

Trabajo Práctico – Seguridad en Sistemas Operativos

Alumnos:

Herrera Gonzalez Franco Esteban – fh745543@gmail.com

Gonzalez Rivero Anthony José – agonrz7@gmail.com

Materia: Sistemas Operativos

Profesor: Ing. Ricardo Martínez

Fecha de Entrega: 05 de junio de 2025

Índice

1. Introducción
2. Marco Teórico
3. Caso Práctico
4. Metodología Utilizada
5. Resultados Obtenidos
6. Conclusiones
7. Bibliografía
8. Anexos

Introducción

En un entorno digital cada vez más interdependiente, donde los servicios críticos de organizaciones públicas, privadas y gubernamentales dependen del funcionamiento estable y seguro de sus infraestructuras informáticas, la protección de los sistemas operativos constituye una prioridad estratégica. Las amenazas a la seguridad no solo han crecido en volumen, sino también en complejidad, haciendo indispensable adoptar enfoques preventivos y activos para reducir los riesgos que afectan a la confidencialidad, integridad y disponibilidad de la información.

Linux, por su diseño modular, su orientación a entornos multiusuario y su carácter abierto y auditado por la comunidad, es uno de los sistemas operativos más utilizados en servidores, dispositivos embebidos, infraestructura de red y entornos académicos. Esta adopción masiva, sin embargo, también lo convierte en un objetivo frecuente para atacantes que buscan explotar debilidades en su configuración, errores del software o fallas humanas.

Marco Teórico

La seguridad en sistemas Linux depende en gran medida de la correcta configuración, mantenimiento y actualización del sistema operativo y sus servicios. Las amenazas a estos entornos no provienen únicamente de errores de programación o exploits sofisticados, sino también de factores humanos, configuraciones débiles y exposición a redes inseguras. A continuación, se desarrollan en profundidad cuatro de las vulnerabilidades más relevantes en entornos Linux actuales, junto con sus respectivas

estrategias de mitigación, sustentadas en herramientas concretas y buenas prácticas de administración.

1. Explotación de vulnerabilidades del sistema

Muchas brechas de seguridad en Linux se originan en fallas del software instalado, especialmente en componentes críticos como sudo, systemd, o polkit. Estas vulnerabilidades, si no son corregidas a tiempo, permiten a un atacante escalar privilegios, ejecutar código arbitrario o tomar control del sistema.

Para mitigar esta amenaza, se recomienda:

- Aplicar procesos de endurecimiento para reducir vulnerabilidades y superficie de ataque (hardening)
- Activar actualizaciones de seguridad automáticas con “unattended-upgrades”.
- Utilizar herramientas de auditoría como **Lynis** (cisofy.com) para identificar configuraciones inseguras, puertos abiertos, permisos incorrectos y servicios innecesarios.

Permisos de Archivos y Usuarios.

El modelo de permisos en Linux es un pilar fundamental de la seguridad del sistema. Cada archivo y directorio tiene un propietario, un grupo asociado y permisos para tres categorías de usuarios: el propietario, el grupo y *otros* (resto de usuarios). Los permisos definen la capacidad de leer (r), escribir (w) o ejecutar (x) el recurso.

Por ello, **solo se deben conceder permisos a quienes verdaderamente los requieren**, minimizando el acceso de *otros* usuarios en archivos sensibles.

En servidores Linux existe un superusuario **root** con privilegios absolutos, por lo que usar el mismo para tareas cotidianas no es aconsejable, ya que cualquier error o intrusión bajo esta cuenta puede comprometer todo el sistema. Lo recomendable es crear usuarios con privilegios limitados para el trabajo diario y otorgar privilegios administrativos (por ejemplo, mediante **sudo**) solo cuando sea necesario; De esta forma si fuere el caso una cuenta con privilegios ha sido vulnerada el daño potencialmente se reduce. Adicionalmente, **deshabilitar el inicio de sesión directo de root** (especialmente vía SSH) es una buena práctica que obliga a los administradores a autenticarse primero con un usuario normal y luego elevar privilegios, añadiendo una capa de seguridad extra. En resumen, una correcta gestión de permisos de archivos y usuarios —respetando el mínimo privilegio y evitando el uso innecesario de cuentas privilegiadas— es esencial para fortalecer la seguridad de un servidor Linux.

Firewalls

Un cortafuegos (firewall) que por definición es esencial para controlar el tráfico de red que entra y sale del servidor, permitiendo solo las conexiones necesarias y bloqueando las no autorizadas.

En Linux, la funcionalidad de firewall está integrada en el núcleo (kernel) a través del subsistema **Netfilter**, que decide el destino (aceptar, rechazar, dropear, etc.) de los paquetes según reglas definidas. Es importante antes de activar el Firewall que por defecto está inactivo, configurar antes las conexiones permitidas, habilitando al menos un acceso por SSH como se describe anteriormente para evitar perder el acceso al servidor.

La herramienta clásica de espacio de usuario para gestionar estas reglas es **Iptables**, que permite a los administradores definir políticas de filtrado de paquetes (por ejemplo, bloquear o permitir puertos, IPs, protocolos), manipula directamente las tablas de Netfilter y ofrece un control muy granular sobre el tráfico. Sin embargo, su uso puede ser complejo para tareas comunes.

En entornos Ubuntu y otras distribuciones orientadas a facilidad de administración, se suele emplear **UFW (Uncomplicated Firewall)** como interfaz simplificada para gestionar el firewall. UFW es básicamente un *frontend* de iptables que provee comandos sencillos para tareas habituales, como abrir o cerrar puertos, sin necesitar escribir reglas iptables complejas. Por ejemplo, con UFW se pueden permitir conexiones SSH o HTTP con comandos cortos (`ufw allow ssh` o `ufw allow 80/tcp`), mientras que internamente estas acciones se traducen en reglas iptables. UFW viene instalado por defecto en Ubuntu (aunque inicialmente deshabilitado).

