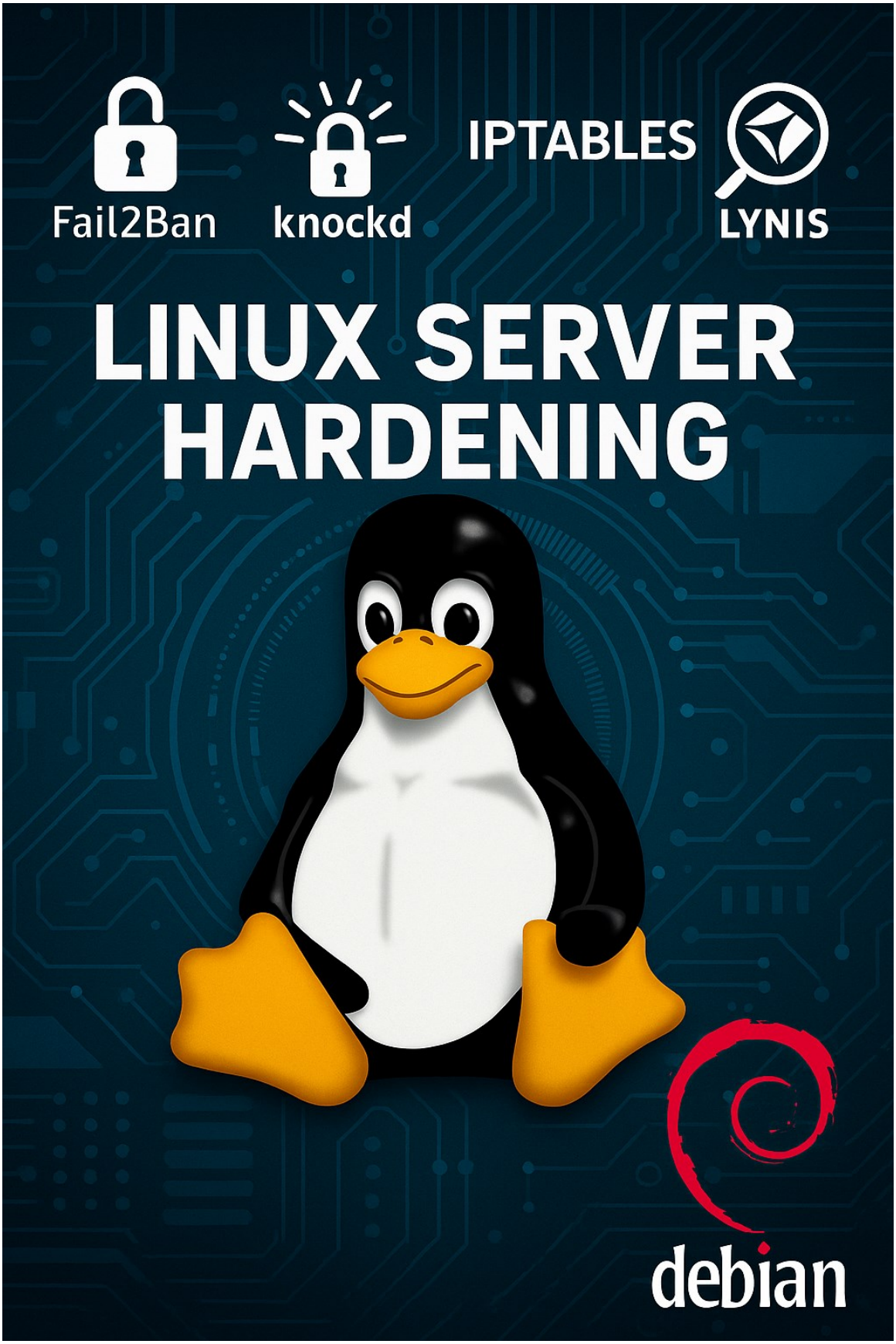


Autor: Javier Cuenca Pérez
Proyecto personal para publicar en Linkedin

🗂️ Índice de la Documentación Técnica

Autor: Javier Cuenca Pérez
Proyecto personal para publicar en Linkedin

1. Objetivo del proyecto

El presente proyecto tiene como objetivo aplicar una serie de **medidas prácticas de hardening** (endurecimiento) sobre un servidor Debian autogestionado, que ya cuenta con los siguientes servicios desplegados previamente:

- Nextcloud en contenedores Docker
- Acceso remoto cifrado mediante VPN WireGuard

Estas medidas están orientadas a proteger el sistema frente a accesos no autorizados, vulnerabilidades comunes y ataques automatizados, mejorando así la seguridad de entornos personales, de laboratorio o pequeñas empresas sin necesidad de soluciones comerciales externas.

