# Más allá del chmod: Seguridad avanzada con SELinux y Auditd


**Alumno: Lautaro Uriel Emalhao**
**Profesor: Leonardo de - Matteis**
**Materia: Seguridad en Sistemas**
**Carrera: Licenciatura en Ciencias de la Computación**

# ¿Qué es el Control de Acceso Discrecional (DAC)?

El Control de Acceso Discrecional (Discretionary Access Control, DAC) es el modelo tradicional de seguridad utilizado en sistemas operativos como Linux y UNIX. En este modelo, el propietario de un recurso (como un archivo o directorio) tiene la capacidad de decidir quién puede acceder a él y qué operaciones puede realizar (lectura, escritura, ejecución).

Los permisos de acceso se gestionan mediante herramientas del sistema como `chmod` (cambiar permisos), `chown` (cambiar propietario) y `chgrp` (cambiar grupo). Por ejemplo, un usuario puede permitir que otros usuarios lean su archivo, pero no lo modifiquen, o restringir el acceso completamente.

Este modelo se denomina "discrecional" porque las decisiones de acceso dependen exclusivamente del criterio del propietario del recurso, y el sistema operativo no impone restricciones adicionales si los permisos lo permiten. Inclusive el usuario root tiene la capacidad de acceder a todos los recursos del sistema sin limitaciones impuestas por DAC.

Un ejemplo para visualizar el Control de Acceso Discrecional es:

```
-rw-r--r-- usuario usuario archivo.txt
```

En este caso:

- El propietario (usuario) puede leer y escribir el archivo.
- El grupo y otros usuarios pueden solo leerlo.
- Estos permisos pueden ser modificados libremente por el propietario del archivo.

DAC es sencillo y flexible, presenta limitaciones importantes en entornos donde se requiere alta seguridad, puesto que no previene que un proceso comprometido acceda a recursos sensibles si tiene permisos DAC adecuados, y además depende completamente de las decisiones de los usuarios, lo que puede generar configuraciones inseguras por error o desconocimiento.

# ¿Qué es el Control de Acceso Mandatorio (MAC)?

El Control de Acceso Mandatorio (Mandatory Access Control, MAC) es un modelo de seguridad en el que las reglas de acceso a los recursos del sistema son definidas y aplicadas por el sistema operativo, y no pueden ser modificadas ni ignoradas por los usuarios, incluyendo al usuario root.

A diferencia del modelo discrecional (DAC), donde el propietario de un archivo decide quién puede acceder y cómo, en MAC las políticas son centralizadas, estrictas y obligatorias. Estas políticas controlan quién puede acceder a qué recursos, bajo qué condiciones, y con qué permisos, sin depender de la voluntad de los usuarios.

Una característica distintiva del modelo MAC es que las políticas de seguridad no pueden ser anuladas por los usuarios ni por los programas que ejecutan, lo que proporciona una protección robusta contra accesos no autorizados, incluso en caso de que una cuenta privilegiada sea comprometida.

Este tipo de control es especialmente útil en entornos que requieren altos niveles de seguridad, como servidores críticos, infraestructuras gubernamentales o sistemas bancarios. Su implementación permite minimizar los efectos de ataques y prevenir que procesos o servicios maliciosos accedan a recursos fuera de su ámbito permitido.

En el ecosistema Linux, SELinux (Security-Enhanced Linux) es una implementación concreta del modelo MAC. Mediante el uso de políticas de seguridad y contextos de acceso, SELinux define con precisión qué operaciones puede realizar cada proceso o usuario, más allá de los permisos tradicionales del sistema de archivos.

# ¿Qué es SELinux?

Security-Enhanced Linux (SELinux) es una implementación de una arquitectura flexible de control de acceso mandatorio integrada en el kernel de Linux, diseñada para proporcionar una capa adicional de seguridad que restringe el acceso a recursos del sistema incluso para usuarios privilegiados.

Fue desarrollado originalmente por la NSA (Agencia de Seguridad Nacional de EE.UU.) en conjunto con la comunidad de software libre, con el objetivo de mejorar la seguridad de los sistemas Linux frente a fallos o accesos indebidos, siendo integrado a la rama principal del núcleo Linux desde la versión 2.6, en agosto de 2003.

Algunas distribuciones, como Red Hat Enterprise Linux, CentOS, Rocky Linux, GNU/Linux Fedora y Scientific Linux incorporan SELinux habilitado por defecto. Android incorpora SELinux de manera definitiva a partir de la versión 5.0 Lollipop. Anteriormente lo incorporó en la versión 4.3 Jelly Bean, pero en modo permisivo. En Android 4.4 KitKat se incluye activado parcialmente (para installd, netd, vold y zygote).

A diferencia del modelo tradicional de seguridad de Linux basado en DAC (Control de Acceso Discrecional), SELinux aplica políticas de seguridad estrictas que no pueden ser modificadas por los usuarios o procesos individuales, ni siquiera por el superusuario (root).

Estas políticas se basan en tres conceptos principales:

- ➢ Type Enforcement (TE): El mecanismo central de SELinux, que define qué tipos de procesos pueden interactuar con qué tipos de objetos (archivos, sockets, puertos, etc.).
- ➢ Role-Based Access Control (RBAC): Permite asociar roles a los usuarios y controlar qué dominios pueden usar.
- ➢ Multi-Level Security (MLS): Implementa niveles de seguridad jerárquicos para sistemas con clasificación de información (por ejemplo, confidencial, secreto, top secret).

SELinux utiliza contextos de seguridad etiquetados en cada archivo, proceso y recurso del sistema, y evalúa cada operación (como abrir, leer, ejecutar, etc.) en función de una política cargada.

Gracias a este enfoque, incluso si un servicio como Apache es comprometido, SELinux puede impedir que acceda a archivos o recursos que no estén expresamente permitidos en su política, reduciendo significativamente el alcance de un posible ataque.

## Arquitectura y paquetes

Security-Enhanced Linux (SELinux) se implementa como una arquitectura modular integrada al kernel de Linux, y se compone de varios niveles y componentes que colaboran para reforzar la seguridad del sistema, tanto desde el núcleo como desde el espacio de usuario.

El módulo central de SELinux se encuentra embebido en el kernel de Linux. Su función es interceptar cada intento de acceso a recursos del sistema (archivos, procesos, puertos, etc.) y

consultar una política de seguridad cargada para decidir si el acceso debe ser permitido o denegado. Para mejorar la eficiencia, utiliza una caché de decisiones conocida como Access Vector Cache (AVC).

