

3.3-RESTRICCIONES DE SEGURIDAD PARA TAREAS PROGRAMADAS (W-L)

Cristóbal Suárez Abad

ADMINISTRACIÓN DE SISTEMAS OPERATIVOS - 2º ASIR

Índice

| | |
|---|----|
| LINUX | 3 |
| ▪ Auditoría:..... | 3 |
| Windows | 13 |
| Tienes que identificar: | 14 |
| - Tareas que usan contraseña almacenada en claro..... | 14 |
| - Tareas configuradas con “Ejecutar con privilegios más altos”..... | 15 |
| - Scripts que se ejecutan desde rutas de usuario (riesgo de manipulación).... | 16 |
| - Triggers sospechosos (eventos de sistema, logon, idle)..... | 17 |
| Generar un informe equivalente al de Linux..... | 18 |
| Fortificación de scripts y rutas. Creación de usuarios de servicio..... | 22 |
| Linux..... | 22 |
| Identificar scripts en: | 22 |
| Aplicar medidas de hardening obligatorias: | 23 |
| Añadir cabecera “trap” para evitar interrupciones: (trap "echo 'ERROR: Execution interrupted' >> /var/log/script.log" ERR INT TERM) | 24 |
| Reemplazar rutas relativas por absolutas..... | 25 |
| Crear un directorio securizado para automatizaciones en el que el propietario sea root..... | 26 |
| Crear un usuario de servicio sin Shell (llamado autosvc). | 27 |
| Configurar sudoers para permitir ejecutar solo un comando concreto | 28 |
| Conclusión de hardening | 31 |
| Windows | 32 |
| Crear un directorio restringido para scripts | 32 |
| Configurar ACL estrictas:..... | 33 |
| Firmar los scripts PowerShell con un certificado local: | 34 |
| Crear un usuario de servicio..... | 35 |
| Modificar una tarea programada para usar este usuario SIN contraseña | 36 |
| Verificaciones finales | 38 |
| Detección de manipulación | 40 |
| Linux..... | 40 |
| Activar auditoría de ejecución de scripts (auditctl)..... | 40 |

| | |
|---|----|
| Crear un sistema de logs centralizado para estos scripts (logger) | 42 |
| Comprobación y análisis de logs (ausearch)..... | 43 |
| Correlación auditd + logger | 44 |
| Pruebas de detección de manipulación | 45 |
| Windows..... | 46 |
| Activar transcripción completa de PowerShell (Set-ItemProperty)..... | 46 |
| 2. Activar registro de scripts firmados y no firmados..... | 48 |
| Activar política de ejecución recomendada | 49 |
| Comprobación mediante el Visor de eventos | 50 |
| Ataque simulado..... | 55 |
| Linux:..... | 55 |
| Modificación de un script protegido | 55 |
| Inserción de PATH malicioso | 56 |
| Ejecución del script como root sin autorización..... | 57 |
| Sustitución del script original | 58 |
| Windows | 59 |
| Modificación de un script PowerShell protegido | 59 |
| Inserción de PATH malicioso | 60 |
| Ejecución del script como administrador | 61 |
| Sustitución de un archivo firmado..... | 62 |
| Conclusión del ataque simulado..... | 63 |

LINUX

■ Auditoría:

Realizar una auditoría completa de tareas programadas:

Tienes que identificar:

- *Scripts sin permisos coherentes.*
- *Tareas que se ejecutan como root sin necesidad.*
- *Rutas no absolutas (riesgo de PATH injection).*
- *Variables de entorno heredadas peligrosas.*
- *Generar un informe inicial enumerando vulnerabilidades reales encontradas.*

Para cada tarea deberías analizar (como mínimo):

- *Objetivo de cada tarea.*
- *Trigger y periodicidad.*
- *Seguridad:*
 - *¿bajo qué usuario corre?*
 - *¿puede implicar riesgo?*
- *Logs generados.*

Identificamos las tareas programadas:

Cron del sistema

ls -l /etc/cron.*

Nos muestra el directorio de cron, donde en ejercicios anteriores hemos puestos los scripts para que se ejecuten según el intervalo de tiempo que queramos (cada hora, diario, semanal y mensual).

```
[root@server2asir usuario]$ls -l /etc/cron.*  
/etc/cron.d:  
total 8  
-rw-r--r-- 1 root root 201 ene  8  2022 e2scrub_all  
-rw-r--r-- 1 root root 377 jun  3  2025 zfsutils-linux  
  
/etc/cron.daily:  
total 28  
-rwxr-xr-x 1 root root 376 nov 11  2019 apport  
-rwxr-xr-x 1 root root 1478 abr  8  2022 apt-compat  
-rwxr-xr-x 1 root root 384 nov 19  2019 cracklib-runtime  
-rwxr-xr-x 1 root root 123 dic  5  2021 dpkg  
-rw-r--r-- 1 root root 171 dic 16 10:16 limpieza_sistema.sh  
-rwxr-xr-x 1 root root 377 may 25  2022 logrotate  
-rwxr-xr-x 1 root root 1330 mar 17  2022 man-db  
  
/etc/cron.hourly:  
total 4  
-rw-r--r-- 1 root root 181 dic 16 10:12 registro_recursos.sh  
  
/etc/cron.monthly:  
total 4  
-rw-r--r-- 1 root root 169 dic 16 10:24 backup_config.sh  
  
/etc/cron.weekly:  
total 8  
-rwxr-xr-x 1 root root 1020 mar 17  2022 man-db  
-rw-r--r-- 1 root root 282 dic 16 10:21 reporte_usuarios.sh  
  Cristobal CON ROOT jueves 18 diciembre 2025 11:29  
[root@server2asir usuario]$
```

Cron del root

crontab -l

Nos muestra el contenido del archivo crontab, donde se establecen comandos y el intervalo de tiempo para que se ejecuten.

```
[root@server2asir usuario]$crontab -l
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow   command
00 12 * * * tar -czf /opt/backup_descargas_$(date +\%F).tar.gz /home/usuario/Descargas
```

Cron de otros usuarios

```
for u in $(cut -f1 -d: /etc/passwd); do  
    crontab -u $u -l 2>/dev/null  
  
done
```

