Herramientas Esenciales de Seguridad

1. Firewall (Cortafuegos): UFW (Uncomplicated Firewall)

UFW es la interfaz predeterminada y fácil de usar para configurar el firewall **Netfilter** de Linux. Viene preinstalado en Ubuntu, pero a menudo está deshabilitado por defecto. Es esencial para restringir el tráfico de red entrante y saliente.

- **¿Por qué usarlo?** Bloquea las conexiones no deseadas a tu sistema, lo que es vital tanto para escritorios como para servidores.
- **Comandos Sencillos:**

  sudo ufw enable

  Habilita el firewall.

  sudo ufw default deny incoming

  Establece la política predeterminada para denegar todas las conexiones entrantes (esto es clave).

  sudo ufw allow ssh

  Permite el acceso SSH (si lo necesitas).

  sudo ufw status verbose

  Muestra el estado y las reglas configuradas.

2. Detección de Rootkits y Malware: rkhunter o Chkrootkit

Estas herramientas escanean tu sistema en busca de **rootkits** y otro **malware**.

¿Qué es un Rootkit?

Un **Rootkit** es un conjunto de herramientas de software malicioso diseñado para **ocultar la presencia** de *malware* u otro código malicioso en un sistema operativo, permitiendo a un atacante

mantener un **acceso persistente** con privilegios elevados (normalmente a nivel de root o Administrador) sin ser detectado.

La palabra se deriva de dos términos:

1. ***Root***: Se refiere al usuario administrador o de máximo privilegio en sistemas Linux/Unix.
2. ***Kit***: Se refiere al conjunto de herramientas que utiliza el atacante.

Su función principal **no es dañar** el sistema directamente, sino actuar como una **capa de invisibilidad** (o camuflaje) para que otros programas maliciosos (como registradores de teclado, *bots* o *backdoors*) puedan operar libremente.

Mecanismo de Funcionamiento (Ocultamiento)

Un *rootkit* manipula las funciones estándar del sistema operativo para "mentirle" al usuario y a las herramientas de seguridad. Por ejemplo:

- **Manipulación de comandos:** Si un atacante quiere ocultar un archivo llamado backdoor.sh, el *rootkit* modifica comandos como ls, ps, netstat, y find para que, cuando el usuario los ejecute, omitan mostrar los archivos y procesos del atacante.
- **Manipulación del Núcleo (*Kernel*):** Los *rootkits* más peligrosos operan inyectando código directamente en el **núcleo (kernel)** del sistema operativo, el corazón de Linux o Windows, para modificar cómo el sistema gestiona la memoria, los procesos y la red.

El Peligro de los Rootkits

El principal peligro de un *rootkit* reside en su capacidad para **otorgar control total y sigiloso** sobre el sistema.

1. Dificultad Extrema de Detección

- **Evaden el antivirus:** Muchos programas antivirus o *antimalware* tradicionales se basan en escanear el sistema pidiendo una lista de archivos y procesos al sistema operativo. Si un *rootkit* ha manipulado estas funciones, el sistema operativo le miente al antivirus, informando que el *malware* no existe.

- **Afectan la auditoría de integridad (como AIDE):** Un *rootkit* bien diseñado podría manipular las funciones de *hashing* del sistema para que un programa como AIDE piense que sus propios archivos de configuración y binarios no han sido modificados, comprometiendo así toda la seguridad.

2. Control Total y Persistente

Una vez instalado, el atacante tiene:

- **Acceso Permanente:** El atacante puede conectarse y desconectarse cuando quiera sin ser detectado.
- **Máximo Privilegio:** Pueden ejecutar cualquier comando, modificar cualquier archivo o permiso, e instalar cualquier *software* adicional.

3. Usos Comunes por Atacantes

Los *rootkits* se utilizan típicamente para:

- **Espionaje Corporativo:** Robar información sensible, propiedad intelectual o credenciales.
- **Lanzar Ataques:** Usar el equipo comprometido como una plataforma para lanzar ataques DDoS (Denegación de Servicio Distribuido) o campañas de *spam* (convirtiendo el equipo en un "zombi" o *bot*).
- **Evasión Fiscal o Fraude:** Ocultar software que altera datos financieros.

