**SEGURIDAD DEL SERVIDOR**

**Índice**

[Introducción 3](#_Toc206720682)

[Prerrequisitos 4](#_Toc206720683)

[Acceso y SSH (sin root, sin contraseñas, solo llaves) 4](#_Toc206720684)

[Sudo más estricto y trazabilidad 6](#_Toc206720685)

[Parches de seguridad automáticos 7](#_Toc206720686)

[Logs persistentes, compresión y retención 8](#_Toc206720687)

[Firewall “deny-by-default” con nftables 9](#_Toc206720688)

[Parámetros del kernel (sysctl) para red y memoria 12](#_Toc206720689)

[Sistema de archivos: /tmp y /var/tmp sin ejecución 14](#_Toc206720690)

[AppArmor en modo “enforce” 15](#_Toc206720691)

[Fail2ban (bloqueo reactivo por logs) 15](#_Toc206720692)

[Auditoría básica con auditd 16](#_Toc206720693)

[Integridad de archivos con AIDE 17](#_Toc206720694)

[Pruebas de validación 18](#_Toc206720695)

[Acceso y SSH 18](#_Toc206720696)

[Sudo y logs 19](#_Toc206720697)

[Parches automáticos 19](#_Toc206720698)

[Logs persistentes 19](#_Toc206720699)

[Firewall 20](#_Toc206720700)

[Sysctl 20](#_Toc206720701)

[FS no ejecutable en /tmp 20](#_Toc206720702)

[AppArmor 21](#_Toc206720703)

[Fail2ban 21](#_Toc206720704)

[Auditd 21](#_Toc206720705)

[AIDE 22](#_Toc206720706)

# Introducción

En este documento se presentan medidas de hardening aplicadas a un servidor Ubuntu Server 24.04, abarcando control de acceso, endurecimiento de servicios críticos, firewall restrictivo, auditoría de eventos y detección de anomalías en archivos. Cada apartado incluye explicación, comandos necesarios y pruebas de validación, con el objetivo de dejar un servidor más seguro y confiable para entornos de desarrollo, pruebas o producción.

# Prerrequisitos

1. Servidor base instalado

* Ubuntu Server 24.04 limpio, actualizado (sudo apt update && sudo apt upgrade -y).

1. Acceso administrativo

* Una cuenta con privilegios de sudo (ej: admin).
* Acceso SSH funcionando (inicialmente con contraseña o llave).

1. Red básica funcionando

* El servidor debe tener conectividad a Internet para instalar paquetes.
* La interfaz de red configurada con IP fija o DHCP estable.

1. Paquetes de seguridad requeridos (se instalarán a lo largo del documento)

* nftables, fail2ban, auditd, aide, unattended-upgrades, apparmor y utilidades de sistema.
* build-essential (para compilar y probar bloqueo de ejecución en /tmp).

1. Servicios previos (opcional según proyecto)

* Si vas a aplicar hardening sobre un stack que ya tiene HAProxy, Docker/Swarm, MariaDB, GlusterFS o un CMS, esos servicios deben estar ya instalados antes de empezar con este documento.
* Este documento no instala esas aplicaciones, solo asegura y endurece el sistema que las aloja.

# Acceso y SSH (sin root, sin contraseñas, solo llaves)

El acceso remoto es uno de los puntos más críticos en un servidor, ya que es la puerta principal de administración. Si alguien logra romper una contraseña débil o entrar como root, puede comprometer todo el sistema. Por eso, la buena práctica es deshabilitar el acceso con contraseña, prohibir que root entre directamente y obligar a usar llaves SSH, que son mucho más seguras. Además, limitamos qué usuarios pueden conectarse y aplicamos parámetros de seguridad adicionales para reducir ataques de fuerza bruta.

En nuestro servidor se hará lo siguiente: generar llaves SSH desde la máquina del administrador, copiarlas al servidor, y luego configurar sshd para que solo el usuario admin pueda acceder usando llaves, nunca con contraseña.

**Paso previo:** Verificar que el servidor ya permite acceso SSH con usuario administrativo (ejemplo: ssh jose-admin@IP). Una vez confirmada la conexión, se procede a configurar el acceso seguro con llaves y a deshabilitar contraseñas.

