



Universidad Autónoma de México
Facultad de Ingeniería



Exposición
"Esquemas de Seguridad "

Sistemas operativos

Hernández Saldívar Héctor Saúl 319276017

Ing. Gunnar Eyal Wolf Iszaevich

Grupo: 6

Semestre 2025-1

Fecha de entrega:
8 de octubre de 2024



Índice:

1. Introducción.....	2
2. Esquemas de seguridad en archivos en Linux.....	2
1. Modelo permisos POSIX.....	3
2. Permisos avanzados (Setuid, Setgid y Sticky bit)	4
3. Listas de Control de Acceso (ACLs)	5
4. SELinux y AppArmor.....	6
5. Auditoría de seguridad.....	7
3. Esquemas de seguridad en archivos en Windows.....	9
1. Listas de Control de Acceso (ACLs)	9
2. Permisos heredados.....	11
3. Auditorías (SACLs)	12
4. Cifrado de Archivos (EFS y Bitlocker)	14
5. Políticas de Grupo y Active directory.....	16
4. Comparaciones.....	17
1. Modelos de permisos.....	17
2. Control de acceso avanzado.....	18
3. Cifrado de Archivos.....	19
4. Auditorias.....	19
5. Conclusión.....	20
6. Referencias.....	21



1. Introducción:

Hoy en día vivimos en una era digital, esto conlleva a tener una variedad muy basta de beneficios para nosotros, sin embargo, esto también genera varios problemas entre ellos uno de mucha importancia: la información puede llegar a ser más vulnerable pudiendo terminar en sitios no muy agradables o que pueda ser utilizada para fines contrarios a los pensados.

Es por eso que debemos tener siempre presente la seguridad de la información como una de nuestras principales prioridades a lo hora de manipular los datos dentro de un sistema operativo. Todo usuario debe de aplicar medidas técnicas y organizativas que garanticen la integridad de la información para evitar así su comprometimiento, es por ello que los esquemas de seguridad en archivos juegan un papel fundamental a la hora de salvaguardar nuestros datos.

Estos esquemas abarcan diversas estrategias y técnicas que nos ayudan a prevenir un acceso no autorizado a la información. Si bien este concepto puede aplicarse a todos los sistemas operativos existentes, es evidente que la forma en que estos esquemas se aplican en cada tipo de sistema operativo van a ser muy diferentes entre sí, haciendo interesante la comparación entre cómo cada uno los maneja respecto al otro para así notar las diferencias y similitudes de sistema a sistema. Sin embargo eso abarcaría mucho tiempo, además de requerir bastante información que puede resultar abrumadora para la gente, por ello sólo podremos comparar entre dos sistemas operativos muy conocidos y los que son más usados: Windows y Linux.

2. Esquemas de seguridad en Archivos en linux

Linux es un sistema operativo caracterizado principalmente por ser de código abierto, tiene un fuerte enfoque en la seguridad por su arquitectura más robusta. Al ser de código abierto la comunidad de desarrolladores puede identificar y solucionar problemas más rápido, además de tener un modelo de permisos más estricto que impide a los usuarios normales poseer privilegios de administrador.

Debido a que no es tan popular entre la mayoría de personas que poseen un equipo de cómputo, no posee una gran cantidad de



virus y malware, aun así esto no hace que el sistema sea completamente seguro.

1. Modelo permisos POSIX

El estándar POSIX (Portable Operating System Interface) se encarga de especificar una serie de interfaces de programación comandos de línea de comandos y utilidades, así como formatos de archivo y convenciones de interacción entre programas y el sistema operativo. Estas especificaciones están basadas en características y funcionalidades que podemos encontrar en los sistemas operativos UNIX tradicionales.

En Linux podemos encontrar los permisos almacenados en **inodes**, los cuales son estructuras de datos asociadas a cada archivo estos contienen cierta información como: el número de inodo, las IDs del propietario (UID) y grupo (GID) que poseen el archivo, los permisos, el tamaño del archivo y las marcas de tiempo.

El acceso a estos archivos se gestiona mediante una máscara de 9 bits que controla la lectura(r), la escritura(w) y la ejecución(x). Cada archivo tiene permisos organizados en tres grupos. En notación binaria cuando el acceso es 1 está activo, cuando se encuentra un cero significa restringido.

