items of note in this example are in **bold**:

```
node=host.example.com type=AVC msg=audit(12/13/2006 11:28:14.395:952) : avc:
denied { getattr } for pid=7236 comm=vsftpd name=public_html dev=dm-0 ino=9601649
scontext=system_u:system_r:ftpd_t:s0 tcontext=system_u:object_r:httpd_sys_content_t:s0
tclass=dir

node=host.example.com type=SYSCALL msg=audit(12/13/2006 11:28:14.395:952) : arch=i386
syscall=lstat64 success=no exit=0
a0=8495230 a1=849c830 a2=874ff4 a3=328d28 items=0 ppid=7234 pid=7236 auid=dwalsh
uid=dwalsh gid=dwalsh euid=dwalsh suid=dwalsh fsuid=dwalsh egid=dwalsh
sgid=dwalsh fsgid=dwalsh tty=(none) comm=vsftpd exe=/usr/sbin/vsftpd
subj=system_u:system_r:ftpd_t:s0 key=(null)
```

This AVC message consists of two records, the actual **AVC** record and the **SYSCALL** record. The kernel generates both of these records when the SELinux system denies access. This AVC message indicates that SELinux prevented **/usr/sbin/vsftpd** from performing the **getattr** access on a **dir** named **public_html**.

But which particular public_html directory has the problem?

The AVC records themselves do not show the full path to the problematic **public_html** directory in question. This is done because of performance reasons. Since AVC messages should occur infrequently, having the kernel report the full path is not considered to be worth the overhead. The AVC records do however include the device (**dm-0**), and the inode in question (**9601649**). You can use this device and inode information to find the correct path if the file and/or directory still exists.

The slow way:

```
# find / -inum 9601649
```

A better way:

The *setroubleshoot* tools can use the **locate** command to attempt to re-assemble the path in question:

```
# locate -r /public_html$

/home/dwalsh/public_html
/home/obama/public_html
...(continues)
```

setroubleshoot then uses **stat** to get the inode of each file returned and compares it to the values in the AVC message; if they match, it analyzes the full path. Of course, you need to have the *mlocate* package installed and running via *cr* on to gather these paths.

Refer to <http://danwalsh.livejournal.com/34903.html> for further details on this topic.

Q: I installed on a system with a pre-existing **/home** partition, and now I can't log in. What do I do?

A: Your **/home** partition is not labeled correctly. You can easily fix this two different ways.

If you just want to relabel **/home** recursively:

```
/sbin/restorecon -v -R /home
```

If you want to be sure there are no other files incorrectly labeled, you can relabel the entire file system:

```
/sbin/fixfiles relabel
```

You must have the **polycoreutils** package installed to use **fixfiles**.

Q: After relabeling my **/home** using **setfiles** or **fixfiles**, am I still be able to read **/home** with a non-SELinux-enabled system?

A: You can read the files from a non-SELinux distribution, or one with SELinux disabled. However, files created by a system not using SELinux systems do not have a security context, nor do any files you remove and recreate. This could be a challenge with files such as **~/ .bashrc**. You may have to relabel **/home** when you reboot the SELinux enabled Fedora system.

Q: How do I share directories using NFS between Fedora and non-SELinux systems?

A: Just as NFS transparently supports many file system types, it can be used to share directories between SELinux and non-SELinux systems.

When you mount a non-SELinux file system via NFS, by default SELinux treats all the files in the share as having a context of **nfs_t**. You can override the default context by setting it manually, using the **context=** option. The following command makes the files in the NFS mounted directory appear to have a context of **system_u:object_r:tmp_t** to SELinux:

```
mount -t nfs -o context=system_u:object_r:tmp_t server:/shared/foo /mnt/foo
```

When SELinux exports a file system via NFS, newly created files have the context of the directory they were created in. In other words, the presence of SELinux on the remote mounting system has no effect on the local security contexts.

Q: How can I create a new Linux user account with the user's home?

A: You can create your new user with the standard **useradd** command. First you must become root.

For the targeted policy:

```
su - root
id -Z
root:system_r:unconfined_t
useradd auser
ls -Z /home
drwx----- auser auser root:object_r:user_home_dir_t /home/auser
```

The initial context for a new user directory has an identity of **root**. Subsequent relabeling of the file system changes the identity to **system_u**. These are functionally the same since the role and type are identical (**object_r:user_home_dir_t**.)

Q: Does the **su** command change my SELinux identity and role?