Cabe mencionar que existen otras alternativas según la distribución, como **Firewalld** en el mundo Red Hat, o herramientas más avanzadas como *Shorewall* para configuraciones complejas, pero todas se basan en la misma infraestructura subyacente de Netfilter/iptables.

Actualizaciones de seguridad

Mantener el sistema actualizado es una de las medidas de seguridad más importantes. Los desarrolladores de Linux (y de las aplicaciones que corren en el servidor) publican actualizaciones periódicas, a menudo para corregir vulnerabilidades descubiertas o errores de seguridad. **No aplicar estos parches de seguridad de manera oportuna deja al servidor expuesto** a amenazas conocidas para las cuales ya existe solución.

Se recomienda habilitar, en la medida de lo posible, **actualizaciones automáticas** al menos para los parches de seguridad críticos, de modo que se reduzca la ventana de exposición desde que se anuncia una vulnerabilidad hasta que el servidor queda protegido, esto implica vigilar de cerca las actualizaciones del propio sistema operativo (kernel, bibliotecas GNU, etc.) así como de los servicios desplegados (servidores web, bases de datos, etc.).

En distribuciones como Ubuntu, por ejemplo, puede utilizarse la herramienta **unattended-upgrades** para instalar automáticamente parches de seguridad. Red Hat y otras distros empresariales ofrecen también servicios de **Live Patching** (como *KernelCare* o *Livepatch* de Canonical) que permiten aplicar parches al kernel en vivo, sin reiniciar, lo cual es muy útil para servidores que requieren alta disponibilidad.

Otro aspecto importante es actualizar también las dependencias y librerías de aplicaciones, ya que un servidor seguro puede verse comprometido por una aplicación web vulnerable corriendo sobre él. Herramientas de gestión de paquetes como APT, YUM/DNF, Snap, etc., facilitan esta tarea. Asimismo, existen **herramientas de escaneo de vulnerabilidades** específicas para servidores que ayudan a identificar paquetes obsoletos o con fallos conocidos (por ejemplo, *Lynis*, *OpenSCAP*, etc.).

Auditoría y monitoreo

Las capacidades de auditoría de Linux permiten registrar eventos relevantes de seguridad para su análisis. En particular, el subsistema de auditoría (herramienta auditd) registra eventos clave del sistema para propósitos de seguridad. Auditd puede configurarse con reglas que monitorean acciones específicas, como cambios a archivos críticos (ej. modificaciones en `/etc/passwd` o en la configuración de SSH) y eventos de autenticación. Cada evento registrado incluye información detallada (usuario, proceso, fecha, tipo de acción), lo que proporciona un rastro (*audit trail*) invaluable para investigar incidentes a posteriori o cumplir requisitos de cumplimiento normativo. De hecho, auditd es instrumental para mantener la **auditabilidad** exigida por marcos regulatorios, ya que garantiza que cualquier acceso o cambio importante quede registrado. Además de auditd, existen herramientas IDS/IPS (*Intrusion Detection/Prevention Systems*) que se pueden desplegar en el servidor para analizar logs en tiempo real en busca de patrones maliciosos (ejemplos: *OSSEC*, *Fail2Ban*, *Tripwire* para integridad de archivos, etc.).

En cuanto al **monitoreo**, es recomendable supervisar de forma continua los *logs* del sistema y servicios. Linux registra eventos en archivos bajo `/var/log/` (como `auth.log` para autenticaciones, `syslog` para eventos del sistema, logs de aplicaciones, etc.). Revisar estos registros permite identificar, por ejemplo, intentos reiterados de login fallidos, escaneos de puertos, errores inusuales, o la presencia de malware (detectada por alertas de herramientas como *rkhunter* o *ClamAV* si se tienen instaladas). Dado que revisar manualmente los logs es tedioso, se pueden usar herramientas de resumen y alerta como **Logwatch**, que envía reportes diarios de actividad relevante, o sistemas más avanzados de monitoreo centralizado/SIEM cuando se administra una infraestructura grande.

Un buen esquema de monitoreo incluirá también **verificaciones de integridad** (para detectar si alguien modificó archivos críticos), monitorización de rendimiento (a veces un pico inesperado de carga puede indicar actividad maliciosa) y alertas en tiempo real ante ciertos eventos (por ejemplo, envío de un correo/alarma si hay 100 intentos fallidos de SSH en una hora, lo cual sugiere un ataque de fuerza bruta).

2. Malware y virus en sistemas Linux

Aunque Linux presenta una menor tasa de infecciones comparado con otros sistemas operativos, existen múltiples variantes de malware dirigidas a servidores y escritorios Linux. Los rootkits, troyanos, backdoors y criptomineros se infiltran comúnmente

mediante scripts maliciosos, aplicaciones comprometidas o vulnerabilidades no parcheadas.

Herramientas como **ClamAV** (clamav.net) permiten escaneos de archivos y directorios en busca de firmas conocidas de malware, mientras que **Rootkit Hunter (rkhunter)** (rkhunter.sourceforge.net) detecta modificaciones sospechosas en archivos de sistema, cambios de permisos y procesos ocultos que podrían indicar una infección avanzada.

Actualizar las bases de datos (comando freshclam) y realizar escaneos periódicos (clamscan, rkhunter --check) fortalece la detección temprana y previene la persistencia de malware.