En concreto, se busca:

- Aplicar configuraciones básicas de refuerzo de seguridad del sistema operativo
- Implementar protección frente a ataques de fuerza bruta
- Restringir y controlar el acceso remoto mediante firewall y técnicas avanzadas como port-knocking
- Realizar auditorías básicas para evaluar el estado de seguridad del sistema
- Documentar todo el proceso paso a paso con ejemplos, configuraciones y capturas de pantalla

Este documento sirve tanto como guía técnica para replicar el entorno, como demostración práctica de habilidades en **administración de sistemas Linux, ciberseguridad básica y despliegue seguro de servicios autogestionados**.

---

2. Requisitos previos

Antes de aplicar las medidas de hardening descritas en este proyecto, es necesario contar con un entorno base funcional. A continuación se detallan los requisitos mínimos recomendados:

2.1. Sistema operativo

- Debian 12 (Bookworm) o versión estable equivalente
- Instalación mínima del sistema (sin entorno gráfico)

2.2. Usuario y acceso

- Usuario con privilegios de sudo correctamente configurado
- Acceso SSH funcional al sistema
- Acceso local o temporal como root (solo para configuraciones puntuales)

2.3. Servicios ya desplegados (preexistentes)

- **Nextcloud** desplegado en contenedores Docker
- **WireGuard** configurado como VPN para acceso remoto seguro
- Acceso desde el móvil o PC al entorno a través de la VPN

2.4. Conectividad

- Conexión a Internet funcional para instalación de paquetes
- Herramientas de red básicas instaladas (curl, ping, netstat, etc.)

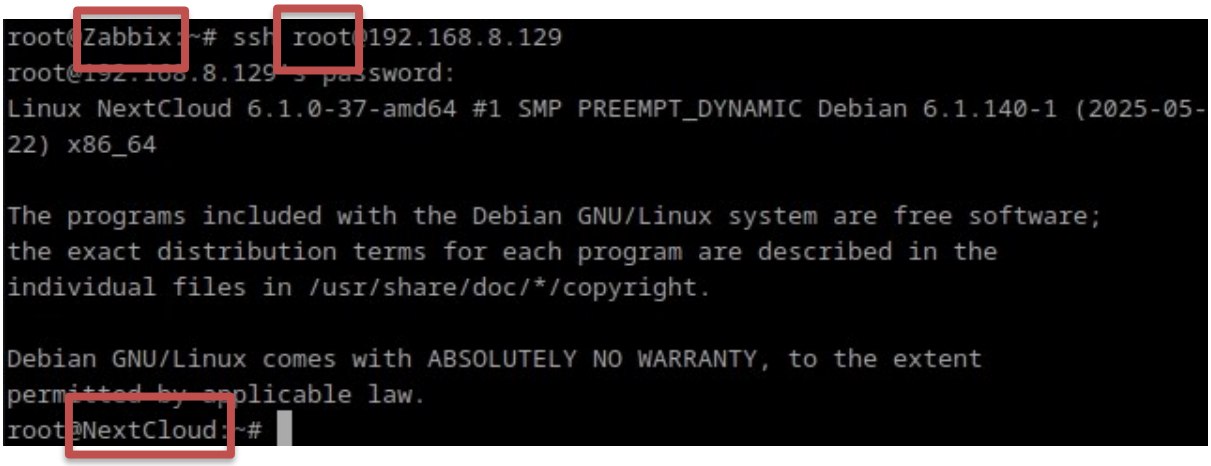
2.5. Recomendaciones adicionales

- Snapshot de la máquina virtual antes de comenzar (si se usa VirtualBox o similar)
- Conexión alternativa (como consola local o modo recovery) por si se pierde acceso SSH

3. Desactivar el acceso SSH como root

Una de las primeras y más efectivas medidas de seguridad en un sistema Linux es **prohibir el acceso remoto directo como usuario root**. Esto evita que atacantes puedan intentar conectarse usando el nombre de usuario más privilegiado del sistema.

Primero verificaremos si esta habilitado el acceso a root por ssh, aprovecharemos que tenemos la maquina de Zabbix (192.168.8.130) para conectar a la máquina de NextCloud (192.168.8.129)



Como tiene acceso vamos a modificar el archivo /etc/ssh/sshd_config para cambiarle algunos valores

También modificaremos el PasswordAuthentication para impedir acceso con claves y el PermitEmptyPassword. Guardamos con Ctrl+O y salimos con Ctrl+X

```
  GNU nano 7.2                                    /etc/ssh/sshd_config *
# To disable tunneled clear text passwords, change to no here!
PasswordAuthentication no
PermitEmptyPasswords no
```

Reiniciamos le servicio sshd

```
root@NextCloud:~# systemctl restart sshd
root@NextCloud:~#
```

Volvemos a verificar la conexión ssh desde Zabbix

Solo sin acceso de root

```
root@Zabbix:~# ssh root@192.168.8.129
root@192.168.8.129's password:
Permission denied, please try again.
```

