Linux Security

SELINUX

Table of Contents

1. SELinux

1.1. Introduction

SELinux was developed as an additional Linux security solution that uses the security framework in the Linux kernel. The purpose was to allow for a more granular security policy that goes beyond what is offered by the default existing permissions of Read, Write, and Execute, and beyond assigning permissions to the different capabilities that are available on Linux. SELinux does this by trapping all system calls that reach the kernel, and denying them by default. This means that on a system that has SELinux enabled and nothing else configured, nothing will work. To allow your system to do anything, as an administrator you will need to write rules and put them in a policy.

An example explains why a solution such as SELinux (or its counterpart AppArmor) is needed :

One morning, I found out that my server was hacked. The server was running a fully patched SLES installation. A firewall was configured on it and no unnecessary services were offered by this server. Further analysis learned that the hacker had come in through a flaky PHP script that was a part of one of the Apache virtual hosts that were running on this server. The intruder had managed to get access to a shell, using the wwwrun account that was used by the Apache Web server. As this wwwrun user, the intruder had created several scripts in the /var/tmp and the /tmp directories, which were a part of a botnet that was launching a Distributed Denial of Service attack against several servers.

The interesting thing about this hack is that it occurred on a server where nothing was really wrong. All permissions where set OK, but the intruder had managed to get into the system. What becomes clearly evident from this example is that in some cases additional security is needed—a security that goes beyond what is offered by using SELinux. As a less complete and less complex alternative, AppArmor can be used.

AppArmor confines specific processes in their abilities to read/write and execute files (and other things). Its view is mostly that things that happen inside a process cannot escape.

SELinux instead uses labels attached to objects (for example, files, binaries, network sockets) and uses them to determine privilege boundaries, thereby building up a level of confinement that can span more than a process or even the whole system.
SELinux was developed by the US National Security Agency (NSA), and since the beginning Red Hat has been heavily involved in its development. The first version of SELinux was offered in the era of, around the year 2006. In the beginning, it offered support for essential services only, but over the years it has developed into a system that offers many rules that are collected in policies to offer protection to a broad range of services.

SELinux was developed in accordance with some certification standards like Common Criteria and FIPS 140. Because some customers specifically requested solutions that met these standards, SELinux rapidly became relatively popular.

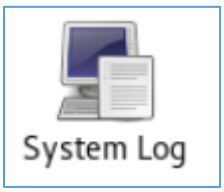
As an alternative to SELinux, Immunix, a company that was purchased by Novell in 2005, had developed AppArmor. AppArmor was built on top of the same security principles as SELinux, but took a completely different approach, where it was possible to restrict services to exactly what they needed to do by using an easy to use wizard-driven procedure. Nevertheless, AppArmor has never reached the same status as SELinux, even if there are some good arguments to secure a server with AppArmor rather than with SELinux.

Because many organizations are requesting SELinux to be in the Linux distributions they are using, SUSE is offering support for the SELinux framework in SUSE Linux Enterprise Server. This does not mean that the default installation of SUSE Linux Enterprise Server will switch from AppArmor to SELinux in the near future.

1.2. Selinux modes

Selinux knows three modes: enforcing, permissive and disabled. The enforcing mode will enforce policies, and may deny access based on selinux rules. The permissive mode will not enforce policies, but can still log actions that would have been denied in enforcing mode. The disabled mode disables selinux.

2. Logging



SLES 12 doesn't use syslog-ng anymore.
It now uses rsyslog (http://www.rsyslog.com)

Verify that rsyslog is running and activated on boot to enable logging of deny messages in /var/log/messages.

```
linsrv1:~ # rcsyslog status
Usage: /sbin/rcsyslog {start|stop|status|try-restart|restart|force-reload|reload}
● rsyslog.service - System Logging Service
   Loaded: loaded (/usr/lib/systemd/system/rsyslog.service; enabled; vendor preset: disabled)
   Active: active (running) since Thu 2016-12-15 11:38:42 CET; 29min ago
 Main PID: 1158 (rsyslogd)
    Tasks: 5 (limit: 512)
   CGroup: /system.slice/rsyslog.service
           └─1158 /usr/sbin/rsyslogd -n
```

```
linsrv1:~ # systemctl status rsyslog.service
● rsyslog.service - System Logging Service
   Loaded: loaded (/usr/lib/systemd/system/rsyslog.service; enabled; vendor preset: disabled)
   Active: active (running) since Thu 2016-12-15 11:38:42 CET; 39min ago
 Main PID: 1158 (rsyslogd)
    Tasks: 5 (limit: 512)
   CGroup: /system.slice/rsyslog.service
           └─1158 /usr/sbin/rsyslogd -n

Dec 15 11:38:42 linsrv1 systemd[1]: Starting System Logging Service...
Dec 15 11:38:42 linsrv1 systemd[1]: Started System Logging Service.
```

```
linsrv1:~ # systemctl is-active rsyslog.service
active
```