3. Ataques de fuerza bruta

Los ataques de fuerza bruta consisten en intentos automáticos y masivos para adivinar contraseñas, generalmente contra el servicio SSH. La falta de medidas de protección, como el uso de contraseñas débiles o puertos por defecto, facilita estos ataques.

La herramienta **Fail2Ban** (fail2ban.org) supervisa los logs del sistema y, al detectar múltiples intentos fallidos de acceso, bloquea temporal o permanentemente las IP ofensivas utilizando “iptables”.

Además, modificar la configuración de sshd_config para:

- Cambiar el puerto por defecto (22),
- Deshabilitar el acceso como root (PermitRootLogin no),
- Habilitar autenticación mediante claves SSH y deshabilitar contraseñas, es una práctica clave de seguridad.

4. Ingeniería social y autenticación débil

La ingeniería social implica manipular a usuarios para que revelen información confidencial, otorguen acceso o instalen software malicioso. Aunque se basa en errores humanos y la prevención va de la mano de la educación de los usuarios y las metodologías de trabajo, puede mitigarse con medidas técnicas como:

- Reforzar políticas de contraseñas usando PAM (Pluggable Authentication Modules), específicamente “libpam-pwquality”, que impone requisitos mínimos de complejidad en contraseñas: longitud, diversidad de caracteres y no repetición.
- Implementar autenticación de dos factores (2FA) con Google Authenticator (github.com/google/google-authenticator), que introduce un token dinámico

adicional al momento del login SSH, reduciendo el riesgo de acceso con credenciales comprometidas.

En conjunto, este análisis busca demostrar cómo una adecuada configuración del sistema operativo, junto con un enfoque proactivo en la gestión de vulnerabilidades, puede reducir de forma significativa el riesgo de ataques y mejorar la postura de seguridad de cualquier entorno basado en Linux.

Caso Práctico

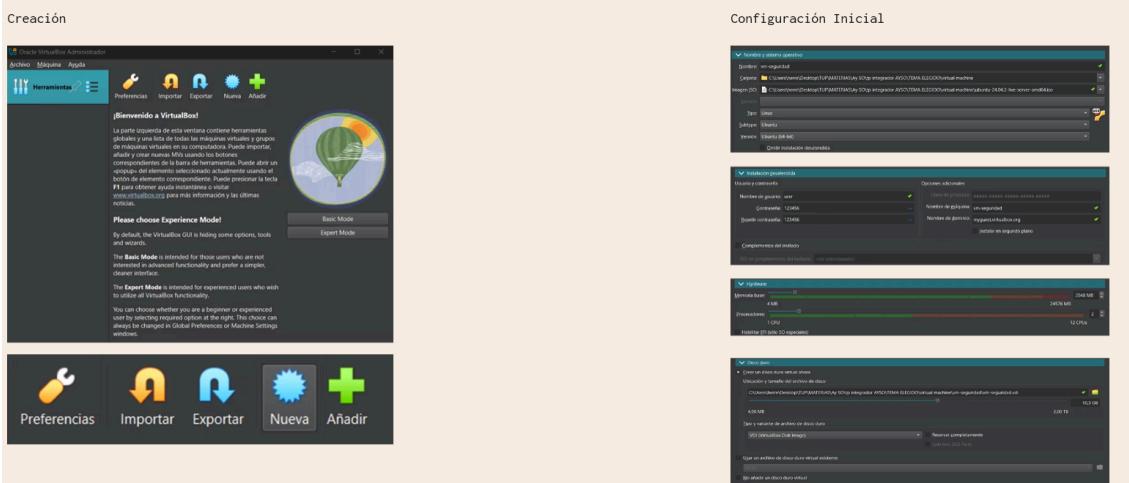
Configuración Segura de un Servidor Ubuntu, instalación de paquetes de seguridad, auditoría de vulnerabilidades y autenticación doble factor

A continuación, se presenta un caso práctico de configuración paso a paso para reforzar la seguridad de un servidor Linux (Ubuntu Server).

Instalación y configuración: Virtual Box y Ubuntu Server



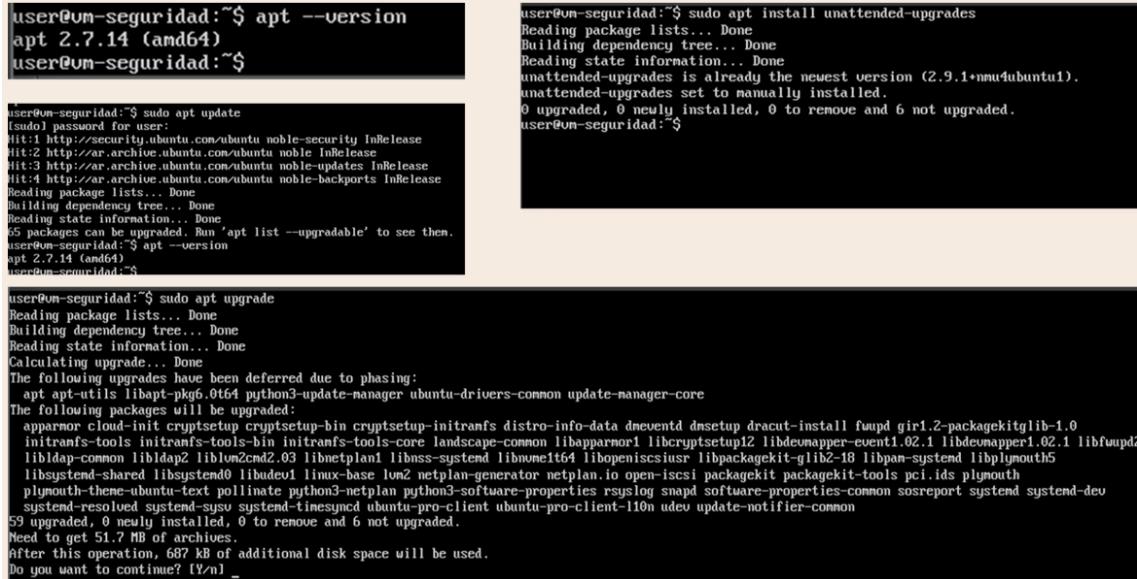
Instalación y configuración: Virtual Box y Ubuntu Server