# (En tu máquina local) genera una llave si no tienes

ssh-keygen -t ed25519 -a 100 -f ~/.ssh/id\_ed25519

# (En el servidor) crea el directorio y pega tu clave pública

sudo -u admin mkdir -p /home/admin/.ssh

echo 'PEGA\_AQUI\_TU\_CLAVE\_PUBLICA' | sudo tee -a /home/admin/.ssh/authorized\_keys

sudo chown -R admin:admin /home/admin/.ssh

sudo chmod 700 /home/admin/.ssh

sudo chmod 600 /home/admin/.ssh/authorized\_keys

# Crea un archivo de hardening de SSH (sin tocar el archivo principal)

sudo tee /etc/ssh/sshd\_config.d/01-hardening.conf >/dev/null <<'EOF'

PermitRootLogin no

PasswordAuthentication no

KbdInteractiveAuthentication no

ChallengeResponseAuthentication no

UsePAM yes

X11Forwarding no

AllowUsers admin

ClientAliveInterval 300

ClientAliveCountMax 2

LoginGraceTime 20

MaxAuthTries 3

# Cifrados y MACs modernos

Ciphers chacha20-poly1305@openssh.com,aes256-gcm@openssh.com

MACs hmac-sha2-512-etm@openssh.com,hmac-sha2-256-etm@openssh.com

EOF

sudo systemctl restart ssh

# Sudo más estricto y trazabilidad

El comando sudo permite que un usuario normal ejecute acciones de administrador. Si no se controla bien, puede ser un punto de entrada para abusos o errores graves. Vamos a aplicar dos medidas:

1. **Trazabilidad completa:** cada vez que alguien use sudo, quedará registrado en un log dedicado (/var/log/sudo.log). Esto ayuda a auditar quién hizo qué en el servidor.
2. **TTY obligatorio:** obligamos a que sudo solo pueda ejecutarse desde una terminal interactiva (TTY). Así evitamos que scripts maliciosos o automatizados lo usen de manera silenciosa.

Con esto se mejora la seguridad y se reduce el riesgo de escaladas no controladas.

# Log dedicado de sudo y TTY obligatorio

echo 'Defaults logfile="/var/log/sudo.log"' | sudo EDITOR='tee -a' visudo

echo 'Defaults requiretty' | sudo EDITOR='tee -a' visudo

# Parches de seguridad automáticos

Uno de los puntos más críticos en la seguridad de un servidor es mantenerlo actualizado. Muchas vulnerabilidades explotadas en servidores son fallos que ya tenían parches disponibles, pero que no fueron aplicados a tiempo.

Para mitigar ese riesgo:

1. **Activamos actualizaciones automáticas** con unattended-upgrades.
2. **Definimos reinicios controlados** de madrugada (cuando no hay usuarios trabajando), evitando que el servidor quede vulnerable o que un reinicio inesperado interrumpa servicios.

sudo apt update && sudo apt install -y unattended-upgrades apt-listchanges

# Habilita y configura

sudo dpkg-reconfigure -plow unattended-upgrades

# Refuerza la configuración (seguridad + reinicio nocturno)

sudo tee /etc/apt/apt.conf.d/51-hardening >/dev/null <<'EOF'

Unattended-Upgrade::Automatic-Reboot "true";

Unattended-Upgrade::Automatic-Reboot-Time "03:30";

APT::Periodic::Update-Package-Lists "1";

APT::Periodic::Unattended-Upgrade "1";

EOF

# Logs persistentes, compresión y retención

Los logs del sistema (que administra systemd-journald) son esenciales para investigar fallos y ataques. Si no se configuran bien, se pueden perder después de un reinicio o crecer tanto que llenen todo el disco.

Con esto aseguramos tres cosas:

1. **Persistencia** → los logs sobreviven reinicios (quedan guardados en disco).
2. **Compresión** → los registros viejos ocupan menos espacio.
3. **Límite de tamaño** → no dejar que crezcan sin control (máximo 500 MB en este caso).