La política de SELinux define las reglas de acceso entre sujetos (como procesos) y objetos (como archivos o dispositivos). Estas políticas están precompiladas y pueden ser personalizadas o ampliadas mediante módulos adicionales. Algunas de las políticas más comunes son:

- ➢ `targeted`: restringe únicamente ciertos servicios específicos.
- ➢ `mls`: aplica seguridad multinivel sobre todos los procesos y objetos.

SELinux utiliza etiquetas para identificar el contexto de seguridad de cada objeto y proceso. Estas etiquetas tienen la siguiente estructura: **`usuario:rol:tipo:nivel`**

Por ejemplo, un archivo podría tener el contexto **`system_u:object_r:httpd_sys_content_t:s0`**, lo que determina cómo puede interactuar con él un proceso determinado.

Por último, SELinux cuenta con una serie de utilidades que permiten administrar políticas, visualizar contextos, restaurar etiquetas por defecto y analizar eventos de auditoría. Estas herramientas son esenciales para la operación diaria y la solución de problemas relacionados con SELinux.

## Estados y modos

SELinux puede operar en diferentes modos que determinan su nivel de intervención en el sistema. Estos modos permiten adaptar el comportamiento de SELinux según el entorno (por ejemplo, producción, pruebas o desarrollo) y facilitan la transición entre una configuración permisiva y una estricta.

SELinux puede funcionar en tres modos diferentes: enforcing (reforzado), permissive (permisivo) y disabled (deshabilitado).

El modo enforcing es el modo predeterminado y recomendado para la mayoría de los sistemas, donde SELinux aplica activamente la política de seguridad cargada en el sistema, controlando y restringiendo el acceso conforme a las reglas definidas. En este modo, todas las acciones que no estén permitidas son bloqueadas.

En el modo permissive, SELinux no bloquea ninguna operación; sin embargo, continúa etiquetando los objetos del sistema y registra en los logs todas las acciones que, de estar en modo enforcing, habrían sido denegadas. Este modo es especialmente útil para el desarrollo y la depuración de políticas, pero no se recomienda su uso en entornos productivos debido a que no impide realmente accesos indebidos.

El modo disabled desactiva completamente SELinux, dejando de aplicar la política y además evitando que los objetos del sistema sean etiquetados con contextos de seguridad. Esto puede

complicar la reactivación de SELinux en el futuro, por lo que se desaconseja su uso salvo en casos excepcionales.

## Archivos y comandos básicos

La configuración principal de SELinux se encuentra en el archivo `/etc/selinux/config`, donde se definen dos variables fundamentales para su funcionamiento: `SELINUX` y `SELINUXTYPE`. La variable `SELINUX` determina el modo de operación de SELinux, pudiendo tomar los valores `enforcing` (modo activo), `permissive` (modo permisivo) o `disabled` (desactivado). Por su parte, `SELINUXTYPE` especifica la política de seguridad activa en el sistema, como por ejemplo la política `targeted` o `mls`. Para que los cambios realizados en este archivo tengan efecto, es necesario reiniciar el sistema.

Para verificar el estado actual de SELinux y el modo en que está operando, se utilizan comandos como `sestatus` y `getenforce`. El primero ofrece información detallada sobre si SELinux está habilitado, el modo en ejecución y la política aplicada; mientras que el segundo muestra de forma concisa el modo actual, que puede ser Enforcing, Permissive o Disabled.

El modo de operación puede modificarse temporalmente mediante el comando `setenforce`, empleando el valor `1` para establecer el modo `enforcing` y `0` para el modo `permissive`. Esta modificación es válida hasta el próximo reinicio del sistema.

SELinux utiliza contextos de seguridad para identificar y controlar los accesos a archivos y procesos. Para visualizar estos contextos, se emplea el comando `ls -Z`, que muestra la etiqueta de seguridad asociada a un archivo o proceso. Para modificar temporalmente el contexto de un archivo, se utiliza `chcon`, mientras que el comando `restorecon` permite restaurar el contexto original definido por la política de seguridad.

Además, SELinux ofrece herramientas para la gestión avanzada de su comportamiento mediante booleanos de política, que son opciones configurables en tiempo de ejecución. El comando `semanage boolean -l` lista todos los booleanos disponibles, y mediante `semanage boolean -m --on <nombre_booleano>` es posible activar alguno de ellos para ajustar las reglas de acceso según las necesidades específicas del sistema.

Finalmente, SELinux registra en los archivos de auditoría, generalmente en `/var/log/audit/audit.log`, todas las acciones denegadas por la política. Para analizar estos registros y comprender los motivos de los bloqueos, se pueden emplear herramientas como `ausearch` y `audit2why`. Esta última facilita la interpretación de los eventos y, junto con `audit2allow`, permite generar excepciones que habiliten acciones previamente bloqueadas, siempre con la debida precaución.

Como dato de color, en Red Hat Enterprise Linux, es posible poner dominios individuales en modo permisivo mientras el sistema se ejecuta en modo forzoso. Por ejemplo, para hacer que el dominio httpd_t sea permisivo: `# semanage permissive -a httpd_t`

Hay que tener en cuenta que los dominios permisivos son una herramienta poderosa, pero que puede comprometer la seguridad del sistema si no se aplica con cuidado. Red Hat recomienda utilizar los dominios permisivos con precaución, por ejemplo, al depurar un escenario específico.

Cuando un sistema está en modo permisivo con SELinux, pueden empezar a aparecer etiquetas mal asignadas en distintos archivos del sistema. Esto pasa porque, aunque SELinux no bloquea nada en ese modo, sigue funcionando en segundo plano y permite que se creen objetos con etiquetas que no siempre son correctas. Además, si el sistema estuvo en modo deshabilitado antes, los archivos que se crearon en ese tiempo ni siquiera tienen etiquetas SELinux, lo que puede generar conflictos cuando después se pasa al modo enforcing, que sí aplica la política con todas las reglas.

Para evitar estos problemas, lo que se hace es re-etiquetar automáticamente el sistema de archivos cuando se cambia de disabled a permissive o enforcing. Esto asegura que todos los archivos tengan el contexto correcto para que SELinux funcione como se espera. Si estamos en modo permisivo y queremos forzar ese re-etiquetado en el próximo reinicio, lo que hay que hacer es ejecutar como root el comando: `fixfiles -F onboot.` Este comando crea un archivo llamado `/.autorelabel`, que le indica al sistema que al reiniciar debe volver a etiquetar todo correctamente.

