SELinux Documentation

Versión 0.1.0

Rafael Rodriguez Gayoso

19 de septiembre de 2017

Índice general

1. Capítulo 1. Conceptos de SELinux **1**
- 1.1. Aplicando más seguridad a Linux . . . 1
- 1.2. Uso de módulos de seguridad en Linux . . . 2
- 1.3. Ampliando DAC con SELinux . . . 2
- 1.4. Restricción de los privilegios de root . . . 3
- 1.5. Reducir el impacto de las vulnerabilidades . . . 3
- 1.6. Etiquetando objetos y recursos . . . 3
- 1.7. Análisis del contexto SELinux . . . 4
- 1.8. Fortificando el acceso con *tipos* . . . 5
- 1.9. Conceder acceso al dominio mediante roles . . . 5
- 1.10. Limitar roles a través de usuarios . . . 6
- 1.11. Control del flujo de información a través de sensibilidades . . . 6
- 1.12. Diseño y distribución de políticas . . . 7
- 1.13. Escritura de políticas SELinux . . . 7
- 1.14. Distribución de políticas a través de módulos . . . 8
- 1.15. Agrupación de módulos en un almacén de políticas . . . 8

2. Capítulo 2 **9**

3. Licencia **11**
- 3.1. Licencia . . . 11

CAPÍTULO 1

Capítulo 1. Conceptos de SELinux

Security Enhanced Linux (SELinux) aporta medidas de seguridad a un sistema Linux con el objetivo de proteger sus recursos.

Aplicando más seguridad a Linux

Los administradores de sistemas deben depositar cierta confianza en los usuarios y procesos de sus sistemas para mantenerlos seguros. Los usuarios pueden intentar explotar vulnerabilidades del software que se ejecuta en el sistema. El control de acceso por defecto que está activo en un sistema Linux es de tipo *discrecional*; depende en parte del comportamiento de los usuarios lo eficaz que resultan estas medidas de protección.

El mecanismo **Discretionary Access Control (DAC)** se basa en la información de usuario y/o grupo de cada proceso y se comprueba con la información de usuario/grupo de cada fichero, o directorio, al que se pretenda acceder. Tomando como ejemplo, el fichero `/etc/shadow`, que contiene información sobre las credenciales de las cuentas de usuario locales:

```
$ ls -l /etc/shadow
-rw------- 1 root root 1010 Apr 25 22:05 /etc/shadow
```

Sin un mecanismo adicional de acceso, este fichero puede ser leído y modificado por cualquier proceso que tenga como propietario a `root`, sin importar el propósito de dicho proceso en el sistema.

Otro ejemplo de como DAC precisa de la confianza en sus usuarios es cuando tenemos almacenada una base de datos en nuestro sistema. Los ficheros con bases de datos sólo son accesibles desde usuarios del DBMS, y también por `root`. Un sistema correctamente configurado sólo permitirá el acceso a un conjunto de usuarios a dichos ficheros a través de una serie de comandos definidos. Dichos usuarios, pueden analizar los ficheros de bases de datos, y potencialmente, acceder a información confidencial en la base de datos sin acceder a través del DBMS.

Sin embargo, los usuarios regulares de un sistema Linux no es la única razón por la que proteger nuestro sistema. Multitud de *daemons* se ejecutan como `root`, o bien, tienen privilegios excesivos en el sistema. Cualquier error de programación en dicho software puede afectar a nuestro sistema, o bien durante la gestión por parte de un usuario sin privilegios de dichos *daemons* puede poner en peligro el sistema.

SELinux proporciona una capa de control de acceso adicional a DAC. A diferencia de DAC, SELinux ofrece un **Mandatory Access Control (MAC)** a nuestro sistema, dando un control total al administrador sobre lo que está permitido y lo que no.

Mandatory significa que el control de acceso se aplica por el sistema operativo y definido a través de las políticas que ha habilitado el administrador. Los usuarios y los procesos no tienen permisos para modificar las reglas de seguridad, evitando que la seguridad dependa de sus operaciones.