Verify that auditd is running and activated on boot to enable logging of easier to read messages in /var/log/audit/audit.log.

```
linsrv1:~ # systemctl status auditd
● auditd.service - Security Auditing Service
   Loaded: loaded (/usr/lib/systemd/system/auditd.service; disabled; vendor preset: disabled)
   Active: active (running) since Thu 2016-12-15 13:05:29 CET; 27min ago
 Main PID: 10104 (auditd)
    Tasks: 2 (limit: 512)
   CGroup: /system.slice/auditd.service
           └─10104 /sbin/auditd -n

Dec 15 13:05:29 linsrv1 systemd[1]: Starting Security Auditing Service...
Dec 15 13:05:29 linsrv1 auditctl[10105]: No rules
Dec 15 13:05:29 linsrv1 auditctl[10105]: AUDIT_STATUS: enabled=0 flag=1 pid=0 rate_limit=0 backlog_limit=320 lost=0 backlog=0
Dec 15 13:05:29 linsrv1 systemd[1]: Started Security Auditing Service.
Dec 15 13:05:29 linsrv1 auditd[10104]: Started dispatcher: /sbin/audispd pid: 10113
Dec 15 13:05:29 linsrv1 auditd[10104]: Init complete, auditd 2.3.6 listening for events (startup state enable)
```

If not activated, then run auditctl -e 1

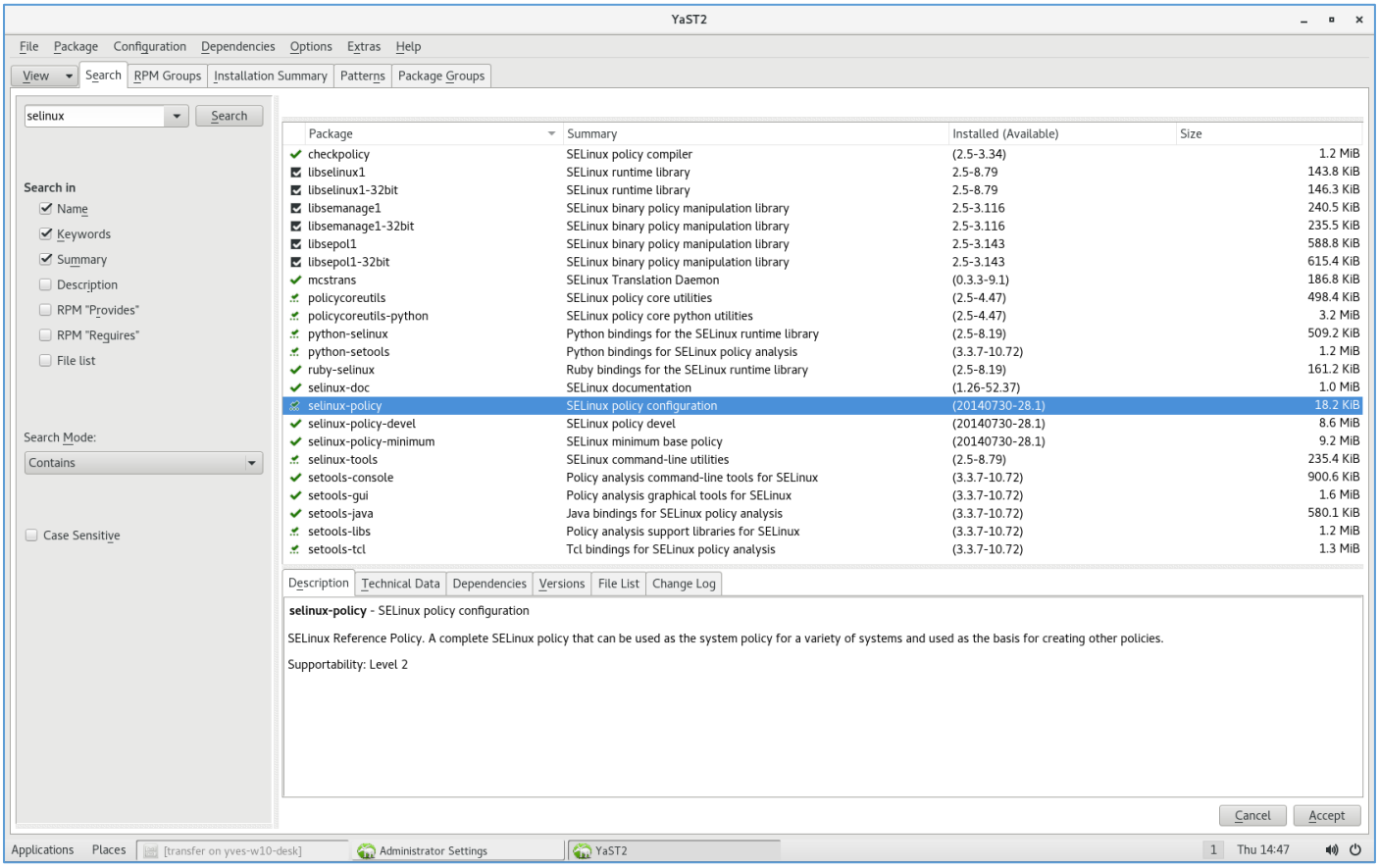
```
linsrv1:~ # auditctl -e 1
AUDIT_STATUS: enabled=1 flag=1 pid=26915 rate_limit=0 backlog_limit=320 lost=0 backlog=1
```

