# Cyber Security SE Linux Exercise

## Goal

Familiarize yourselves with the SE Linux tools. Exercise SELinux type enforcement to confine a program and perhaps users.

## Your Environment

Each machine has a Linux VM (Fedora Core 12) with the appropriate packages installed. Work in that VM.

### *Some Introductory Exercises*

First, work through a few basic scenarios on the basic targeted policy. These exercises are from the "Fedora 12 Security-Enhanced Linux User Guide" which can be downloaded from http://docs.fedoraproject.org/

Basic file and process security contexts.

1. Log in as alice. Run "ls -l" and "ls -Z". What is the security context of the files in Alice's directory?
2. Run "id -Z" as Alice. What is Alice's security context?

How is password changing constrained?

1. Run "ls -Z /usr/bin/passwd" and "ls -Z /etc/shadow".
2. Run passwd as Alice. Stop at the password prompt. In another shell window run "ps -eZ | grep passwd". What type is the password process running under? What type enforcement rules do you think direct password access?

How is a basic service confined?

1. As root, run "touch /var/www/html/testfile" and "ls -Z /var/www/html/testfile".
2. Start web server, "/etc/init.d/httpd/start".
3. As Alice run "wget http://localhost/testfile". Did it work?
4. As root run "chcon -t samba_share_t /var/www/html/testfile". This should change the type of the file at least until the next reboot.
5. Have Alice run wget again. Did it work?
6. Look at the logs at the end of /var/log/messages.

Create a confined user.

1. By default, users in targeted policy are unconfined. Start "SE Linux Management" GUI from the System>Admin menu. Look at the users and the user mappings.
2. From the shell as root, use "useradd test-guest" to create a new user. "passwd test-guest" to give the new user a password (test).

3. From the SE Linux Management GUI, add a new login mapping entry that maps test-guest to xguest_u.
4. Login as test-guest.
5. Try to access http://illinois.edu three ways: via firefox, via wget, and via telnet with port 80 specified as the third argument. Which, if any approaches worked?
6. Try to invoke "su – alice". Did it work? What is the security context of test-guest and /bin/su?

## Scenario

CosaNostra Pizza is interested in SE Linux to provide the same kind of separation you implemented using ACL's in Windows. They are also interested in using SE Linux to provide isolation for a network service they need to deploy.

As a reminder of their file separation requirements, this company has three sets of employees:

- Deliverators
- 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.

Their network service sits on port 10001 and accepts TCP requests. It needs to be able to read a configuration file (responses.txt) and write a log file (magic8.log).

Implement the following:

1. Build a type enforcement policy module to implement the network service confinement.

2. Build a type enforcement policy and set up users to provide at least two group separation.

### *Policy editing*

With FC5 and beyond, 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 policy/support directory. You may need to read up on M4 (see references) to understand what existing macros do.

## *Confining magic8d*

There should be a magic8.tar.gz file in alice's home directory. Unpack and build this program. There is a magic8d, which is the fortune telling server and magic8, the fortune telling client. Launch magic8d in one window. From another window run "magic8 127.0.0.1 "Am I happy?". It should return with a message chosen from response.txt.

As root move the magic8 subdirectory to /.

We will work to confine the server program, so magic8d can only access the files response.txt and magic8.log directly.

Use the "Selinux Policy Generation Tool" to get a starting point for the new magic8d policy module. This starts you through a wizard

1. The first page has you select the "type of application". Select "User application"
2. On the second page enter "magic8d" as the name. Select /magic8/magic8d as the executable.
3. On the third page entry 10001 for the TCP selected port that magic8d listens on.
4. Leave the fourth page blank.
5. Leave the fifth page blank.
6. Add the file /magic8/response.txt as a file that the application manages.
7. Do not select any booleans.
8. Create a /magic8/magic8-policy directory to generate the policy in.
9. Select apply on the last page and it will create four files in the directory you selected in the last step.
10. Review the generated files. As root invoke the generated shell file. My version had a syntax error I could comment out (line 24 of the .te file).
11. Once the module successfully loads, launch magic8d. Use "ps -eZ" to verify that magic8d is running under the special confined type magic8d.

12. Connect as a client. Look in /var/log/messages. Is there anything interesting? The module is generated to run permissive. Change policy and reload to get rid of errors that look detrimental. Once things look good, comment out the permissive statement in the .te file and reload. Does it work? Are there any other log messages?
13. Use chcon to change the type of the magic8.log file and see how it affects the execution of magic8d when running in enabled mode.

This policy could be tightened up. For example, magic8d only needs to read and not write response.txt. You could try tightening the policy further.

## *Creating Confined Users*

To do a proof of concept of using type enforcement to implement the delivery, financial, and administrative separations that we implemented in the Windows ACL lab, we will create a new SE linux type to represent the delivery users. Again, we will use the Policy Generation Tool to start the process:

1. Select “login user” as the type of application or user
2. Enter “delivery” as the name
3. Skip this step.
4. Skip this step.
5. Select all TCP and all UDP.
6. Skip this step.
7. Hit apply to create the policy.
8. Review the generated files. The .te file creates a new delivery_r role and delivery_u user. It does not create any file types though. You can copy the magic8d_rw_t references from the magic8d policy and use those as the basis to create delivery_rw_t types and related rules.
9. Invoke the generated .sh file to compile and load the module.
10. Create users bob and carol. Use passwd to set their passwords to the standard ones.
11. Use the semanage GUI to map bob to the delivery_u selinux user and map carol to the user_u selinux user.
12. Create /delivery directory. Set the discretionary permissions to read, write, execute for everyone. Use chcon to set the type to “delivery_rw_t”.
13. Use ssh to login as bob. Enter the /delivery directory. Create a file in that directory. What is the type of the new file.
14. Use ssh to login as carol. Enter the /delivery directory. What happened?

This policy is not as refined as the one specified in the Windows ACL exercise. How would you expand this policy to have the policy, private, and read-only areas for each group?

### *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

### *Interesting directory and files*

The virtual machines have the targeted modular policy installed. I think I have the corresponding policy source installed on the virtual machine.

- /selinux – The root of the proc file system that controls how the selinux kernel module operates.
- /etc/selinux/config – Identifies the policy installed and the enforcing mode used at boot time.
- /usr/share/selinux/devel/include - The root of the reference policy module sources.
- /usr/share/selinux – Installed policy packages

## Interesting Commands

- GUI semanage from the “System>Administration” 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.
- Chcon – Change the types associated with files. See the man page for details.
- Newrole – Change the role and/or type that a user is running under.

## References


- Configuring the SELinux Policy, http://www.nsa.gov/research/_files/selinux/papers/policy2-abs.shtml - 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
- 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