A sysadmin's guide to SELinux: 42 answers to the big questions

By Alex Callejas ⋮ 9-11 minutes

Get answers to the big questions about life, the universe, and everything else about Security-Enhanced Linux.

> "It is an important and popular fact that things are not always what they seem…"
>
> —Douglas Adams, *The Hitchhiker's Guide to the Galaxy*

Security. Hardening. Compliance. Policy. The Four Horsemen of the SysAdmin Apocalypse. In addition to our daily tasks—monitoring, backup, implementation, tuning, updating, and so forth—we are also in charge of securing our systems. Even those systems where the third-party provider tells us to disable the enhanced security. It seems like a job for *Mission Impossible*'s Ethan Hunt.

Faced with this dilemma, some sysadmins decide to take the blue pill because they think they will never know the answer to the big question of life, the universe, and everything else. And, as we all know, that answer is **42**.

In the spirit of *The Hitchhiker's Guide to the Galaxy*, here are the 42 answers to the big questions about managing and using SELinux with your systems.

1. SELinux is a LABELING system, which means every process has a LABEL. Every file, directory, and system object has a LABEL. Policy rules control access between labeled processes and labeled objects. The kernel enforces these rules.
2. The two most important concepts are: *Labeling* (files, process, ports, etc.) and *Type enforcement* (which isolates processes from each other based on types).
3. The correct Label format is `user:role:type:level` (*optional*).
4. The purpose of *Multi-Level Security (MLS) enforcement* is to control processes (*domains*) based on the security level of the data they will be using. For example, a secret process cannot read top-secret data.
5. *Multi-Category Security (MCS) enforcement* protects similar processes from each other (like virtual machines, OpenShift gears, SELinux sandboxes, containers, etc.).
6. Kernel parameters for changing SELinux modes at boot:
   - `autorelabel=1` → forces the system to relabel
   - `selinux=0` → kernel doesn't load any part of the SELinux infrastructure
   - `enforcing=0` → boot in permissive mode
7. If you need to relabel the entire system:

   ```
   # touch /.autorelabel
   # reboot
   ```

   If the system labeling contains a large amount of errors, you might need to boot in permissive mode in order for the autorelabel to succeed.
8. To check if SELinux is enabled:

   ```
   # getenforce
   ```
9. To temporarily enable/disable SELinux:

   ```
   # setenforce [1|0]
   ```
10. SELinux status tool:

    ```
    # sestatus
    ```
11. Configuration file:

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

12. How does SELinux work? Here's an example of labeling for an Apache Web Server:
    - Binary: `/usr/sbin/httpd`→`httpd_exec_t`
    - Configuration directory: `/etc/httpd`→`httpd_config_t`
    - Logfile directory: `/var/log/httpd` → `httpd_log_t`
    - Content directory: `/var/www/html` → `httpd_sys_content_t`
    - Startup script: `/usr/lib/systemd/system/httpd.service` → `httpd_unit_file_d`
    - Process: `/usr/sbin/httpd -DFOREGROUND` → `httpd_t`
    - Ports: `80/tcp, 443/tcp` → `httpd_t, http_port_t`

A process running in the `httpd_t` context can interact with an object with the `httpd_something_t` label.

13. Many commands accept the argument `-Z` to view, create, and modify context:
    - `ls -Z`
    - `id -Z`
    - `ps -Z`
    - `netstat -Z`
    - `cp -Z`
    - `mkdir -Z`

Contexts are set when files are created based on their parent directory's context (with a few exceptions). RPMs can set contexts as part of installation.

14. There are four key causes of SELinux errors, which are further explained in items 15-21 below:
    - Labeling problems
    - Something SELinux needs to know
    - A bug in an SELinux policy/app
    - Your information may be compromised

15. *Labeling problem:* If your files in `/srv/myweb` are not labeled correctly, access might be denied. Here are some ways to fix this:
    - If you know the label:

      ```
      # semanage fcontext -a -t httpd_sys_content_t '/srv/myweb(/.*)?'
      ```

    - If you know the file with the equivalent labeling:

      ```
      # semanage fcontext -a -e /srv/myweb /var/www
      ```

    - Restore the context (for both cases):

      ```
      # restorecon -vR /srv/myweb
      ```

16. *Labeling problem:* If you move a file instead of copying it, the file keeps its original context. To fix these issues:

    - Change the context command with the label:

      ```
      $ sudo chcon -t httpd_system_content_t /var/www/html/index.html
      ```

    - Change the context command with the reference label:

      ```
      $ sudo chcon --reference /var/www/html/ /var/www/html/index.html
      ```

    - Restore the context (for both cases):

      ```
      $ sudo restorecon -vR /var/www/html/
      ```

17. If *SELinux needs to know* HTTPD listens on port 8585, tell SELinux:

```
$ sudo semanage port -a -t http_port_t -p tcp 8585
```

18. *SELinux needs to know* booleans allow parts of SELinux policy to be changed at runtime without any knowledge of SELinux policy writing. For example, if you want httpd to send email, enter:

```
$ sudo setsebool -P httpd_can_sendmail 1
```

19. *SELinux needs to know* Booleans are just off/on settings for SELinux:
    - To see all booleans: `# getsebool -a`
    - To see the description of each one: `# semanage boolean -l`
    - To set a boolean execute: `# setsebool [_boolean_] [1|0]`
    - To configure it permanently, add `-P`. For example:

      ```
      # setsebool httpd_enable_ftp_server 1 -P
      ```

20. SELinux policies/apps can have bugs, including:
    - Unusual code paths
    - Configurations
    - Redirection of `stdout`
    - Leaked file descriptors
    - Executable memory
    - Badly built libraries

21. *Your information may be compromised* if you have confined domains trying to:
    - Load kernel modules
    - Turn off the enforcing mode of SELinux
    - Write to `etc_t/shadow_t`
    - Modify iptables rules

22. SELinux tools for the development of policy modules:

```
$ yum -y install setroubleshoot setroubleshoot-server
```

    Reboot or restart `auditd` after you install.