## Confinamiento de usuarios

SELinux no solo controla el acceso entre procesos y archivos, sino que también puede confinar a los usuarios del sistema, limitando lo que pueden hacer, incluso si tienen privilegios elevados. Esta característica permite asignar dominios de seguridad específicos a cada usuario, estableciendo políticas que restringen sus acciones dentro del sistema operativo.

Hay dos tipos de usuarios desde la perspectiva de SELinux:

- **Usuarios no confinados (unconfined)**
  - Por defecto, los usuarios del sistema (como root o usuarios estándar) suelen operar en el dominio `unconfined_t`, lo que significa que no están sujetos a restricciones estrictas por parte de SELinux. Pueden ejecutar comandos y acceder a recursos sin que se apliquen controles detallados.
- **Usuarios confinados (confined)**
  - Un usuario confinado opera en un dominio restringido (como `staff_t, user_t` o `guest_t`), y está sujeto a una política específica que limita sus permisos. Por ejemplo, un usuario confinado puede estar impedido de acceder a ciertos archivos del sistema, cambiar configuraciones sensibles o ejecutar binarios no autorizados.

Para ver la asignación de usuarios de SELinux en el sistema, se utiliza el comando `semanage login -l` como root. Otros comandos útiles para realizar las tareas de gestión de confinamiento son:

- ➢ `semanage login -a -s user_u juan`
  - Este comando agrega un nuevo mapeo de usuario, asociando el usuario de Linux (juan) con el usuario SELinux (user_u). El usuario SELinux define el nivel de confinamiento y qué roles y dominios puede usar.
- ➢ `semanage login -m -s staff_u juan`
  - Este comando modifica un mapeo ya existente. En este caso, se está cambiando el usuario SELinux asignado a juan, pasándolo de, por ejemplo, user_u a staff_u.
- ➢ `restorecon -R -v /home/juan`
  - Este comando restaura los contextos de seguridad SELinux en los archivos y carpetas bajo el directorio /home/juan, aplicando el contexto correcto según las políticas.
  - -R: lo hace recursivo.
  - -v: muestra en pantalla qué está haciendo.
  - Es importante ejecutarlo después de cambiar el mapeo del usuario, para que los archivos del usuario usen las nuevas etiquetas correctas.

Un cuadro con las diferentes opciones vistas en los comandos anteriormente mencionados son:

| **Opción** | **Significado** | **Para qué sirve** |
|---|---|---|
| `-l` | **list** | Lista los usuarios del sistema y sus perfiles SELinux asignados |
| `-a` | **add** | Agrega un nuevo mapeo entre un usuario de Linux y un perfil SELinux |
| `-s` | **SELinux user** | Indica qué perfil SELinux (como `user_u`, `staff_u`, etc.) se quiere asignar |
| `-m` | **modify** | Modifica un mapeo ya existente (por ejemplo, cambiar el perfil SELinux asignado) |
| `-R` | **recursive** | Aplica la acción de forma recursiva sobre subdirectorios y archivos (`restorecon`) |
| `-v` | **verbose** | Muestra en pantalla qué está haciendo el comando (`restorecon`, `fixfiles`, etc.) |

Es posible listar los usuarios de SELinux con el comando `seinfo`, proporcionado por el paquete `setools-console`, que no está instalado por defecto.

Aunque un usuario de Linux no esté confinado por SELinux (es decir, esté en el dominio unconfined_t), si ejecuta una aplicación que está definida por la política de SELinux para funcionar en su propio dominio confinado, esa aplicación sigue estando restringida por ese dominio. Esto significa que, aunque el usuario no esté limitado directamente, la aplicación sí lo está, lo que representa una capa extra de seguridad. En caso de que esa aplicación tenga una vulnerabilidad, SELinux puede limitar el daño que podría causar, porque el proceso no tiene total libertad.

Algo similar ocurre con los usuarios que sí están confinados. Cada usuario confinado tiene su propio dominio, con restricciones claras definidas en la política. Además, puede haber una transición a un dominio de destino más específico, dependiendo del tipo de tarea o aplicación que esté ejecutando. Lo importante es que los usuarios confinados operan siempre bajo límites bien definidos, y SELinux les asigna privilegios especiales según el rol que tengan en el sistema.

## Seguridad Multinivel

SELinux permite aplicar un modelo de seguridad mucho más estricto llamado MLS (Multi-Level Security). Este modelo se basa en niveles de sensibilidad, donde tanto los archivos como los procesos del sistema tienen etiquetas que indican qué tan sensible es la información y quién puede acceder a ella.

A diferencia de las políticas comunes (como la targeted), donde se controla qué procesos pueden hacer qué cosas, con MLS se suma una dimensión más: el nivel de clasificación. Por ejemplo, un archivo puede estar marcado como confidencial y otro como secreto, y SELinux puede impedir que un proceso en un nivel más bajo acceda a los archivos de un nivel superior. Esto se basa en el modelo Bell-LaPadula, que sigue reglas como:

- **No leer arriba**: un proceso no puede leer información de un nivel más alto al suyo.
- **No escribir abajo**: un proceso no puede escribir en un archivo de nivel más bajo (para evitar filtraciones).

Este tipo de configuración se utiliza en entornos donde es necesario aislar estrictamente la información, como en sistemas militares o gubernamentales. Para que funcione, el sistema debe estar configurado con una política MLS activa, y tanto el kernel como las herramientas de SELinux deben tener soporte para ese modelo.

No es algo que venga activado por defecto en todas las distribuciones, pero demuestra el nivel de control y granularidad que SELinux puede ofrecer cuando se necesita un entorno verdaderamente seguro.

Nota:

Aunque la mayoría de los kernels modernos incluyen el soporte básico para SELinux, no todos vienen compilados con las extensiones necesarias para manejar MLS. Por ejemplo, las versiones de kernel que distribuyen Red Hat Enterprise Linux, Rocky Linux, AlmaLinux y Fedora sí traen habilitado el código para rangos MLS/MCS, de modo que permiten cargar la política mls sin modificaciones. En cambio, los kernels estándar de distribuciones como Debian, Ubuntu o Arch suelen compilarse solo con el soporte "básico" de SELinux: es posible activar la política targeted, pero si se intenta cargar mls vas a obtener errores porque el kernel no entiende los campos de nivel y categoría. En esos casos, la única forma de usar MLS es recompilar el kernel con las opciones correspondientes (o instalar uno alternativo que ya las incluya). Así que, en la práctica, un sistema puede ejecutar SELinux perfectamente y aun así no ser capaz de aplicar seguridad multinivel.