Esto protege tanto la disponibilidad (no se llena el disco) como la trazabilidad (no se pierden evidencias).

sudo sed -i 's/^#\?Storage=.\*/Storage=persistent/' /etc/systemd/journald.conf

sudo sed -i 's/^#\?Compress=.\*/Compress=yes/' /etc/systemd/journald.conf

sudo sed -i 's/^#\?SystemMaxUse=.\*/SystemMaxUse=500M/' /etc/systemd/journald.conf

sudo systemctl restart systemd-journald

# Firewall “deny-by-default” con nftables

Un firewall “deny-by-default” parte de bloquear todo y luego permitir solo lo imprescindible; así se reduce al mínimo la superficie de ataque y se limita el movimiento lateral. Con nftables definimos sets de IPs/rangos confiables y reglas claras: aceptar conexiones ya establecidas, permitir SSH solo desde IPs de administración, abrir 80/443 para el balanceador/web, y autorizar puertos internos estrictamente necesarios para Swarm, GlusterFS, MariaDB y VRRP; todo lo demás cae con logging para auditoría. Esto se aplica a nivel de sistema (persistente) y se puede ajustar por entorno cambiando los rangos en los sets.

**Prerrequisitos mínimos**

* Ubuntu/Debian con nftables disponible (kernel ≥ 4.x).
* Conocer tus rangos reales: IPs de admin/bastión, subred de apps, subred de nodos.

sudo apt-get update && sudo apt-get install -y nftables

sudo systemctl enable --now nftables

sudo tee /etc/nftables.conf >/dev/null <<'EOF'

table inet filter {

sets {

cluster\_nodes { type ipv4\_addr; flags interval; elements = { 10.0.0.0/24, 192.168.1.0/24 } }

admin\_ips { type ipv4\_addr; elements = { 203.0.113.10 } } # ← AJUSTA: IP(s) de administración/Bastión

app\_subnet { type ipv4\_addr; elements = { 10.10.0.0/16 } } # ← AJUSTA: red de aplicaciones

}

chain input {

type filter hook input priority 0; policy drop;

ct state { established, related } accept

iifname "lo" accept

ip protocol icmp accept limit rate 5/second

# SSH solo desde admin

ip saddr @admin\_ips tcp dport 22 ct state new limit rate 10/minute accept

# HTTP/HTTPS públicos

tcp dport { 80, 443 } accept

# Swarm entre nodos

ip saddr @cluster\_nodes tcp dport { 2377, 7946 } accept

ip saddr @cluster\_nodes udp dport { 7946, 4789 } accept

# GlusterFS

ip saddr @cluster\_nodes tcp dport { 24007, 24008, 49152-49251 } accept

# MariaDB solo desde apps

ip saddr @app\_subnet tcp dport 3306 accept

# VRRP (Keepalived)

ip saddr @cluster\_nodes ip protocol vrrp accept

log prefix "DROP\_IN " flags all counter drop

}

chain forward {

type filter hook forward priority 0; policy drop;

ct state { established, related } accept

log prefix "DROP\_FWD " flags all counter drop

}

chain output {

type filter hook output priority 0; policy accept;

}

}

EOF

sudo systemctl reload nftables

# Parámetros del kernel (sysctl) para red y memoria

El kernel es el núcleo del sistema operativo y controla tanto la red como la memoria. A través de sysctl podemos ajustar parámetros de bajo nivel para reforzar la seguridad. Con estas reglas endurecemos la pila de red contra ataques comunes:

* **Redirects y source routing**: desactivados para que un atacante no pueda redirigir tráfico a voluntad.
* **SYN cookies**: activados para mitigar ataques SYN flood (DoS por conexiones medias abiertas).
* **Log de paquetes sospechosos (martians)**: permite registrar tráfico con IP inválida.
* **IPv6**: también se protege contra redirects.
* **Colas de red y memoria**: se limitan para que un atacante no sature la RAM creando miles de conexiones.
* **Core dumps de binarios con SUID/SGID**: deshabilitados para que no se filtren datos sensibles si un proceso con privilegios falla.
* **ASLR (Address Space Layout Randomization)**: activado para dificultar exploits de memoria.

sudo tee /etc/sysctl.d/90-hardening.conf >/dev/null <<'EOF'