Más información:

- Trusted Computer System Evaluation Criteria

Uso de módulos de seguridad en Linux

Volvamos al ejemplo anterior del fichero `shadow`. Un sistema MAC puede ser configurado para permitir el acceso a un número limitado de procesos para leer y/o escribir en el mismo. En un sistema configurado de este modo, un usuario conectado como `root` no puede acceder directamente al fichero ni modificarlo:

```
# id
uid=0(roo) gid=0(root)
# cat /etc/shadow
cat: /etc/shadow: Permission denied
# chmod a+r /etc/shadow
chmod: changing permission of `/etc/shadow': Permission denied
```

Esto se hace cumplir a través de reglas que definen cuando se puede leer el contenido de un fichero. Con SELinux, estas reglas son definidas en la política de SELinux y se cargan durante el inicio del sistema. Es el propio kernel Linux el encargado de aplicar dichas reglas.

Ampliando DAC con SELinux

SELinux no modifica la configuración DAC, ni puede revocar las restricciones impuestas. Si un sistema (sin SELinux) prohibe un acceso concreto, no hay nada que SELinux pueda hacer para revocar esa decisión. Esto es así, porque los *hooks* de LSM son lanzados después del chequeo de los permisos DAC.

Por ejemplo, si es necesario permitir el acceso a un fichero a un usuario concreto, no es posible declarar una política SELinux para llevarlo a cabo. En su lugar, debemos hacer uso de las características de Linux, como puede ser el uso de ACL's:

```
$ setfacl -m u:lisa:rw /path/to/file

$ getfacl /path/to/file
# file: file
# owner: swift
# group: swift
user::rw-
user::lisa:rw-
group::r--
mask::r--
other::r--
```

Restricción de los privilegios de root

Cuando usamos un sistema DAC existe un usuario que tiene todos los poderes: `root`. Es decir, este usuario es capaz de llevar a cabo cualquier acción en el sistema, desde gestionar la red al control de acceso, o la modificación de los ID's de usuarios. Esto es soportado a través de un concepto que se conoce como **capabilities**. SELinux es capaz de restringir el acceso a esas *capacidades* de un modo detallado.

Derivado de este hecho, incluso el usuario `root` puede ser controlado sin impactar en las operaciones sobre el sistema.

Reducir el impacto de las vulnerabilidades

Uno de los principales beneficios de SELinux es su capacidad de reducir el impacto de una vulnerabilidad.

Una política de SELinux bien diseñada es capaz de controlar a las aplicaciones para que sólo puedan realizar un número reducido de actividades. Este modelo de *privilegios reducidos* se asegura de que cualquier comportamiento anormal no sólo es detectado, sino que es bloqueado.

Sin embargo, hay dos conceptos erróneos acerca de la capacidad de SELinux para frustar los ataques, que son, el impacto de las políticas y los posibles *exploits* sobre las mismas.

Si la política no sigue el concepto de *privilegios reducidos*, entonces SELinux puede considerar este comportamiento anómalo como normal y permitir su ejecución. Esto significa, que las políticas tienen que ser diseñadas con mucho detalle, lo que se traduce en que se invertirá una gran cantidad de tiempo; hay más de 80 clases y sobre 200 permisos gestionados por SELinux, y las reglas deben tener en cuenta todas esas clases y permisos para cada interacción entre dos objetos o recursos.

El segundo concepto equivocado es el *exploit*. Si una vulnerabilidad permite a un usuario no autenticado usar los servicios de una aplicación como si estuviera autorizado, entonces SELinux no aplicará ninguna medida para reducir el impacto de la vulnerabilidad. Si dicha aplicación accede a aquellos ficheros a los que tiene permiso, SELinux tampoco realizará ninguna medida correctora. Todo es normal.

Digamos que SELinux sólo entra en acción cuando la aplicación tiene un comportamiento no esperado, o errático. SELinux puede controlar una **RCE (Remote Command Execution)** contra una aplicación, pero nada puede hacer contra un secuestro de sesión, o una inyección de SQL.