## Políticas de seguridad personalizadas

Una política de seguridad en SELinux es un conjunto de reglas que define cómo pueden interactuar los procesos y los recursos del sistema. Estas políticas son cargadas en el kernel mediante herramientas de espacio de usuario, y son las que determinan qué operaciones están permitidas. De manera predeterminada, SELinux bloquea todos los accesos, y solo permite aquellos que están explícitamente autorizados por las reglas de la política activa.

Cada regla establece una relación concreta entre un proceso y un recurso. Por ejemplo:

```
ALLOW apache_process apache_log:FILE READ;
```

Esta línea indica que el proceso etiquetado como apache_process puede leer un archivo con la etiqueta apache_log. Estas etiquetas no se basan en el nombre del archivo o el usuario del sistema operativo, sino en atributos de seguridad que SELinux asigna a los objetos.

Las etiquetas se almacenan como atributos extendidos del sistema de archivos, y pueden visualizarse con comandos como:

```
$ ls -Z /etc/passwd

system_u:object_r:passwd_file_t:s0 /etc/passwd
```

Donde:

- `system_u` representa al usuario SELinux,
- `object_r` al rol asignado,
- `passwd_file_t` es el tipo de recurso (tipo de archivo),
- `s0` corresponde al nivel de seguridad.

La política que viene por defecto con SELinux (selinux-policy) incluye reglas para los servicios y aplicaciones integradas en Red Hat Enterprise Linux 8. Sin embargo, si instalo una aplicación que no está contemplada por esa política, SELinux no aplicará restricciones sobre ella. Para

resolver esto, se pueden crear módulos de política personalizados, que permiten añadir nuevas reglas y extender el comportamiento de SELinux.

Una herramienta útil para este propósito es sepolicy, que permite generar políticas personalizadas basadas en una aplicación o servicio determinado. Además, los scripts que genera incluyen el uso del comando restorecon, que sirve para corregir el etiquetado de archivos, asegurando que cada uno tenga el contexto de seguridad adecuado según la política.

## Políticas personalizadas para contenedores

Red Hat Enterprise Linux 8 incorpora la herramienta udica, diseñada específicamente para generar políticas SELinux personalizadas para contenedores. Esta herramienta permite controlar de forma precisa el acceso que un contenedor tiene a los recursos del sistema anfitrión, como el almacenamiento, los dispositivos o la red. Al hacerlo, se mejora la seguridad general del entorno y se facilita el cumplimiento de normativas.

A diferencia de la política genérica (container_t) aplicada por defecto a todos los contenedores, udica permite crear un dominio SELinux dedicado para cada contenedor, limitando su comportamiento con base en lo que realmente necesita. Esto permite aplicar el principio de mínimo privilegio sin desactivar SELinux.

El proceso de generación de una política SELinux con udica consta de tres etapas principales:

- Análisis del contenedor:
    - Se inspecciona un archivo JSON del contenedor (por ejemplo, generado por podman inspect) para detectar:
        - Capacidades del kernel requeridas
        - Puertos de red utilizados
        - Directorios montados desde el host

- Asignación de reglas SELinux:
    - Con base en la información obtenida, udica selecciona las reglas adecuadas. Por ejemplo:
        - Si el contenedor requiere acceso al directorio /home, se incorpora un bloque que permite ese acceso.
        - Si se detecta uso de un puerto específico, udica consulta las bibliotecas de SELinux para aplicar la etiqueta correcta.

- Generación de la política final:
    - La política se construye utilizando bloques heredados del lenguaje intermedio común de SELinux (CIL). Cada bloque corresponde a una acción (como acceder a logs o comunicarse con Xserver), y se combinan para formar una política compacta y precisa.

El resultado es un módulo de política listo para instalar con semodule, que confina al contenedor dentro de su propio dominio SELinux, sin interferir con el resto del sistema o con otros contenedores.

## Automatización y despliegue uniforme de configuraciones en múltiples sistemas

Mantener una configuración SELinux coherente en varios sistemas es crucial para garantizar una política de seguridad uniforme y facilitar el cumplimiento normativo en entornos empresariales. Red Hat Enterprise Linux permite exportar e importar configuraciones SELinux, incluyendo módulos de política personalizados, etiquetas, booleans y archivos de configuración. Estas configuraciones pueden aplicarse manualmente o automatizarse para lograr despliegues rápidos y consistentes.

Para simplificar esta automatización, se puede utilizar Ansible, una herramienta de gestión de configuraciones y orquestación que permite ejecutar tareas de forma repetible y escalable en múltiples servidores. La versión open source de Ansible es gratuita y puede instalarse fácilmente en distribuciones como Rocky Linux, facilitando el despliegue y mantenimiento de configuraciones SELinux a gran escala.

Por otro lado, Red Hat Ansible Automation Platform es la versión comercial que ofrece soporte técnico empresarial, acceso a actualizaciones certificadas y funcionalidades adicionales como Ansible Tower para gestión centralizada. Esta suscripción es ideal para organizaciones que requieren soporte profesional y funcionalidades avanzadas, aunque para muchos casos la versión open source de Ansible es suficiente.

Así, combinando las herramientas de exportación e importación de SELinux con la automatización de Ansible, se puede asegurar que todos los sistemas ejecuten una configuración SELinux consistente, reduciendo errores, aumentando la seguridad y facilitando la administración en entornos de múltiples hosts.

# Auditd e integración con SELinux

El demonio auditd es el componente en espacio de usuario del sistema de auditoría de Linux. Su función principal es registrar eventos relevantes desde el punto de vista de la seguridad, como accesos a archivos sensibles, cambios en permisos, modificaciones de usuarios o denegaciones generadas por SELinux. Toda esta información se guarda en archivos de log, generalmente ubicados en /var/log/audit/audit.log, permitiendo un control detallado sobre lo que ocurre en el sistema.