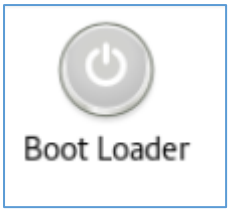
3. Configuring SELinux

3.1. Installing SELinux Packages and Modifying GRUB 2

Software > Software Management > View > Search => selinux

Selecting all SELinux Packages in Yast





After installing the SELinux packages, you need to modify the GRUB 2 boot loader. Do this from YaST, select System > Boot Loader > Kernel Parameters. Now add the following parameters to the Optional Kernel Command Line Parameters :

```
security=selinux selinux=1 enforcing=0
```

These options are used for the following purposes:

- security=selinux

This option tells the kernel to use SELinux and not AppArmor

- selinux=1

This option switches on SELinux

- enforcing=0

This option puts SELinux in permissive mode. In this mode, SELinux is fully functional, but does not enforce any of the security settings in the policy. Use this mode for configuring your system. To switch on SELinux protection, when the system is fully operational, change the option to enforcing=1 and add SELINUX=enforcing in/etc/selinux/config.
After installing the SELinux packages and enabling the SELinux GRUB 2 boot options, reboot your server to activate the configuration.

The /var/log/messages log file will tell you that selinux is disabled or enabled

```
linsrv1:~ # grep -i SELinux /var/log/messages
```

```
2016-12-15T15:06:23.735089+01:00 linsrv1 kernel: [    0.005330] SELinux:  Initializing.
2016-12-15T15:06:23.735092+01:00 linsrv1 kernel: [    0.005338] SELinux:  Starting in permissive mode
2016-12-15T15:06:23.735133+01:00 linsrv1 kernel: [    0.083520] evm: security.selinux
2016-12-15T15:06:23.735379+01:00 linsrv1 kernel: [    1.201961] SELinux:  Registering netfilter hooks
2016-12-15T15:06:23.735643+01:00 linsrv1 kernel: [    3.956447] SELinux: 32768 avtab hash slots, 102495 rules.
2016-12-15T15:06:23.735644+01:00 linsrv1 kernel: [    3.967754] SELinux: 32768 avtab hash slots, 102495 rules.
2016-12-15T15:06:23.735644+01:00 linsrv1 kernel: [    3.985864] SELinux:  8 users, 98 roles, 4615 types, 282 bools, 1 sens, 1024 cats
2016-12-15T15:06:23.735646+01:00 linsrv1 kernel: [    3.985864] SELinux:  83 classes, 102495 rules
2016-12-15T15:06:23.735646+01:00 linsrv1 kernel: [    3.988813] SELinux:  Class netlink_iscsi_socket not defined in policy.
2016-12-15T15:06:23.735647+01:00 linsrv1 kernel: [    3.988816] SELinux:  Class netlink_fib_lookup_socket not defined in policy.
2016-12-15T15:06:23.735647+01:00 linsrv1 kernel: [    3.988817] SELinux:  Class netlink_connector_socket not defined in policy.
2016-12-15T15:06:23.735647+01:00 linsrv1 kernel: [    3.988817] SELinux:  Class netlink_netfilter_socket not defined in policy.
2016-12-15T15:06:23.735648+01:00 linsrv1 kernel: [    3.988820] SELinux:  Class netlink_generic_socket not defined in policy.
2016-12-15T15:06:23.735648+01:00 linsrv1 kernel: [    3.988820] SELinux:  Class netlink_scsitransport_socket not defined in policy.
2016-12-15T15:06:23.735650+01:00 linsrv1 kernel: [    3.988821] SELinux:  Class netlink_rdma_socket not defined in policy.
2016-12-15T15:06:23.735650+01:00 linsrv1 kernel: [    3.988821] SELinux:  Class netlink_crypto_socket not defined in policy.
2016-12-15T15:06:23.735650+01:00 linsrv1 kernel: [    3.988833] SELinux:  Permission audit_read in class capability2 not defined in policy.
2016-12-15T15:06:23.735651+01:00 linsrv1 kernel: [    3.988834] SELinux:  Class binder not defined in policy.
2016-12-15T15:06:23.735651+01:00 linsrv1 kernel: [    3.988835] SELinux: the above unknown classes and permissions will be allowed
2016-12-15T15:06:23.735652+01:00 linsrv1 kernel: [    3.988839] SELinux:  Completing initialization.
2016-12-15T15:06:23.735653+01:00 linsrv1 kernel: [    3.988839] SELinux:  Setting up existing superblocks.
```

```
linsrv1:~ # grep SELinux /var/log/messages | grep -i Init
2016-12-15T15:06:23.735089+01:00 linsrv1 kernel: [    0.005330] SELinux:  Initializing.
2016-12-15T15:06:23.735652+01:00 linsrv1 kernel: [    3.988839] SELinux:  Completing initialization.
```