Nos muestra el mismo archivo de antes, pero también con aquellas entradas que hayan hecho otros usuarios.

```
[root@server2asir usuario]$for u in $(cut -f1 -d: /etc/passwd); do  
    crontab -u $u -l 2>/dev/null  
done  
# Edit this file to introduce tasks to be run by cron.  
#  
# Each task to run has to be defined through a single line  
# indicating with different fields when the task will be run  
# and what command to run for the task  
#  
# To define the time you can provide concrete values for  
# minute (m), hour (h), day of month (dom), month (mon),  
# and day of week (dow) or use '*' in these fields (for 'any').  
#  
# Notice that tasks will be started based on the cron's system  
# daemon's notion of time and timezones.  
#  
# Output of the crontab jobs (including errors) is sent through  
# email to the user the crontab file belongs to (unless redirected).  
#  
# For example, you can run a backup of all your user accounts  
# at 5 a.m every week with:  
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/  
#  
# For more information see the manual pages of crontab(5) and cron(8)  
#  
# m h  dom mon dow   command  
00 12 * * * tar -czf /opt/backup_descargas_${(date +\%F)}.tar.gz /home/usuario/Descargas  
@daily echo "$(date) ¿Ha salido Half-Life 3?: NO" #Registro de si ha salido Half-Life 3  
[root@server2asir ~]#
```

systemd timers

systemctl list-timers --all

Nos muestra los temporizadores de systemd.

| NEXT | LEFT | LAST | PASSED | UNIT | ACTIVATES |
|-----------------------------|---------------|-----------------------------|-------------------|--------------------------------|----------------------------------|
| Thu 2025-12-18 11:36:15 UTC | 1min 56s left | n/a | n/a | systemd-tmpfiles-clean.timer | systemd-tmpfiles-clean.service |
| Thu 2025-12-18 11:52:54 UTC | 18min left | Tue 2025-12-16 10:13:16 UTC | 2 days ago | apt-daily-upgrade.timer | apt-daily-upgrade.service |
| Thu 2025-12-18 13:09:28 UTC | 1h 35min left | Tue 2025-12-16 10:58:06 UTC | 2 days ago | fwupd-refresh.timer | fwupd-refresh.service |
| Thu 2025-12-18 13:36:50 UTC | 2h 2min left | Wed 2025-12-10 12:03:42 UTC | 1 week 0 days ago | motd-news.timer | motd-news.service |
| Thu 2025-12-18 16:50:51 UTC | 5h 16min left | Tue 2025-12-09 11:43:41 UTC | 1 week 1 day ago | man-db.timer | man-db.service |
| Thu 2025-12-18 20:24:00 UTC | 8h left | Wed 2025-12-10 12:24:42 UTC | 1 week 0 days ago | apt-daily.timer | apt-daily.service |
| Fri 2025-12-19 00:00:00 UTC | 12h left | n/a | n/a | dpkg-db-backup.timer | dpkg-db-backup.service |
| Fri 2025-12-19 00:00:00 UTC | 12h left | Thu 2025-12-18 11:21:38 UTC | 12min ago | logrotate.timer | logrotate.service |
| Fri 2025-12-19 11:26:39 UTC | 23h left | Thu 2025-12-18 11:26:39 UTC | 7min ago | update-notifier-download.timer | update-notifier-download.service |
| Sun 2025-12-21 03:10:00 UTC | 2 days left | Tue 2025-12-16 09:55:35 UTC | 2 days ago | e2scrub_all.timer | e2scrub_all.service |
| Mon 2025-12-22 00:41:19 UTC | 3 days left | Tue 2025-12-16 10:25:21 UTC | 2 days ago | fstrim.timer | fstrim.service |
| Tue 2025-12-23 15:30:32 UTC | 5 days left | Tue 2025-12-16 11:12:08 UTC | 2 days ago | update-notifier-motd.timer | update-notifier-motd.service |
| n/a | n/a | n/a | n/a | apport-autoreport.timer | apport-autoreport.service |
| n/a | n/a | n/a | n/a | snapd.snap-repair.timer | snappy.snap-repair.service |
| n/a | n/a | n/a | n/a | ua-timer.timer | ua-timer.service |

15 timers listed.

Análisis de tareas:

- 00 12 * * * tar -czf /opt/backup_descargas_\$(date +\%F).tar.gz
/home/usuario/Descargas

Objetivo de la tarea

El objetivo es realizar un **respaldo automatizado y comprimido** de la carpeta de descargas de un usuario. Al incluir la fecha en el nombre del archivo (`$(date +\%F)`), permite mantener un historial de copias (versionado diario) en lugar de un único archivo que se sobrescribe.

Trigger y Periodicidad

El "disparador" es el demonio **cron** (o crond), que revisa las tablas de tareas cada minuto.

- **Periodicidad:** Diaria.
- **Momento exacto:** Cada día a las 12:00:00 PM.
- **Sintaxis:** 00 (minuto) 12 (hora) * * * (cualquier día del mes/mes/día de la semana).

Seguridad

- **¿Bajo qué usuario corre?** Depende de dónde se haya configurado:
 - Si está en /etc/crontab, se especifica un usuario (normalmente root).
 - Si se usó crontab -e como usuario normal, corre como ese **usuario**.
 - **Importante:** Para escribir en /opt/, el usuario necesita permisos de escritura en ese directorio. Si el usuario "usuario" no tiene permisos ahí, la tarea fallará a menos que la ejecute root.
- **¿Puede implicar riesgo?**
 - **Llenado de disco:** Al crear un archivo .tar.gz nuevo cada día en /opt/, si la carpeta de Descargas es grande, el disco se llenará rápidamente. **Es el riesgo principal.**
 - **Privilegios:** Si corre como root, hay un riesgo menor de seguridad si alguien logra manipular el contenido de /home/usuario/Descargas para explotar alguna vulnerabilidad de tar (aunque es poco común).
 - **Exposición de datos:** El archivo resultante en /opt/ podría ser legible por otros usuarios si no se ajusta el umask o los permisos tras la creación.

Logs generados

Por defecto, cron no genera un archivo de log específico para el "output" (salida) de la tarea, pero sí registra la ejecución:

- **Log del sistema:** En distribuciones tipo Debian/Ubuntu, verás la ejecución en /var/log/syslog. En RHEL/CentOS, en /var/log/cron.
- **Salida de error/estándar:** Si el comando genera un error (por ejemplo, "disco lleno"), cron intentará enviar un correo electrónico al usuario local.
- **Recomendación:** Para tener logs propios, se suele modificar el comando así: 00 12 * * * comando >> /var/log/backup.log 2>&1

- @daily echo "\$(date) ¿Ha salido Half-Life 3?: NO" #Registro de si ha salido Half-Life 3

Objetivo de la tarea

El objetivo es puramente **informativo y de registro histórico (log)**. Se encarga de escribir una línea de texto en la salida estándar (que cron capture) confirmando que, un día más, Valve no ha lanzado Half-Life 3. Es una tarea de monitoreo de estado manual/humorística.

Trigger y Periodicidad

El disparador es el alias especial de cron.

- **Periodicidad:** Diaria.
- **Momento exacto:** Generalmente se ejecuta a las **00:00 (medianoche)** de cada día.
- **Sintaxis:** @daily es un atajo equivalente a 0 0 * * *.