Para consultar y analizar estos registros, se utilizan herramientas como ausearch (para búsquedas detalladas de eventos) o aureport (para generar resúmenes y reportes). Esto resulta fundamental para investigar incidentes, hacer auditorías de cumplimiento, o incluso para depurar el comportamiento de políticas SELinux.

La configuración de las reglas que determinan qué eventos deben auditarse se realiza mediante el comando auditctl. Al iniciarse el sistema, auditctl lee automáticamente las reglas definidas en el archivo /etc/audit/audit.rules. Estas reglas pueden especificar, por ejemplo, que se audite cada vez que se accede a un determinado archivo, que se ejecute un comando en particular, o que se cambien privilegios de usuarios.

Por otro lado, el comportamiento general del demonio auditd (como el tamaño máximo de los archivos de log, la acción a tomar si el disco se llena, o la ubicación de los archivos de auditoría) se configura en el archivo /etc/audit/auditd.conf. Esto permite a los administradores ajustar la auditoría a las necesidades específicas del sistema o del entorno de seguridad en el que se trabaja.

En conjunto con SELinux, auditd juega un rol clave: cada vez que SELinux deniega una operación por razones de política, dicha denegación se registra como un evento AVC (Access Vector Cache) que es recogido por auditd. De este modo, se obtiene una trazabilidad completa que no solo informa qué acción fue bloqueada, sino también quién la intentó, bajo qué contexto y qué tipo de recurso estaba involucrado.

Además de su configuración básica, el demonio auditd permite una personalización avanzada que mejora tanto la seguridad como la eficiencia en entornos Linux. Según Red Hat (s.f.), una de las mejores prácticas es establecer reglas específicas y persistentes que limiten los eventos auditados a aquellos realmente relevantes desde el punto de vista de la seguridad. Esto evita una sobrecarga innecesaria de logs y mejora el análisis posterior. Por ejemplo, se recomienda auditar accesos a archivos críticos como /etc/shadow o comandos privilegiados como passwd y sudo.

También es importante definir correctamente los límites de tamaño de los archivos de log, así como su rotación, para evitar que llenen el disco. Esto se configura en el archivo /etc/audit/auditd.conf. Allí se pueden establecer parámetros como `max_log_file`, `max_log_file_action` y `space_left_action`, que determinan el tamaño máximo del archivo, qué hacer cuando se alcanza ese límite y qué acción tomar cuando queda poco espacio en disco, respectivamente.

Por otro lado, Red Hat destaca la importancia de habilitar la protección contra la manipulación del sistema de auditoría, activando el modo "inmutable". Este modo impide modificar las reglas de auditoría una vez aplicadas, incluso por usuarios con privilegios, reforzando así la integridad del sistema.

Este enfoque convierte al sistema de auditoría no solo en un mecanismo de registro, sino también en una herramienta clave de defensa activa frente a posibles compromisos o abusos del sistema.

# Actividad Práctica - Registro de logs y cambio de políticas

En el servidor Apache usado en la actividad de hardening de la materia, dentro del sistema operativo Rocky Linux, se configuró SELinux para que esté en modo Enforcing desde el comienzo. En la actividad práctica sobre SELinux y Auditd, se van a utilizar ambas herramientas y cambiar algunos permisos.
Primero hay que revisar los logs, por lo que se utiliza la herramienta de ausearch y audit2why para tener una explicación de lo que sucede. El comando a utilizar es:
`ausearch -m AVC -ts recent | audit2why`

1. `Ausearch`
    a. Herramienta que permite buscar eventos registrados por SELinux y el sistema de auditoría en el archivo `/var/log/audit/audit.log`.
2. `-m AVC`
    a. Filtro por tipo de mensaje.
    b. Muestra únicamente los eventos AVC (Access Vector Cache), que son aquellos donde SELinux denegó un acceso.
3. `-ts recent`
    a. Filtra los eventos a partir del momento más reciente.
    b. Solo se muestran los eventos actuales (evita mostrar eventos antiguos).
4. `| (pipe)`
    a. Redirige la salida de ausearch como entrada al siguiente comando (audit2why), formando una cadena de análisis automatizado.
5. `Audit2why`
    a. Interpreta los eventos AVC.
    b. Explica en lenguaje claro por qué SELinux bloqueó la acción.

El primer log que se ve es:

```
type=AVC msg=audit(1749948906.047:486): avc:  denied  { name_connect } for
pid=1068  comm="php-fpm"  dest=80  scontext=system_u:system_r:httpd_t:s0
tcontext=system_u:object_r:http_port_t:s0 tclass=tcp_socket permissive=0

        Was caused by:
        One of the following booleans was set incorrectly.
        Description:
        Allow httpd to can network connect

        Allow access by executing:
        # setsebool -P httpd_can_network_connect 1
        Description:
        Allow httpd to graceful shutdown
```

```
        Allow access by executing:
        # setsebool -P httpd_graceful_shutdown 1
        Description:
        Allow httpd to can network relay

        Allow access by executing:
        # setsebool -P httpd_can_network_relay 1
        Description:
        Allow nis to enabled

        Allow access by executing:
        # setsebool -P nis_enabled 1
```

PHP-FPM (FastCGI Process Manager) es una implementación alternativa de PHP que permite manejar de manera eficiente aplicaciones web bajo servidores como Apache o Nginx. Su principal función es gestionar procesos PHP de forma separada, permitiendo un mejor rendimiento, control de procesos y consumo de recursos.

En lugar de que cada solicitud genere un nuevo proceso, PHP-FPM mantiene procesos persistentes que responden a múltiples solicitudes, lo que reduce la sobrecarga y mejora la velocidad. Esto es especialmente útil en entornos de producción con alta carga o servicios como WordPress, donde se requiere una respuesta rápida y continua del backend PHP.

En el contexto de este trabajo, PHP-FPM es utilizado por el servidor Apache para ejecutar scripts PHP del sitio web WordPress. El servicio corre bajo su propio contexto de seguridad (httpd_t cuando SELinux está activado), y puede ser afectado por políticas de acceso que impiden, por ejemplo, conexiones de red o lectura de ciertos archivos. Por esta razón, es fundamental revisar los permisos de SELinux asociados a PHP-FPM, ya que un mal ajuste puede interrumpir la funcionalidad del sitio o de servicios asociados como bases de datos o paneles de monitoreo como Uptime Kuma.