4. Dificultad de Eliminación

Debido a que residen en niveles tan bajos (a veces en el *firmware* o el *kernel*), la eliminación de un *rootkit* es extremadamente compleja. En muchos casos, la única forma segura de restaurar la integridad del sistema es realizar una **reinstalación completa del sistema operativo desde cero**.

Tipos Principales de Rootkits

Los *rootkits* se clasifican según dónde residen en el sistema, siendo la regla general: cuanto más bajo, más peligroso:

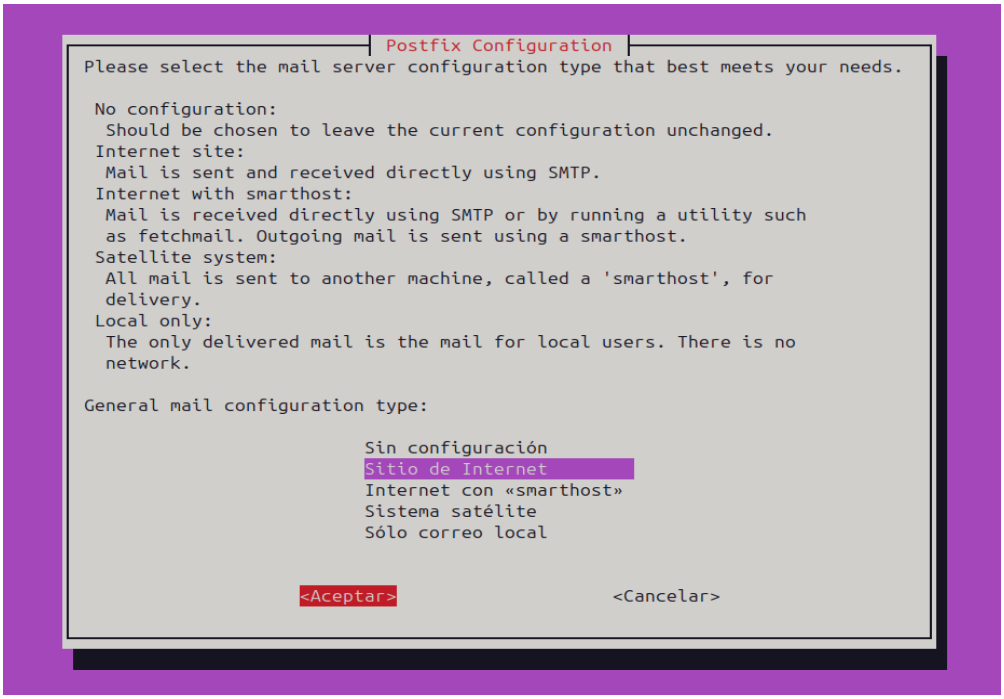
1. **Rootkits de Nivel de Aplicación:** Modifican bibliotecas o *APIs* de aplicaciones específicas. Son los más fáciles de detectar.
2. **Rootkits de Nivel de Núcleo (*Kernel*):** Modifican el *kernel* del sistema operativo (el corazón del SO) para ocultar procesos y archivos a un nivel que es invisible para la mayoría de las herramientas de usuario. Estos son extremadamente peligrosos.
3. **Rootkits de Hipervisor (Virtualización):** Residen por debajo del sistema operativo y lo cargan a él mismo en una máquina virtual. Esto le da al atacante una visión y control totales sobre todo el sistema, y son casi imposibles de detectar desde el sistema operativo comprometido.

Rootkit Hunter

- **rkhunter (Rootkit Hunter):** Es muy popular por su base de datos de firmas conocida de rootkits.