Seguridad

- **¿Bajo qué usuario corre?**
 - Suele correr bajo el usuario que haya editado su crontab (crontab -e). No requiere privilegios de root ya que el comando echo y date son inofensivos y accesibles para cualquier usuario.
- **¿Puede implicar riesgo?**
 - **Riesgo nulo:** El comando no modifica archivos del sistema, no abre puertos y no consume recursos de CPU o memoria significativos. Es una de las tareas más seguras posibles.
 - **Inundación de correo (Mail flooding):** Si el servidor tiene configurado un agente de correo (MTA) y no se redirige la salida, el usuario recibirá un email diario con esa frase, lo cual podría ser molesto a largo plazo.

Logs generados

- **Registro de ejecución:** Quedará constancia de que el comando se ejecutó en los logs del sistema (/var/log/syslog o /var/log/cron).
- **Salida del comando:** * Tal como está escrito, la frase [Fecha] ¿Ha salido Half-Life 3?: NO se envía a la **salida estándar (stdout)**.
 - **Importante:** Cron capture esa salida y, por defecto, la envía por correo interno al usuario. Si quieras que se guarde en un archivo real para verlo

después, deberíasadir una redirección: @daily echo "\$(date) ¿Ha
salido...?" >> /home/usuario/hl3_status.log

c) Vulnerabilidades detectadas:

No se han encontrado ningún caso, así que pondré ejemplos:

- **Scripts sin permisos coherentes**

-rwxrwxr-x admin admin clean.sh

Riesgo: modificación por otros usuarios.

- **Ejecución innecesaria como root**

0 3 * * * /home/admin/scripts/clean.sh

- **Rutas no absolutas**

rm temp/*

Riesgo de **PATH injection**.

- **Variables de entorno heredadas**

PATH=/home/admin/bin:\$PATH

Windows

*Exportar todas las tareas programadas: (schtasks /Query /XML /TN * > all_tasks.xml)*

Tienes que identificar:

- *Tareas que usan contraseña almacenada en claro.*
- *Tareas configuradas con “Ejecutar con privilegios más altos”.*
- *Scripts que se ejecutan desde rutas de usuario (riesgo de manipulación).*
- *Triggers sospechosos (eventos de sistema, logon, idle).*

Generar un informe equivalente al de Linux.

- Para descargar el XML usa:

schtasks /Query /XML > all_tasks.xml

- Para un CSV usa:

Get-ScheduledTask | Export-Csv -Path "all_tasks.csv" -NoTypeInformation

- Para generar una carpeta con cada tarea en un XML independiente:

This creates a folder and saves each task as its own XML file

New-Item -ItemType Directory -Force -Path ".\TaskBackups"

Get-ScheduledTask | ForEach-Object {

\$xml = Export-ScheduledTask -TaskName \$_.TaskName -TaskPath \$_.TaskPath

\$filename = (\$_.TaskName -replace '[\\V:*?"<>|]', '_') + ".xml"

\$xml | Out-File -FilePath ".\TaskBackups\\$filename"

}

Tienes que identificar:

- Tareas que usan contraseña almacenada en claro.

¿Qué buscar en el XML?

```
<Principals>
<Principal id="Author">
<UserId>USUARIO</UserId>
<LogonType>Password</LogonType>
<RunLevel>HighestAvailable</RunLevel>
</Principal>
</Principals>
```

Indicador de riesgo:

- <LogonType>Password</LogonType>

Esto significa:

- La tarea se ejecuta con un usuario concreto
- Windows **almacena la contraseña** (cifrada, pero recuperable por el sistema)

Riesgo:

- Robo de credenciales si el sistema se compromete
- Mala práctica si no es estrictamente necesario

- Tareas configuradas con “Ejecutar con privilegios más altos”.

Qué buscar

En el mismo bloque <Principal>:

<RunLevel>HighestAvailable</RunLevel>

Indicador de riesgo:

- HighestAvailable = ejecutar como administrador

No siempre es malo, pero:

- Es crítico si ejecuta scripts modificables
- O si el trigger es automático (logon, evento...)

- Scripts que se ejecutan desde rutas de usuario (riesgo de manipulación).

Qué buscar

Bloque <Actions> → <Exec>:

```
<Actions>
  <Exec>
    <Command>C:\Users\juan\Desktop\backup.bat</Command>
  </Exec>
</Actions>
```

Rutas peligrosas:

- C:\Users\
- C:\Temp\
- C:\Downloads\
- Escritorio del usuario

Riesgo:

- El usuario puede modificar el script
- Si la tarea corre como admin → **Escalada de privilegios**

Buena práctica:

- C:\Scripts\
- C:\Program Files\
- C:\Windows\System32\

- Triggers sospechosos (eventos de sistema, logon, idle).

Qué buscar

Bloque <Triggers>:

Al iniciar sesión

<LogonTrigger>

Al arrancar el sistema

<BootTrigger>

Por eventos del sistema

<EventTrigger>

Cuando el sistema está inactivo.

<IdleTrigger>

Riesgo:

- Persistencia (malware)
- Ejecuciones invisibles al usuario
- Automatización privilegiada

Generar un informe equivalente al de Linux.

- Tarea analizada:

Nombre de la tarea: Backup_Descargas_Usuario

Comando ejecutado:

`powershell.exe -ExecutionPolicy Bypass -File "C:\Scripts\backup_descargas.ps1"`

Contenido relevante del script:

```
$fecha = Get-Date -Format "yyyy-MM-dd"  
$origen = "C:\Users\usuario\Downloads"  
$destino = "C:\Backups\backup_descargas_$fecha.zip"  
Compress-Archive -Path $origen -DestinationPath $destino
```

Objetivo de la tarea

El objetivo de la tarea es realizar un **respaldo automatizado y comprimido** de la carpeta **Descargas** de un usuario de Windows.