Por ello, se procede a cambiar los permisos de esta manera: para permitir que httpd (y por ende php-fpm, que está bajo su contexto) realice conexiones de red salientes (por ejemplo, a un servidor externo o incluso a localhost:80), hay que habilitar el booleano:

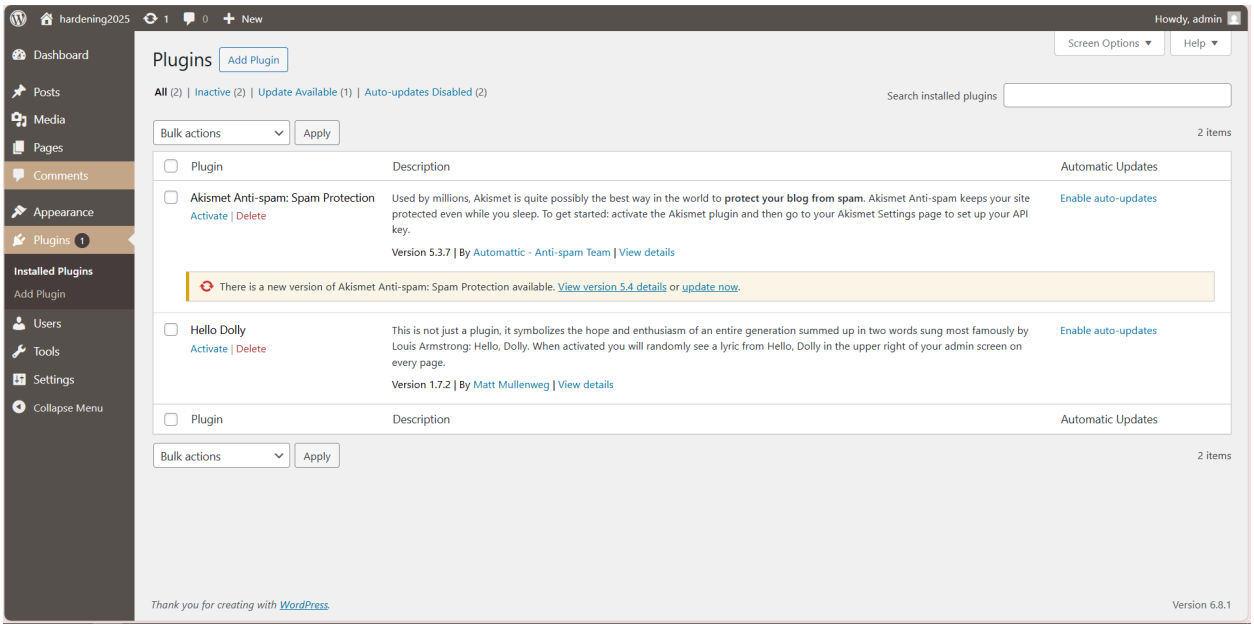
```
setsebool -P httpd_can_network_connect 1
```

Del lado del servidor, es posible confirmar que los permisos se cambiaron:

```
[root@localhost ~]# [ 7167.896506] SELinux:  Converting 389 SID table entries...
[ 7167.920096] SELinux:  policy capability network_peer_controls=1
[ 7167.920132] SELinux:  policy capability open_perms=1
[ 7167.920186] SELinux:  policy capability extended_socket_class=1
[ 7167.920268] SELinux:  policy capability always_check_network=0
[ 7167.920348] SELinux:  policy capability cgroup_seclabel=1
[ 7167.920449] SELinux:  policy capability nnp_nosuid_transition=1
[ 7167.920500] SELinux:  policy capability genfs_seclabel_symlinks=1
```

```
[root@hardening ~]# getsebool httpd_can_network_connect
httpd_can_network_connect --> on
```

Gracias al cambio, ahora es posible descargar actualizaciones para Wordpress.




Otro log que surge a raíz del cambio es:

```
type=AVC msg=audit(1749951496.923:743): avc: denied { write } for
pid=1069 comm="php-fpm" name="html" dev="dm-3" ino=8460822
scontext=system_u:system_r:httpd_t:s0
tcontext=system_u:object_r:httpd_sys_content_t:s0 tclass=dir permissive=0
        Was caused by:
        The boolean httpd_unified was set incorrectly.
        Description:
        Allow httpd to unified

        Allow access by executing:
        # setsebool -P httpd_unified 1
```

Siguiendo el consejo dado, se ingresa el comando `setsebool -P httpd_unified 1` para que PHP-FPM (y otros procesos httpd_t) escriban en esos directorios con contexto httpd_sys_content_t. Esto es útil dado que el servidor web o PHP necesita escribir dentro de su árbol de archivos y cambiar el permiso no presenta riesgos de seguridad.

Cuando está habilitado, este booleano permite que el dominio httpd_t (Apache) tenga acceso completo a todos los tipos asociados a httpd (es decir, puede ejecutar, leer o escribir archivos con tipo httpd_sys_content_t). Sin embargo, cuando está deshabilitado, se aplica una separación entre el contenido web de solo lectura, escritura o ejecución.

Desactivar este booleano proporciona un nivel adicional de seguridad, pero agrega una carga administrativa extra, ya que se debe etiquetar individualmente cada script u otro contenido web según el tipo de acceso (lectura, escritura o ejecución) que necesite.

Al igual que antes, vemos:

```
[11494.926666] SELinux:  Converting 390 SID table entries...
[11494.940784] SELinux:  policy capability network_peer_controls=1
[11494.940850] SELinux:  policy capability open_perms=1
[11494.940975] SELinux:  policy capability extended_socket_class=1
[11494.944678] SELinux:  policy capability always_check_network=0
[11494.944911] SELinux:  policy capability cgroup_seclabel=1
[11494.945156] SELinux:  policy capability nnp_nosuid_transition=1
[11494.945358] SELinux:  policy capability genfs_seclabel_symlinks=1
```

Luego, ya no aparecen los logs de PHP-FPM.