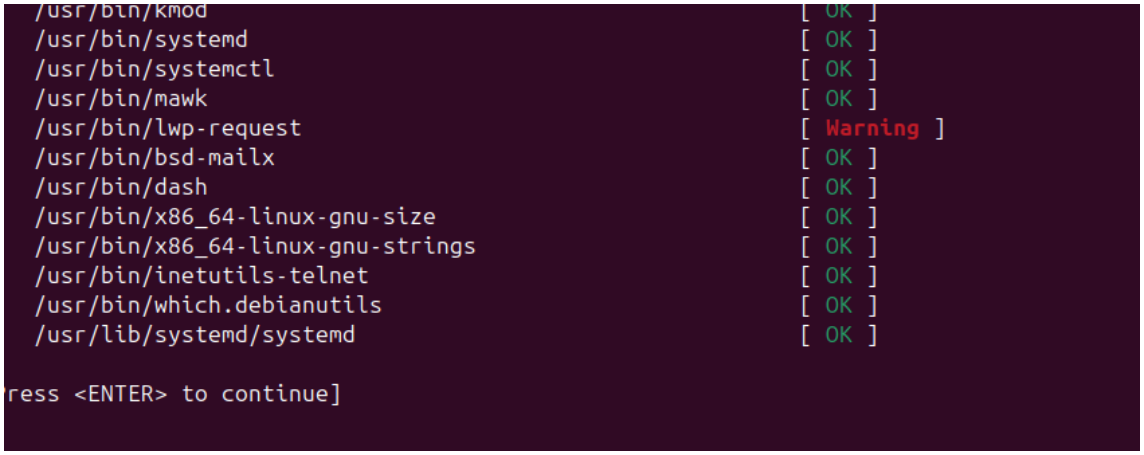
    - **Instalación:**

    sudo apt install rkhunter



- **Uso:**

sudo rkhunter --check ← dos --



Ese aviso es una **advertencia benigna (falso positivo)** emitida por **rkhunter** (Rootkit Hunter). No suele indicar un problema real de seguridad, sino una anomalía que el programa considera sospechosa.

¿Por qué aparece esta advertencia?

1. **Naturaleza del Archivo:** El programa rkhunter espera que ciertos comandos del sistema sean archivos **binarios compilados**, ya que los *rootkits* a menudo sustituyen binarios legítimos por *scripts* maliciosos para ocultarse.
2. **lwp-request es un *script***: El comando /usr/bin/lwp-request es una utilidad legítima para realizar peticiones web (parte del paquete libwww-perl). Sin embargo, en muchas distribuciones de Linux (especialmente Debian/Ubuntu), este programa está implementado como un ***script* de Perl**, no como un binario tradicional.
3. **Conflicto de Expectativas:** Como rkhunter encuentra un *script* donde esperaba un binario (o una implementación diferente), lo marca con una **[Warning]** por precaución.

¿Cómo puedo quitar el aviso? (Solución)

Dado que este es un archivo legítimo instalado por el gestor de paquetes de tu sistema, puedes decirle a rkhunter que lo ignore de forma segura incluyéndolo en la lista blanca de *scripts* permitidos (**SCRIPTWHITELIST**).

1. Edita el archivo de configuración local de rkhunter:

Se recomienda usar el archivo .local para no perder los cambios en las actualizaciones.

sudo nano /etc/rkhunter.conf.local

2. **Añade la siguiente línea** al final del archivo:
3. SCRIPTWHITELIST=/usr/bin/lwp-request
4. **Guarda y cierra** el archivo (Ctrl+O, Enter, Ctrl+X en nano).
5. **Vuelve a ejecutar rkhunter** y el aviso para /usr/bin/lwp-request ya no debería aparecer.

```
Performing group and account checks
  Checking for passwd file                              [ Found ]
  Checking for root equivalent (UID 0) accounts         [ None found ]
  Checking for passwordless accounts                    [ None found ]
  Checking for passwd file changes                      [ Warning ]
  Checking for group file changes                       [ Warning ]
  Checking root account shell history files             [ None found ]

Performing system configuration file checks
  Checking for an SSH configuration file                [ Not found ]
  Checking for a running system logging daemon          [ Found ]
  Checking for a system logging configuration file      [ Found ]
  Checking if syslog remote logging is allowed          [ Not allowed ]

Performing filesystem checks
  Checking /dev for suspicious file types               [ None found ]
  Checking for hidden files and directories             [ Warning ]
```

Esos dos avisos, **Checking for passwd file changes** y **Checking for group file changes**, son advertencias muy comunes y en la mayoría de los casos son **falsos positivos benignos** en rkhunter.