- rwx para el propietario del archivo, ocupa los Bits 1-3.
- rwx para el grupo, ocupa los Bits 4-6.
- rwx para otros, ocupa los Bits 7-9

Por ejemplo si tenemos los permisos **rwxr-xr--**.

En binario:**111101100**.

Damos a entender que el propietario tiene los tres permisos activados, mientras que el grupo solo tiene acceso a leer y ejecutar el archivo, dejando a los otros únicamente con el permiso de lectura.

También contamos con comandos que nos servirán para gestionar los permisos.

→ **chmod**: Cambia los permisos, por ejemplo:

`chmod 754 archivo.txt` → Asigna los permisos `rwxr-xr--` al archivo `archivo.txt`.

→ **chown**: Cambia el propietario, por ejemplo:



`chown usuario archivo.txt → Cambia el propietario del archivo archivo.txt al usuario especificado.`

➔ **chgrp:** Cambia el grupo, por ejemplo:

`chgrp grupo archivo.txt → Asigna el archivo archivo.txt al grupo especificado.`

2. Permisos avanzados (Setuid, Setgid y Sticky bit)

- **Setuid (Set User ID):** permite que un archivo ejecutable se ejecute con los privilegios del propietario en lugar del usuario que lo ejecuta. Esta representado en el inode por el valor 4xxx. Ejemplo:

`rwsr-xr-x → Permisos octales: 4755.`

El bit Setuid está activado, el propietario puede leer, escribir y ejecutar, mientras que el grupo y otros pueden leer y ejecutar.

- **Setgid:** Se asegura que los archivos creados en un directorio hereden el grupo del directorio padre, puede ser muy útil para realizar acciones en entornos colaborativos.

Esta representado en el inode por el valor 2xxx. Ejemplo:

`rwxr-sr-x → Permisos octales: 2755.`

El bit Setgid está activado, y el grupo ejecutará el archivo con los privilegios del grupo propietario.

- **Sticky bit:** Este es un permiso especial aplicado generalmente a directorios y garantiza que solo el propietario de un archivo pueda eliminarlo o renombrarlo, incluso si otros tienen permisos de escritura en el directorio. Está representado en el inode por el valor 1xxx. Ejemplo:

`rwxrwxrwt → Permisos octales: 1777.`

El Sticky Bit está activado, lo que significa que en este directorio solo el propietario de un archivo o root puede eliminarlo, a pesar de que todos los usuarios tienen permisos de escritura en el directorio.



3. Listas de Control de Acceso (ACLs)

Son una extensión del modelo POSIX de Linux, solo que las listas pueden asignar permisos de forma más específica y precisa a archivos y/o directorios para usuarios o grupos adicionales, además de que no se limitan a las tres categorías básicas del modelo POSIX (propietario, grupo y otros). En cada archivo o directorio donde se esté usando ACLs tiene un conjunto de entradas que especifican los permisos adicionales, las ACLs se almacenan junto a los inodes del archivo o del directorio.

Cada ACL está definida por una serie de Entradas de Control de Acceso(ACEs), estas definen:

- Tipo de entrada: especifica si la entrada aplica para un usuario específico(u), grupo específico(g), usuarios que no están en el grupo(m) y otros(o).
- Permisos: lectura(r), escritura(w) y ejecución(x) asociados a la entrada.
- Identificador(UID y GID): El ID de usuario(UID) o el ID del grupo(GID) a los que se les serán aplicados los permisos

Funcionamiento:

Tenemos archivo.txt con permisos rwxr-----

Para asignar los permisos haciendo uso de ACLs sin alterar la configuración básica podemos utilizar los siguientes comandos

→ **setfacl -m u:jose:rwx archivo.txt**-Se encarga de añadir o modificar la asignación adicional

→ **getfacl archivo.txt**- Este se encarga de mostrar las ACLs de un archivo o directorio, incluyendo los permisos POSIX. La salida con ambos comandos ejecutados debería ser

user::rwx

group::r-x

other::---

user:jose:rwx # ACL adicional para el usuario "jose"

→ **setfacl -x u:jose archivo.txt**-Se encarga de eliminar la asignación adicional

→ **setfacl -m m::rwx archivo.txt**-Máscara de permisos, define los permisos máximos que se

pueden aplicar a las ACEs de grupo y usuarios adicionales (excluyendo al propietario)

- **setfacl -b archivo.txt**-Remueve todas las ACLs de un archivo, restaurando todo a los permisos básicos POSIX
- **getfacl archivo_origen.txt|setfacl --set-file=archivo_destino.txt**-Esto copia ACLs de un archivo a otro.