Otro log que aparece reiteradas veces es el siguiente:

```
type=AVC msg=audit(1749955668.217:1027): avc:  denied  { read } for  pid=1
comm="systemd"          name="pm2.pid"          dev="dm-0"          ino=621323
scontext=system_u:system_r:init_t:s0
tcontext=system_u:object_r:admin_home_t:s0 tclass=file permissive=0

        Was caused by:
                Missing type enforcement (TE) allow rule.

                  You can use audit2allow to generate a loadable module to
allow this access.
```

¿Qué significa? El proceso systemd (PID 1) intentó leer el archivo pm2.pid. Ese archivo está etiquetado con el tipo admin_home_t, que indica que se encuentra en el home de un administrador. El contexto del proceso es init_t (el tipo de systemd), que no tiene permiso para leer archivos etiquetados como admin_home_t. SELinux bloquea esa acción y además avisa: **`Missing type enforcement (TE) allow rule`**. Esto significa que no existe una regla SELinux que permita explícitamente esta acción, y que no se puede resolver simplemente con un booleano como en los casos anteriores.

pm2.pid es el manejador de procesos principal de pm2, que es el que está ejecutando Uptime Kuma.

Hay cuatro posibles soluciones, lo que no significa que todas sean válidas:

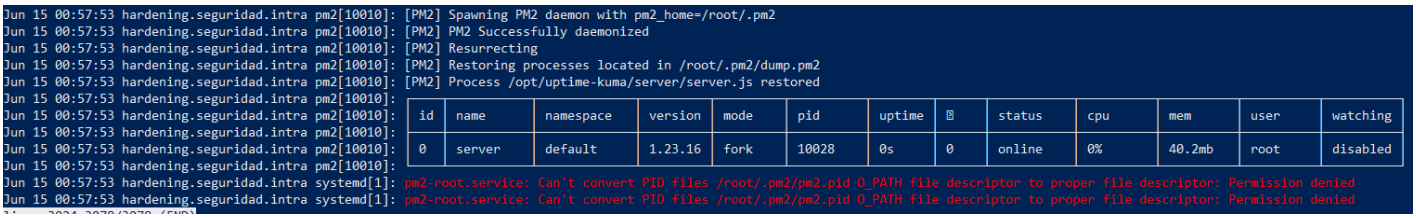
1. Mover el archivo a un lugar apropiado para evitar conflictos con accesos a home no autorizados.
2. Cambiar el contexto SELinux del archivo con `chcon` o `semanage`

3. Crear una política personalizada con audit2allow como indica el log. Esto no es recomendado del todo debido a que cambia de manera global las políticas de SELinux al permitir que systemd lea el archivo aunque no pueda. Se puede hacer de la siguiente manera:
    a. `grep pm2.pid /var/log/audit/audit.log | audit2allow -M allow_pm2`
    b. `semodule -i allow_pm2.pp`
    c. `.pp` significa "Policy Package". Es un archivo binario generado por la herramienta audit2allow o por compiladores de políticas como checkmodule/semodule_package. Contiene la regla de permiso específica que se puede instalar en el sistema con semodule.
4. Ignorar el log porque no es crítico ni afecta el funcionamiento del programa.

Para la demostración, se van a probar todas las opciones posibles para comparar qué efectos tienen cada una. La modificación del contexto de SELinux en archivos o directorios específicos, mediante herramientas como chcon (temporal) o semanage fcontext (permanente), se considera una solución aceptable y segura en la mayoría de los casos. Sin embargo, se la clasifica como una opción intermedia por los siguientes motivos:

- Cambiar el contexto SELinux es efectivo, pero debe hacerse correctamente. Usar el tipo incorrecto puede llevar a nuevos errores o incluso a brechas de seguridad si se otorgan permisos demasiado amplios. Por ejemplo, etiquetar con public_content_rw_t a un archivo que no debe ser escrito desde el exterior puede permitir acciones no deseadas.
- Puede generar inconsistencias si no se hace de forma permanente.
- Reubicar archivos en rutas estándar con contextos ya definidos por la política (como /var/run, /var/lib, etc.) es considerado más ordenado y menos propenso a errores a futuro. Cambiar contextos, en cambio, introduce excepciones personalizadas que, si no se documentan bien, pueden dificultar la auditoría.

Primero, se cambió el archivo con el error a una carpeta en el directorio /var y se le dio contexto var_t se probaron los cambios de manera temporal para ver qué efectos produce, pero tuvieron que deshacerse porque trajo problemas.




Se tuvo que reiniciar manualmente Uptime Kuma con el comando: `pm2 start /opt/uptime-kuma/server/server.js --name uptime-kuma`

Alinear la configuración de PM2 y systemd con las prácticas estándar del sistema operativo, asegurando que los archivos de estado y PID se ubiquen en rutas compatibles con la política de SELinux, evita conflictos de permisos y mantiene la seguridad del sistema. Sin embargo, para llevar a cabo esta configuración se deben cambiar muchos archivos de lugar y

configuraciones ya preestablecidas. Cambiar los directorios de lugar, dada la cantidad de dependencias de archivos que tiene el programa, no es una tarea sencilla.

El problema podría deberse a un error de diseño, porque la carpeta en la que quedó alojada PM2 es /root/.pm2. Archivos como .pid, si van a ser accedidos por systemd, deberían estar en lugares como /run/pm2/, /var/run/pm2/, etc., que tienen contexto SELinux adecuado (var_run_t) y no bloquean a procesos del sistema.