Sin acceso de root y sin acceso con claves ssh

```
root@Zabbix:~# ssh root@192.168.8.129
root@192.168.8.129: Permission denied (publickey).
root@Zabbix:~#
```

Ahora verificamos que se puede entrar con un usuario normal

```
root@Zabbix:~# ssh javi@192.168.8.129
javi@192.168.8.129's password:
Linux NextCloud 6.1.0-37-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.140-1 (2025-05-
22) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon Jun  2 00:15:11 2025 from 192.168.8.130
javi@NextCloud:~$
```


4. Instalación y configuración de Fail2Ban

Fail2Ban es una herramienta que analiza los archivos de log en busca de intentos de acceso fallidos y aplica acciones de bloqueo (como reglas de firewall) para proteger el sistema ante ataques de fuerza bruta, especialmente en servicios expuestos como SSH.

Instalación del paquete

```
root@NextCloud:~# apt update && apt install fail2ban -y
Obj:1 http://deb.debian.org/debian bookworm InRelease
Obj:2 http://security.debian.org/debian-security bookworm-security InRelease
```

Comprobamos que el servicio se activa automáticamente, aunque con errores de código, ahora vamos a solucionarlo

```
root@NextCloud:~# systemctl status fail2ban
× fail2ban.service - Fail2Ban Service
     Loaded: loaded (/lib/systemd/system/fail2ban.service; enabled; preset: enabled)
     Active: failed (Result: exit-code) since Tue 2025-06-03 13:38:29 CEST; 1min 25s ago
   Duration: 160ms
       Docs: man:fail2ban(1)
    Process: 4684 ExecStart=/usr/bin/fail2ban-server -xf start (code=exited, status=255/EXCEPTION)
   Main PID: 4684 (code=exited, status=255/EXCEPTION)
        CPU: 111ms

jun 03 13:38:29 NextCloud systemd[1]: Started fail2ban.service - Fail2Ban Service.
jun 03 13:38:29 NextCloud fail2ban-server[4684]: 2025-06-03 13:38:29,156 fail2ban.configreader   [4684]: WARNIN
jun 03 13:38:29 NextCloud fail2ban-server[4684]: 2025-06-03 13:38:29,170 fail2ban               [4684]: ERROR
jun 03 13:38:29 NextCloud fail2ban-server[4684]: 2025-06-03 13:38:29,174 fail2ban               [4684]: ERROR
jun 03 13:38:29 NextCloud systemd[1]: fail2ban.service: Main process exited, code=exited, status=255/EXCEPTION
jun 03 13:38:29 NextCloud systemd[1]: fail2ban.service: Failed with result 'exit-code'.
```

Copiaremos el archivo original de configuración y crearemos uno de manera local para no modificar el original

```
root@NextCloud:~# cp /etc/fail2ban/jail.conf /etc/fail2ban/jail.local
root@NextCloud:~#
```

Buscamos la sección que no esté comentada(activa) del sshd y agregamos estas líneas

```
 GNU nano 7.2                                /etc/fail2ban/jail.local *

[sshd]

# To use more aggressive sshd modes set filter parameter "mode" in jail.local:
# normal (default), ddos, extra or aggressive (combines all).
# See "tests/files/logs/sshd" or "filter.d/sshd.conf" for usage example and details.
#mode   = normal
port    = ssh
backend = systemd
enabled = true
maxretry = 3
bantime = 1h
findtime = 10m
```


maxretry: número máximo de intentos
findtime: Periodo de observación de los intentos
Si se hacen 3 intentos en un periodo inferior a 10 minutos te bloquea
bandtime: Duracion del bloqueo

```
root@NextCloud:~# systemctl restart fail2ban
```

Verificamos el servicio

```
root@NextCloud:~# systemctl status fail2ban
● fail2ban.service - Fail2Ban Service
     Loaded: loaded (/lib/systemd/system/fail2ban.service; enabled; preset: enabled)
     Active: active (running) since Tue 2025-06-03 14:06:47 CEST; 26s ago
       Docs: man:fail2ban(1)
   Main PID: 5518 (fail2ban-server)
      Tasks: 5 (limit: 4613)
     Memory: 17.1M
        CPU: 161ms
     CGroup: /system.slice/fail2ban.service
             └─5518 /usr/bin/python3 /usr/bin/fail2ban-server -xf start
```

Verificamos el fail2ban

```
root@NextCloud:~# fail2ban-client status sshd
Status for the jail: sshd
|- Filter
|  |- Currently failed: 0
|  |- Total failed:     0
|  `- Journal matches:  _SYSTEMD_UNIT=sshd.service + _COMM=sshd
`- Actions
   |- Currently banned: 0
   |- Total banned:     0
   `- Banned IP list:
root@NextCloud:~#
```