El programa rkhunter está diseñado para detectar si los *rootkits* han modificado archivos críticos del sistema.

¿Por qué aparecen estas advertencias?

rkhunter calcula un *hash* (una huella digital única) de los archivos /etc/passwd y /etc/group la primera vez que se ejecuta (o cuando se actualiza su base de datos) y **almacena ese valor.**

Cualquiera de las siguientes acciones hará que el *hash* actual del archivo **no coincida** con el valor almacenado, generando la advertencia:

1. **Creación de un Nuevo Usuario:** Cada vez que añades un nuevo usuario al sistema con adduser, se modifica el archivo /etc/passwd.
2. **Creación de un Nuevo Grupo:** Cada vez que añades un nuevo grupo con addgroup, se modifica el archivo /etc/group.
3. **Cambios en Usuarios/Grupos Existentes:** Cambiar la *shell* de un usuario, su directorio principal, o su pertenencia a grupos.
4. **Actualizaciones del Sistema:** Algunas actualizaciones del sistema operativo añaden usuarios o grupos de servicio por defecto (por ejemplo, para bases de datos o servicios web).

Si sabes que has hecho cambios legítimos (como crear un usuario o grupo) **entre el escaneo anterior y el actual**, puedes ignorar la advertencia.

Solución: Actualizar la Base de Datos de Propiedades

Para que rkhunter deje de mostrar estas advertencias, debes decirle que **acepte los cambios actuales** como la nueva referencia válida.

Ejecuta el siguiente comando para actualizar la base de datos de propiedades de rkhunter:

sudo rkhunter --propupd

Pasos

1. Ejecuta el comando anterior.
2. rkhunter te preguntará si deseas actualizar los *hashes* de los archivos que han cambiado (incluyendo /etc/passwd y /etc/group).
3. Responde **"S"** (Sí) a la pregunta para actualizar la base de datos.
4. La próxima vez que ejecutes sudo rkhunter --check, esos dos avisos deberían aparecer como **[OK]**.

Checking for hidden files and directories [Warning], indica que **Rootkit Hunter (rkhunter)** ha encontrado archivos o directorios ocultos sospechosos en ubicaciones del sistema donde no se esperan.

¿Por qué aparece esta advertencia?

Los atacantes y los *rootkits* suelen utilizar archivos y directorios ocultos (aquellos cuyos nombres comienzan con un punto, como .oculto o .bashrc) para esconderse. rkhunter escanea directorios críticos, principalmente /dev, /usr/bin, /usr/sbin, y el directorio principal de *root* (/root), en busca de estos elementos.

La advertencia se genera porque rkhunter encontró uno o más de estos archivos o directorios ocultos.

Pasos a Seguir para Investigar y Solucionar

Esta advertencia requiere un paso adicional de investigación, ya que podría ser un falso positivo (archivos legítimos creados por el sistema o por programas) o una señal de un problema real.

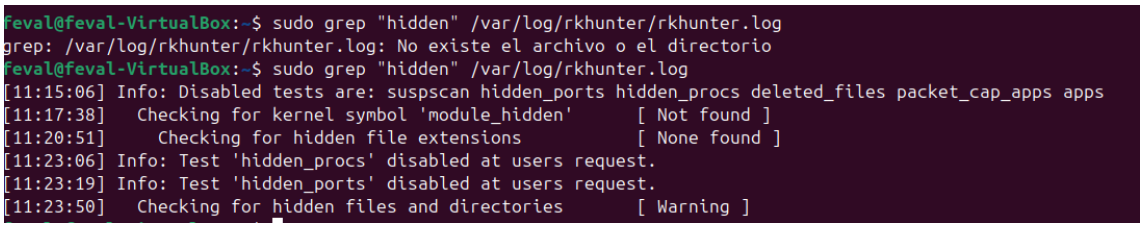
1. Revisar el Archivo de Registro

El primer paso es ver el informe completo generado por rkhunter, donde detallará qué archivos ocultos exactos encontró.