4. SELinux y AppArmor

- SELinux(Security-Enhanced Linux):
Implementa un modelo de Control de Acceso Obligatorio (MAC). En el modelo MAC, los permisos y accesos son determinados por políticas de seguridad definidas por el administrador del sistema y no pueden ser alterados por los usuarios individuales o procesos.

Asigna etiquetas de seguridad a archivos, procesos y recursos, además de verificar cada una de estas mismas en cada operación de acceso. Ejemplo:

`user_u:object_r:httpd_sys_content_t:s0`

- **user_u**: Rol de usuario
- **object_r**: El tipo de objeto
- **httpd_sys_content_t**: El tipo de archivo, lo que indica que es un archivo del servidor web.
- **s0**: El nivel de seguridad

Las políticas de SELinux son su núcleo de funcionamiento, definen acciones que están permitidas entre procesos, archivos, puertos y otros recursos basándose en las etiquetas de seguridad

Tiene las siguientes herramientas:

- **comando getenforce**: Muestra el modo en el que está operando SELinux (enforcing, permissive o disabled).
- **comando semanage**: Permite gestionar las políticas de SELinux, modificar las etiquetas de seguridad de archivos y cambiar los permisos de acceso.



- AppArmor:

Es una alternativa a SELinux, en lugar de las etiquetas, hacemos uso de perfiles predefinidos que controlan qué recursos y archivos puede acceder un proceso específico. Por ejemplo, el perfil puede especificar que un proceso de servidor web solo puede acceder a archivos en un directorio específico (como /var/www), y no puede leer ni modificar otros archivos del sistema, lo que aisla las actividades del servidor web. También se permite que los administradores creen sus perfiles propios personalizados para aplicaciones específicas.

Podemos destacar dos modos:

- a. Perfiles estrictos: En este modo, se aplican restricciones de acceso definidas para los perfiles y deniega cualquier intento de acceso que no esté permitido en el perfil.
- b. Complain mode: En este modo no se deniega el acceso, pero registra intentos de acceso no permitidos en los logs. Esto es útil para identificar problemas en los perfiles sin interrumpir el funcionamiento normal.

Herramientas:

- **comando aa-status:** Muestra el estado de AppArmor y los perfiles que están cargados y activos.
- **comando aa-complain:** Cambia un perfil al modo de queja, permitiendo registrar los intentos de acceso sin bloquearlos.
- **comando aa-enforce:** Cambia un perfil al modo estricto, bloqueando los accesos no permitidos.

5. Auditoría de seguridad

Es un componente clave para garantizar la integridad del sistema y detectar accesos no autorizados o actividades sospechosas en tiempo real

Audit(Audit Daemon): Es el demonio responsable de capturar y registrar eventos de seguridad en el sistema. Permite que los administradores del sistema auditen y monitorean acciones críticas, como

intentos de acceso a archivos, cambios en los permisos, ejecuciones de comandos específicos, y cualquier otra acción que pueda ser relevante desde el punto de vista de la seguridad. Registra eventos en un archivo de log (generalmente ubicado en /var/log/audit/audit.log). Los eventos que se registran son definidos por reglas de auditoría, que determinan qué acciones o tipos de eventos deben ser monitoreados.

Los eventos que se pueden auditar son:

- ✓ **Acceso a archivos:** Puedes auditar quién accede, lee, escribe o ejecuta un archivo específico.
- ✓ **Cambios de permisos:** Registra cualquier intento de cambiar los permisos de un archivo o directorio.
- ✓ **Modificación de configuraciones críticas:** Puedes auditar si se realizan cambios en archivos de configuración importantes como /etc/passwd o /etc/shadow.
- ✓ **Accesos fallidos:** Intentos de acceso a recursos para los cuales el usuario no tiene los permisos adecuados.
- ✓ **Ejecuciones de comandos:** Puedes auditar la ejecución de comandos específicos para controlar qué usuarios pueden ejecutar ciertas aplicaciones.

Los comandos principales son:

- ➔ **auditctl:** Se encarga de crear, modificar o eliminar las reglas de auditoría. Ejemplo:
`auditctl -w /etc/passwd -p wa -k passwd_change`

-w /etc/passwd: Especifica el archivo que se va a auditar, en este caso, el archivo /etc/passwd.
-p wa: Define las operaciones que deben ser auditadas. En este caso, la escritura (w) y los cambios de atributos (a).
-k passwd_change: Es una etiqueta (key) que puedes usar para identificar esta regla en los logs.