3.2. Verifying that SELinux is functional

```
linsrv1:~ # sestatus -v
SELinux status:                 enabled
SELinuxfs mount:                /sys/fs/selinux
SELinux root directory:         /etc/selinux
Loaded policy name:             minimum
Current mode:                   permissive
Mode from config file:          permissive
Policy MLS status:              enabled
Policy deny_unknown status:     allowed
Max kernel policy version:      30

Process contexts:
Current context:                unconfined_u:unconfined_r:unconfined_t:s0
Init context:                   system_u:system_r:kernel_t:s0
/usr/sbin/sshd                  system_u:system_r:kernel_t:s0

File contexts:
Controlling terminal:           unconfined_u:object_r:devpts_t:s0
/etc/passwd                     system_u:object_r:unlabeled_t:s0
/etc/shadow                     system_u:object_r:unlabeled_t:s0
/bin/bash                       system_u:object_r:unlabeled_t:s0
/bin/login                      system_u:object_r:unlabeled_t:s0
/bin/sh                         system_u:object_r:unlabeled_t:s0 -> system_u:object_r:unlabeled_t:s0
/sbin/agetty                    system_u:object_r:unlabeled_t:s0 -> system_u:object_r:unlabeled_t:s0
/sbin/init                      system_u:object_r:unlabeled_t:s0 -> system_u:object_r:unlabeled_t:s0
/sbin/mingetty                  system_u:object_r:unlabeled_t:s0
/usr/sbin/sshd                  system_u:object_r:unlabeled_t:s0
/lib/libc.so.6                  system_u:object_r:unlabeled_t:s0 -> system_u:object_r:unlabeled_t:s0
/lib/ld-linux.so.2              system_u:object_r:unlabeled_t:s0 -> system_u:object_r:unlabeled_t:s0
```

3.3. Configuring SELinux

At this point you have a completely functional SELinux system and it is time to further configure it.In the current status, SELinux is operational but not in enforcing mode. This means that it does not limit you in doing anything, it logs everything that it should be doing if it were in enforcing mode.
This is good, because based on the log files you can find what it is that it would prevent you from doing. As a first test, put SELinux in enforcing mode and find out if you can still use your server after doing so : check that the optionenforcing=1 is set in the GRUB 2 configuration file, while SELINUX=enforcing is set in /etc/selinux/config.
Reboot your server and see if it still comes up the way you expect it to. If it does, leave it like that and start modifying the server in a way that everything works as expected. However, you may not even be able to boot the server properly. In that case, switch back to the mode where SELinux is not enforcing and start tuning your server.

3.3.1. getenforce

In SELinux, three different modes can be used:

- Enforcing:

This is the default mode. SELinux protects your server according to the rules in the policy, and SELinux logs all of its activity to the audit log.

- Permissive:

This mode is useful for troubleshooting. If set to Permissive, SELinux does not protect your server, but it still logs everything that happens to the log files.

- Disabled:

In this mode, SELinux is switched off completely and no logging occurs. The file system labels however are not removed from the file system.
You have already read how you can set the current SELinux mode from GRUB 2 while booting using the enforcing boot parameter.

Use getenforce to verify whether selinux is enforced, disabled or permissive.

```
linsrv1:~ # getenforce
Permissive
```

The /selinux/enforce file contains 1 when enforcing, and 0 when permissive mode is active.

```
root@fedora13 ~# cat /selinux/enforce
1root@fedora13 ~#
```

3.3.2. setenforce

You can use setenforce to switch between the Permissive or the Enforcing state once selinux is activated.

```
linsrv1:~ # setenforce Enforcing
linsrv1:~ # getenforce
Enforcing
linsrv1:~ # setenforce Permissive
linsrv1:~ # getenforce
Permissive
```

3.3.3. sestatus

You can see the current selinux status and policy with the sestatus command.

```
linsrv1:~ # sestatus
SELinux status:                 enabled
SELinuxfs mount:                /sys/fs/selinux
SELinux root directory:         /etc/selinux
Loaded policy name:             minimum
Current mode:                   permissive
Mode from config file:          permissive
Policy MLS status:              enabled
Policy deny_unknown status:     allowed
Max kernel policy version:      30
```

3.4. SELinux Policy

Most Red Hat server will have the targeted policy. Only NSA/FBI/CIA/DOD/HLS use the mls policy.
The targted policy will protect hundreds of processes, but lets other processes run 'unconfined' (= they can do anything).

The policy is an essential component of SELinux. SUSE Linux Enterprise Server 12 SP2 includes the minimum SELinux reference policy in the package selinux-policy-minimum.