El uso de la fecha en el nombre del archivo (yyyy-MM-dd) permite:

- Mantener un histórico de copias diarias
- Evitar la sobrescritura de backups anteriores
- Facilitar la recuperación de versiones anteriores

Trigger y periodicidad

Disparador configurado: TimeTrigger

```
<CalendarTrigger>  
  <ScheduleByDay>  
    <DaysInterval>1</DaysInterval>  
  </ScheduleByDay>  
  <StartBoundary>2025-03-01T12:00:00</StartBoundary>  
</CalendarTrigger>
```

Características:

- **Periodicidad:** Diaria
- **Momento exacto:** Cada día a las 12:00 PM
- **Mecanismo:** Servicio **Task Scheduler (Programador de tareas)** de Windows
- **Equivalencia con cron:**
00 12 * * *

Seguridad

Usuario que ejecuta la tarea

```
<Principal>
<UserId>USUARIO-PC\usuario</UserId>
<LogonType>Password</LogonType>
<RunLevel>HighestAvailable</RunLevel>
</Principal>
```

- La tarea se ejecuta bajo el usuario usuario
- Usa credenciales almacenadas
- Se ejecuta con privilegios elevados

Riesgos de seguridad

Uso de credenciales almacenadas

- Windows guarda la contraseña de forma cifrada
- Si el sistema se ve comprometido, puede facilitar movimientos laterales

Privilegios elevados

- RunLevel=HighestAvailable
- Si el script es modificable por el usuario → **riesgo de escalada de privilegios**

Ruta del script

C:\Scripts\backup_descargas.ps1

- **Ruta segura si los permisos están bien configurados**
- **Riesgo si usuarios no administradores pueden modificar el script**

Llenado de disco

- **Se genera un archivo .zip nuevo cada día**
- **Si la carpeta Downloads es grande → consumo rápido de espacio**

Exposición de datos

- El directorio C:\Backups\ puede ser accesible por otros usuarios
- Riesgo si no se ajustan ACLs (permisos NTFS)

Logs generados

Logs del sistema

- Visor de eventos
 - Registro de Windows → Sistema
 - Microsoft-Windows-TaskScheduler/Operational

Se registra:

- Inicio de la tarea
- Finalización correcta o fallida
- Código de error

Salida del script

Por defecto:

- PowerShell **no guarda la salida en archivo**
- Solo queda reflejada como éxito o error en el Programador de tareas

Recomendación de logging (equivalente a >> backup.log 2>&1)

Modificar la acción de la tarea:

```
powershell.exe -ExecutionPolicy Bypass -File "C:\Scripts\backup_descargas.ps1" `>> "C:\Logs\backup_descargas.log" 2>&1
```

O dentro del script:

```
Start-Transcript -Path "C:\Logs\backup_descargas.log" -Append
```

Conclusión

La tarea cumple correctamente su función de respaldo automático, pero presenta **riesgos potenciales**:

- Uso de credenciales almacenadas
- Ejecución con privilegios elevados
- Posible llenado de disco
- Exposición de datos respaldados

Medidas recomendadas

- Ejecutar la tarea con una **cuenta de servicio**
- Evitar HighestAvailable si no es imprescindible
- Limitar permisos NTFS sobre scripts y backups
- Implementar rotación o limpieza automática de backups
- Habilitar logging persistente

Fortificación de scripts y rutas. Creación de usuarios de servicio

Linux

Identificar scripts en:

/usr/local/bin

/opt/scripts

/home/admin/scripts

Comandos de identificación

```
find /usr/local/bin /opt/scripts /home/admin/scripts -type f -name "*.sh" -o -perm /111
```

o

```
find / -type f -name "*.sh" -o -perm /111
```

Se revisan:

- Scripts ejecutables
- Scripts usados por cron o automatizaciones
- Propietario y permisos

Aplicar medidas de hardening obligatorias:

- **Permisos 700 para scripts propiedad del usuario que los ejecuta.**

```
chown autosvc:autosvc /opt/scripts/backup.sh
```

```
chmod 700 /opt/scripts/backup.sh
```

Solo el propietario puede:

- Leer
- Escribir
- Ejecutar

Ningún acceso para grupo u otros

- **No permitir scripts en home de usuarios normales.**

Ruta insegura:

```
/home/admin/scripts
```

Medidas aplicadas:

- Migración de scripts a ruta segura:

```
mv /home/admin/scripts/*.sh /opt/secure-automation/
```

Eliminación del directorio:

```
rmdir /home/admin/scripts
```

Justificación de seguridad:

- El HOME de usuarios es modificable
- Riesgo de escalada de privilegios si cron ejecuta scripts desde ahí

Añadir cabecera “trap” para evitar interrupciones: (trap "echo 'ERROR: Execution interrupted' >> /var/log/script.log" ERR INT TERM)

Cabecera obligatoria:

```
#!/bin/bash
```

```
set -e
```

```
trap "echo \"$(date '+%F %T') ERROR: Execution interrupted\" >> /var/log/script.log" ERR INT TERM
```

Protección frente a:

- Interrupciones manuales (CTRL+C)
- Señales del sistema
- Errores no controlados

Mejora la trazabilidad y auditoría

Reemplazar rutas relativas por absolutas.

Incorrecto:

```
tar -czf backup.tar.gz data/
```

Correcto:

```
/bin/tar -czf /opt/backups/backup.tar.gz /opt/data
```

Motivo:

- Evita ataques de PATH hijacking
- Garantiza ejecución predecible desde cron

Crear un **directorio securizado** para automatizaciones en el que el propietario sea root.

Directorio recomendado

/opt/secure-automation

Configuración de seguridad

mkdir /opt/secure-automation

chown root:root /opt/secure-automation

chmod 750 /opt/secure-automation

Características:

- Propietario: root
- Solo root puede modificar
- Usuarios de servicio pueden ejecutar si se les concede grupo

Crear un usuario de servicio sin Shell (llamado autosvc).

Usuario: autosvc

```
useradd -r -s /usr/sbin/nologin -d /nonexistent autosvc
```

Verificación:

```
getent passwd autosvc
```

Características del usuario de servicio:

- Sin shell interactiva
- Sin HOME real
- No puede iniciar sesión
- Uso exclusivo para automatizaciones

Configurar sudoers para permitir ejecutar **solo un comando concreto**

Requisito:

Permitir ejecutar **solo un comando concreto**

Edición segura de sudoers

visudo

Regla aplicada

autosvc ALL=(root) NOPASSWD: /opt/secure-automation/backup.sh

Restricción:

- autosvc NO puede ejecutar ningún otro comando
- NO acceso a hell root
- Solo ese script exacto

Reconfigurar cron para que ejecute el script bajo el usuario autosvc

Crontab del usuario autosvc

crontab -u autosvc -e

Ejemplo:

0 2 * * * /opt/secure-automation/backup.sh >> /var/log/backup.log 2>&1

Se ejecuta:

- Bajo usuario autosvc
- Con permisos limitados
- Con rutas absolutas
- Con logging

Validaciones finales de seguridad

- Comprobaciones

```
ls -l /opt/secure-automation
```

```
crontab -u autosvc -l
```

```
sudo -l -U autosvc
```

- Verificación de ejecución

```
grep CRON /var/log/syslog
```

Conclusión de hardening

Las automatizaciones quedan protegidas frente a:

- Manipulación de scripts
- Escalada de privilegios
- Interrupciones no controladas
- Abuso de sudo
- Persistencia insegura

Se cumple el principio de:

Menor privilegio + ejecución controlada + trazabilidad

Windows

Crear un directorio restringido para scripts

Directorio seguro recomendado

C:\SecureScripts

Creación:

```
New-Item -ItemType Directory -Path C:\SecureScripts
```

Motivo:

Evitar la ejecución de scripts desde rutas de usuario (C:\Users\...), que son fácilmente manipulables.