De todas formas, se intentó cambiar la ubicación de pm2, pero no solo sigue marcando el mismo error sino que dejó de funcionar Uptime Kuma:

```
[root@hardening ~]# mkdir -p /var/lib/pm2
[root@hardening ~]# mv /root/.pm2 /var/lib/pm2
[root@hardening ~]# chown -R root:root /var/lib/pm2
[root@hardening ~]# chmod -R 700 /var/lib/pm2
[root@hardening ~]# semanage fcontext -a -t var_lib_t "/var/lib/pm2(/.*)?"
[root@hardening ~]# restorecon -Rv /var/lib/pm2
Relabeled /var/lib/pm2/.pm2 from unconfined_u:object_r:admin_home_t:s0 to unconfined_u:object_r:var_lib_t:s0
Relabeled /var/lib/pm2/.pm2/module_conf.json from unconfined_u:object_r:admin_home_t:s0 to unconfined_u:object_r:var_lib_t:s0
Relabeled /var/lib/pm2/.pm2/touch from unconfined_u:object_r:admin_home_t:s0 to unconfined_u:object_r:var_lib_t:s0
Relabeled /var/lib/pm2/.pm2/pm2.log from unconfined_u:object_r:admin_home_t:s0 to unconfined_u:object_r:var_lib_t:s0
Relabeled /var/lib/pm2/.pm2/pm2.pid from system_u:object_r:admin_home_t:s0 to system_u:object_r:var_lib_t:s0
Relabeled /var/lib/pm2/.pm2/pub.sock from system_u:object_r:admin_home_t:s0 to system_u:object_r:var_lib_t:s0
Relabeled /var/lib/pm2/.pm2/dump.pm2 from unconfined_u:object_r:admin_home_t:s0 to unconfined_u:object_r:var_lib_t:s0
Relabeled /var/lib/pm2/.pm2/dump.pm2.bak from unconfined_u:object_r:admin_home_t:s0 to unconfined_u:object_r:var_lib_t:s0
Relabeled /var/lib/pm2/.pm2/rpc.sock from system_u:object_r:admin_home_t:s0 to system_u:object_r:var_lib_t:s0
Relabeled /var/lib/pm2/.pm2/logs from unconfined_u:object_r:admin_home_t:s0 to unconfined_u:object_r:var_lib_t:s0
Relabeled /var/lib/pm2/.pm2/logs/server-out.log from unconfined_u:object_r:admin_home_t:s0 to unconfined_u:object_r:var_lib_t:s0
Relabeled /var/lib/pm2/.pm2/logs/server-error.log from unconfined_u:object_r:admin_home_t:s0 to unconfined_u:object_r:var_lib_t:s0
Relabeled /var/lib/pm2/.pm2/pids from unconfined_u:object_r:admin_home_t:s0 to unconfined_u:object_r:var_lib_t:s0
Relabeled /var/lib/pm2/.pm2/modules from unconfined_u:object_r:admin_home_t:s0 to unconfined_u:object_r:var_lib_t:s0
[root@hardening ~]# nano /etc/systemd/system/pm2-root.service
[root@hardening ~]# systemctl daemon-reload
[root@hardening ~]# systemctl restart pm2-root
[root@hardening ~]# pm2 status
[PM2] Spawning PM2 daemon with pm2_home=/root/.pm2
[PM2] PM2 Successfully daemonized

 id  name  mode  ↺  status  cpu  memory

[root@hardening ~]# pm2 start
[PM2][ERROR] File ecosystem.config.js not found
[root@hardening ~]# pm2 start /opt/uptime-kuma/server/server.js --name uptime-kuma
[PM2] Starting /opt/uptime-kuma/server/server.js in fork_mode (1 instance)
[PM2] Done.

 id  name  mode  ↺  status  cpu  memory
 0  uptime-kuma  fork  0  online  0%  41.4mb

[root@hardening ~]# pm2 status

 id  name  mode  ↺  status  cpu  memory
 0  uptime-kuma  fork  17  online  100%  73.3mb

[root@hardening ~]# ss -tuln | grep 3001
[root@hardening ~]# ausearch -m AVC -ts recent | audit2why
type=AVC msg=audit(1752415260.534:149): avc:  denied  { read } for  pid=1 comm="systemd" name="pm2.pid" dev="dm-0" ino=621323 scontext=system_u:system_r:init_t:s0 tcontext=system_u:object_r:admin_home_t:s0 tclass=file permissive=0

        Was caused by:
                Missing type enforcement (TE) allow rule.

                You can use audit2allow to generate a loadable module to allow this access.

type=AVC msg=audit(1752415260.534:150): avc:  denied  { read } for  pid=1 comm="systemd" name="pm2.pid" dev="dm-0" ino=621323 scontext=system_u:system_r:init_t:s0 tcontext=system_u:object_r:admin_home_t:s0 tclass=file permissive=0
```

Otra opción sería agregar una nueva regla de SELinux que solucione el error. Esto se hace de manera sencilla, de esta forma:

```
[root@hardening ~]# grep pm2.pid /var/log/audit/audit.log | audit2allow -M allow_pm2
******************** IMPORTANT ***********************
To make this policy package active, execute:

semodule -i allow_pm2.pp

[root@hardening ~]# semodule -i allow_pm2.pp
[root@hardening ~]# systemctl daemon-reload
[root@hardening ~]# semodule -l
abrt
accountsd
acct
afs
afterburn
aiccu
aide
ajaxterm
allow_pm2
```

A pesar de haber aplicado una nueva política a partir de los logs de auditd, sigue persistiendo el mismo problema.

Después de todo lo visto, la última opción es ignorar el error dado que Uptime Kuma funciona correctamente. Las alertas vistas no impactan en el correcto funcionamiento del servicio gestionado por pm2. Este tipo de alertas suelen ser falsos positivos o ruido en los logs, reflejando bloqueos que no afectan la operatividad ni la seguridad real del sistema.

Modificar las políticas para eliminarlas podría reducir la seguridad y generar efectos no deseados. Por ello, se recomienda no hacer cambios inmediatos en la política SELinux y continuar con un monitoreo periódico para asegurar que no surjan problemas funcionales asociados. En resumen, en ciertos casos, es más prudente no intervenir en alertas SELinux cuando estas no afectan el sistema, evitando así riesgos innecesarios.

# Bibliografía

**Oracle.** (s.f.). *Discretionary Access Control (DAC)*. Oracle Help Center. Recuperado el 11 de julio de 2025, de https://docs.oracle.com/cd/E56339_01/html/E53985/ugintro-8.html

**Linux.die.net.** (s.f.). *auditd(8) - Linux man page*. Recuperado el 11 de julio de 2025, de https://linux.die.net/man/8/auditd

**Red Hat.** (s.f.). *Using SELinux*. Red Hat Enterprise Linux 8 documentation. Recuperado el 11 de julio de 2025, de https://docs.redhat.com/es/documentation/red_hat_enterprise_linux/8/html/using_selinux/index

**Michael Kerrisk / man7.org.** (s.f.). *selinux(8) - SELinux Command Line documentation*. Recuperado el 11 de julio de 2025, de https://man7.org/linux/man-pages/man8/selinux.8.html

**Red Hat.** (s.f.). *How to configure Linux auditing with auditd*. Red Hat Blog. Recuperado el 11 de julio de 2025, de https://www.redhat.com/en/blog/configure-linux-auditing-auditd

**PHP Documentation Group.** (n.d.). *Installation of PHP with FPM*. PHP Manual. https://www.php.net/manual/en/install.fpm.php

**Red Hat.** (s.f.). *SELinux users and administrators guide*. Red Hat, Inc. Recuperado el 14 de julio de 2025, de https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/7/html/selinux_users_and_administrators_guide/index