Probando su funcionamiento
Vamos hacer una breve prueba del comportamiento del fail2ban con fallos repetidos de ssh desde la maquina Zabbix

```
Connection to 192.168.8.129 closed.
root@Zabbix:~# ssh javier@192.168.8.129
javier@192.168.8.129's password:
Permission denied, please try again.
javier@192.168.8.129's password:
Permission denied, please try again.
javier@192.168.8.129's password:
^Z
[5]+  Stopped                 ssh javier@192.168.8.129
```


Tras 3 intentos a un usuario que no existe el fail2ban banea mi ip durante 1h

```
root@NextCloud:~# fail2ban-client status sshd
Status for the jail: sshd
|- Filter
|  |- Currently failed: 0
|  |- Total failed:     3
|  `- Journal matches:  _SYSTEMD_UNIT=sshd.service + _COMM=sshd
`- Actions
   |- Currently banned: 1
   |- Total banned:     1
   `- Banned IP list:   192.168.8.130
root@NextCloud:~#
```

Ahora meteremos el comando para desbanear la ip

```
root@NextCloud:~# fail2ban-client set sshd unbanip 192.168.8.130
1
root@NextCloud:~#
```

Y listo, la ip baneada fue borrada de la lista “negra”

```
root@NextCloud:~# fail2ban-client status sshd
Status for the jail: sshd
|- Filter
|  |- Currently failed: 0
|  |- Total failed:     3
|  `- Journal matches:  _SYSTEMD_UNIT=sshd.service + _COMM=sshd
`- Actions
   |- Currently banned: 0
   |- Total banned:     1
   `- Banned IP list:
root@NextCloud:~#
```


5. Configuración del firewall con iptables

El firewall es una de las herramientas más críticas para la seguridad de un servidor. En este proyecto se utilizará iptables, el sistema clásico de filtrado de paquetes en Linux, para:

- Restringir conexiones a puertos no autorizados
- Permitir solo servicios necesarios: SSH, HTTP/HTTPS, VPN
- Aplicar una política por defecto de denegación

Al hacer #iptables -L -n -v //vemos un **gran fallo de seguridad** porque la política por defecto de INPUT es aceptar todo tipo de conexiones por el puerto que sea.

```
root@NextCloud:~# iptables -L -n -v
# Warning: iptables-legacy tables present, use iptables-legacy to see them
Chain INPUT (policy ACCEPT 81655 packets, 12M bytes)
 pkts bytes target     prot opt in     out     source               destination
   44  7912 f2b-sshd   6    --  *      *       0.0.0.0/0            0.0.0.0/0           multiport dports 22
  863 66164 ACCEPT     17   --  *      *       0.0.0.0/0            0.0.0.0/0           udp dpt:51820
```

Vamos a bloquearlo todo y crear reglas para aceptar solo las que nos interesa

```
root@NextCloud:~# iptables -P INPUT DROP
```

Ya esta definido por defecto como DROP el INPUT, como ya tenimaos definidas l as trglas del 22(ssh) y el 51820(WireGuard), vamos a añadir las reglas del puerto 80(http) y 443(https) para futuros proyectos

```
Chain INPUT (policy DROP 178 packets, 15529 bytes)
 pkts bytes target     prot opt in     out     source               destination
   44  7912 f2b-sshd   6    --  *      *       0.0.0.0/0            0.0.0.0/0           multiport dports 22
  909 69620 ACCEPT     17   --  *      *       0.0.0.0/0            0.0.0.0/0           udp dpt:51820
```

INCIDENCIA: Al hacer el DROP general del INPUT estamso rechazando tb las respuesta del DNS con lo cual no podemos tener salida a internet, vamos a solucionarlo metiendo manualmente las reglas
(detectando fallo)

```
root@NextCloud:~# ping -c 3 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
^Z
[7]+  Detenido                 ping -c 3 8.8.8.8
```

Añadiendo reglas para permitir todas respuestas legitimas a conexiones que hayamos hecho

```
root@NextCloud:~# iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
```


Permitir tambien el trafico de manera local

```
root@NextCloud:~# iptables -A INPUT -i lo -j ACCEPT
```

Acceso a internet recuperado

```
root@NextCloud:~# ping -c 3 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=116 time=51.5 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=116 time=38.6 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=116 time=54.7 ms

--- 8.8.8.8 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2019ms
```

Acceso local

```
root@NextCloud:~# ping -c 3 192.168.8.130
PING 192.168.8.130 (192.168.8.130) 56(84) bytes of data.
64 bytes from 192.168.8.130: icmp_seq=1 ttl=64 time=0.368 ms
64 bytes from 192.168.8.130: icmp_seq=2 ttl=64 time=0.465 ms
64 bytes from 192.168.8.130: icmp_seq=3 ttl=64 time=0.478 ms

--- 192.168.8.130 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2084ms
```

Bien, ya podemos agregar las reglas para el 80 y el 443

```
root@NextCloud:~# iptables -A INPUT -p tcp --dport 80 -j ACCEPT
root@NextCloud:~# iptables -A INPUT -p tcp --dport 443 -j ACCEPT
```