Para eliminar una regla hay que ejecutar el siguiente comando

```
auditctl -W /etc/passwd
```

→ **ausearch:** Se encarga de realizar la búsqueda de los eventos creados y registrados. Se puede realizar las busquedas por:

logs:

```
ausearch -f /etc/passwd
```

Filtrar por clave:

```
ausearch -k passwd_change
```

Buscar eventos por usuarios:

```
ausearch -ua <user_ID>
```

→ **auditd:** Inicia, detiene o reinicia el demonio de auditoría en el sistema.

3. Esquemas de seguridad en Archivos en Windows

No es secreto para nadie que windows es el sistema operativo más popular del mercado y por ello el más usado por la mayoría de usuarios de un equipo de computo. A diferencia de Linux, Windows es un software propietario, por ello su código no está disponible para que cualquiera lo pueda revisar o modificar.

Es muy conocido por ser más fácil de usar y por la gran cantidad de software que existe para él, sin embargo esto también lo hace un sistema bastante atacado y con una gran variedad de virus y malware.

1. Listas de Control de Acceso (ACLs)

Son un mecanismo avanzado para gestionar permisos de archivos y carpetas de forma granular. Contienen

entradas de control de acceso (ACEs), donde cada ACE define los permisos otorgados o denegados a un usuario o grupo en particular.

Los componentes clave de una ACE SON:



SID (Security Identifier): El Identificador de Seguridad es un valor único que identifica a un usuario o grupo. El SID se usa para asociar la ACE a un usuario o grupo específico.

Máscara de acceso (Access Mask): La máscara de acceso es un conjunto de bits que define los permisos que la ACE concede o deniega al usuario o grupo asociado.

- **0x001: Lectura** (READ_DATA o FILE_READ_DATA para archivos, FILE_LIST_DIRECTORY para directorios). Este bit permite al usuario leer los datos de un archivo o listar los contenidos de un directorio.
- **0x002: Escritura** (WRITE_DATA o FILE_WRITE_DATA para archivos, FILE_ADD_FILE para directorios). Este bit permite al usuario modificar los datos de un archivo o agregar archivos a un directorio.
- **0x004: Ejecución** (EXECUTE o FILE_EXECUTE para archivos, FILE_TRAVERSE para directorios). Este bit permite ejecutar un archivo o acceder a un subdirectorio.

Además también incluye permisos avanzados como DELETE, WRITE_ATTRIBUTES y TAKE_OWNERSHIP.

Permisos: Cada ACE define permisos específicos, como lectura, escritura, ejecución, eliminar, modificar atributos, etc.

Tipo de ACE: Puede ser permitir o denegar, lo que indica si la acción está permitida o no para el usuario o grupo.

Comandos:

→ **icacls**: nos permite ver y modificar las ACLs.

para ver: *icacls archivo.txt*

para modificar:

icacls archivo.txt /grant usuario1:(RX)

para eliminar:

icacls archivo.txt /remove usuario1



→ **PowerShell:** ofrece una forma más avanzada y flexible de gestionar las ACLs en Windows. El comando Get-ACL te permite recuperar la información detallada sobre los permisos de un archivo o carpeta.
para ver: *Get-ACL archivo.txt*

para modificar:

```
$acl = Get-ACL archivo.txt
$perm = "usuario1", "Write", "Allow"
$ace = New-Object
System.Security.AccessControl.FileSystemAccessRule $perm
$acl.SetAccessRule($ace) Set-ACL archivo.txt $acl
```

2. Permisos heredados

Es un mecanismo que permite que los permisos asignados a una carpeta o directorio padre se transmitan automáticamente a sus subdirectorios y archivos, facilitando así la administración de permisos en estructuras de carpetas complejas. Para que estos permisos se hereden basta con que los archivos o subcarpetas estén dentro de la carpeta o directorio padre.

Hay dos tipos de herencia:

Herencia explícita: Son los permisos que se asignan directamente a un archivo o carpeta sin depender de la herencia de un directorio superior.

Herencia implícita: Son los permisos que un archivo o carpeta hereda de su directorio padre.