# IPv4

net.ipv4.conf.all.accept\_redirects = 0

net.ipv4.conf.default.accept\_redirects = 0

net.ipv4.conf.all.send\_redirects = 0

net.ipv4.conf.default.send\_redirects = 0

net.ipv4.conf.all.accept\_source\_route = 0

net.ipv4.conf.default.accept\_source\_route = 0

net.ipv4.tcp\_syncookies = 1

net.ipv4.conf.all.log\_martians = 1

net.ipv4.conf.default.log\_martians = 1

# IPv6 (endurecido, sin deshabilitarlo)

net.ipv6.conf.all.accept\_redirects = 0

net.ipv6.conf.default.accept\_redirects = 0

# Limita uso de memoria por colas de red (evita DoS por buffers)

net.core.somaxconn = 1024

net.core.netdev\_max\_backlog = 250000

# Desactiva core dumps de SUID/SGID

fs.suid\_dumpable = 0

kernel.randomize\_va\_space = 2

EOF

sudo sysctl --system

# Sistema de archivos: /tmp y /var/tmp sin ejecución

Los directorios /tmp y /var/tmp en Linux se utilizan para almacenar archivos temporales que generan los programas y servicios. El problema es que, por defecto, cualquier usuario (incluso sin permisos especiales) puede escribir ahí. Si un atacante logra subir un script o binario malicioso en estas carpetas, podría ejecutarlo directamente y comprometer el sistema.

Estos directorios con opciones restrictivas:

* **noexec**: bloquea la ejecución de binarios desde ahí.
* **nosuid**: impide que archivos con el bit SUID se ejecuten con privilegios.
* **nodev**: evita que se creen dispositivos especiales dentro.
* **mode=1777**: mantiene el comportamiento estándar de “directorio público temporal” (todos pueden escribir, pero solo el dueño puede borrar lo suyo).

# Monta /tmp y /var/tmp como tmpfs endurecido

echo 'tmpfs /tmp tmpfs defaults,rw,nosuid,nodev,noexec,mode=1777 0 0' | sudo tee -a /etc/fstab

echo 'tmpfs /var/tmp tmpfs defaults,rw,nosuid,nodev,noexec,mode=1777 0 0' | sudo tee -a /etc/fstab

sudo mount -o remount /tmp || sudo mount /tmp

sudo mount -o remount /var/tmp || sudo mount /var/tmp

# AppArmor en modo “enforce”

AppArmor es un sistema de seguridad del kernel que aplica perfiles de control a cada aplicación para limitar qué archivos, recursos o redes puede usar. Sirve para reducir el impacto de un ataque, porque aunque un servicio como Apache o MariaDB sea comprometido, el atacante queda confinado y no puede acceder a todo el servidor. En nuestro caso lo usaremos verificando que AppArmor esté activo y en modo **enforce**, asegurando que las restricciones se apliquen de forma obligatoria; si no lo está, lo activamos con sudo systemctl enable --now apparmor.

sudo aa-status | head -n 5

# Si aparece "profiles are in enforce mode" y perfiles cargados, ok.

# Habilita en caso de estar inactivo

sudo systemctl enable --now apparmor

# Fail2ban (bloqueo reactivo por logs)

Analiza los logs del sistema en busca de intentos de acceso fallidos o sospechosos (por ejemplo, múltiples intentos de contraseña en poco tiempo). Su propósito es bloquear automáticamente la IP atacante durante un tiempo definido, evitando ataques de fuerza bruta y accesos no autorizados. En nuestro caso lo usamos sobre SSH, configurando que si un usuario falla 5 veces en 10 minutos, la IP quede bloqueada por 1 hora; esto refuerza la seguridad sin intervención manual.

sudo apt install -y fail2ban

# Jail básica para sshd (ban 1 hora si 5 fallos en 10 min)

sudo tee /etc/fail2ban/jail.d/sshd-hardening.conf >/dev/null <<'EOF'

[sshd]

enabled = true

port = ssh

filter = sshd

logpath = /var/log/auth.log

maxretry = 5

findtime = 10m

bantime = 1h

backend = systemd

EOF

sudo systemctl enable --now fail2ban

sudo fail2ban-client status sshd

# Auditoría básica con auditd

Auditd es un servicio que permite auditar y registrar eventos críticos del sistema, como cambios en usuarios, contraseñas, permisos o archivos sensibles. Su propósito es tener trazabilidad completa de acciones importantes, útil para detectar intrusiones o malas configuraciones. En nuestro caso, lo usamos para monitorear archivos clave como /etc/passwd, /etc/shadow, /etc/sudoers y el directorio webroot, de modo que cualquier modificación quede registrada en los logs de auditoría.

sudo apt install -y auditd audispd-plugins

sudo systemctl enable --now auditd

# Reglas de auditoría focalizadas

sudo tee /etc/audit/rules.d/hardening.rules >/dev/null <<'EOF'