Verificación

```
root@NextCloud:~# iptables -L -n -v
# Warning: iptables-legacy tables present, use iptables-legacy to see them
Chain INPUT (policy DROP 1356 packets, 119K bytes)
 pkts bytes target     prot opt in     out     source               destination
   80 10384 f2b-sshd   6    --  *      *       0.0.0.0/0            0.0.0.0/0           multiport dports 22
  970 74324 ACCEPT     17   --  *      *       0.0.0.0/0            0.0.0.0/0           udp dpt:51820
 1427  531K ACCEPT     0    --  *      *       0.0.0.0/0            0.0.0.0/0           ctstate RELATED,ESTABLISHED
    1    84 ACCEPT     0    --  lo     *       0.0.0.0/0            0.0.0.0/0
    0     0 ACCEPT     6    --  *      *       0.0.0.0/0            0.0.0.0/0           tcp dpt:80
    0     0 ACCEPT     6    --  *      *       0.0.0.0/0            0.0.0.0/0           tcp dpt:443
```

Haciendo persistente las reglas

```
root@NextCloud:~# netfilter-persistent save
run-parts: executing /usr/share/netfilter-persistent/plugins.d/15-ip4tables save
# Warning: iptables-legacy tables present, use iptables-legacy-save to see them
run-parts: executing /usr/share/netfilter-persistent/plugins.d/25-ip6tables save
```


Conclusión:

El servidor ahora tiene una **política de seguridad fuerte y realista**:

- Bloquea todo lo innecesario
- Deja pasar solo lo que tú has definido explícitamente
- Está listo para producción o pruebas reales

6. Escaneo de seguridad con Lynis

Lynis es una herramienta de auditoría de seguridad para sistemas Linux. Permite evaluar el nivel de fortaleza del sistema y proporciona recomendaciones prácticas para mejorar la configuración y seguridad general.
Este escaneo sirve como **punto de control** para verificar que las medidas aplicadas (como el hardening SSH, el firewall y Fail2Ban) están siendo detectadas y valoradas correctamente.

Instalamos Lynis

```
root@NextCloud:~# apt update && apt install lynis -y
Obj:1 http://security.debian.org/debian-security bookworm-security InRelease
Obj:2 http://deb.debian.org/debian bookworm InRelease
```

Ejecutando una auditoria básica, este procedimiento puede tardar de 1 a 2 minutos, saldrán muchas linear de comprobación y al finalizar Lynis nos mostrara un resultado dando una puntuación, recomendaciones, advertencias detectadas etc..

```
root@NextCloud:~# lynis audit system

[ Lynis 3.0.8 ]

################################################################################
```

Resultado:

```
Lynis security scan details:

Hardening index : 64 [############        ]
Tests performed : 259
Plugins enabled : 1

Components:
- Firewall               [V]
- Malware scanner        [X]

Scan mode:
Normal [V]  Forensics [ ]  Integration [ ]  Pentest [ ]

Lynis modules:
- Compliance status      [?]
- Security audit         [V]
- Vulnerability scan     [V]

Files:
- Test and debug information      : /var/log/lynis.log
- Report data                     : /var/log/lynis-report.dat
```


Conclusion:

Tras ejecutar el análisis con lynis audit system, se obtuvo el siguiente resumen:

- **Hardening index**: 64
- **Tests realizados**: 259
- **Plugins cargados**: 1
- **Firewall**: Detectado y activo ✅
- **Malware scanner**: No detectado ❌
- **Módulos activos**:
  - Security audit: ✅
  - Vulnerability scan: ✅
  - Compliance status: (requiere configuración adicional)

Un índice de 64/100 es adecuado para un entorno personal o de pruebas.
Se pueden mejorar varios aspectos según las sugerencias generadas por Lynis, especialmente la falta de un escáner antimalware, que se abordará en el siguiente punto del proyecto.

---

7. Auditoría con chkrootkit y rkhunter

Una buena práctica en cualquier proceso de hardening es realizar un escaneo del sistema en busca de **rootkits y malware** que pudieran haber pasado desapercibidos. Para ello se utilizan dos herramientas clásicas y complementarias:

- chkrootkit: escaneo rápido de rootkits comunes
- rkhunter (Rootkit Hunter): escaneo más profundo de binarios, permisos y anomalías del sistema

Instalando herramientas

```
root@NextCloud:~# apt update && apt install chkrootkit rkhunter -y
Obj:1 http://deb.debian.org/debian bookworm InRelease
Obj:2 http://security.debian.org/debian-security bookworm-security InRelease
```