23. Use `journalctl` for listing all logs related to `setroubleshoot`:

```
$ sudo journalctl -t setroubleshoot --since=14:20
```

24. Use `journalctl` for listing all logs related to a particular SELinux label. For example:

```
$ sudo journalctl _SELINUX_CONTEXT=system_u:system_r:policykit_t:s0
```

25. Use `setroubleshoot` log when an SELinux error occurs and suggest some possible solutions. For example, from `journalctl`:

```
Jun 14 19:41:07 web1 setroubleshoot: SELinux is preventing httpd from getattr access on
the file /var/www/html/index.html. For complete message run: sealert -l 12fd8b04-0119-
4077-a710-2d0e0ee5755e

# sealert -l 12fd8b04-0119-4077-a710-2d0e0ee5755e
SELinux is preventing httpd from getattr access on the file /var/www/html/index.html.

***** Plugin restorecon (99.5 confidence) suggests ************************

If you want to fix the label,
/var/www/html/index.html default label should be httpd_syscontent_t.
Then you can restorecon.
Do
# /sbin/restorecon -v /var/www/html/index.html
```

26. Logging: SELinux records information all over the place:
    - `/var/log/messages`
    - `/var/log/audit/audit.log`
    - `/var/lib/setroubleshoot/setroubleshoot_database.xml`
27. Logging: Looking for SELinux errors in the audit log:

```
$ sudo ausearch -m AVC,USER_AVC,SELINUX_ERR -ts today
```

28. To search for SELinux Access Vector Cache (AVC) messages for a particular service:

```
$ sudo ausearch -m avc -c httpd
```

29. The `audit2allow` utility gathers information from logs of denied operations and then generates SELinux policy-allow rules. For example:
    - To produce a human-readable description of why the access was denied: `# audit2allow -w -a`
    - To view the type enforcement rule that allows the denied access: `# audit2allow -a`
    - To create a custom module: `# audit2allow -a -M mypolicy`

      The `-M` option creates a type enforcement file (.te) with the name specified and compiles the rule into a policy package (.pp): `mypolicy.pp mypolicy.te`

    - To install the custom module: `# semodule -i mypolicy.pp`
30. To configure a single process (domain) to run permissive: `# semanage permissive -a httpd_t`
31. If you no longer want a domain to be permissive: `# semanage permissive -d httpd_t`
32. To disable all permissive domains:

```
$ sudo semodule -d permissivedomains
```

33. Enabling SELinux MLS policy:

```
$ sudo yum install selinux-policy-mls
```

    In `/etc/selinux/config`:

```
SELINUX=permissive
SELINUXTYPE=mls
```

    Ensure that SELinux is running in permissive mode:

```
$ sudo setenforce 0
```

    Use the `fixfiles` script to ensure that files are relabeled upon the next reboot:

```
$ sudo fixfiles -F onboot
$ sudo reboot
```

34. Create a user with a specific MLS range:

```
$ sudo useradd -Z staff_u tux
```

Using the `useradd` command, map the new user to an existing SELinux user (in this case, `staff_u`).

35. To view the mapping between SELinux and Linux users:

```
$ sudo semanage login -l
```

36. Define a specific range for a user:

```
$ sudo semanage login --modify --range s2:c100 tux
```

37. To correct the label on the user's home directory (if needed):

```
$ sudo chcon -R -l s2:c100 /home/tux
```

38. To list the current categories:

```
$ sudo chcat -L
```

39. To modify the categories or to start creating your own, modify the file as follows:

```
/etc/selinux/_<selinuxtype>_/setrans.conf
```

40. To run a command or script in a specific file, role, and user context:

```
$ sudo runcon -t initrc_t -r system_r -u user_u yourcommandhere
```

    - -t is the *file context*
    - -r is the *role context*
    - -u is the *user context*

41. Containers running with SELinux disabled:
    - With Podman: `# podman run --security-opt label=disable …`
    - With Docker: `# docker run --security-opt label=disable …`

42. If you need to give a container full access to the system:
    - With Podman: `# podman run --privileged …`
    - With Docker: `# docker run --privileged …`

And with this, you already know the answer. So please: **Don't panic, and turn on SELinux**.

Sources:

- SELinux by Dan Walsh
- Your visual how-to guide for SELinux policy enforcement also by Dan Walsh
- Security Enhanced Linux for mere mortals by Thomas Cameron
- The SELinux Coloring Book by Máirín Duffy
- SELinux User's and Administrator's Guide—Red Hat Enterprise Linux 7