El archivo de registro se encuentra generalmente en /var/log/rkhunter/rkhunter.log.

Utiliza el comando grep para filtrar las líneas que contienen la palabra "hidden" (oculto) en el registro:

```
sudo grep "hidden" /var/log/rkhunter.log
```



El problema es que el comando grep "hidden" /var/log/rkhunter.log **solo te está mostrando las líneas que contienen la palabra "hidden"**, pero el registro real de los archivos ocultos encontrados por rkhunter está en las líneas que siguen a la sección de **"Hidden File and Directory Checks"** (Verificaciones de archivos y directorios ocultos).

rkhunter solo usa la palabra "hidden" para el encabezado de la sección y para las líneas que muestran qué pruebas están desactivadas. Necesitas ver las líneas que tienen la palabra **"Found"** o la ruta exacta del archivo/directorio.

Para encontrar los archivos o directorios ocultos específicos que causaron la advertencia, utiliza un comando de búsqueda más amplio, buscando la palabra **Found** o **Hidden** dentro de esa sección del registro:

```
sudo grep -E 'Found|Hidden' /var/log/rkhunter.log
```

```
[11:23:35]   Checking for a running system logging daemon    [ Found ]
[11:23:35] Info: Found an rsyslog configuration file: /etc/rsyslog.conf
[11:23:35] Info: Found a systemd configuration file: /etc/systemd/journald.conf
[11:23:35]   Checking for a system logging configuration file [ Found ]
[11:23:50] Warning: Hidden file found: /etc/.resolv.conf.systemd-resolved.bak: ASCII text
[11:23:50] Warning: Hidden file found: /etc/.updated: ASCII text
```

Los dos archivos que generaron el aviso [Warning] en la sección Checking for hidden files and directories son:

1. **/etc/.resolv.conf.systemd-resolved.bak**
2. **/etc/.updated**

Ambos son **falsos positivos** y son archivos temporales o de configuración legítimos en sistemas basados en **Debian/Ubuntu 24.04.3 LTS**

Análisis y Solución de los Avisos

1. /etc/.resolv.conf.systemd-resolved.bak

Este archivo es una copia de seguridad temporal de la configuración de resolución de nombres de dominio (DNS) manejada por **systemd-resolved**. Se crea automáticamente durante la gestión de la red y es completamente normal.

2. /etc/.updated

Este es un archivo de marcador de posición utilizado por el sistema de paquetes (APT) de Debian/Ubuntu. Se usa para indicar que el sistema se ha actualizado recientemente y no representa una amenaza.

Cómo Eliminar las Advertencias

Dado que ambos son archivos de sistema legítimos, la forma más segura de detener la advertencia es añadirlos a la lista blanca de archivos ocultos permitidos de rkhunter.

1. **Edita el archivo de configuración local de rkhunter:**

sudo nano /etc/rkhunter.conf

2. **Añade las siguientes líneas** al final del archivo:

Archivos ocultos de sistema legítimos en Ubuntu/Debian

ALLOWHIDDENFILE=/etc/.resolv.conf.systemd-resolved.bak

ALLOWHIDDENFILE=/etc/.updated

3. **Guarda y cierra** el archivo (Ctrl+O, Enter, Ctrl+X en nano).
4. **Vuelve a ejecutar la comprobación de rkhunter** para confirmar que la sección Checking for hidden files and directories ahora muestra **[OK]**.

sudo rkhunter --check

Con estos pasos, mantienes la seguridad de tu sistema sin tener que preocuparte por falsos positivos conocidos.

2. Determinar si son Falsos Positivos

Una vez que tengas la lista de archivos/directorios ocultos, debes comprobar su legitimidad.

Archivo Oculto Típico	**Legitimidad**
/dev/.udev o /dev/.shm	**Falso Positivo.** Estos son archivos/directorios de servicio creados legítimamente por el sistema operativo.