3.4.1. /etc/selinux/config

The main configuration file for selinux is /etc/selinux/config. When in permissive mode, the file looks like this.
The targeted policy is selected in /etc/selinux/config.

```
linsrv1:~ # cat /etc/selinux/config
# 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=permissive
# SELINUXTYPE= can take one of these two values:
#     targeted - Targeted processes are protected,
#     mls - Multi Level Security protection.
#     minimum - Modification of targeted policy. Only selected processes are protected.
SELINUXTYPE=minimum
```

12.9. DAC or MAC

Standard Unix permissions use Discretionary Access Control to set permissions on files. This means that a user that owns a file, can make it world readable by typing chmod 777 $file.
With selinux the kernel will enforce Mandatory Access Control which strictly controls what processes or threads can do with files (superseding DAC). Processes are confined by the kernel to the minimum access they require.
SELinux MAC is about labeling and type enforcing! Files, processes, etc are all labeled with an SELinux context. For files, these are extended attributes, for processes this is managed by the kernel.
The format of the labels is as follows:

```
user:role:type:(level)
```

We only use the type label in the targeted policy.

12.10. ls -Z

To see the DAC permissions on a file, use ls -l to display user and group owner and permissions.
For MAC permissions there is new -Z option added to ls. The output shows that file in /root have a XXXtype of admin_home_t.

```
[root@centos65 ~]# ls -Z
-rw-------. root root system_u:object_r:admin_home_t:s0 anaconda-ks.cfg
-rw-r--r--. root root system_u:object_r:admin_home_t:s0 install.log
-rw-r--r--. root root system_u:object_r:admin_home_t:s0 install.log.syslog
[root@centos65 ~]# useradd -m -s /bin/bash pol
[root@centos65 ~]# ls -Z /home/pol/.bashrc
-rw-r--r--. pol pol unconfined_u:object_r:user_home_t:s0 /home/pol/.bashrc
```

12.11. -Z

There are also some other tools with the -Z switch:

```
mkdir -Z
cp -Z
ps -Z
netstat -Z
...
```


12.12. /selinux

When selinux is active, there is a new virtual file system named /selinux. (You can compare it to /proc and /dev.)

```
[root@centos65 ~]# ls -l /selinux/
total 0
-rw-rw-rw-. 1 root root 0 Apr 12 19:40 access
dr-xr-xr-x. 2 root root 0 Apr 12 19:40 avc
dr-xr-xr-x. 2 root root 0 Apr 12 19:40 booleans
-rw-r--r--. 1 root root 0 Apr 12 19:40 checkreqprot
dr-xr-xr-x. 83 root root 0 Apr 12 19:40 class
--w-------. 1 root root 0 Apr 12 19:40 commit_pending_bools
-rw-rw-rw-. 1 root root 0 Apr 12 19:40 context
-rw-rw-rw-. 1 root root 0 Apr 12 19:40 create
-r--r--r--. 1 root root 0 Apr 12 19:40 deny_unknown
--w-------. 1 root root 0 Apr 12 19:40 disable
-rw-r--r--. 1 root root 0 Apr 12 19:40 enforce
dr-xr-xr-x. 2 root root 0 Apr 12 19:40 initial_contexts
-rw-------. 1 root root 0 Apr 12 19:40 load
-rw-rw-rw-. 1 root root 0 Apr 12 19:40 member
-r--r--r--. 1 root root 0 Apr 12 19:40 mls
crw-rw-rw-. 1 root root 1, 3 Apr 12 19:40 null
-r--------. 1 root root 0 Apr 12 19:40 policy
dr-xr-xr-x. 2 root root 0 Apr 12 19:40 policy_capabilities
-r--r--r--. 1 root root 0 Apr 12 19:40 policyvers
-r--r--r--. 1 root root 0 Apr 12 19:40 reject_unknown
-rw-rw-rw-. 1 root root 0 Apr 12 19:40 relabel
```

```
-r--r--r--. 1 root root 0 Apr 12 19:40 status
-rw-rw-rw-. 1 root root 0 Apr 12 19:40 user
```
Although some files in /selinux appear wih size 0, they often contain a boolean value. Check /selinux/enforce to see if selinux is running in enforced mode.
```
[root@RHEL5 ~]# ls -l /selinux/enforce
-rw-r--r-- 1 root root 0 Apr 29 08:21 /selinux/enforce
[root@RHEL5 ~]# echo $(cat /selinux/enforce)
1
```
12.13. identity
The SELinux Identity of a user is distinct from the user ID. An identity is part of a security context, and (via domains) determines what you can do. The screenshot shows user root having identity user_u.
```
[root@rhel55 ~]# id -Z
user_u:system_r:unconfined_t
```
12.14. role
The selinux role defines the domains that can be used. A role is denied to enter a domain, unless the role is explicitely authorized to do so.