Windows permite detener la herencia o modificar los permisos heredados para casos específicos:

- ✓ Haz clic derecho sobre el archivo o carpeta, selecciona Propiedades.
- ✓ En la pestaña de Seguridad, haz clic en Opciones avanzadas.
- ✓ Haz clic en Deshabilitar herencia.
- ✓ Elige entre Convertir permisos heredados en permisos explícitos o Eliminar todos los permisos heredados.

Para modificar un permiso heredado en un archivo específico, puedes otorgar permisos de escritura en un archivo individual que ha heredado solo permisos



de lectura. En este caso, puedes hacer que el archivo deje de heredar permisos.

Además de que también podemos aplicar esta misma herencia con estructuras ACEs, ya que tienen un bit especial que indica si los permisos son heredados o no.

En cada ACE, se pueden definir las siguientes propiedades para controlar la herencia:

OI (Object Inherit): Si este bit está activado, los archivos dentro del directorio heredarán los permisos definidos en la ACE.

CI (Container Inherit): Si este bit está activado, las subcarpetas dentro del directorio heredarán los permisos definidos en la ACE.

IO (Inherited Only): Indica que la ACE solo es aplicable a objetos heredados (subcarpetas o archivos), no al objeto actual.

Comandos:

→ **icacls** puede gestionar los permisos heredados desde la línea de comandos

Ver herencia: `icacls "C:\Carpeta"`

Eliminar herencia:

`icacls "C:\Carpeta" /inheritance:d`

→ **PowerShell** también ofrece una forma avanzada de ver y modificar los permisos heredados

`Get-ACL "C:\Carpeta"`

3. Auditorías (SACLS)

Las Listas de Control de Acceso del Sistema (SACLS) son listas especiales que definen qué eventos relacionados con el acceso o modificación de un archivo o carpeta deben ser registrados en los logs de auditoría del sistema. Estas listas permiten realizar un seguimiento detallado de las acciones de los usuarios sobre archivos y carpetas.

Cada entrada puede definir: Quién debe ser auditado (usuario o grupo), qué tipo de acceso o acciones

deben ser auditadas (lectura, escritura, modificación, etc.) y si el acceso es exitoso o fallido. También se pueden auditar los cambios y permisos, así como la creación o eliminación de archivos.

Todos los eventos registrados por las SACLs se almacenan en los logs de auditoría del sistema, estos se pueden ver mediante el visor de eventos Windows, los administradores pueden configurar el visor para mostrar eventos relacionados con:

- **Acceso a archivos:** Quién accedió, cuándo, y desde qué cuenta.
- **Intentos de modificación:** Quién intentó cambiar un archivo o directorio, y si fue exitoso o fallido.
- **Accesos fallidos:** Los intentos fallidos de acceder a archivos o directorios también se registran, lo que puede ser útil para identificar intentos de intrusión.

Los comandos para la configuración de auditorías son:

→ **auditpol:** es utilizado para configurar y gestionar las políticas de auditoría en todo el sistema. Estas políticas dictan qué acciones se auditarán y qué eventos se registrarán en los logs.

Para ver la configuración actual:
`auditpol /get /category:*`

Habilitar la auditoría de acceso a archivos y carpetas
`auditpol /set /subcategory:"Object Access" /success:enable /failure:enable`

Habilitar auditoría para cambios de permisos
`auditpol /set /subcategory:"File System" /success:enable /failure:enable`

→ **Wvtutil:** Permite gestionar y consultar los eventos registrados en el sistema, incluidos

los generados por las políticas de auditoría configuradas a través de SACLs.

Ver los eventos registrados para un log en específico:

WEvtutil qe Security /f:text

Eliminar registros antiguos:

WEvtutil cl Security

Exportar registros de eventos:

WEvtutil epl Security C:\path\logs.evtx

4. Cifrado de Archivos (EFS y BitLocker)

EFS (Encrypting File System): Es una funcionalidad integrada en los sistemas de archivos NTFS que permite cifrar archivos y carpetas de manera individual. Es transparente para el usuario, lo que significa que los usuarios que tienen los permisos adecuados pueden acceder a los archivos cifrados sin realizar pasos adicionales.

Se cifran archivos y carpetas utilizando una clave única de cifrado para cada archivo. Esta clave, conocida como FEK (File Encryption Key), es responsable de cifrar el contenido del archivo. La FEK se cifra a su vez utilizando la clave pública del usuario autorizado, y solo ese usuario puede descifrar el archivo utilizando su clave privada.