Configurar ACL estrictas:

- Solo Administradores + un usuario de servicio.
- Prohibir escritura a cualquier otro usuario.

Deshabilitar herencia de permisos

PowerShell: icacls C:\SecureScripts /inheritance:r

Asignar permisos explícitos

Suponiendo que el usuario de servicio se llama autosvc:

icacls C:\SecureScripts /grant "Administradores:(OI)(CI)F"

icacls C:\SecureScripts /grant "autosvc:(OI)(CI)RX"

Permisos otorgados:

- Administrators: Control total
- autosvc: Solo lectura y ejecución

Verificación

icacls C:\SecureScripts

Justificación de hardening

- Ningún usuario estándar puede modificar scripts
- El usuario de servicio **solo puede ejecutar**
- Se previene escalada de privilegios por manipulación de scripts

Firmar los scripts PowerShell con un certificado local:

```
Set-AuthenticodeSignature -FilePath "C:\SecureScripts\script.ps1" -Certificate (Get-ChildItem cert:\CurrentUser\My\XXXX)
```

Creación de certificado autofirmado (si no existe)

```
$cert = New-SelfSignedCertificate ` 
-Subject "CN=SecureScriptsSigning" ` 
-CertStoreLocation cert:\CurrentUser\My ` 
-Type CodeSigning
```

Obtener el Thumbprint:

```
$cert.Thumbprint
```

Firmar el script

```
Set-AuthenticodeSignature ` 
-FilePath "C:\SecureScripts\script.ps1" ` 
-Certificate $cert
```

Verificación de la firma

```
Get-AuthenticodeSignature C:\SecureScripts\script.ps1
```

Debe mostrar:

Status : Valid

Beneficios de la firma

- Garantiza integridad del script
- Impide ejecución si es modificado
- Permite aplicar políticas AllSigned o RemoteSigned

Crear un usuario de servicio

Usuario de servicio: autosvc

New-LocalUser "autosvc"

-**NoPassword**

-**AccountNeverExpires**

-**Description "Usuario de servicio para tareas automatizadas"**

Restringir inicio de sesión interactivo

Directiva de seguridad local →

Directivas locales →

Asignación de derechos de usuario →

Denegar inicio de sesión local

| Directiva | Configuración de seguri... |
|--|----------------------------|
| Actuar como parte del sistema operativo | Administradores |
| Administrar registro de seguridad y auditoría | SERVICIO LOCAL,Servici... |
| Agregar estaciones de trabajo al dominio | Administradores,Operad... |
| Ajustar las cuotas de la memoria para un proceso | Usuarios |
| Apagar el sistema | Administradores,Windo... |
| Aumentar el espacio de trabajo de un proceso | |
| Aumentar prioridad de programación | |
| Bloquear páginas en la memoria | SERVICIO LOCAL,Admin... |
| Cambiar la hora del sistema | SERVICIO LOCAL,Admin... |
| Cambiar la zona horaria | Administradores |
| Cargar y descargar controladores de dispositivo | SERVICIO LOCAL,Servici... |
| Crear objetos compartidos permanentes | Administradores |
| Crear objetos globales | Administradores |
| Crear un archivo de paginación | |
| Crear un objeto símbolo (token) | Administradores |
| Crear vínculos simbólicos | |
| Denegar el acceso a este equipo desde la red | |
| Denegar el inicio de sesión como servicio | |
| Denegar el inicio de sesión como trabajo por lotes | |
| Denegar el inicio de sesión local | |
| Denegar inicio de sesión a través de Servicios de Escritorio re... | |

Añadir:

autosvc

Resultado:

- El usuario NO puede iniciar sesión
- Solo puede ser usado por servicios/tareas

Modificar una tarea programada para usar este usuario SIN contraseña

Requisito clave

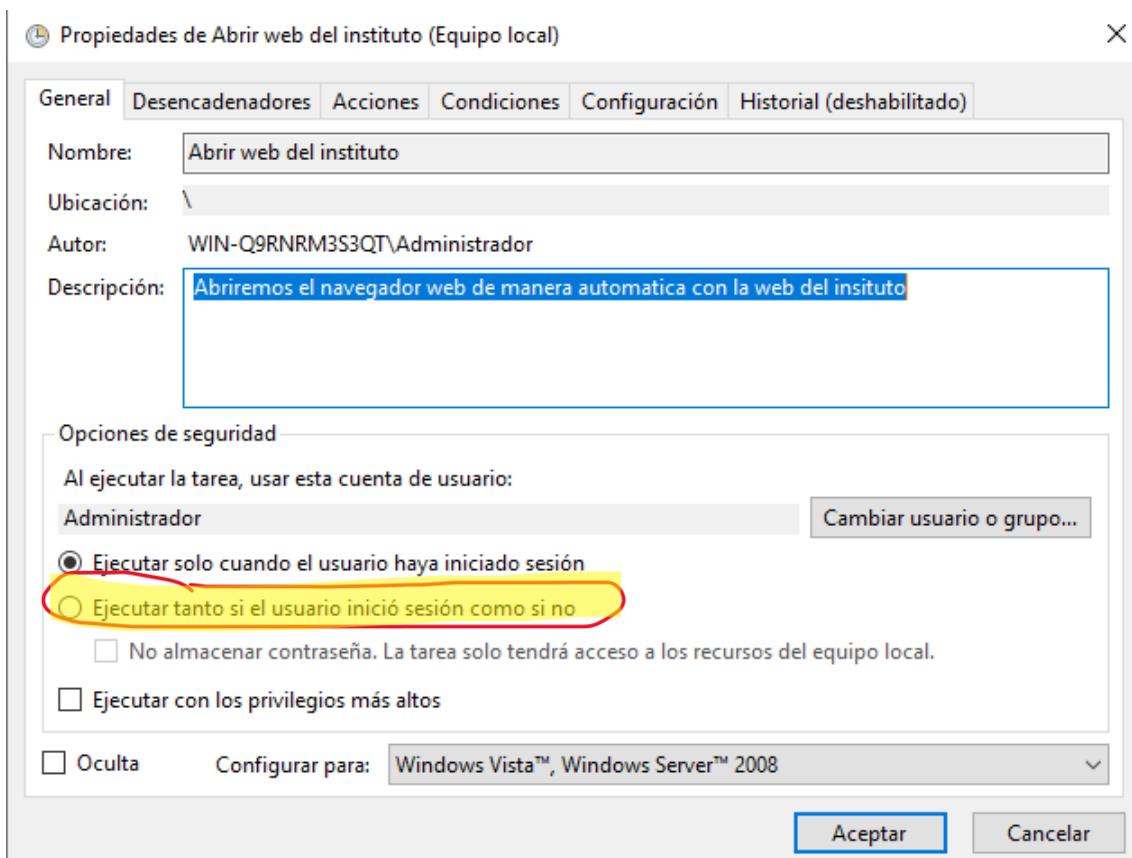
Ejecutar una tarea sin contraseña almacenada