12.15. type (or domain)
The selinux context is the security context of a process. An selinux type determines what a process can do. The screenshot shows init running in type init_t and the mingetty's running in type getty_t.
```
[root@centos65 ~]# ps fax -Z | grep /sbin/init
system_u:system_r:init_t:s0 1 ? Ss 0:00 /sbin/init
[root@centos65 ~]# ps fax -Z | grep getty_t
system_u:system_r:getty_t:s0 1307 tty1 Ss+ 0:00 /sbin/mingetty /dev/tty1
system_u:system_r:getty_t:s0 1309 tty2 Ss+ 0:00 /sbin/mingetty /dev/tty2
system_u:system_r:getty_t:s0 1311 tty3 Ss+ 0:00 /sbin/mingetty /dev/tty3
system_u:system_r:getty_t:s0 1313 tty4 Ss+ 0:00 /sbin/mingetty /dev/tty4
system_u:system_r:getty_t:s0 1320 tty5 Ss+ 0:00 /sbin/mingetty /dev/tty5
system_u:system_r:getty_t:s0 1322 tty6 Ss+ 0:00 /sbin/mingetty /dev/tty6
```
The selinux type is similar to an selinux domain, but refers to directories and files instead of processes.
Hundreds of binaries also have a type:
```
[root@centos65 sbin]# ls -lZ useradd usermod userdel httpd postcat postfix
-rwxr-xr-x. root root system_u:object_r:httpd_exec_t:s0 httpd
-rwxr-xr-x. root root system_u:object_r:postfix_master_exec_t:s0 postcat
-rwxr-xr-x. root root system_u:object_r:postfix_master_exec_t:s0 postfix
-rwxr-x---. root root system_u:object_r:useradd_exec_t:s0 useradd
-rwxr-x---. root root system_u:object_r:useradd_exec_t:s0 userdel
-rwxr-x---. root root system_u:object_r:useradd_exec_t:s0 usermod
```
Ports also have a context.
```
[root@centos65 sbin]# netstat -nptlZ | tr -s ' ' | cut -d' ' -f6-
Foreign Address State PID/Program name Security Context
LISTEN 1096/rpcbind system_u:system_r:rpcbind_t:s0
LISTEN 1208/sshd system_u:system_r:sshd_t:s0-s0:c0.c1023
LISTEN 1284/master system_u:system_r:postfix_master_t:s0
LISTEN 1114/rpc.statd system_u:system_r:rpcd_t:s0
LISTEN 1096/rpcbind system_u:system_r:rpcbind_t:s0
LISTEN 1666/httpd unconfined_u:system_r:httpd_t:s0
LISTEN 1208/sshd system_u:system_r:sshd_t:s0-s0:c0.c1023
LISTEN 1114/rpc.statd system_u:system_r:rpcd_t:s0
LISTEN 1284/master system_u:system_r:postfix_master_t:s0
```
You can also get a list of ports that are managed by SELinux:
```
[root@centos65 ~]# semanage port -l | tail
xfs_port_t tcp 7100
xserver_port_t tcp 6000-6150
zabbix_agent_port_t tcp 10050
zabbix_port_t tcp 10051
zarafa_port_t tcp 236, 237
zebra_port_t tcp 2600-2604, 2606
zebra_port_t udp 2600-2604, 2606
```

```
zented_port_t tcp 1229
zented_port_t udp 1229
zope_port_t tcp 8021
```


12.16. security context

The combination of identity, role and domain or type make up the selinux security context.
The id will show you your security context in the form identity:role:domain.

```
[paul@RHEL5 ~]$ id | cut -d' ' -f4
context=user_u:system_r:unconfined_t
```

The ls -Z command shows the security context for a file in the form identity:role:type.

```
[paul@RHEL5 ~]$ ls -Z test
-rw-rw-r-- paul paul user_u:object_r:user_home_t test
```

The security context for processes visible in /proc defines both the type (of the file in /proc) and the domain (of the running process). Let's take a look at the init process and /proc/1/ .
The init process runs in domain init_t.

```
[root@RHEL5 ~]# ps -ZC init
LABEL PID TTY TIME CMD
system_u:system_r:init_t 1 ? 00:00:01 init
```

The /proc/1/ directory, which identifies the init process, has type init_t.

```
[root@RHEL5 ~]# ls -Zd /proc/1/
dr-xr-xr-x root root system_u:system_r:init_t /proc/1/
```

It is not a coincidence that the domain of the init process and the type of /proc/1/ are both init_t.
Don't try to use chcon on /proc! It will not work.

12.17. transition

An selinux transition (aka an selinux labelling) determines the security context that will be assigned. A transition of process domains is used when you execute a process. A transition of file type happens when you create a file.
An example of file type transition.

```
[pol@centos65 ~]$ touch test /tmp/test
[pol@centos65 ~]$ ls -Z test
-rw-rw-r--. pol pol unconfined_u:object_r:user_home_t:s0 test
[pol@centos65 ~]$ ls -Z /tmp/test
-rw-rw-r--. pol pol unconfined_u:object_r:user_tmp_t:s0 /tmp/test
```