Primer inicio: Virtual Box y Ubuntu Server



apt:



lynis:

```
user@vm-seguridad:~$ sudo apt install lynis
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
```

```
[2007-02-21, CISQI - https://ciscofy.com/login/]
Enterprise support available (compliance, plugins, interface and tools)

[*) Initializing program
-- Detecting OS...
-- Checking profiles... [ HOME ] [ HOME ]

Program version: 3.0.9
Operating system: Linux
Operating system name: Ubuntu
Operating system version: 24.04
Kernel version: 6.1.0
Machine platform: x86_64
Hostname: user-separated

Profiles: /etc/linus/default.prf
Log file: /var/log/linus.log
Report file: /var/log/linus/report.dat
Report version: 1.0
Plugin directory: /etc/linus/plugins

Auditor: [ Not Specified ]
Language: en
Test category: all
Test group: all

-- Program update status... [ NO UPDATE ]

[*) System tools
-- Scanning available tools...
-- Checking system binaries...

[!] Plugins (phase 1)
Note: plugins have more extensive tests and may take several minutes to complete
-- Plugin: debian
[*) Debian Tests
-- Checking for system binaries that are required by Debian Tests...
-- Checking /bin... [ ERROR ]
```

```
user@um-seguridad:~$ sudo lynis audit system
```

```
[WARNING]: Test DEB-0001 had a long execution: 35.526751 seconds

  - libpam-totpauth           [ Not Installed ]
  - File System Checks:
    - /etc/Crypt, Cryptsetup & Cryptmount:
      - Checking / on /dev/sda2   [ NOT ENCRYPTED ]
  - Software:
    - apt-listchanges          [ Not Installed ]
    - apt-listchanges           [ Not Installed ]
    - needrestart               [ Installed ]
    - fail2ban                  [ Not Installed ]

[+] Boot and services
-----
  - Service Manager           [ systemct ]
  - Checking UEFI boot        [ DISABLED ]
  - Checking presence GRUB2    [ FOUND ]
  - Checking for password protection [ MUNG ]
  - Check running services (systemctl)
    - Result: found 22 running services [ DONE ]
  - Check enabled services at boot (systemctl)
    - Result: found 49 enabled services [ DONE ]

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

```
suggestion[]=-DEB-0200]Install libpam-tmpdir to set $TMP and $TMPDIR for PAM sessions]-|-  
suggestion[]=-DEB-0010]Install apt-listbugs to display a list of critical bugs prior to each  
suggestion[]=-DEB-0011]Install apt-listchanges to display any significant changes prior to a  
suggestion[]=-DEB-0008]Install fail2ban to automatically ban hosts that commit multiple auth  
plugins_enabled=1
```

Clamv y Rkhunter:

```
Running kernel seems to be up-to-date.  
No services need to be restarted.  
No containers need to be restarted.  
No user sessions are running outdated binaries.  
No UM guests are running outdated hypervisor (qemu) binaries on this host.  
user@un-seguridad:~$
```

freshclam:

```
user@um-seguridad:~$ sudo freshclam
```

```
user@vm-seguridad:~$ sudo freshclam
ERROR: Failed to lock the log file /var/log/clamav/freshclam.log: Resource temporarily unavailable
ERROR: Problem with internal logger (UpdateLogFile = /var/log/clamav/freshclam.log).
ERROR: initialize: libfreshclam init failed.
ERROR: Initialization error!
user@vm-seguridad:~$ _
```

```
user@vm-seguridad:~$ sudo systemctl stop clamav-freshclam
user@vm-seguridad:~$ sudo freshclam
ClamAV update process started at Fri May 30 23:42:49 2025
Fri May 30 23:42:49 2025 -> daily.cud database is up-to-date (version: 27653, sigs: 2075704, f-level: 90, builder: raynman)
Fri May 30 23:42:49 2025 -> main.cud database is up-to-date (version: 62, sigs: 6647427, f-level: 90, builder: sigmgr)
Fri May 30 23:42:49 2025 -> bytecode.cud database is up-to-date (version: 336, sigs: 83, f-level: 90, builder: mrandolp)
user@vm-seguridad:~$
```

```
ser@um-seguridad:~$ sudo systemctl start clamav-freshclam  
ser@um-seguridad:~$ _
```

clamscan:

```
user@um-seguridad:~$ pwd
/home/user
user@um-seguridad:~$ sudo clamscan -r /home
Loading: 27s, ETA: 0s [=====>] 8.71M/8.71M sigs
Compiling: 5s, ETA: 0s [=====>] 41/41 tasks

/home/user/.bashrc: OK
/home/user/.profile: OK
/home/user/.cache/motd.legal-displayed: Empty file
/home/user/.bash_logout: OK
/home/user/.ssh/authorized_keys: Empty file
/home/user/.sudo_as_admin_successful: Empty file