El cifrado ocurre en segundo plano, por lo que los usuarios que tienen los permisos adecuados pueden trabajar con los archivos cifrados como si fueran archivos normales. Además, si un usuario no autorizado intenta acceder a un archivo cifrado, el archivo será ilegible, ya que no tiene acceso a la clave privada necesaria para descifrar el contenido.

Las FEK se gestionan de la siguiente manera:

- **Clave del archivo (FEK):** Cada archivo tiene una clave de cifrado única que cifra su contenido.
- **Cifrado de la FEK:** La clave del archivo (FEK) se cifra con la clave pública del usuario que



cifra el archivo. Solo el usuario autorizado, con su clave privada, puede descifrar esta clave del archivo y, por lo tanto, acceder al contenido.

Para poder cifrar un archivo con EFS se debe hacer lo siguiente:

- ✓ Haz clic derecho sobre el archivo o carpeta.
- ✓ Selecciona Propiedades.
- ✓ En la pestaña General, haz clic en Opciones avanzadas.
- ✓ Marca la casilla Cifrar contenido para proteger los datos.
- ✓ Haz clic en Aceptar.

Bitlocker: Es una herramienta de cifrado más robusta que EFS, diseñada para proporcionar cifrado de disco completo. Cifra todo el contenido de una partición o disco, proporcionando seguridad integral para todos los datos almacenados en esa unidad.

Puede usar un TPM (Trusted Platform Module), un chip que almacena de manera segura las claves de cifrado. Si el dispositivo cuenta con un chip TPM, este almacena las claves de cifrado de forma segura y evita que el disco sea descifrado si el equipo ha sido manipulado o movido a otro sistema. Si el dispositivo no tiene un TPM, BitLocker puede configurarse para requerir una contraseña o una unidad flash USB que contenga una clave de recuperación para desbloquear el disco durante el inicio.

Se hace uso de una clave maestra de cifrado para proteger toda la unidad. Esta clave se almacena de forma segura en el TPM pero también se puede proteger mediante una clave de recuperación en caso de que el acceso normal al sistema se encuentre bloqueado.

Para activar Bitlocker debemos hacer lo siguiente

- ✓ Abre el Panel de control y selecciona Sistema y seguridad.
- ✓ Haz clic en Cifrado de unidad BitLocker.

- ✓ Selecciona la unidad que deseas cifrar y haz clic en Activar BitLocker.
- ✓ Sigue las instrucciones para elegir un método de autenticación (por ejemplo, usar un TPM, una contraseña o una unidad flash USB).
- ✓ BitLocker comenzará a cifrar la unidad.

5. Políticas de Grupo y Active directory.

Active directory: Es una base de datos estructurada que almacena información sobre los usuarios, grupos, equipos y otros recursos dentro de una red. AD proporciona un marco para la autenticación y autorización, permitiendo a los administradores gestionar de manera eficiente quién tiene acceso a qué recursos en la organización.

Los elementos pueden ser almacenados como objetos, estos pueden ser:

- **Usuarios:** Cuentas que representan individuos en la red.
- **Grupos:** Colecciones de usuarios que pueden ser administradas como una única entidad.
- **Equipos:** Dispositivos dentro de la red.
- **Unidades Organizativas (OUs):** Contenedores que agrupan objetos para simplificar la gestión y aplicación de políticas.

Políticas de Grupo: Son una característica de Active Directory que permite a los administradores aplicar configuraciones y restricciones centralizadas en todos los equipos y usuarios de una organización. Se utilizan para gestionar el entorno operativo de los usuarios y los equipos, incluyendo la configuración de seguridad, la administración de software, y el control de acceso a recursos.

Al crear una política de grupo, un administrador puede especificar configuraciones para:

- Permisos de archivos.
- Configuraciones de seguridad.
- Configuraciones de software.

Las políticas de grupo permiten implementar configuraciones para asegurar archivos, realizar auditorías, y aplicar otras medidas de seguridad en múltiples sistemas al mismo tiempo. Un administrador puede:

- ✓ Configurar auditorías de acceso a archivos para que todos los intentos de acceso (exitosos y fallidos) sean registrados en un log de eventos.
- ✓ Aplicar configuraciones de cifrado para todos los equipos en la red, asegurando que los datos sensibles están protegidos sin necesidad de configuraciones individuales.