Ejecutando escaneo del chkrootkit con #chkrootkit

```
WARNING: Output from ifpromisc:
lo: not promisc and no packet sniffer sockets
enp0s3: not promisc and no packet sniffer sockets
wg0: not promisc and no packet sniffer sockets
br-32184a78c904: not promisc and no packet sniffer sockets
docker0: not promisc and no packet sniffer sockets

Checking `w55808'...                                not found
Checking `wted'...                                  not found
Checking `scalper'...                               not found
Checking `slapper'...                               not found
Checking `z2'...                                    not found
Checking `chkutmp'...                               not found
Checking `OSX_RSPLUG'...                            not tested
root@NextCloud:~#
```


El resultado mostró lo siguiente:

- Todas las interfaces de red (como lo, enp0s3, wg0, docker0) **no están en modo promiscuo**, lo que indica que no hay sniffers activos escuchando tráfico.
- Todos los módulos probados (wted, scalper, slapper, z2, chkutmp, etc.) fueron reportados como **"not found"**, lo cual es el comportamiento esperado en un sistema limpio.
- Algunos elementos no fueron probados porque no aplican al sistema (not tested), como OSX_RSPLOG

Para el rkhunter lo primero es actualizar su base de datos

```
root@NextCloud:~# rkhunter --update
Invalid WEB_CMD configuration option: Relative pathname: "/bin/false"
root@NextCloud:~#
```

INCIDENCIA: El problema esta en su archivo .conf donde la variable WEB_CMD esta mal definida, esto ocurre cuando rkhunter intenta ejecutar una "comprobación online", vamos arreglarlo de una manera simple.

En el archivo .conf buscamos el WEB_CMD y lo comentamos con # lo guardamos con Ctrl+O y salimos

```
  GNU nano 7.2                                        /etc/rkhunter.conf *
#
#WEB_CMD="/bin/false"
```

Ahora si hizo le intento de actualizar, pero por lo visto los repositorios están desactualizados y habría que hacerlo a mano, vamos a continuar igualmente con el escaneo que aun así nos dará información relevante.

```
root@NextCloud:~# rkhunter --update
[ Rootkit Hunter version 1.4.6 ]

Checking rkhunter data files...
  Checking file mirrors.dat                                  [ Skipped ]
  Checking file programs_bad.dat                             [ Update failed ]
  Checking file backdoorports.dat                            [ Update failed ]
  Checking file suspscan.dat                                 [ Update failed ]
  Checking file i18n versions                                [ Update failed ]

Please check the log file (/var/log/rkhunter.log)
```


Con #rkhunter –check nos mostrara un sinfín de salidas que podría tardar unos 5min

```
System checks summary
=====================

File properties checks...
    Files checked: 144
    Suspect files: 1

Rootkit checks...
    Rootkits checked : 497
    Possible rootkits: 0

Applications checks...
    All checks skipped

The system checks took: 4 minutes and 12 seconds

All results have been written to the log file: /var/log/rkhunter.log

One or more warnings have been found while checking the system.
Please check the log file (/var/log/rkhunter.log)

root@NextCloud:~#
```

Uno de los binarios (/usr/bin/lwp-request) fue marcado como **[Warning]**. Esto es común en binarios que pueden usarse para tareas automatizadas o descargas (como wget, curl, lwp-request, etc.), pero **no implica que haya una infección real**.

```
/usr/bin/lwp-request                                      [ Warning ]
/usr/bin/mail.mailutils                                   [ OK ]
/usr/bin/dash                                             [ OK ]
```


8. Extra: Protección de acceso SSH con Knockd (port-knocking)

Knockd es un demonio que escucha los intentos de conexión a puertos específicos y, al recibir una secuencia correcta, ejecuta un script. En este caso, se usara para **abrir temporalmente el puerto 22 (SSH)** solo si el usuario "llama" correctamente, ocultando así el servicio SSH de escaneos y ataques automatizados.

El objetivo de esta practica es mantener el puerto 22 cerrado mediante iptables y abrirla tras la secuencia del knock, para ello se realizara un script que abrirá el acceso del puerto 22 solo a la ip que realizo el knock

Instalación del Knockd

```
root@NextCloud:~# apt update && apt install knockd -y
```

Habilitamos el knockd en nuestra maquina Debian con nano y modificamos a =1 , poned también vuestra interface cual tengáis, guardamos y cerramos.

```
  GNU nano 7.2                                          /etc/default/knockd
# control if we start knockd at init or not
# 1 = start
# anything else = don't start
# PLEASE EDIT /etc/knockd.conf BEFORE ENABLING
START_KNOCKD=1

# command line options
KNOCKD_OPTS="-i enp0s3"
```

Editamos el archivo .conf de knockd, asegurate de añadir la ip del que va hacer el knock.
La secuencia seria:

Abri ssh: 10001,10002,10003

Cerrar ssh: 30003,30002,30001

Cuando se haga el knock, según la secuencia de puertos elegirá una script u otro.

```
  GNU nano 7.2                                    /etc/knockd.conf *
[options]
        UseSyslog

[openSSH]
        sequence    = 10001,10002,10003
        seq_timeout = 5
        command     = /usr/local/bin/openSSH.sh 192.168.8.130
        tcpflags    = syn

[closeSSH]
        sequence    = 30003,30002,30001
        seq_timeout = 5
        command     = /usr/local/bin/closeSSH.sh 192.168.8.130
        tcpflags    = syn
```