En Windows esto se logra usando:

- **Cuenta local**
- Opción: “Run whether user is logged on or not”
- **Sin marcar “Store password”**

Configuración desde el Programador de tareas

1. Abrir **Task Scheduler**
2. Editar la tarea
3. Pestaña **General**:
 - Usuario: autosvc
 - *Run whether user is logged on or not*
 - *Do NOT store password*
 - *Do NOT use highest privileges (salvo necesidad)*



Ejecución del script firmado

Acción:

Program/script:

powershell.exe

Arguments:

-ExecutionPolicy AllSigned -File "C:\SecureScripts\script.ps1"

Importante:

- Se fuerza política AllSigned
- Solo scripts firmados pueden ejecutarse

Verificaciones finales

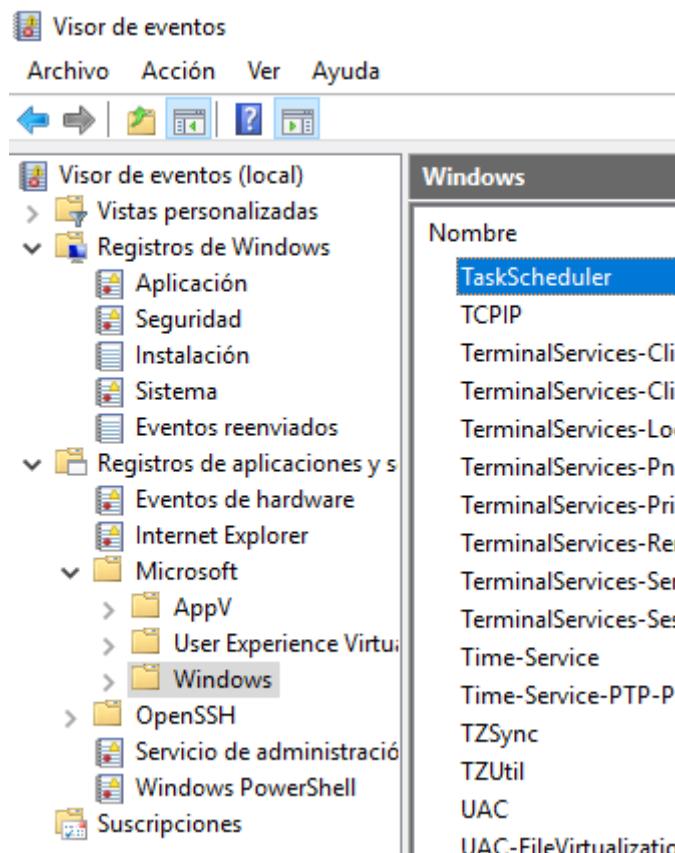
Comprobar identidad de ejecución

Dentro del script:

whoami >> C:\Logs\script.log

Revisar eventos

- Visor de eventos →
 - Microsoft → Windows → TaskScheduler → Operativo



Eventos clave:

- ID 100: tarea iniciada
- ID 102: tarea completada
- ID 101: error

Conclusión de hardening (Windows)

Con estas medidas se consigue:

- ✓ Ejecución desde rutas protegidas
- ✓ Control total por ACL
- ✓ Integridad mediante firma digital
- ✓ Usuario de servicio sin acceso interactivo
- ✓ Sin contraseñas almacenadas
- ✓ Principio de mínimo privilegio

Equivalente funcional y de seguridad a:

- chmod 700
- root + usuario de servicio
- sudoers restrictivo
- cron hardened en Linux

Detección de manipulación

Linux

Activar auditoría de ejecución de scripts (auditctl)

Objetivo

Detectar:

- Ejecución de scripts
- Modificaciones
- Uso indebido de automatizaciones
- Posibles intentos de escalada de privilegios

Activar el servicio de auditoría

apt update

apt install auditd -y

systemctl enable auditd

systemctl start auditd

Reglas de auditoría para scripts críticos

Verificación:

auditctl -s

Reglas de auditoría para scripts críticos

Suponiendo scripts en: **/opt/secure-automation**

Auditoría de ejecución

auditctl -a always,exit -F arch=b64 -S execve -F dir=/opt/secure-automation -k secure_scripts_exec

Auditoría de modificaciones

auditctl -w /opt/secure-automation -p wa -k secure_scripts_modify

Qué se audita:

- execve → ejecución de scripts
- w → escritura
- a → cambios de atributos

Persistencia de las reglas

Para que sobrevivan a reinicios:

nano /etc/audit/rules.d/secure-scripts.rules

Contenido:

```
-a always,exit -F arch=b64 -S execve -F dir=/opt/secure-automation -k  
secure_scripts_exec  
  
-w /opt/secure-automation -p wa -k secure_scripts_modify
```

Aplicar:

augenrules --load

Crear un sistema de logs centralizado para estos scripts (logger)

Objetivo

Tener trazabilidad propia **independiente de auditd**, similar a logs de aplicaciones.

Modificación del script auditado

Cabecera recomendada:

```
#!/bin/bash

set -e

SCRIPT_NAME=$(basename "$0")

logger -t SECURE_SCRIPT "Inicio ejecución: $SCRIPT_NAME - Usuario: $(whoami)"

trap "logger -t SECURE_SCRIPT \"ERROR: Ejecución interrumpida en
$SCRIPT_NAME\\"" ERR INT TERM
```

Ejemplo de logging de eventos

logger -t SECURE_SCRIPT "Backup iniciado en \$(date '+%F %T')"

logger -t SECURE_SCRIPT "Backup finalizado correctamente"

Ubicación de los logs

Dependiendo de la distro:

- **Debian / Ubuntu**

/var/log/syslog

Comprobación y análisis de logs (ausearch)

Buscar ejecuciones de scripts

```
ausearch -k secure_scripts_exec
```

Buscar modificaciones sospechosas

```
ausearch -k secure_scripts_modify.
```

Filtro por usuario

```
ausearch -k secure_scripts_exec -ua autosvc
```

Interpretación típica

Salida relevante:

```
type=EXECVE msg=audit(1700000000.123:456):  
    argc=2 a0="/bin/bash" a1="/opt/secure-automation/backup.sh"  
    uid=1001 auid=1001
```

Campos clave:

- uid → usuario efectivo
- auid → usuario original
- a0/a1 → comando ejecutado
- msg=audit → timestamp

Correlación auditd + logger

Ejemplo de investigación

Detectar ejecución:

```
ausearch -k secure_scripts_exec
```

Correlacionar con logs:

```
grep SECURE_SCRIPT /var/log/syslog
```

Permite:

- Saber **quién ejecutó**
- **Cuando**
- **Qué script**
- **Resultado**

Pruebas de detección de manipulación

Intento de modificación no autorizada

```
echo "malicious code" >> /opt/secure-automation/backup.sh
```

Resultado:

```
ausearch -k secure_scripts_modify
```

Evento registrado

Ejecución manual del script:

```
/opt/secure-automation/backup.sh
```

Resultado:

- Evento en auditd
- Entrada en syslog vía logger

Conclusión de detección de manipulación

Con esta configuración se consigue:

- ✓ Auditoría de ejecución y modificación
- ✓ Detección temprana de manipulación
- ✓ Correlación de eventos (auditd + logs)
- ✓ Evidencia forense válida
- ✓ Equivalente funcional a HIDS básico

Cumple los principios de:

Trazabilidad – Integridad – No repudio

Windows

Activar transcripción completa de PowerShell (Set-ItemProperty)

Objetivo

Registrar **todo lo que se ejecuta en PowerShell**:

- Comandos
- Scripts
- Parámetros
- Usuario
- Fecha y hora

Equivalente a:

auditctl + logger en Linux

1.1 Crear directorio seguro para transcripciones

```
New-Item -ItemType Directory -Path C:\PSLogs
```

```
icacls C:\PSLogs /inheritance:r
```

```
icacls C:\PSLogs /grant "Administrators:F"
```

Motivo:

Evitar manipulación de logs por usuarios estándar.

1.2 Activar transcripción vía registro (Set-ItemProperty)

```
Set-ItemProperty `  

-Path HKLM:\Software\Policies\Microsoft\Windows\PowerShell\Transcription `  

-Name EnableTranscripting `  

-Value 1 `  

-Type DWord
```

Configurar ruta de logs:

```
Set-ItemProperty `  

-Path HKLM:\Software\Policies\Microsoft\Windows\PowerShell\Transcription `  

-Name OutputDirectory `  

-Value "C:\PSLogs"
```

Incluir invocación de scripts:

```
Set-ItemProperty `n-Path HKLM:\Software\Policies\Microsoft\Windows\PowerShell\Transcription `n-Name IncludeInvocationHeader `n-Value 1 `n-Type DWord
```

Resultado

Cada ejecución de PowerShell genera un archivo .txt con:

- Usuario
- Script ejecutado
- Comandos exactos
- Fecha y hora

2. Activar registro de scripts firmados y no firmados

Objetivo

Detectar:

- Scripts PowerShell ejecutados
- Scripts **firmados vs no firmados**
- Código potencialmente malicioso

2.1 Activar Script Block Logging

```
Set-ItemProperty`
`-Path HKLM:\Software\Policies\Microsoft\Windows\PowerShell\ScriptBlockLogging
`-Name EnableScriptBlockLogging
`-Value 1
`-Type DWord
```

(Opcional, más agresivo):

```
Set-ItemProperty`
`-Path HKLM:\Software\Policies\Microsoft\Windows\PowerShell\ScriptBlockLogging
`-Name EnableScriptBlockInvocationLogging
`-Value 1
`-Type DWord
```

Qué se registra

- Código PowerShell completo
- Incluso código ofuscado
- Scripts firmados y NO firmados
- Ejecuciones desde memoria

Activar política de ejecución recomendada

(No es detección, pero refuerza el control)

Set-ExecutionPolicy AllSigned -Scope LocalMachine

Efecto:

- Solo se ejecutan scripts firmados
- Todo intento no firmado queda registrado

Comprobación mediante el Visor de eventos

4.1 Eventos de PowerShell

Ruta en el Visor de eventos

Registro de Aplicaciones y servicios

└─ PowerShell

Eventos relevantes

- **4104** → Script Block Logging (código ejecutado)
- **4103** → Módulos cargados
- **400 / 403** → Inicio / fin de sesión PowerShell

📌 Aquí puedes ver:

- Script completo
- Usuario
- Si estaba firmado o no

4.2 Auditoría de tareas programadas (Security Log)

Requisito

Debe estar activada la directiva:

Audit Object Access

Audit Other Object Access Events

secpol.msc → Configuración de Seguridad → Directivas locales → Directiva de auditoría

The screenshot shows the Windows Local Security Policy snap-in (secpol.msc). The left pane displays a tree view of security policies, including 'Configuración de seguridad', 'Directivas de cuenta', 'Directivas locales' (selected), 'Directiva de auditoría' (selected), 'Asignación de derechos de usuario', 'Opciones de seguridad', 'Windows Defender Firewall con seguridad', 'Directivas de Administrador de listas', 'Directivas de clave pública', 'Directivas de restricción de software', 'Directivas de control de aplicaciones', 'Directivas de seguridad IP en Equipo', and 'Configuración de directiva de auditoría' (selected). The right pane lists audit policies with their current configuration status:

| Directiva | Configuración de seguridad |
|---|----------------------------|
| Auditar el acceso a objetos | Sin auditoría |
| Auditar el acceso al servicio de directorio | Sin auditoría |
| Auditar el cambio de directivas | Sin auditoría |
| Auditar el seguimiento de procesos | Sin auditoría |
| Auditar el uso de privilegios | Sin auditoría |
| Auditar eventos de inicio de sesión | Sin auditoría |
| Auditar eventos de inicio de sesión de cuenta | Sin auditoría |
| Auditar eventos del sistema | Sin auditoría |
| Auditar la administración de cuentas | Sin auditoría |

Eventos solicitados

Evento Significado

4698 Creación de tarea programada

4699 Eliminación de tarea

4700 Tarea deshabilitada

4701 Tarea habilitada

Ruta en el Visor de eventos

Visor de eventos

 └─ Registros de Windows

 └─ Seguridad

The screenshot shows the Windows Event Viewer interface. The left pane displays a navigation tree with the following structure:

- Visor de eventos (local)
 - Vistas personalizadas
 - Registros de Windows
 - Aplicación
 - Seguridad** (selected)
 - Instalación
 - Sistema
 - Eventos reenviados
 - Registros de aplicaciones y s...
 - Suscripciones

The right pane is titled "Seguridad" and shows a list of 11,352 events. The table has three columns: "Palabras clave", "Fecha y hora", and "Orig". All events listed are of type "Auditoría correcta" and occurred at 28/12/2025 16:09:15 or 16:08:43.

