

SE Linux Class Overview

Version 2

The purpose of this exercise is to introduce you to using SE Linux features and writing SE Linux policy. We will be using type enforcement, MLS, and MCS policies.

I had hoped to use the new modular reference policy. However, that feature was too bleeding edge for me to get working reliably, so we will stay with the older targeted and strict reference policies.

Reference Documents

Configuring the SELinux Policy, <http://www.nsa.gov/selinux/papers/policy2-abs.cfm> - Up to date reference on the core policy language statements, but it does not address modular policy, MLS, or MCS.

SELinux and MLS Putting the Pieces Together, <http://www.tcs-sec.com/images/SELinuxandMLS.pdf> - Describes the design and implementation of the MLS policy mechanism in SELinux but it does not go into great detail on the commands and files.

Getting Started with Multi-Category Security, <http://james-morris.livejournal.com/8228.html> - Describes the goals of MCS and goes into good detail on how to actually create and use a MCS policy. However, the description assumes the new modular policy framework.

Getting started with SE Linux HOWTO: the new SELinux - <http://www.lurking-grue.org/GettingStartedWithNewSELinuxHOWTO.pdf> - This is a good tutorial to walk you through different aspects of selinux. However, things change so quickly some of the information is out of date or just not applicable to our environment. For example, completely ignore the Installation section 3. The policy paths in Sections 5 and 6 are also wrong, but the file names are accurate.

Writing SE Linux HOWTO - <http://www.lurking-grue.org/writingselinuxpolicyHOWTO.html> - A good description of the layout of the macros and type enforcement files for the monolithic policy.

RedHat SELinux Guide - <http://www.centos.org/docs/4/html/rhel-selg-en-4/>

Enabling SELinux

The /selinux portion of the file system is mapped to the runtime memory of the SELinux system much like the /proc file system maps out controls to the rest of the Linux system. The /selinux/enforce file controls whether the security server really enforces the policy or not. I have the lab systems configured to operate in *permissive* mode. This mean the

security server will be run, but the results will never really restrict access. Instead only the error message will be logged, but the operation will be permitted.

You can directly change the values in the `/selinux/enforce` file to change between permissive and enforcing mode. Or you can use the **getenforce** and **setenforce** commands. If you change to enforcing mode, the system will actively use the results to restrict access. On reboot, the `/selinux/enforce` will be reset to 0. This mode is valuable when developing policy. If you end up with a too restrictive policy, a reboot will return you to a state where you can fix things. So while, you can set the system to be in a persistent enabled mode, please do not do this on the lab machines

User Level View

As a user in an active SE Linux system, you don't (can't) interact with the policy. Someone else has set up the policy rules to associate types, categories, sensitivity labels, and rules with you. Also, some initial security contexts have been associated with all files and many other objects in the system.

Use the **id** command to determine the security context associated with current running process.

While a user can be assigned the capability of multiple role and role can have multiple domains associated with it, a given process at any point in time is associated with exactly one role and one type. The **newrole** command is used to transition between types, domains, and levels.

Many commands have been augmented with a **-Z** argument to show the new SELinux attributes. For example **ls -Z** will show the security contexts associated with each file. **ps -Z** does the same thing with processes.

When you create a file, the type enforcement rules will use your current domain and the type of the enclosing directory to determine the type of the new file. You can change the existing label of a file (assuming you are a privileged user) using the **chcon** command.

Looking at audit messages

SE Linux audit messages are placed in the kernel buffer which you can see from the **dmesg** command or by looking at `/var/log/messages`. Look for the prefix "avc: denied" (two spaces) to find the access denied logs.

Virtual terminals

In particular, the strict policy does not work well with the window system. When working with the enabled strict policy, you will need to work from the 6 virtual terminals provided by linux. From the windows environment type Ctl-Alt-F1 through Ctl-Alt-F6 to reach one of the 6 virtual terminals. Once in the virtual terminals, Alt-F1 through Alt-F6 will switch you between terminals. Alt-F7 will return you to the windowing environment.

Manipulating policies

The policies are installed at `/etc/selinux`.

The targeted policy is installed, and the source for the targeted policy is stored at `/etc/selinux/targeted/src/policy`. Let's make a new version of the targeted policy. First copy the `/etc/selinux/targeted` directory to your personal directory, say `skhpol`:

```
cp -r /etc/selinux/targeted /etc/selinux/skhpol
```

You need to change to values to instruct the system to use the new `skhpol`. First in `/etc/selinux/skhpol/src/policy/Makefile`, change the defined of `TYPE` from `targeted` to `skhpol`. Then in `/etc/selinux/config`, change the `SELINUXTYPE` from `targeted` to `skhpol`.

Now you need to compile and load the new policy. Go to the `/etc/selinux/skhpol/src/policy` directory. Type "make policy". This will compile the policy. If that looks good, then type "make install". This should load the new policy.

If you are loading a significantly different policy, you may need to relabel some or all of the file system. The **fixfiles** command with the **relabel** option should help with this.

There are several main types of files in the policy directory

- context files – These define default contexts for files and processes on system start
- user definition – The users file
- Type enforcement files – Much of the type enforcement logic is implemented in m4 macros. These files are in the macros subdirectory. Other files are in the domains and types directories. These files have `.te` extension
- Policy.conf – This file is created during policy compilation. It is the combined output of all `.te` files after the macro processor has been run. It is useful for debugging, when you don't know why a particular operation is being allowed (or denied).

Make Targets

The makefile in the policy source directory (e.g., `/etc/selinux/skhpolicy/src/policy`) controls the compilation and loading of the policy. You will be using the following targets:

- make policy or the default target: Compiles the policy.
- make install: Links the policy and prepares it to load.
- make load: Loads the policy. Calls install and policy as needed. This is the target that changes the running policy.
- make relabel: Relabels all files in the system to be consistent with the file context prescribed by the policy. Necessary if you are loading a wildly different policy. Commands like **restorecon** do something similar, and these commands let you specify a portion of the file system to relabel rather than relabeling the entire disk.

- `make enableaudit`: Call before compiling the policy. It will remove explicit `dontaudit` commands. I'm having problems getting audit messages to show up in my test logs. Perhaps this call will help.

MLS and MCS

If you want to enable MLS or MCS in your policy, you will need to do the following:

- Set MLS or MCS to Y in the makefile.
- Run “`make mlsconvert`” or “`make mcsconvert`” to set default sensitivity labels (should only be done once for a given directory).
- Then compile and load as normal

MLS and strict policy do not get along. The strict policy does not have the logic to deal with TTY's being accessed from levels other than `s0`.

MLS is also configured in this environment, and you have probably noticed the fourth sensitivity label in the security contexts we have looked at. Clearances of `s0` through `s15` are defined and categories from `c0` through `c255`. The commands let you enable aliases for these values, but when I try to set aliases the policy compile crashes.

By default, everything is set at `s0`. You can use the `-l` argument to `newrole` to adjust the current level of your process to another value within the range. Similarly `chcon` lets the privileged user adjust the sensitivity label on files.

Try adjusting your user level. Create files and directories at different levels and see if the access constraints meet your expectations.

Other tools

Various tools have been developed to help analyze and create SE Linux policies. `Apol` is one of the tools and it is installed on the lab machines. It analyzes the policy and allows the user to