A: The **su** command performs full domain transitions and changes your role. This is easier than using the **newrole** command as **newrole** requires you to enter two passwords - one to identify as the user, and another to identify as root.

Other forms of Linux/UNIX® identity change, for example **setuid(2)**, do not cause an SELinux identity change.

Q: I'm having troubles with **avc** errors filling my logs for a particular program. How do I choose not to audit the access for it?

A: If you wanted to not audit **dmesg**, for example, you would put this in your **dmesg.te** file:

```
dontaudit dmesg_t userdomain:fd { use };
```

This eliminates the error output to the terminal for all user domains, including user, staff and sysadm.

Q: Even running in permissive mode, I'm getting a large number of **avc denied** messages.

A: In a non-enforcing mode, you should actually receive *more* messages than in enforcing mode. The kernel logs each access denial as if you were in an enforcing mode. Since you are not restricted by policy enforcement, you can perform more actions, which results in more denials being logged.

If an application running under an enforcing mode is denied access to read a number of files in a directory, it is stopped once at the beginning of the action. In a non-enforcing mode, the application is not stopped from traversing the directory tree, and generates a denial message for each file read in the directory.

Q: I get a specific permission denial only when SELinux is in enforcing mode, but I don't see any audit messages in **/var/log/messages** (or **/var/log/audit/audit.log** if using the audit daemon). How can I identify the cause of these silent denials?

A: The most common reason for a silent denial is when the policy contains an explicit **dontaudit** rule to suppress audit messages. The **dontaudit** rule is often used this way when a benign denial is filling the audit logs.

To look for your particular denial, enable auditing of all **dontaudit** rules:

```
semodule -b /usr/share/selinux/targeted/enableaudit.pp
```



Enabled dontaudit output is verbose

Enabling auditing of all **dontaudit** rules likely produce a large amount of audit information, most of which is irrelevant to your denial.

Use this technique only if you are specifically looking for an audit message for a denial that seems to occur silently. You want to re-enable **dontaudit** rules as soon as possible.

Once you have found your problem you can reset to the default mode by executing

```
semodule -b /usr/share/selinux/targeted/base.pp
```

Q: Why do I not see the output when I run certain daemons in debug or interactive mode?

A: SELinux intentionally disables access to the tty devices to stop daemons from communicating back with the controlling terminal. This communication is a potential security hole because such daemons could insert commands into the controlling terminal. A broken or compromised program could use this hole to cause serious problems.

There are a few ways you can capture standard output from daemons. One method is to pipe the output to the cat command.

```
snmpd -v | cat
```

When debugging a daemon, you may want to turn off the transition of the daemon to its specific domain. You can do this using **system-config-selinux** or **setsebool** on the command line.

A final option is to turn off enforcing mode while debugging. Issue the command **setenforce 0** to turn off enforcing mode, and use the command **setenforce 1** to re-enable SELinux when you are finished debugging.

Q: When I do an upgrade of the policy package (for example, using **yum**), what happens with the policy? Is it updated automatically?

A: Policy reloads itself when the package is updated. This behavior replaces the manual **make load**.

In certain situations, you may need to relabel the file system. This might occur as part of an SELinux bug fix where file contexts become invalid, or when the policy update makes changes to the file **/etc/selinux/targeted/contexts/files/file_contexts**.

After the file system is relabeled, a **reboot** is not required, but is useful in ensuring every process and program is running in the proper domain. This is highly dependent on the changes in the updated policy.

To relabel, you have several options. You may use the **fixfiles** command:

```
fixfiles relabel && reboot
```

Alternately, use the **/.autorelabel** mechanism:

```
touch /.autorelabel && reboot
```

Q: If the policy shipping with an application package changes in a way that requires relabeling, will RPM handle relabeling the files owned by the package?

A: Yes. The security contexts for the files owned by the package are stored in the header data for the package. The file contexts are set directly after the **cpio** copy, as the package files are being put on the disk.

Q: Why do binary policies distributed with Fedora, such as **/etc/selinux/<pollicyname>/policy/policy.<version>**, and those I compile myself have different sizes and MD5 checksums?

A: When you install a policy package, pre-compiled binary policy files are put directly into **/etc/selinux**. The different build environments will make target files that have different sizes and MD5 checksums.

Q: Will new policy packages disable my system?

A: There is a possibility that changes in the policy package or in the policy shipping with an application package can cause errors, more denials, or other unknown behaviors. You can discover which package caused the breakage by reverting policy and application packages one at a time. If you don't want to return to the previous package, the older version of the configuration files will be saved with the extension **.rpmsave**. Use the mailing lists, bugzilla, and IRC to help you work through your problem. If you are able, write or fix policy to resolve your problem.