Centralización: Es proporcionada por Active directory y las políticas de grupo, permite a las organizaciones implementar reglas y configuraciones de seguridad en una escala masiva. Esto no solo ahorra tiempo, sino que también reduce el riesgo de errores humanos al configurar permisos en cada máquina individualmente.

4. Comparaciones entre Linux y Windows

Una vez revisados los esquemas de seguridad en archivos tanto para Linux como para Windows, es posible que se hayan visto diferencias muy marcadas entre los esquemas de un sistema operativo al de otro. Sin embargo es necesario y, hasta cierto punto, demandado realizar una comparación breve entre ambos sistemas para poder observar con mayor claridad cómo cada uno implementa sus esquemas en sus respectivos archivos.

1. Modelos de permisos

Linux:

El modelo de permisos en Linux está basado en POSIX, que es directo y sencillo. Cada archivo y directorio tiene tres conjuntos de permisos: propietario, grupo y otros, y se asignan tres tipos de permisos básicos: lectura (r), escritura (w) y ejecución (x). Este modelo permite un control básico sobre los accesos, pero puede ser expandido con ACLs (Listas de Control de Acceso), que añaden mayor granularidad



al permitir asignar permisos específicos a usuarios o grupos adicionales. Además, Linux ofrece herramientas avanzadas como SELinux y AppArmor, que permiten un control más detallado sobre el acceso a los recursos del sistema.

Windows:

Desde su origen, ha adoptado un enfoque más granular al gestionar permisos, utilizando ACLs de forma predeterminada. Las DACLs (Discretionary Access Control Lists) permiten definir permisos detallados para cualquier usuario o grupo sobre archivos, directorios y recursos del sistema. Windows también incluye las SACLs (System Access Control Lists), que se utilizan para configurar auditorías y registrar eventos relacionados con el acceso a recursos. El modelo de permisos de Windows es más flexible y granular desde el inicio, lo que facilita un control más específico y detallado sobre los accesos.

2. Control de acceso avanzado

Linux:

Ofrece control de acceso discrecional (DAC) mediante el modelo POSIX, pero además cuenta con herramientas avanzadas como SELinux (Security-Enhanced Linux) y AppArmor, que proporcionan control de acceso obligatorio (MAC, Mandatory Access Control). MAC añade una capa de seguridad adicional al imponer políticas que controlan el acceso a los recursos del sistema, independientemente de los permisos asignados a los archivos. SELinux es muy utilizado en entornos empresariales y críticos, ya que permite definir políticas estrictas que controlan qué procesos pueden interactuar con qué archivos o recursos.

Windows:

El control de acceso se gestiona principalmente a través de ACLs y SACLs, lo que proporciona una gestión detallada y granular de los permisos y auditorías. Sin embargo, Windows no incluye un sistema MAC como SELinux para imponer políticas de seguridad adicionales. En lugar de ello, Windows utiliza Políticas de Grupo (Group Policies), que



permiten gestionar de manera masiva los permisos y la configuración de seguridad en una red de equipos. A través de estas políticas, los administradores pueden implementar y mantener configuraciones de seguridad de manera uniforme en todos los sistemas de la organización.

3. Cifrado de Archivos

Linux:

El cifrado en Linux no está integrado directamente en el sistema operativo para archivos individuales o discos completos, y requiere el uso de herramientas externas. Para el cifrado de discos completos, se utiliza LUKS (Linux Unified Key Setup), que protege el contenido de discos y particiones enteras. Para cifrar archivos individuales, herramientas como GPG (GNU Privacy Guard) son comunes. Estas soluciones proporcionan un nivel alto de seguridad, pero su configuración y uso suelen requerir un conocimiento técnico más avanzado.

Windows:

Ofrece soluciones de cifrado nativas directamente integradas en el sistema operativo. Para archivos individuales, se utiliza EFS (Encrypting File System), que permite cifrar carpetas y archivos de manera transparente para el usuario autorizado. Para el cifrado de discos completos, Windows utiliza BitLocker, que protege todo el contenido de una unidad y puede hacer uso del TPM (Trusted Platform Module) para almacenar las claves de cifrado de manera segura. Estas herramientas están diseñadas para ser fáciles de configurar y usar, incluso para usuarios sin experiencia técnica avanzada.