Creando los scripts de apertura y cierre

Abrir

```
 GNU nano 7.2                                    /usr/local/bin/openSSH.sh *
#!/bin/bash
iptables -I INPUT -p tcp --dport 22 -s $1 -j ACCEPT
```

Cerrar

```
 GNU nano 7.2                                   /usr/local/bin/closeSSH.sh *
#!/bin/bash
iptables -D INPUT -p tcp --dport 22 -s $1 -j ACCEPT
```

Damos los permisos de ejecución

```
root@NextCloud:~# chmod +x /usr/local/bin/openSSH.sh && chmod +x /usr/local/bin/closeSSH.sh
root@NextCloud:/usr/local/bin# ls -l
total 8
-rwxr-xr-x 1 root root 64 jun  3 16:15 closeSSH.sh
-rwxr-xr-x 1 root root 64 jun  3 16:14 openSSH.sh
root@NextCloud:/usr/local/bin#
```

Abriendo puertos en iptables para las secuencias 10001,10002,10003

```
root@NextCloud:/usr/local/bin# iptables -A INPUT -p tcp --dport 10001 -j ACCEPT
root@NextCloud:/usr/local/bin# iptables -A INPUT -p tcp --dport 10002 -j ACCEPT
root@NextCloud:/usr/local/bin# iptables -A INPUT -p tcp --dport 10003 -j ACCEPT
root@NextCloud:/usr/local/bin#
```

Y lo mismo para la secuencia de cerrar el ssh(22) 30001,30002,30001

```
root@NextCloud:/usr/local/bin# iptables -A INPUT -p tcp --dport 30003 -j ACCEPT
iptables -A INPUT -p tcp --dport 30002 -j ACCEPT
iptables -A INPUT -p tcp --dport 30001 -j ACCEPT
root@NextCloud:/usr/local/bin#
```

Iniciar y habilitar el servicio knockd

```
root@NextCloud:/usr/local/bin# systemctl enable knockd
Synchronizing state of knockd.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable knockd
Created symlink /etc/systemd/system/multi-user.target.wants/knockd.service → /lib/systemd/system/knockd.service.
root@NextCloud:/usr/local/bin# systemctl start knockd
```


Verificación de servicio

```
root@NextCloud:/usr/local/bin# systemctl status knockd
● knockd.service - Port-Knock Daemon
     Loaded: loaded (/lib/systemd/system/knockd.service; enabled; preset: enabled)
     Active: active (running) since Tue 2025-06-03 16:32:55 CEST; 7s ago
       Docs: man:knockd(1)
   Main PID: 196995 (knockd)
      Tasks: 1 (limit: 4613)
     Memory: 524.0K
        CPU: 9ms
     CGroup: /system.slice/knockd.service
             └─196995 /usr/sbin/knockd -i enp0s3

jun 03 16:32:55 NextCloud systemd[1]: Started knockd.service - Port-Knock Daemon.
jun 03 16:32:55 NextCloud knockd[196995]: starting up, listening on enp0s3
```

Probemos el knock desde la maquina de Zabbix, pero antes… hay que cerrar el puerto 22 que nuestro fail2ban abre por defecto

```
root@NextCloud:~# iptables -L INPUT -n --line-numbers
# Warning: iptables-legacy tables present, use iptables-legacy to see them
Chain INPUT (policy DROP)
num  target     prot opt source               destination
1    f2b-sshd   6    --  0.0.0.0/0            0.0.0.0/0            multiport dports 22
2    ACCEPT     17   --  0.0.0.0/0            0.0.0.0/0            udp dpt:51820
3    ACCEPT     0    --  0.0.0.0/0            0.0.0.0/0            ctstate RELATED,ESTABLISHED
```

Eliminamos la regla numero 1 NOTA: Esto no quiere decir que fail2ban no seguirá activo, simplemente que mientras el puerto este cerrado no tiene donde escuchar, una vez se abra con el knock funcionará igual.

```
root@NextCloud:~# iptables -D INPUT 1
root@NextCloud:~# iptables -L INPUT -n --line-numbers
# Warning: iptables-legacy tables present, use iptables-legacy to see them
Chain INPUT (policy DROP)
num  target     prot opt source               destination
1    ACCEPT     17   --  0.0.0.0/0            0.0.0.0/0            udp dpt:51820
2    ACCEPT     0    --  0.0.0.0/0            0.0.0.0/0            ctstate RELATED,ESTABLISHED
```