12.18. extended attributes

Extended attributes are used by selinux to store security contexts. These attributes can be viewed with ls when selinux is running.

```
[root@RHEL5 home]# ls --context
drwx------ paul paul system_u:object_r:user_home_dir_t paul
drwxr-xr-x root root user_u:object_r:user_home_dir_t project42
drwxr-xr-x root root user_u:object_r:user_home_dir_t project55
[root@RHEL5 home]# ls -Z
drwx------ paul paul system_u:object_r:user_home_dir_t paul
drwxr-xr-x root root user_u:object_r:user_home_dir_t project42
drwxr-xr-x root root user_u:object_r:user_home_dir_t project55
[root@RHEL5 home]#
```

When selinux is not running, then getfattr is the tool to use.

```
[root@RHEL5 etc]# getfattr -m . -d hosts
# file: hosts
security.selinux="system_u:object_r:etc_t:s0\000"
```

12.19. process security context

A new option is added to ps to see the selinux security context of processes.

```
[root@RHEL5 etc]# ps -ZC mingetty
LABEL PID TTY TIME CMD
system_u:system_r:getty_t 2941 tty1 00:00:00 mingetty
system_u:system_r:getty_t 2942 tty2 00:00:00 mingetty
```

12.20. chcon

Use chcon to change the selinux security context.

This example shows how to use chcon to change the type of a file.

```
[root@rhel55 ~]# ls -Z /var/www/html/test42.txt
-rw-r--r-- root root user_u:object_r:httpd_sys_content_t /var/www/html/test4\
2.txt
[root@rhel55 ~]# chcon -t samba_share_t /var/www/html/test42.txt
[root@rhel55 ~]# ls -Z /var/www/html/test42.txt
-rw-r--r-- root root user_u:object_r:samba_share_t /var/www/html/test42.txt
```

Be sure to read man chcon.

12.21. an example

The Apache2 webserver is by default targeted with SELinux. The next screenshot shows that any file created in /var/www/html will by default get the httpd_sys_content_t type.

```
[root@centos65 ~]# touch /var/www/html/test42.txt
[root@centos65 ~]# ls -Z /var/www/html/test42.txt
-rw-r--r--. root root unconfined_u:object_r:httpd_sys_content_t:s0 /var/www/h\
tml/test42.txt
```

Files created elsewhere do not get this type.

```
[root@centos65 ~]# touch /root/test42.txt
[root@centos65 ~]# ls -Z /root/test42.txt
-rw-r--r--. root root unconfined_u:object_r:admin_home_t:s0 /root/test42.txt
```

Make sure Apache2 runs.

```
[root@centos65 ~]# service httpd restart
Stopping httpd: [ OK ]
Starting httpd: [ OK ]
```

Will this work ? Yes it does.

```
[root@centos65 ~]# wget http://localhost/test42.txt
--2014-04-12 20:56:47-- http://localhost/test42.txt
Resolving localhost... ::1, 127.0.0.1
Connecting to localhost|::1|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 0 [text/plain]
Saving to: “test42.txt”
...
```

Why does this work ? Because Apache2 runs in the httpd_t domain and the files in /var/www/html have the httpd_sys_content_t type.

```
[root@centos65 ~]# ps -ZC httpd | head -4
LABEL PID TTY TIME CMD
unconfined_u:system_r:httpd_t:s0 1666 ? 00:00:00 httpd
unconfined_u:system_r:httpd_t:s0 1668 ? 00:00:00 httpd
unconfined_u:system_r:httpd_t:s0 1669 ? 00:00:00 httpd
```


So let's set SELinux to enforcing and change the type of this file.

```
[root@centos65 ~]# chcon -t samba_share_t /var/www/html/test42.txt
[root@centos65 ~]# ls -Z /var/www/html/test42.txt
-rw-r--r--. root root unconfined_u:object_r:samba_share_t:s0 /var/www/html/t\
est42.txt
[root@centos65 ~]# setenforce 1
[root@centos65 ~]# getenforce
Enforcing
```

There are two possibilities now: either it works, or it fails. It works when selinux is in permissive mode, it fails when in enforcing mode.

```
[root@centos65 ~]# wget http://localhost/test42.txt
--2014-04-12 21:05:02-- http://localhost/test42.txt
Resolving localhost... ::1, 127.0.0.1
Connecting to localhost|::1|:80... connected.
HTTP request sent, awaiting response... 403 Forbidden
2014-04-12 21:05:02 ERROR 403: Forbidden.
```

The log file gives you a cryptic message...

```
[root@centos65 ~]# tail -3 /var/log/audit/audit.log
type=SYSCALL msg=audit(1398200702.803:64): arch=c000003e syscall=4 succ\
```

```
ess=no exit=-13 a0=7f5fbc334d70 a1=7fff553b4f10 a2=7fff553b4f10 a3=0 it\
ems=0 ppid=1666 pid=1673 auid=500 uid=48 gid=48 euid=48 suid=48 fsuid=4\
8 egid=48 sgid=48 fsgid=48 tty=(none) ses=1 comm="httpd" exe="/usr/sbin\
/httpd" subj=unconfined_u:system_r:httpd_t:s0 key=(null)
type=AVC msg=audit(1398200702.804:65): avc: denied { getattr } for p\
id=1673 comm="httpd" path="/var/www/html/test42.txt" dev=dm-0 ino=26324\
1 scontext=unconfined_u:system_r:httpd_t:s0 tcontext=unconfined_u:objec\
t_r:samba_share_t:s0 tclass=file
type=SYSCALL msg=audit(1398200702.804:65): arch=c000003e syscall=6 succ\
ess=no exit=-13 a0=7f5fbc334e40 a1=7fff553b4f10 a2=7fff553b4f10 a3=1 it\
ems=0 ppid=1666 pid=1673 auid=500 uid=48 gid=48 euid=48 suid=48 fsuid=4\
8 egid=48 sgid=48 fsgid=48 tty=(none) ses=1 comm="httpd" exe="/usr/sbin\
/httpd" subj=unconfined_u:system_r:httpd_t:s0 key=(null)
```