----- SCAN SUMMARY -----
Known viruses: 8707471
Engine version: 1.0.8
Scanned directories: 4
Scanned files: 3
Infected files: 0
Data scanned: 0.00 MB
Data read: 0.00 MB (ratio 0.00:1)
Time: 32.769 sec (0 m 32 s)
Start Date: 2025:05:30 23:45:33
End Date: 2025:05:30 23:46:05
user@um-seguridad:~$ _
```

rkhunter:

```
user@um-seguridad:~$ sudo rkhunter --check_
```

File	Description	Status	Details
/etc/ldlinux.so	ld-linux.so Rootkit	[Not found]	Checking the network...
/etc/low_worm	Low Worm	[Not found]	Performing checks on the network ports
/etc/lockit	Lockit / L3K Rootkit	[Not found]	Checking for backdoor ports
/etc/malware	Malware	[Not found]	[None found]
/etc/malware2	Root-MT Rootkit	[Not found]	Performing checks on the network interfaces
/etc/malware3	MRK Rootkit	[Not found]	Checking for promiscuous interfaces
/etc/malware4	MTR Rootkit	[Not found]	[None found]
/etc/malware5	Other Rootkit	[Not found]	Checking the local host...
/etc/optic_kit	Optic Kit (Tx) Worm	[Not found]	Performing system host checks
/etc/phala_worm	Phala Worm	[Not found]	Checking for local host name
/etc/phala_rootkit	Phala Rootkit	[Not found]	[Found]
/etc/phala_worm	Phala WORM (extended tests)	[Not found]	Checking for system startup files
/etc/phala_worm	Phala WORM (extended tests)	[Not found]	[Found]
/etc/phala_worm	Phala WORM (extended tests)	[Not found]	Checking system startup files for salure
/etc/phala_worm	Phala WORM (extended tests)	[Not found]	[None found]
/etc/phala_worm	Phala WORM (extended tests)	[Not found]	Performing group and account checks
/etc/phala_worm	Phala WORM (extended tests)	[Not found]	Checking for passwd file
/etc/phala_worm	Phala WORM (extended tests)	[Not found]	[Found]
/etc/phala_worm	Phala WORM (extended tests)	[Not found]	Checking for root equivalent (UID 0) accounts
/etc/phala_worm	Phala WORM (extended tests)	[Not found]	[None found]
/etc/phala_worm	Phala WORM (extended tests)	[Not found]	Checking for passwordless accounts
/etc/phala_worm	Phala WORM (extended tests)	[Not found]	[None found]
/etc/phala_worm	Phala WORM (extended tests)	[Not found]	Checking for passwd file changes
/etc/phala_worm	Phala WORM (extended tests)	[Not found]	[None found]
/etc/phala_worm	Phala WORM (extended tests)	[Not found]	Checking for group file changes
/etc/phala_worm	Phala WORM (extended tests)	[Not found]	[None found]
/etc/phala_worm	Phala WORM (extended tests)	[Not found]	Checking root account shell history files
/etc/sia	Sia Rootkit	[Not found]	Performing system configuration file checks
/etc/slapper	Slapper Worm	[Not found]	Checking for an SSH configuration file
/etc/sneakin	Sneakin Rootkit	[Not found]	[Not found]
/etc/sneakin	Sneakin Rootkit	[Not found]	Checking for a running system logging daemon
/etc/sneakin	Sneakin Rootkit	[Not found]	[Found]
/etc/sockit	Sockit Rootkit	[Not found]	Checking for a system logging configuration file
/etc/sockit	Sockit Rootkit	[Not found]	[Not allowed]
/etc/superkit	Superkit Rootkit	[Not found]	Checking if syslog remote logging is allowed
/etc/superkit	Superkit Rootkit	[Not found]	[None found]
/etc/telnetkit	TelnetKit Rootkit	[Not found]	Performing filesystem checks
/etc/tor	Tor Rootkit	[Not found]	Checking for an /etc/ssh configuration file
/etc/tor	Tor Rootkit	[Not found]	[Found]
/etc/trojanit	Trojanit Rootkit	[Not found]	Checking for a running system logging daemon
/etc/trojanit	Trojanit Rootkit	[Not found]	[Found]
/etc/trojanit	Trojanit Rootkit	[Not found]	Checking for a system logging configuration file
/etc/trojanit	Trojanit Rootkit	[Not found]	[Not allowed]
/etc/trootkit	Troot Rootkit	[Not found]	Checking if syslog remote logging is allowed
/etc/trootkit	Troot Rootkit	[Not found]	[None found]
/etc/trootkit	Troot Rootkit	[Not found]	Performing filesystem checks
/etc/trootkit	Troot Rootkit	[Not found]	Checking /dev for suspicious file types
/etc/trootkit	Troot Rootkit	[Not found]	[Found]
/etc/xbitkit	Xbitkit Rootkit	[Not found]	Checking for hidden files and directories
/etc/xbitkit	Xbitkit Rootkit	[Not found]	[Warning]
/etc/z8k	Z8k Rootkit	[Not found]	[None found]

(Press QUIT to continue) (Press <ENTER> to continue)

rkhunter:

```
System checks summary
=====
File properties checks...
    Files checked: 141
    Suspect files: 0

Rootkit checks...
    Rootkits checked : 477
    Possible rootkits: 0

Applications checks...
    All checks skipped

The system checks took: 7 minutes and 43 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)

user@vm-seguridad:~$
```

fail2ban:

```
user@vm-seguridad:~$ sudo install fail2ban_
user@vm-seguridad:~$ sudo install fail2ban
install: missing destination file operand after 'fail2ban'
Try 'install --help' for more information.
user@vm-seguridad:~$ _

user@vm-seguridad:~$ sudo apt install fail2ban
user@vm-seguridad:~$ sudo systemctl enable fail2ban
Synchronizing state of fail2ban.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable fail2ban
user@vm-seguridad:~$ 

user@vm-seguridad:~$ sudo systemctl start fail2ban
user@vm-seguridad:~$ sudo fail2ban-client status
Status
[- Number of jail:      1
`- Jail list:   sshd
user@vm-seguridad:~$ sudo 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:
user@vm-seguridad:~$
```

Cambio del puerto 22:

```
user@vm-seguridad:~$ sudo nano /etc/ssh/sshd_config
```

```
GNU nano 7.2
PermitRootLogin no
```

```
GNU nano 7.2
PermitRootLogin no
Port 2222
PasswordAuthentication no
```

```
user@vm-seguridad:~$ sudo systemctl restart ssh
```

libpam-pwquality:

```
user@vm-seguridad:~$ sudo apt install libpam-pwquality
```

```
user@vm-seguridad:~$ sudo apt install libpam-pwquality
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  cracklib-runtime libcrack2 libpam-pwquality libpwquality-common libpwquality1 wamerican
The following NEW packages will be installed:
  cracklib-runtime libcrack2 libpam-pwquality libpwquality-common libpwquality1 wamerican
0 upgraded, 2 newly installed, 0 to remove and 6 not upgraded.
Need to get 446 kB of archives.
After this operation, 1,332 kB of additional disk space will be used.
Do you want to continue? [Y/n] -
```

```
user@vm-seguridad:~$ sudo nano /etc/pam.d/common-password
```

```
GNU nano 7.2
/etc/pam.d/common-password
# This file is included from other service-specific PAM config files,
# and should contain a list of modules that define the services to be
# used to change user passwords. The default is pam_unix.
#
# Explanation of pam_unix options:
#   The "try_first_pass" option allows pam_unix to try to
#   hashed passwords using the yescrypt algorithm, introduced in Debian
#   9.1. If this option is present, pam_unix will use yescrypt by default.
#   If this option is not present, or if it is set to "no", pam_unix will
#   use the option "shadow"; if a shadow password hash will be shared
#   between Debian 9.1 and older releases replace "yescrypt" with "shadow".
#   (See also the "obscure" and "try_first_pass" options)
#   OBSOLETE_ECRYPT option in login.defs. See the pam_unix manpage
#   for other options.
#
# As of pam 1.0.1-5, this file is managed by pam-auth-update by default.
# To maintain compatibility of this, it is recommended that you configure any
# local modules either before or after the default block, and use
# pam-auth-update(8) to automatically resection of other modules. See
# pam-auth-update(8) for details.
#
# Here are the per-package modules (the "Primary" block)
password    requisite        pam_permit.so
password    requisite        pam_unix.so retry=3
# here's the fallback if no module succeeds          pam_unix.so obscure use_authok try_first_pass yescrypt
password    requisite        pam_deny.so
# sets the status to positive return value (success)
# if all the modules above succeed (they have already)
# this avoids us returning an error just because nothing sets a success code
# and the module below will each just JUMP over us
password    required        pam_permit.so
# and here are more per-package modules (the "Additional" block)
# end of pam-auth-update config
pam_permit.so
pam_unix.so obscure use_authok try_first_pass yescrypt
pam_deny.so
```

libpam-pwquality:

```
user@vm-seguridad:~$ passwd user
Changing password for user.
Current password:
New password:
BAD PASSWORD: The password is too similar to the old one
New password:
```

```
Changing password for user.
Current password:
New password:
BAD PASSWORD: The password fails the dictionary check - it is too simplistic/systematic
New password:
```

```
user@vm-seguridad:~$ passwd user
Changing password for user.
Current password:
New password:
BAD PASSWORD: The password is too similar to the old one
New password:
BAD PASSWORD: The password is shorter than 12 characters
New password:
```

```
user@vm-seguridad:~$ passwd user
Changing password for user.
Current password:
New password:
BAD PASSWORD: The password is too similar to the old one
New password:
BAD PASSWORD: The password is shorter than 12 characters
New password:
BAD PASSWORD: The password is shorter than 12 characters
password: Have exhausted maximum number of retries for service
password: password unchanged
user@vm-seguridad:~$ _
```

F2A:

```
user@vm-seguridad:~$ sudo apt install libpam-google-authenticator
[sudo] password for user:
```

```
Do you want authentication tokens to be time-based (y/n)
warning: pasting the following URL into your browser exposes the OTP secret to Google:
https://www.google.com/chart?chs=200x200&chld=tototo&user=vm-seguridad&sec
```



```
Your new secret key is: FESPAQWSTSDXEMBCTZ699
```

```
Do you want me to update your "/home/user/.google_authenticator" file? (y/n) y
```

```
Do you want to disallow multiple uses of the same authentication token? This restricts you to one login about every 30s, but it increases your chances to notice or even prevent man-in-the-middle attacks (y/n) y
```

```
By default, a new token is generated every 30 seconds by the mobile app. In order to compensate for possible time-skew between the client and the server, we allow an extra token before and after the current time. This allows for a time skew of up to 30 seconds between authentication server and client. If you experience problems with poor time synchronization, you can increase the window from its default size of 3 permitted codes (one previous code, the current code, the next code) to 17 permitted codes (the 8 previous codes, the current code, and the 8 next codes). This will permit for a time skew of up to 4 minutes between client and server.
```

```
Do you want to do so? (y/n) y
```

```
If the computer that you are logging into isn't hardened against brute-force login attempts, you can enable rate-limiting for the authentication module. By default, this limits attackers to no more than 3 login attempts every 30s.
```