-w /etc/passwd -p wa -k identity

-w /etc/group -p wa -k identity

-w /etc/shadow -p wa -k identity

-w /etc/sudoers -p wa -k sudo

-w /etc/sudoers.d/ -p wa -k sudo

-w /var/www/html/ -p wa -k webroot

-w /var/log/auth.log -p wa -k auth

EOF

sudo augenrules --load

sudo systemctl restart auditd

# Integridad de archivos con AIDE

Es una herramienta que verifica la integridad de archivos críticos del sistema, comparando su estado actual con una base de datos inicial. Sirve para detectar cambios inesperados en binarios o configuraciones sensibles que podrían indicar un ataque o manipulación maliciosa. En este caso lo instalamos, generamos la base de datos inicial y programamos una tarea diaria que revisa automáticamente si hubo modificaciones sospechosas en los archivos vigilados.

sudo apt install -y aide

sudo aideinit

sudo mv /var/lib/aide/aide.db.new /var/lib/aide/aide.db

# Tarea diaria

echo '0 4 \* \* \* root /usr/bin/aide --config=/etc/aide/aide.conf --check > /var/log/aide/last-check.log 2>&1' | sudo tee /etc/cron.d/aide-check

# Pruebas de validación

## Acceso y SSH

Verificamos que solo el usuario autorizado pueda acceder mediante llaves SSH, mientras que root y contraseñas quedan bloqueados

# Debe fallar login de root

ssh root@TU\_SERVIDOR

# Debe fallar contraseña (si intentas forzar password)

ssh -o PreferredAuthentications=password -o PubkeyAuthentication=no admin@TU\_SERVIDOR

# Debe permitir con tu llave

ssh admin@TU\_SERVIDOR

## Sudo y logs

Confirmamos que todo uso de sudo quede registrado en un log dedicado, garantizando trazabilidad.

# Revisar que sudo esté registrando todo en /var/log/sudo.log

sudo tail -n 10 /var/log/sudo.log

## Parches automáticos

Validamos que el sistema aplique actualizaciones de seguridad automáticamente y queden registradas

sudo systemctl status unattended-upgrades --no-pager

grep -E "Unattended-Upgrade" /var/log/dpkg.log | tail

## Logs persistentes

Comprobamos que los logs sobreviven a reinicios y tienen compresión y límites de tamaño configurados

sudo ls -ld /var/log/journal # Debe existir (persistente)

journalctl -b | head -n 3

## Firewall

Revisamos que solo estén abiertos los puertos necesarios y que las reglas “deny-by-default” estén activas

sudo nft list ruleset | less

ss -tulpen | grep -E ':22|:80|:443|:2377|:7946|:4789|:24007|:24008|:4915|:3306' || true

# Desde IP NO autorizada, probar que 22 no responde

## Sysctl

Verificamos que los parámetros del kernel estén endurecidos contra ataques de red y fugas de memoria

sysctl net.ipv4.tcp\_syncookies

sysctl net.ipv4.conf.all.accept\_redirects

sysctl fs.suid\_dumpable

## FS no ejecutable en /tmp

Comprobamos que no se pueda ejecutar código desde temporales, bloqueando vectores comunes de malware

echo 'int main(){return 0;}' > /tmp/t.c && gcc /tmp/t.c -o /tmp/t 2>/dev/null || true

/tmp/t || echo "OK: noexec impide ejecución en /tmp"

## AppArmor

Validamos que AppArmor esté cargado y en modo **enforce**, aplicando perfiles de seguridad

sudo aa-status | sed -n '1,25p'

## Fail2ban

Probamos que se bloqueen IPs que intentan fuerza bruta en SSH tras múltiples intentos fallidos

sudo fail2ban-client status

sudo fail2ban-client status sshd

# Intenta 5 logins fallidos y verifica "Banned" para tu IP

## Auditd

Confirmamos que los cambios críticos en usuarios, sudoers o webroot quedan auditados

# Toca un archivo vigilado y verifica registro

sudo usermod -c "prueba" admin

sudo ausearch -k identity | tail

## AIDE

Probamos que el sistema detecte modificaciones inesperadas en binarios y configuraciones sensibles

sudo /usr/bin/aide --config=/etc/aide/aide.conf --check | head -n 50