| Palabras clave | Fecha y hora | Orig |
|--------------------|---------------------|------|
| Auditoría correcta | 28/12/2025 16:09:15 | Micr |
| Auditoría correcta | 28/12/2025 16:09:15 | Micr |
| Auditoría correcta | 28/12/2025 16:09:01 | Micr |
| Auditoría correcta | 28/12/2025 16:09:01 | Micr |
| Auditoría correcta | 28/12/2025 16:08:46 | Micr |
| Auditoría correcta | 28/12/2025 16:08:46 | Micr |
| Auditoría correcta | 28/12/2025 16:08:43 | Micr |
| Auditoría correcta | 28/12/2025 16:08:43 | Micr |
| Auditoría correcta | 28/12/2025 16:08:43 | Micr |
| Auditoría correcta | 28/12/2025 16:08:42 | Micr |
| Auditoría correcta | 28/12/2025 16:08:42 | Micr |
| Auditoría correcta | 28/12/2025 16:08:22 | Micr |

Ejemplo: Evento 4698

Campos clave:

- **Subject** → Usuario que crea la tarea
- **Task Name** → Nombre de la tarea
- **Task Content** → XML completo (acción, usuario, script)

 **Uso forense:**

- Detectar persistencia
- Detectar creación de tareas maliciosas
- Saber qué script se ejecuta y con qué permisos

Correlación de eventos (detección de manipulación)

Ejemplo de investigación

Detectar creación de tarea:

Evento 4698

Ver qué ejecuta:

Task Content → powershell.exe -File C:\SecureScripts\script.ps1

Ver ejecución real:

- Evento 4104 (PowerShell)
- Archivo de transcripción en C:\PSLogs

Resultado:

- Quién creó la tarea
- Qué script ejecuta
- Qué código real se ejecutó
- Cuándo y con qué usuario

Prueba de detección

6.1 Ejecutar script no firmado

`.\script.ps1`

Resultado:

- Bloqueo (por AllSigned)
- Evento 4104 registrado
- Transcripción generada

6.2 Crear tarea programada

`schtasks /create ...`

✓ Resultado:

- Evento 4698 en Seguridad
- XML visible

Conclusión de detección de manipulación (Windows)

Con esta configuración se consigue:

- ✓ Registro completo de PowerShell
- ✓ Detección de scripts firmados y no firmados
- ✓ Evidencia forense detallada
- ✓ Auditoría de persistencia por tareas
- ✓ Correlación clara de eventos

Equivalente funcional a:

- auditd
- ausearch
- logger
en Linux.

Ataque simulado

Linux:

Modificación de un script protegido

Ataque

Intentar modificar un script crítico:

```
echo "echo MALICIOUS CODE" >> /opt/secure-automation/backup.sh
```

Resultado esperado

-  Modificación **denegada**
- El usuario no tiene permisos de escritura

Detección

```
ausearch -k secure_scripts_modify
```

Evento registrado:

- Usuario que intentó modificar
- Archivo afectado
- Fecha y hora

Conclusión:

El sistema **detecta y registra** el intento de manipulación.

Inserción de PATH malicioso

Ataque

Modificar PATH para interceptar comandos:

```
export PATH=/tmp:$PATH
echo -e '#!/bin/bash\necho MALICIOUS TAR' > /tmp/tar
chmod +x /tmp/tar
```

Ejecutar script:

```
/opt/secure-automation/backup.sh
```

Resultado esperado

-  El ataque **no funciona**
- El script usa rutas absolutas (/bin/tar)

Detección

```
ausearch -k secure_scripts_exec
```

Se observa:

- Ejecución legítima
- Sin uso de binarios manipulados

Conclusión:

El uso de rutas absolutas **neutraliza el PATH hijacking**.

Ejecución del script como root sin autorización

Ataque

Intentar ejecutar el script como root desde un usuario normal:

```
sudo /opt/secure-automation/backup.sh
```

Resultado esperado

usuario is not allowed to execute this command

Detección

```
grep sudo /var/log/auth.log
```



Conclusión:

El sudoers restrictivo impide la escalada de privilegios.

Sustitución del script original

Ataque

Intentar reemplazar el script completo:

```
cp /tmp/malicious.sh /opt/secure-automation/backup.sh
```

Resultado esperado

-  Operación denegada (propietario root)
- Evento de auditoría registrado

```
ausearch -k secure_scripts_modify
```

Conclusión:

El control de permisos + auditd **bloquea y registra** el ataque.

Windows

Modificación de un script PowerShell protegido

Ataque

Modificar un script en C:\SecureScripts como usuario estándar:

```
Add-Content C:\SecureScripts\script.ps1 "Write-Output 'MALICIOUS'"
```

Resultado esperado

Access is denied

Detección

- Evento **4663** (intento de escritura)
- Log de PowerShell si se intenta forzar

❖ Conclusión:

Las ACL NTFS impiden manipulación directa.

Inserción de PATH malicioso

Ataque

`$env:PATH="C:\Temp;" + $env:PATH`

Crear binario falso:

`echo "MALICIOUS" > C:\Temp\tar.exe`

Ejecutar tarea programada.

Resultado esperado

-  No se ejecuta código malicioso
- La tarea usa rutas absolutas (C:\Windows\System32\)

Detección

- Evento **4104** (Script Block Logging)
- Transcripción completa en C:\PSLogs

Conclusión:

Rutas absolutas + logging evitan el ataque y lo registran.

Ejecución del script como administrador

Ataque

Intentar ejecutar manualmente:

Start-Process powershell -Verb RunAs

Ejecutar:

C:\SecureScripts\script.ps1

Resultado esperado

-  Bloqueo si no está firmado
- Política AllSigned aplicada

Detección

- Evento **4104**
- Evento **4688** (creación de proceso)

Conclusión:

La ejecución elevada no evita los controles de firma.

Sustitución de un archivo firmado

Ataque

Reemplazar el script firmado por otro no firmado:

```
Copy-Item C:\Temp\malicious.ps1 C:\SecureScripts\script.ps1 -Force
```

Resultado esperado

-  Acceso denegado por ACL
 - o, si se logra copiar como admin:

File is not digitally signed

Detección

- Evento **4104**
- Transcripción PowerShell
- Firma inválida al ejecutar:

```
Get-AuthenticodeSignature C:\SecureScripts\script.ps1
```

Conclusión:

La **integridad del código** está garantizada por firma digital.

Conclusión del ataque simulado

| Ataque | Linux | Windows |
|-------------------------|----------------------|-------------------|
| Modificación de scripts | ✗ Bloqueado + auditd | ✗ Bloqueado + ACL |
| PATH malicioso | ✗ Neutralizado | ✗ Neutralizado |
| Escalada de privilegios | ✗ sudoers | ✗ AllSigned |
| Sustitución de script | ✗ Detectado | ✗ Detectado |

Veredicto de seguridad

El sistema:

- **Detecta intentos de manipulación**
- **Bloquea escaladas de privilegios**
- **Registra evidencia forense**
- Cumple **hardening defensivo real**

Equivalente a una **prueba de intrusión controlada (pentest básico)** con resultado satisfactorio.