```
Do you want to enable rate-limiting? (y/n) y
```

```
user@vm-seguridad:~$ _
```

```
user@vm-seguridad:~$ sudo nano /etc/pam.d/sshd
```

```
GNU nano 7.2
auth required pam_google_authenticator.so
```

F2A:

```
user@vm-seguridad:~$ sudo nano /etc/ssh/sshd_config
```

```
Port 2222
ChallengeResponseAuthentication yes
UsePAM yes
AuthenticationMethods password,keyboard-interactive
```

```
user@vm-seguridad:~$ sudo nano /etc/pam.d/login_
```

```
GNU nano 7.2
/etc/pam.d/login
# The PAM configuration file for the Shadow 'login' service

# Enforce a minimal delay in case of failure (in microseconds).
# (Replaces the 'FAIL_DELAY' setting from login.defs)
# Note: other modules may have their own minimal delay, for example,
# to disable any delay, you should add the nodelay option to pam_unix
auth optional pam_delay.so delay=300000

# Outputs an issue file prior to each login prompt (Replace the
# 'ISSUE_FILE' option from login.defs)
# auth required pam_issue.so issues/etc/issue

# Disallows other than root logins when /etc/nologin exists
# (Replaces the 'NOLOGINS_FILE' option from login.defs)
auth requisite pam_nologin.so

# Sets the loginuid process attribute
session required pam_loginuid.so

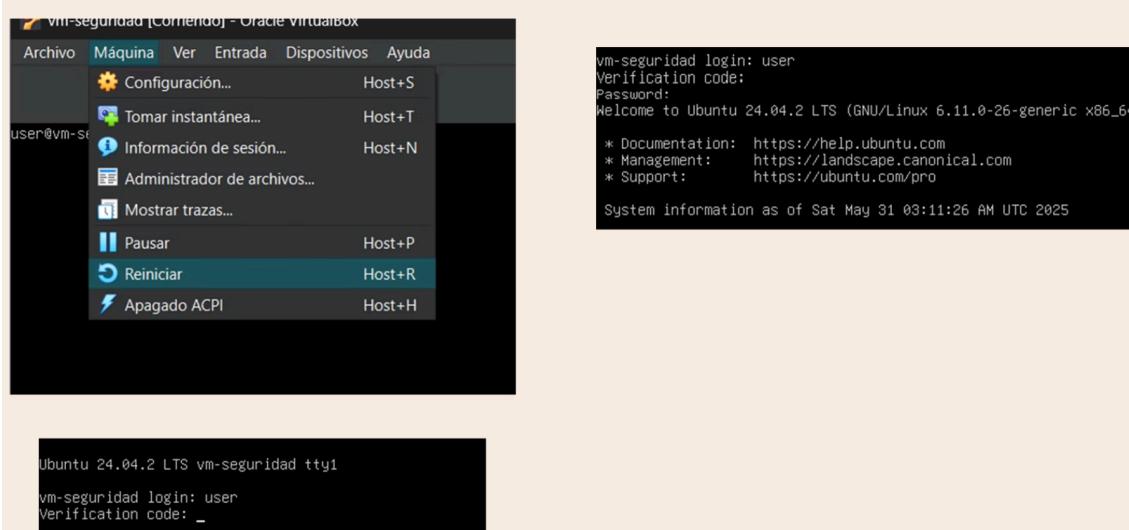
# Prints the message of the day upon successful login.
# (Replaces the 'MOTD_FILE' option from login.defs)
# This includes a dynamic # (Replaces the 'NOLOGINS_FILE' option from login.defs)
# and a static (admin-only) '#'
session optional pam_motd.so chroot=/root
# SELinux needs to be the first session rule. This ensures that any
# module could execute code in the wrong domain.
# When SELinux is disabled, this return success.
session optional pam_selinux.so close
# SELinux needs to be the first session rule. This ensures that any
# module could execute code in the wrong domain.
# When SELinux is present, 'Required' would be sufficient (When SELinux
# is disabled, this returns success)
session [success] ignore|ignore module_unk|ignore default|bad pam_selinux.so close

# SELinux needs to be the first session rule. This ensures that any
# module could execute code in the wrong domain.
# When SELinux is present, 'Required' would be sufficient (When SELinux
# is disabled, this returns success)
session [success] ignore|ignore module_unk|ignore default|bad pam_selinux.so open
# When the module is present, 'Required' would be sufficient (When SELinux
# is disabled, this returns success)