And /var/log/messages mentions nothing of the failed download.

12.22. setroubleshoot

The log file above was not very helpful, but these two packages can make your life much easier.

```
[root@centos65 ~]# yum -y install setroubleshoot setroubleshoot-server
```

You need to reboot for this to work...

So we reboot, restart the httpd server, reactive SELinux Enforce, and do the wget again... and it fails (because of SELinux).

```
[root@centos65 ~]# service httpd restart
Stopping httpd: [FAILED]
Starting httpd: [ OK ]
[root@centos65 ~]# getenforce
Permissive
[root@centos65 ~]# setenforce 1
[root@centos65 ~]# getenforce
Enforcing
[root@centos65 ~]# wget http://localhost/test42.txt
--2014-04-12 21:44:13-- http://localhost/test42.txt
Resolving localhost... ::1, 127.0.0.1
Connecting to localhost|::1|:80... connected.
HTTP request sent, awaiting response... 403 Forbidden
2014-04-12 21:44:13 ERROR 403: Forbidden.
```

The /var/log/audit/ is still not out best friend, but take a look at /var/log/messages.

```
[root@centos65 ~]# tail -2 /var/log/messages
Apr 12 21:44:16 centos65 setroubleshoot: SELinux is preventing /usr/sbin/h\
ttpd from getattr access on the file /var/www/html/test42.txt. For complete \
SELinux messages. run sealert -l b2a84386-54c1-4344-96fb-dcf969776696
Apr 12 21:44:16 centos65 setroubleshoot: SELinux is preventing /usr/sbin/h\
ttpd from getattr access on the file /var/www/html/test42.txt. For complete \
SELinux messages. run sealert -l b2a84386-54c1-4344-96fb-dcf969776696
```

So we run the command it suggests...

```
[root@centos65 ~]# sealert -l b2a84386-54c1-4344-96fb-dcf969776696
SELinux is preventing /usr/sbin/httpd from getattr access on the file /va\
r/www/html/test42.txt.
***** Plugin restorecon (92.2 confidence) suggests ********************
If you want to fix the label.
/var/www/html/test42.txt default label should be httpd_sys_content_t.
Then you can run restorecon.
Do
# /sbin/restorecon -v /var/www/html/test42.txt
...
```


We follow the friendly advice and try again to download our file:

```
[root@centos65 ~]# /sbin/restorecon -v /var/www/html/test42.txt
/sbin/restorecon reset /var/www/html/test42.txt context unconfined_u:objec\
t_r:samba_share_t:s0->unconfined_u:object_r:httpd_sys_content_t:s0
[root@centos65 ~]# wget http://localhost/test42.txt
--2014-04-12 21:54:03-- http://localhost/test42.txt
Resolving localhost... ::1, 127.0.0.1
Connecting to localhost|::1|:80... connected.
HTTP request sent, awaiting response... 200 OK
```

It works!
introduction to SELinux
112

12.23. booleans

Booleans are on/off switches

```
[root@centos65 ~]# getsebool -a | head
abrt_anon_write --> off
abrt_handle_event --> off
allow_console_login --> on
allow_cvs_read_shadow --> off
allow_daemons_dump_core --> on
allow_daemons_use_tcp_wrapper --> off
allow_daemons_use_tty --> on
allow_domain_fd_use --> on
allow_execheap --> off
allow_execmem --> on
```

You can set and read individual booleans.

```
[root@centos65 ~]# setsebool httpd_read_user_content=1
[root@centos65 ~]# getsebool httpd_read_user_content
httpd_read_user_content --> on
[root@centos65 ~]# setsebool httpd_enable_homedirs=1
[root@centos65 ~]# getsebool httpd_enable_homedirs
httpd_enable_homedirs --> on
```

You can set these booleans permanent.

```
[root@centos65 ~]# setsebool -P httpd_enable_homedirs=1
[root@centos65 ~]# setsebool -P httpd_read_user_content=1
```

The above commands regenerate the complete /etc/selinux/targeted directory!

```
[root@centos65 ~]# cat /etc/selinux/targeted/modules/active/booleans.local
# This file is auto-generated by libsemanage
# Do not edit directly.
httpd_enable_homedirs=1
httpd_read_user_content=1
```