Q: My console is being flooded with messages. How do I turn them off?

A: To regain useful control, turn off kernel messages to the console with this command:

```
dmesg -n 1
```

Q: Can I test the default policy without installing the policy source?

A: You can test SELinux default policy by installing just the **selinux-policy-pollicyname** and **pollicycoreutils** packages. Without the policy source installed, the **fixfiles** command automates the file system relabeling.

The command **fixfiles relabel** is the equivalent of **make relabel**. During the relabeling, it will delete all of the files in **/tmp**, cleaning up files which may have old file context labels.

Other commands are **fixfiles check**, which checks for mislabeled files, and **fixfiles restore**, which fixes the mislabeled files but does not delete the files in **/tmp**. The **fixfiles** command does not take a list of directories as an argument, because it relabels the entire file system. If you need to relabel a specific directory path, use **restorecon**.

Q: Why are some of my KDE applications having trouble under SELinux?

A: KDE executables always appear as **kdeinit**, which limits what can be done with SELinux policy. This is because every KDE application runs in the domain for **kdeinit**.

Problems often arise when installing SELinux because it is not possible to relabel **/tmp** and **/var/tmp**. There is no good method of determining which file should have which context.

The solution is to fully log out of KDE and remove all KDE temporary files:

```
rm -rf /var/tmp/kdecache-<username> rm -rf /var/tmp/<other_kde_files>
```

At your next login, your problem should be fixed.

Q: Why does **SELINUX=disabled** not work for me?

A: Be careful of white space in the file **/etc/sysconfig/selinux**. The code is very sensitive to white space, even trailing space.

Q: I have a process running as **unconfined_t**, and SELinux is still preventing my application from running.

A: We have begun to confine the **unconfined_t** domain somewhat. SELinux restricts certain memory protection operation. Following is a list of those denials, as well as possible reasons and solutions for those denials. For more information on these restrictions, see <http://people.redhat.com/drepper/selinux-mem.html>.

These show up in **/var/log/messages** (or **/var/log/audit/audit.log** if using the audit daemon) as avc denials. These can also show up when running programs with errors like

```
error while loading shared libraries: /usr/lib/libavutil.so.49:
cannot restore segment prot after reloc: Permission denied
```

which indicates that the library is trying to perform a text relocation and failing. Text relocations are bad, but can be allowed via the first hint below. Below are the SELinux memory permissions that are denied, as well as hints at how to address these denials.

execmod

This is usually based on a library label. You can permanently change the context on the library with the following commands

```
# /usr/sbin/semanage fcontext -a -t textrel_shlib_t '/usr/lib/libavutil.so.49.0.0'
# /sbin/restorecon -v /usr/lib/libavutil.so.49.0.0
```

with the particular library at fault in place of **/usr/lib/libavutil.so.49.0.0**. Now your application should be able to run. Please report this as a bug in <http://bugzilla.redhat.com>.

execstack

Attempt to **execstack -c LIBRARY**. Now try your application again. If the application now works, the library was mistakenly marked as requiring **execstack**. Please report this as a bug in <http://bugzilla.redhat.com>.

execmem, execheap

A boolean for each one of these memory check errors have been provided. So if you need to run an application requiring either of these permissions, you can set the boolean `allow_exec*` to fix the problem. For instance if you try to run an application and you get an AVC message containing an **execstack** failure. You can set the Boolean with:

```
setsebool -P allow_execstack=1
```

Q: What do these rpm errors mean?

A:

```
restorecon reset /etc/modprobe.conf context system_u:object_r:etc_runtime_t-
>system_u:object_r:modules_conf_t
restorecon reset /etc/cups/ppd/homehp.ppd context user_u:object_r:cupsd_etc_t-
>system_u:object_r:cupsd_rw_etc_t
```

During the update process, the selinux package runs restorecon on the difference between the previously install policy file `_context` and the newly install policy context. This maintains the correct file context on disk.

```
libsepol.sepol_genbools_array: boolean hidd_disable_trans no longer in policy
```

This indicates that the updated policy has removed the boolean from policy.

Q: I want to run a daemon on a non standard port but SELinux will not allow me. How do get this to work?