# This module parses environment configuration files
[ Read 100 lines ]
Help Write Out Where Is Cut Execute Go To Line
[ Read 100 lines ]
Edit Read File Replace Paste Location
[ Read 100 lines ]

```

F2A:



Comandos utilizados:

- **sudo apt update**
#Actualiza la lista de paquetes disponibles desde los repositorios configurados, permitiendo que el sistema conozca las versiones más recientes antes de instalar o actualizar software. No instala ni actualiza paquetes, solo sincroniza el índice.
- **sudo apt upgrade**
#Actualiza todos los paquetes instalados a sus versiones más recientes disponibles, sin instalar ni eliminar otros paquetes.
- **sudo apt install unattended-upgrades**
#Instala el sistema de actualizaciones automáticas para aplicar parches de seguridad sin intervención manual.

- **sudo apt install clamav rkhunter**
#Instala ClamAV (antivirus) y Rootkit Hunter (herramienta para detectar rootkits y exploits).
- **sudo freshclam**
#Actualiza la base de datos de firmas de virus de ClamAV.
- **sudo clamscan -r /home**
#Escanea recursivamente el directorio `/home` en busca de virus utilizando ClamAV.
- **sudo rkhunter --check**
#Ejecuta un chequeo del sistema con Rootkit Hunter para detectar rootkits, puertas traseras y exploits locales.

- `sudo apt install lynis`
#Instala Lynis, una herramienta de auditoría de seguridad para sistemas Unix.
 - `sudo lynis audit system`
#Ejecuta una auditoría de seguridad completa del sistema con Lynis, generando recomendaciones de refuerzo.
 - `sudo apt install fail2ban`
#Instala Fail2Ban, que protege contra ataques de fuerza bruta bloqueando IPs sospechosas.
 - `sudo systemctl enable fail2ban`
#Habilita Fail2Ban para que se inicie automáticamente con el sistema.
 - `sudo systemctl start fail2ban`
#Inicia el servicio Fail2Ban.
-
- `sudo nano /etc/ssh/sshd_config`
#Edita la configuración del servidor SSH para aplicar medidas de seguridad.
 - `PermitRootLogin no`
#Deshabilita el inicio de sesión como usuario root vía SSH.
 - `Port 2222`
#Cambia el puerto SSH por defecto (22) a otro más alto, como medida contra escaneos automáticos.
 - `PasswordAuthentication no`
#Desactiva la autenticación por contraseña; requiere autenticación por clave.
 - `sudo apt install libpam-pwquality`
#Instala el módulo PAM para reforzar la calidad de las contraseñas.
 - `sudo nano /etc/pam.d/common-password`
#Edita las reglas de contraseña del sistema para aplicar políticas de complejidad.
 - `password requisite pam_pwquality.so retry=3 minlen=12 difok=3`
#Establece mínimo 12 caracteres, 3 intentos y 3 diferencias con contraseñas anteriores.

```

·      sudo apt install libpam-google-authenticator
      #Instala el módulo PAM para integrar Google Authenticator como 2FA.

·      google-authenticator
      #Genera claves secretas y configura 2FA para el usuario actual.

·      sudo nano /etc/pam.d/sshd
      #Configura PAM para que SSH requiera autenticación con Google
      #Authenticator.

·      auth required pam_google_authenticator.so
      #Habilita 2FA en sesiones SSH.

·      sudo nano /etc/ssh/sshd_config
      #Ajusta parámetros SSH para habilitar la autenticación multifactor.

·      ChallengeResponseAuthentication yes
      #Permite desafíos interactivos como 2FA.

·      UsePAM yes
      #Habilita PAM para que funcione el módulo de Google Authenticator.

·      PasswordAuthentication yes
      #Necesario si se usa contraseña + 2FA. Si usás claves, puede ser no.

·      AuthenticationMethods password,keyboard-interactive
      #Requiere tanto contraseña como segundo factor (2FA) para autenticarse.

·      sudo nano /etc/pam.d/login
      #Aplica Google Authenticator también para inicios de sesión en consola
      #(TTY o local).

·      auth required pam_google_authenticator.so
      #Habilita 2FA en sesiones locales.

```

Metodología Utilizada

- Se investigaron conceptos en manuales oficiales y en distintas fuentes técnicas.
- Se configuró un entorno Linux (Ubuntu Server).
- Se realizó la instalación de distintos paquetes de seguridad.
- Se utilizaron las herramientas de seguridad instaladas para realizar auditorías y análisis de vulnerabilidades.
- Se realizaron pruebas de acceso y cambio de contraseñas.
- Se modificó el puerto 22.

- Se configuró el acceso de doble factor.
- Se documentaron los comandos.

Resultados Obtenidos

- Se mitigaron vulnerabilidades.
- Se verificó la correcta delegación de privilegios con sudo.
- Se verificó la ausencia de rootkits y softwares maliciosos.
- Se verificó el correcto funcionamiento de los sistemas de acceso mediante doble factor .

Conclusiones

A lo largo de este informe se ha demostrado que, si bien Linux ofrece una base sólida en términos de control de acceso y administración de usuarios, la verdadera fortaleza del sistema depende directamente de la correcta configuración y mantenimiento por parte del administrador.

La implementación de herramientas como firewalls, antivirus (ClamAV), escáneres de rootkits (rkhunter), mecanismos de autenticación robustos como 2FA, y la aplicación de políticas de contraseñas seguras, constituyen una capa de defensa esencial ante amenazas reales como malware, explotación de vulnerabilidades, ataques de fuerza bruta e ingeniería social. Asimismo, la gestión de permisos y el principio de mínimo privilegio resultan determinantes para contener posibles compromisos del sistema y limitar su alcance.

En definitiva, la combinación de buenas prácticas, herramientas adecuadas y conciencia sobre las amenazas actuales permite convertir un sistema Linux en una plataforma altamente resistente frente a ataques. Este enfoque preventivo no solo mejora la postura de seguridad del entorno, sino que también garantiza una mayor estabilidad, integridad y disponibilidad de los servicios que allí se ejecutan.

Bibliografía

Fuentes técnicas:

- ClamAV Documentation: <https://docs.clamav.net/>
- Rootkit Hunter: <https://rkhunter.sourceforge.net/>
- Lynis Auditing Tool: <https://cisofy.com/lynis/>
- CIS Benchmark for Debian Linux:
https://www.cisecurity.org/benchmark/debian_linux
- Fail2Ban Documentation: https://www.fail2ban.org/wiki/index.php/Main_Page
- OpenSSH Best Practices: https://www.ssh.com/academy/ssh/sshd_config

- Google Authenticator for PAM:
<https://github.com/google/google-authenticator-libpam>