

Cyber Security SE Linux Exercise

Goal

Familiarize yourselves with the SE Linux tools in preparation for upcoming SE Linux Lab assignment. Exercise SELinux type enforcement and MCS policy.

Scenario

Bob Co is interested in SE Linux to provide the same kind of separation you implemented using ACL's in Windows. As a reminder this company has three sets of employees:

- Engineers
- Financial folks
- System Administrators

Each set of folks would like a place on the file system where in general they have full control and other folks have read-only access. But each group of folks would like to have the following subareas with different access:

- A public area that gives all users read and write access.
- A private area that blocks everyone except for folks of the same set.

Basic type enforcement policies should provide adequate separation. MCS may also work. Try both mechanisms to get a feel for their expressibility and ease of use.

Your Environment

Each machine has a Linux VM with the appropriate packages installed. Work in that VM. First explore your environment with the `-Z` arguments for `ls` and `ps` and the `id` command to identify the current security contexts.

Feel free to create and delete users as necessary. Use the “`adduser`” command to create new users. You may want to `ssh` into the VM with the new users to test access. Creating new users and getting the full permission set correct under X windows can be difficult.

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 means 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.

Interesting directory and files

The virtual machines have the targeted modular policy installed. We do not have access to the source for these modules. I have downloaded the `refpolicy` source, but this is not what is installed.

- `/selinux` – The root of the proc file system that controls how the selinux kernel module operates.
- `/etc/selinux` – Root of the policy configuration area
- `/etc/selinux/config` – Identifies the policy installed and the enforcing mode used at boot time.
- `/etc/selinux/setrans.conf` – The definition of user names for built in category names.
- `/etc/selinux/refpolicy/src/policy/policy` – The root of the reference policy module sources.

- /usr/share/selinux – Installed policy packages
- /usr/share/doc/ - Contains one subdirectory about selinux and another subdirectory about the policy

Interesting Commands

- GUI semanage from the “System Tools” menu. Wrapper for the command line semanage tool.
- GUI Policy Generation tool from the “System Tools” menu. Creates the start of a new policy module.
- GUI Policy Analysis tool from the “System Tools” menu. Also called apol.
- new -Z arguments – Many standard commands like id, ls, and ps now have a -Z argument that displays the security context associated with the process or file.
- Semanage – Manages logins, users, ports. See the man page and the Gentoo reference below for more details.
- Semodule – Compiles and loads policy modules. See the man page and the RedHat reference below for more details.
- Chcat – Manage categories. See the man page and the Morris reference below for more details.
- Chcon – Change the types associated with files. See the man page for details.
- Newrole – Take on a new role.

Policy editing

With FC5, modular policy is supported. You can use the Policy Generation GUI to get started on creating your own module to define policy for an application, daemon, or set of users. For each module there are potentially three files (actually four created by the policy generation gui).

- foo.te – This is the main and only required file. It includes the AV statements and any supporting type and role definitions.
- foo.if – The interface file. If your policy requires other modules to access what you define, you will need to create an if file. You should not need to create this file, but you will need to create an empty file to satisfy the standard policy makefile.
- foo.fc – The file context. Describes the base labels of files. You will probably need to create this for the magic8 module policy.
- foo.sh – Generated by the policy gui to invoke the appropriate build tools to compile and load your new module.

The type enforcement files use many M4 macros. Most of the macros are defined in the support directory. You may need to read up on M4 (see references) to understand what existing macros do.

References

- 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. The build description also addresses the old monolithic model.
- RedHat documentation on building, compiling, and loading your modular policy - http://www.redhat.com/docs/manuals/enterprise/RHEL-5-manual/Deployment_Guide-en-US/sec-sel-policy-customizing.html
- Dan Walsh's article on writing a new policy module. <http://www.redhatmagazine.com/2007/08/21/a-step-by-step-guide-to-building-a-new-selinux-policy-module/>
- Gentoo documentation on using semanage and semodule - <http://www.gentoo.org/proj/en/hardened/selinux/selinux-handbook.xml?part=3&chap=4>
- James Morris' description of MCS - <http://james-morris.livejournal.com/8228.html>
- Joshua Brindle's description of how modular policies are implemented <http://securityblog.org/brindle/2006/07/05/selinux-policy-module-primer/>
- Exploiting the M4 Macro Language - <http://www.cs.stir.ac.uk/~kjt/research/pdf/expl-m4.pdf>