A: You can use the **semanage** command to define additional ports. So say you want httpd to be able to listen on port 8082. You could enter the command.

```
semanage port -a -p tcp -t http_port_t 8082
```

Q: I am writing a PHP script that needs to create files and possibly execute them. SELinux policy is preventing this. What should I do?

A: First, you should never allow a system service to execute anything it can write. This gives an attacker the ability to upload malicious code to the server and then execute it, which is something we want to prevent.

If you merely need to allow your script to create (non-executable) files, this is possible. That said, you should avoid having system applications writing to the `/tmp` directory, since users tend to use the `/tmp` directory also. It would be better to create a directory elsewhere which could be owned by the apache process and allow your script to write to it. You should label the directory **httpd_sys_script_rw_t**, which will allow apache to read and write files to that directory. This directory could be located anywhere that apache can get to (even `$HOME/public_html/`).

Q: I am setting up swapping to a file, but I am seeing AVC messages in my log files?

A: You need to identify the swapfile to SELinux by setting its file context to **swapfile_t**.

```
chcon -t swapfile_t SWAPFILE
```

Q: Please explain the **relabelto/relabelfrom** permissions?

A: For files, **relabelfrom** means "Can domain D relabel a file from (i.e. currently in) type T1?" and **relabelto** means "Can domain D relabel a file to type T2?", so both checks are applied upon a file relabeling, where T1 is the original type of the type and T2 is the new type specified by the program.

Useful documents to look at:

- Object class and permission summary by Tresys http://tresys.com/selinux/obj_perms_help.shtml
- Implementing SELinux as an LSM technical report (describes permission checks on a per-hook basis) <http://www.nsa.gov/selinux/papers/module-abs.cfm>. This is also available in the selinux-doc package (and more up-to-date there).
- Integrating Flexible Support for Security Policies into the Linux Operating System - technical report (describes original design and implementation, including summary tables of classes, permissions, and what permission checks are applied to what system calls. It is not entirely up-to-date with current implementation, but a good resource nonetheless). <http://www.nsa.gov/selinux/papers/slinux-abs.cfm>

1.4. Deploying SELinux

Q: What file systems can I use for SELinux?

A: The file system must support **xattr** labels in the right *security.** namespace. In addition to ext2/ext3/ext4, XFS has recently added support for the necessary labels.

Note that XFS SELinux support was broken in the upstream Linux kernel versions 2.6.14 and 2.6.15, but fixed (worked around) in version 2.6.16. Your kernel must be of this version or later if you choose to use XFS with SELinux.

Q: How does SELinux impact system performance?

A: This is a metric that is hard to measure, and is heavily dependent on the tuning and usage of the system running SELinux. When performance was last measured, the impact was around 7% for completely untuned code. Subsequent changes in system components such as networking are likely to have made that worse in some cases. SELinux performance tuning continues to be a priority of the development team.

Q: What types of deployments, applications, and systems should I leverage SELinux in?

A: Initially, SELinux has been used on Internet facing servers that are performing a few specialized functions, where it is critical to keep extremely tight security. Administrators typically strip such a box of all extra software and services, and run a very small, focused set of services. A Web server or mail server is a good example.

In these edge servers, you can lock down the policy very tightly. The smaller number of interactions with other components makes such a lock down easier. A dedicated system running a specialized third-party application would also be a good candidate.

In the future, SELinux will be targeted at all environments. In order to achieve this goal, the community and *independent software vendors* (ISVs) must work with the SELinux developers to produce the necessary policy.

Q: How does SELinux affect third-party applications?

A: One goal of implementing a targeted SELinux policy in Fedora is to allow third-party applications to work without modification. The targeted policy is transparent to those unaddressed applications, and it falls back on standard Linux DAC security. These applications, however, will not be running in an extra-secure manner. You or another provider must write policy to protect these applications with MAC security.

It is impossible to predict how every third-party application might behave with SELinux, even running the targeted policy. You may be able to fix issues that arise by changing the policy. You may find that SELinux exposes previously unknown security issues with your application. You may have to modify the application to work under SELinux.

Note that with the addition of *Policy Modules*, it is now possible for third-party developers to include policy modules with their application. If you are a third-party developer or a package-maintainer, please consider including a policy module in your package. This will allow you to secure the behavior of your application with the power of SELinux for any user installing your package.

One important value that Fedora testers and users bring to the community is extensive testing of third-party applications. With that in mind, please bring your experiences to the appropriate mailing list, such as the fedora-selinux list, for discussion. For more information about that list, refer to <https://admin.fedoraproject.org/mailman/listinfo/selinux>.