Comprobamos puerto desde la maquina zabbix

```
root@Zabbix:~# nmap -p 22 192.168.8.129
Starting Nmap 7.93 ( https://nmap.org ) at 2025-06-03 16:45 CEST
Nmap scan report for 192.168.8.129
Host is up (0.00029s latency).

PORT   STATE    SERVICE
22/tcp filtered ssh
MAC Address: 08:00:27:A0:B2:9F (Oracle VirtualBox virtual NIC)
```


No hay respuesta si hacemos un ssh

```
root@Zabbix:~# ssh javi@192.168.8.129
^Z
[6]+  Stopped                 ssh javi@192.168.8.129
```

Ejecutamos knock de apertura de puerto desde la maquina de Zabbix(192.168.8.130)

```
root@Zabbix:~# knock 192.168.8.129 10001 10002 10003
root@Zabbix:~#
```

En la maquina de Nextlcoud(192.168.8.129) hacemos systemctl status knockd para verificar.

Como se puede observar las 3 secuencias llegan perfectamente al knockd y ejecuta el script de openSSH.sh

```
root@NextCloud:~# systemctl status knockd
● knockd.service - Port-Knock Daemon
     Loaded: loaded (/lib/systemd/system/knockd.service; enabled; preset: enabled)
     Active: active (running) since Tue 2025-06-03 16:32:55 CEST; 18min ago
       Docs: man:knockd(1)
   Main PID: 196995 (knockd)
      Tasks: 1 (limit: 4613)
     Memory: 540.0K
        CPU: 14ms
     CGroup: /system.slice/knockd.service
             └─196995 /usr/sbin/knockd -i enp0s3

jun 03 16:32:55 NextCloud systemd[1]: Started knockd.service - Port-Knock Daemon.
jun 03 16:32:55 NextCloud knockd[196995]: starting up, listening on enp0s3
jun 03 16:51:41 NextCloud knockd[196995]: 192.168.8.130: openSSH: Stage 1
jun 03 16:51:41 NextCloud knockd[196995]: 192.168.8.130: openSSH: Stage 2
jun 03 16:51:41 NextCloud knockd[196995]: 192.168.8.130: openSSH: Stage 3
jun 03 16:51:41 NextCloud knockd[196995]: 192.168.8.130: openSSH: OPEN SESAME
jun 03 16:51:41 NextCloud knockd[197225]: openSSH: running command: /usr/local/bin/openSSH.sh 192.168.8.130
```

Comprobamos

```
root@Zabbix:~# ssh javi@192.168.8.129
javi@192.168.8.129's password:
Linux NextCloud 6.1.0-37-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.140-1 (2025-05-
22) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
You have mail.
Last login: Tue Jun  3 14:20:39 2025 from 192.168.8.130
javi@NextCloud:~$
```


Cerrando puerto
(zabbix)

```
root@Zabbix:~# knock 192.168.8.129 30003 30002 30001
```

Comprobacion con systemctl
(Nextcloud)

```
jun 03 16:59:22 NextCloud knockd[196995]: 192.168.8.130: closeSSH: Stage 1
jun 03 16:59:22 NextCloud knockd[196995]: 192.168.8.130: closeSSH: Stage 2
jun 03 16:59:22 NextCloud knockd[196995]: 192.168.8.130: closeSSH: Stage 3
jun 03 16:59:22 NextCloud knockd[196995]: 192.168.8.130: closeSSH: OPEN SESAME
jun 03 16:59:22 NextCloud knockd[197423]: closeSSH: running command: /usr/local/bin/closeSSH.sh 192.168.8.130
```

9. Conclusión

Este proyecto ha implementado un conjunto de medidas prácticas de **hardening en un servidor Debian** orientado a entornos autogestionados. A través de herramientas ligeras y configuraciones accesibles, se ha conseguido aumentar de forma considerable el nivel de seguridad del sistema.

Medidas aplicadas

- **Desactivación de acceso root por SSH**
- **Instalación y configuración de Fail2Ban**
  - Protección frente a intentos de fuerza bruta
- **Firewall personalizado con iptables**
  - Política por defecto de denegación
  - Puertos abiertos explícitamente para servicios esenciales (WireGuard, HTTP/S)
- **Escaneo de seguridad con Lynis**
  - Evaluación del sistema con un índice de hardening de 64/100
- **Auditoría de rootkits con chkrootkit y rkhunter**
  - Sin amenazas graves detectadas
- **Protección avanzada del acceso SSH mediante Knockd (port-knocking)**
  - Ocultación total del puerto 22
  - Acceso controlado por secuencia de knock y scripts personalizados

Valor añadido

El sistema resultante:

- **Reduce superficie de ataque**
- **Evita conexiones no autorizadas**
- **Permite auditorías periódicas**
- **Es adaptable a cualquier entorno Debian/Ubuntu con pocos recursos**

Mejoras futuras (opcional)

- Añadir alertas automáticas por correo tras detección de intentos
- Integrar rkhunter y chkrootkit como tareas programadas con log rotativo
- Aplicar escaneo externo desde servicios como Shodan o SecurityHeaders.com