4. Auditorias

Linux:

Las auditorías de seguridad son gestionadas por herramientas como auditd, que permiten registrar eventos relacionados con el acceso a archivos, cambios en permisos y otros eventos críticos de seguridad. auditd es una herramienta muy flexible y poderosa, pero requiere una configuración manual detallada por parte del administrador. Los logs de



auditóriá generados por auditd pueden revisarse utilizando comandos como ausearch para filtrar y analizar los eventos registrados. Si bien auditd es muy eficaz, su configuración y uso pueden resultar complicados para administradores sin experiencia previa en auditoría de sistemas.

Windows:

Facilita las auditorías a través de su sistema integrado con SACLs. Las SACLs permiten a los administradores configurar auditorías detalladas sobre archivos y recursos, registrando eventos como accesos exitosos, fallidos o modificaciones de permisos. Estos eventos se registran en el Visor de eventos, que ofrece una interfaz gráfica fácil de usar para visualizar y analizar los logs de auditoría. Este enfoque es más amigable para los usuarios y administradores que prefieren una interfaz gráfica en lugar de depender completamente de herramientas de línea de comandos.

5. Conclusión

Muchas personas estarán pensando que al comparar ambos sistemas operativos, estamos viendo que sistema operativo es mejor pero esto no es verdad. El verdadero motivo para comparar los esquemas de seguridad entre Windows y Linux fue meramente para observar el cómo cada sistema se encarga de proteger sus archivos de diferente manera basados en como fueron diseñados y estructurados, así del uso pensado para estos.

Uno podría decir que Linux es mejor por el hecho de que si bien su seguridad es algo simple, tiene permisos muy potentes y herramientas avanzadas para los entornos críticos, además de que al no ser su uso tan amplio entre la gente, tiene menos probabilidad de ser afectado por un ataque o un virus.

También está Windows que tiene un enfoque más flexible y granular que lo hace eficiente para el manejo de redes que tengan un gran volumen de usuarios, además de tener que centralizar y administrar la gestión para permisos en conjunto con las auditorías.



Ya en los párrafos anteriores uno puede ver la verdadera respuesta a la pregunta de ¿Cuál es más seguro?. Pues es que en realidad ambos son seguros dependiendo del entorno en el que te encuentres, si necesitas un control más detallado sobre los permisos que tienes en tus archivos la respuesta es Linux, pero si estás manejando redes con una gran variedad de usuarios lo mejor es Windows.

Muchos piensan en su fanatismo por su SO preferido, que uno debe ser mejor que otro por sus propias razones y preferencias, pero nunca se ponen a ver que ambos tienen sus puntos fuertes así como sus puntos débiles. Además de que tanto Windows como Linux están diseñados para dos enfoques distintos, por lo cual decir que uno es mejor que otro no tiene mucho sentido en realidad.

6. Referencias:

- [1] J. Herrera, "Estándar POSIX, ¿qué es y para qué sirve?," *Guía Hardware*, 29 de junio de 2023.
<https://www.guiahardware.es/estandar-posix/> (Accedido el 27 de septiembre de 2024).
- [2] "The Linux Documentation Project: Recent Updates," *Tldp.org*, 2015. https://tldp.org/sorted_howtos.html (Accedido el 27 de septiembre de 2024).
- [3] R. Love and Addison-Wesley, *Linux Kernel Development*, 3rd ed. Upper Saddle River: Addison-Wesley, 2015. (Accedido el 28 de septiembre de 2024).
- [4] "Documentation for Red Hat Products," *Red Hat Customer Portal*. <https://access.redhat.com/documentation/en-us/> (Accedido el 28 de septiembre de 2024).
- [5] Wibjorn, "Microsoft Learn: adquirir conocimientos que le abran las puertas en su carrera profesional", *learn.microsoft.com*. <https://learn.microsoft.com/es-es/> (Accedido el 1 de octubre de 2024).
- [6] Pavel Yosifovich, D. A. Solomon, and A. Ionescu, *Windows Internals, Part 1*. Microsoft Press, 2017. (Accedido el 3 de octubre de 2024).
- [7] V. Alonso, "¿Qué es más seguro Linux o Windows?," *sistemasoperativos.info*, 22 de mayo de 2023. [¿Qué es más seguro Linux o Windows? | Análisis 2024 \(sistemasoperativos.info\)](https://sistemasoperativos.info/que-es-mas-seguro-linux-o-windows-analisis-2024-sistemasoperativos.info) (Accedido el 3 de octubre de 2024).