Etiquetando objetos y recursos

Cuando SELinux tiene que tomar una decisión sobre si permitir o no una determinada acción, lo hace basándose en el contexto tanto del **sujeto** (que es quien inicia la acción) y el **objeto** (que es el objetivo de la acción). Estes contextos son referenciados en las reglas de SELinux.

El contexto de un proceso es lo que identifica a un proceso para SELinux. SELinux no tiene constancia del propietario de los procesos de Linux, ni le importa el nombre, o su ID. Todo lo que precisa conocer es el contexto del proceso, que es representado a los usuarios y administradores como una **etiqueta**. *Etiqueta* y *contexto* aunque sean conceptos diferentes técnicamente, se pueden usar para referirse al mismo componente.

Un ejemplo de etiqueta: el contexto del usuario actual:

```
$ id -Z
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

En SELinux debemos hacer uso de las etiquetas para gestionar los controles de acceso. Esto se hace de este modo por las siguientes razones:

- El uso de rutas de binarios o ficheros (como se hace en AppArmor) es más sencillo, y fácil de manejar para los administradores, pero se convierte en una dependencia del recurso. Si el fichero es movido, o un proceso llega a tener una visión diferente del sistema de ficheros, entonces los sistemas de control de acceso no se comportarán como se espera. Con un contexto basado en etiquetas, esta información es estática y el sistema puede controlar mejor el recurso.
- Los contextos reflejan el propósito de un proceso. El mismo binario puede ser ejecutado en diferentes escenarios dependiendo del modo de inicio. El valor de su contexto es realmente lo que se necesita conocer desde el punto de vista del administrador, y poder llegar a determinar como ha sido ejecutado y cuál es su propósito.
- Los contextos realizan abstracciones de todo aquello que no es *tangible*, como puede ser una *pipe* o una base de datos.

Como ejemplo, tengamos en cuenta las siguientes políticas:

- Permitir al proceso `httpd` el uso del puerto TCP `80`.
- Permitir a los procesos etiquetados como `httpd_t` hacer uso de los puertos TCP etiquetados como `httpd_port_t`.

En el primer ejemplo, es más difícil la reutilización de la política cuando el servidor web no está usando el binario `httpd`, o cuando queremos tener un acceso HTTP en un puerto diferente. Con la versión *etiquetada*, el nombre de binario ya no importa, sólo que esté etiquetado como `httpd_t`. Y lo mismo ocurre con el puerto.

Análisis del contexto SELinux

Un contexto está formado por al menos tres valores, en ocasiones cuatro. El contexto de un servidor web Apache es el siguiente:

```
$ ps -eZ | grep httpd
system_u:system_r:httpd_t:s0  511  ?   00:00:00 httpd
```

Como se puede observar, el proceso tiene un contexto formado por los siguientes campos:

- `system_u`: Representa al usuario SELinux
- `system_r`: Representa el rol SELinux
- `httpd_t`: Representa el tipo SELinux (también conocido como el dominio de ser un proceso)
- `s0`: Representa el nivel de sensibilidad

La estructura puede ser representada del siguiente modo:

unconfined_u	unconfined_r	unconfined_t	s0-s0:c0.c1023
SELinux user	SELinux role	SELinux type	Sensivity level

Cuando trabajamos con SELinux, todo lo que necesitamos es el contexto. En la mayoría de los casos, el campo más importante es el tercero (dominio o tipo), ya que la mayoría de las políticas (99 %) están formadas por reglas que hacen referencia a la relación entre dos tipos (sin tener en cuenta roles, usuarios o niveles de sensibilidad).

Dentro de cada directorio `/proc/<pid>`, perteneciente a un proceso, podemos encontrar un subdirectorio llamado `attr`:

```
$ ls /proc/$$/attr
current      fscreate        prev
exec         keycreate       sockcreate
```

Estos ficheros puede estar vacíos o contener un contexto SELinux. De estar vacío, significa que la aplicación no tiene asignada un contexto para un uso determinado, y el contexto SELinux vendrá heredado de su padre.

El significado de los ficheros es el siguiente:

- El fichero `current` contiene el contexto SELinux actual del proceso.
- El fichero `exec` contiene el contexto SELinux que se asignará a la aplicación que se ejecute a partir de esta aplicación. Suele estar vacío.
- El fichero `fscreate` contiene el contexto SELinux que será asignado al próximo fichero que sea escrito por la aplicación. Suele estar vacío.
- El fichero `keycreate` contiene el contexto SELinux que será asignado a las llaves almacenadas (cached) en el kernel por la aplicación. Suele estar vacío.
- El fichero `prev` contiene el contexto SELinux anterior para este proceso. Suele ser el contexto del proceso padre.
- El fichero `sockcreate` contiene el contexto SELinux que será asignado al siguiente socket creado por la aplicación. Suele estar vacío.

Fortificando el acceso con *tipos*

El dominio de un proceso (el tercer campo del contexto SELinux) es la base del control de acceso detallado de un proceso respecto de si mismo y de otros tipos. Se suele hacer referencia a SELinux como un sistema de control de acceso *orientado a tipos*; de tal forma que cuando alguna acción no es autorizada, suele estar motivada por los controles de acceso (o la ausencia de ellos) a nivel de tipo.

Con esta orientación, SELinux es capaz de controlar lo que una aplicación puede hacer en base a como fue ejecutada la primera vez: un servidor web que es iniciado de forma interactiva por un usuario se situará en un tipo diferente a un servidor web iniciado a través de `init`, incluso cuando el binario es el mismo. El servidor web iniciado a través de `init` tendrá un nivel mayor de confianza.

Echemos un vistazo a los procesos `dbus-daemon`:

```
# ps -eZ | grep dbus-daemon
system_u:system_r:system_dbusd_t 4531 ?        00:00:00 dbus-daemon
staff_u:staff_r:staff_dbusd_t    5266 ?        00:00:00 dbus-daemon
```

En este ejemplo, un proceso `dbus-daemon` está en ejecución con el tipo `system_dbusd_t`, mientras que el otro lo hace con `staff_dbusd_t`.

A los nombres de los tipos en SELinux se les suele añadir el sufijo `_t`, aunque no es obligatario.

Conceder acceso al dominio mediante roles

El control de acceso basado en roles es un importante método para mantener la seguridad en un sistema, sobre todo evitando usos maliciosos por parte de los usuarios. Los roles de SELinux definen los tipos (dominios) con los que se permite a un proceso ser ejecutado. Los tipos definen los permisos, por lo que, un rol SELinux define lo que un usuario puede hacer o no, al tener acceso a uno o varios roles.

Por convención, los roles SELinux llevan el sufijo `_r`. En la mayoría de los sistemas con SELinux, están disponibles los siguientes roles:

- **user_r** - A este rol sólo le está permitido el uso de procesos relacionados con aplicaciones de usuario final. Privilegios, como el cambio de usuario, no están permitidos para este rol.
- **staff_r** - Es similar al anterior, pero está permitida la capacidad de cambio de rol. Está relacionado con aquellas operaciones que no son críticas para el sistema, y se suele asignar a *operadores*.

- **sysadm_r** - Este rol dispone de privilegios que permiten la ejecución de tareas administrativas del sistema.
- **secadm_r** - Este rol tiene la capacidad de modificar la política SELinux y gestionar los controles SELinux.
- **system_r** - Los procesos en segundo plano y los *daemons* tienen este rol.
- **unconfined_r** - Está asociado a usuarios finales. Desde este rol se pueden usar diferentes *tipos*, los cuales tendrán más o menos privilegios dependiendo de las reglas que se le apliquen en cada caso.

Pueden estar soportados otros roles, como `guest_r` y `xguest_r`, dependiendo de la distribución. Para consultar la lista de roles disponibles debemos hacer uso del comando `seinfo`:

```
# seinfo --role
Roles: 14
  auditadm_r
  dbadm_r
  ...
  unconfined_r
```

Limitar roles a través de usuarios

Un usuario SELinux es diferente a un usuario de Linux. Mientras que un usuario de Linux puede modificar su identidad a través de comandos como `sudo`, SELinux se asegura (por regla general) de que no haya cambios aún cuando haya modificado su usuario en Linux. SELinux se asegura de que se aplican los controles de acceso oportunos, aún cuando haya conseguido mayores privilegios en el sistema.

Un ejemplo de este tipo de control es **(UBAC) User-based Access Control**, que está disponbile en algunas distribuciones, y que evita que los usuarios accedan a los ficheros de otros usuarios SELinux aún cuando hagan uso de controles DAC para poder acceder a otros ficheros.

La característica más importante de los usuarios SELinux, es la restricción de uso de los diferentes roles. Una vez que a un usuario Linux se le asigna un usuario SELinux, no le será posible el uso de un rol que no le pertenezca.

Para los usuarios de SELinux, se usa el sufijo `_u`.

Control del flujo de información a través de sensibilidades

El cuarto elemento del contexto SELinux, la sensibilidad (no siempre está presente), es necesaria para el soporte de **(MLS) Multilevel Security**. Las etiquetas de sensibilidad permiten la clasificación de los recursos y la restricción de accesos a los mismos basándose en credenciales de seguridad. Están formadas por dos partes: un valor de confidencialidad (con prefijo `s`) y una categoría (con prefijo `c`).

El nivel de confidencialidad permite una clasificación de los recursos desde un nivel inferior `0` hasta el valor máximo que configure el administrador. Se puede comparar al modelo *Bell-LaPadula*.

Las categorías permiten que los recursos se etiqueten con una o más categorías, sobre las que poder aplicar control de acceso. De este modo, se puede implementar soporte para *multipropiedad* (por ejemplo, sistemas de hosting de aplicaciones para diferentes clientes), donde tendremos recursos y procesos de un cliente asignados a un conjunto de categorías, mientras que los procesos y recursos de otro cliente tendrán otro conjunto de categorías. Cuando un proceso no tiene ninguna categoría asignada, no podrá hacer nada con aquellos recursos que tienen otra categoría asignada.

Diseño y distribución de políticas

La habilitación de SELinux no activa automáticamente la restricción en los accesos. Si SELinux está activado y no puede encontrar una política, anulará su inicio. Esto es debido a que la *política* define el comportamiento del sistema. Estas políticas son distribuídas generalmente de forma compilada como módulos de políticas. Estos módulos son añadidos a un almacén de políticas y cargado en memoria para que SELinux pueda aplicar las reglas en el sistema.

Escritura de políticas SELinux

Las políticas de SELinux pueden ser registradas de tres modos diferentes (lenguajes):

- En un formato nativo de SELinux, y de fácil lectura por una persona.
- Extendiendo el formato nativo con macros M4 para facilitar el desarrollo de políticas.
- En **Common Intermediate Language (CIL)**, un formato legible por máquinas.

La mayoría de las distribuciones con soporte SELinux basan sus políticas en https://github.com/TresysTechnology/refpolicy/wiki, que es un conjunto de políticas de SELinux disponible como proyecto libre.

Tomando como ejemplo la regla del servidor web vista más arriba, y que permite a los procesos etiquetados como `httpd_t` tomar control de los puertos TCP etiquetados con `http_port_t`.

En el formato nativo de SELinux, se declara del siguiente modo:

```
allow httpd_t http_port_t : tcp_socket { name_bind };
```

Usando un estilo de referencias, esta regla es parte de la siguiente llamada a una macro:

```
corenet_tcp_bind_http_port(httpd_t)
```

En CIL, la regla se declara como sigue:

```
(allow httpd_t http_port_t (tcp_socket (name_bind)))
```

En la mayoría de las declaraciones, podemos destacar lo siguiente:

- El sujeto (responsable de la acción); en este caso, el conjunto de procesos etiquetados con el tipo `httpd_t`.
- El recurso objetivo o objeto; en este caso, el conjunto de sockets TCP (`tcp_socket`) etiquetados con el tipo `http_port_t`.
- La acción; en este caso, la acción de vincularse a un puerto (`name_bind`).
- El resultado que aplicará la política; en este caso, que la acción está permitida (`allow`).

Una política se suele escribir para una aplicación, o conjunto de aplicaciones. En el ejemplo anterior será parte de la política a aplicar a servidores web.

Se suelen crear tres ficheros para cada aplicación:

- Un fichero `.te`, que contiene la reglas de restricción por tipo.
- Un fichero `.if`, que contiene definiciones de intefaz y plantilla. Es similar a los ficheros *header* de otros lenguajes de programación.
- Un fichero `.fc`, que contiene las expresiones de contexto de fichero. Asignan etiquetas a los recursos en el sistema de ficheros.

Una vez finalizada la escritura de la política, debe ser empaquetada en un módulo.

Distribución de políticas a través de módulos

Inicialmente, SELinux utilizaba un esquema monolítico, y guardar todas las reglas posibles en un único fichero de políticas. Esta solución se hizo poco manejable, y se cambió por un modelo modular.

Con este planteamiento, los desarrolladores pueden diseñar políticas para una aplicación en particular, un rol, etc.

Con los espacios de usuario recientes de SELinux, los ficheros `*.pp` se consideran escritos en **high-level language (HLL)**, que a pesar del nombre son binarios.

En la mayoría de las distribuciones, SELinux guarda los módulos `*.pp` en el directorio `/usr/share/selinux`. Eses módulos están preparadas para ser activados por los administradores.

Cuando se activa un módulo, el comando `semodule` copia eses módulos en un directorio dedicado: `/etc/selinux/targeted/modules/active/modules`. Esta localización está determinada por la versión de SELinux, y en las más recientes se usa `/var/lib`. Cuando todos los módulos se encuentran en una única localización, se compilan, guardando el resultado en `/etc/selinux/targeted/policy/policy.30` y cargado en memoria.

Agrupación de módulos en un almacén de políticas

Un almacén sólo contiene una política, y sólo una puede estar activada al mismo tiempo. Los administradores puede cambiar de política, aunque sea necesario reiniciar el sistema.

La política activa en el sistema se puede consultar con el comando `sestatus`:

```
# sestatus | grep "Loaded policy name"
Loaded policy name:             targeted
```

El nombre de la política que se activará en el próximo reinicio se define en `/etc/selinux/config`, en el parámetro `SELINUXTYPE`.

CAPÍTULO 2

Capítulo 2

CAPÍTULO 3

Licencia

Licencia

Reconocimiento-CompartirIgual 3.0 España (CC BY-SA 3.0 ES)

Esto es un resumen inteligible para humanos (y no un sustituto) de la licencia, disponible en los idiomas siguientes: Aranés Asturiano Castellano Euskera Galego

Usted es libre de:

Compartir - copiar y redistribuir el material en cualquier medio o formato

Adaptar - remezclar, transformar y crear a partir del material para cualquier finalidad, incluso comercial

El licenciador no puede revocar estar libertades mientras cumpla los términos de la licencia.

Bajo las condiciones siguientes:

Reconocimiento - Debe reconocer adecuadamente la autoría[1], proporcionar un enlace a la licencia e indicar si se han realizado cambios[2]. Puede hacerlo de cualquier manera razonable, pero no de una manera que sugiera que tiene el apoyo del licenciador, o lo recibe por el uso que hace.

[1] Si se facilitan, debe proporcionar el nombre del creador y las partes a reconocer, un aviso de derechos de explotación, un aviso de licencia, una nota de descargo de responsabilidad y un enlace al material. Las licencias CC anteriores a la versión 4.0 también requieren que facilite el título del material, si le ha sido facilitado, y pueden tener algunas otras pequeñas diferencias.

[2] En la versión 3.0 y anteriores, la indicación de los cambios sólo se requiere si se crea una obra derivada.

Compartir Igual - Si remezcla, transforma o crea a partir del material, deberá difundir sus contribuciones bajo la misma licencia que el original[3].

No hay restricciones adicionales - No puede aplicar términos legales o medidas tecnológicas que legalmente restrinjan realizar aquello que la licencia permite.

[3] También puede utilizar una licencia compatible de la lista de https://creativecommons.org/compatiblelicenses.