Archivo Oculto Típico	Legitimidad
/root/.ssh	**Legítimo.** Contiene claves SSH del usuario *root*.
Archivos ocultos en /tmp o /var/tmp	**Potencialmente problemático.** Aunque los programas los usan, si son binarios ejecutables, pueden ser sospechosos.
Archivos ocultos en /usr/bin o /usr/sbin	**MUY SOSPECHOSO.** Es muy inusual encontrar archivos ocultos legítimos en directorios de binarios del sistema.

3. Eliminar la Advertencia (Whitelist)

Si determinas que los archivos ocultos son **legítimos** (como /dev/.udev), puedes añadir el archivo o directorio a la lista blanca de rkhunter para que no vuelva a alertar.

1. **Edita el archivo de configuración local:**

```
sudo nano /etc/rkhunter.conf.local
```

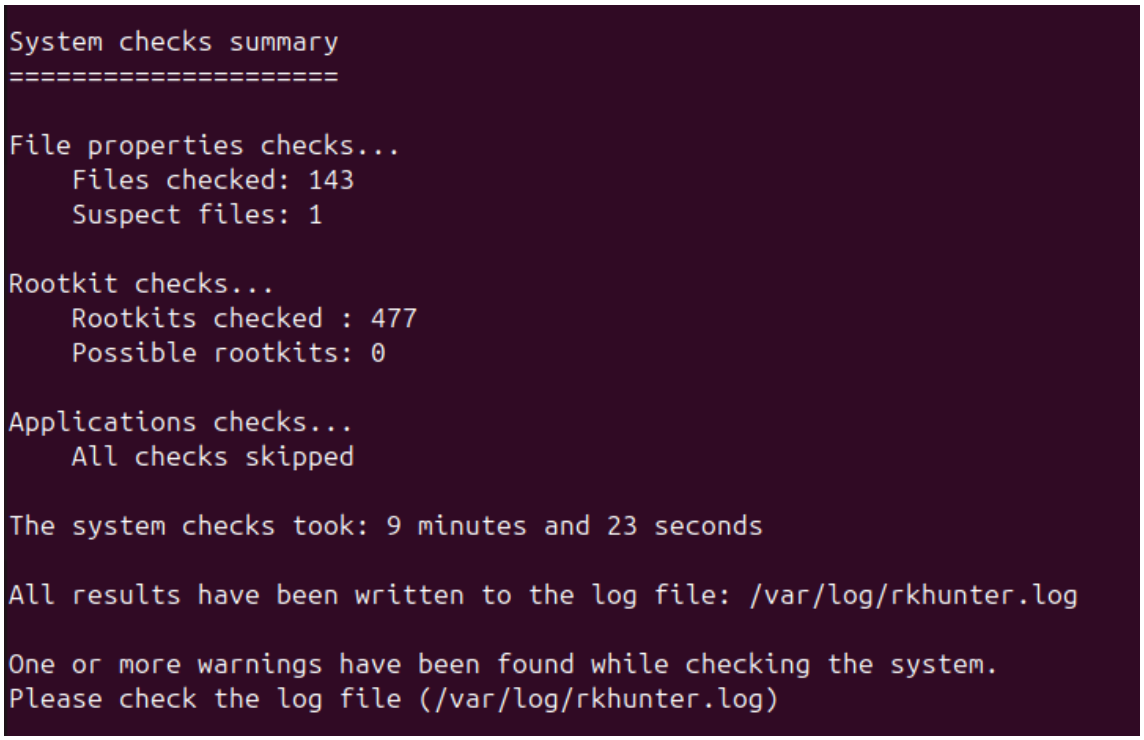
2. **Añade la línea ALLOWHIDDENDIR o ALLOWHIDDENFILE** para cada elemento que quieras ignorar.

Por ejemplo, para ignorar el directorio /dev/.udev:

```
ALLOWHIDDENDIR=/dev/.udev
```

3. **Guarda y cierra** el archivo.
4. Vuelve a ejecutar **sudo rkhunter --check** para confirmar que la advertencia ha desaparecido.

Nota importante: Solo añade a la lista blanca aquellos archivos de los que estés **absolutamente seguro** que son legítimos y parte del sistema operativo. Si encuentras un archivo oculto sospechoso en /usr/bin, deberías investigarlo como una posible intrusión.



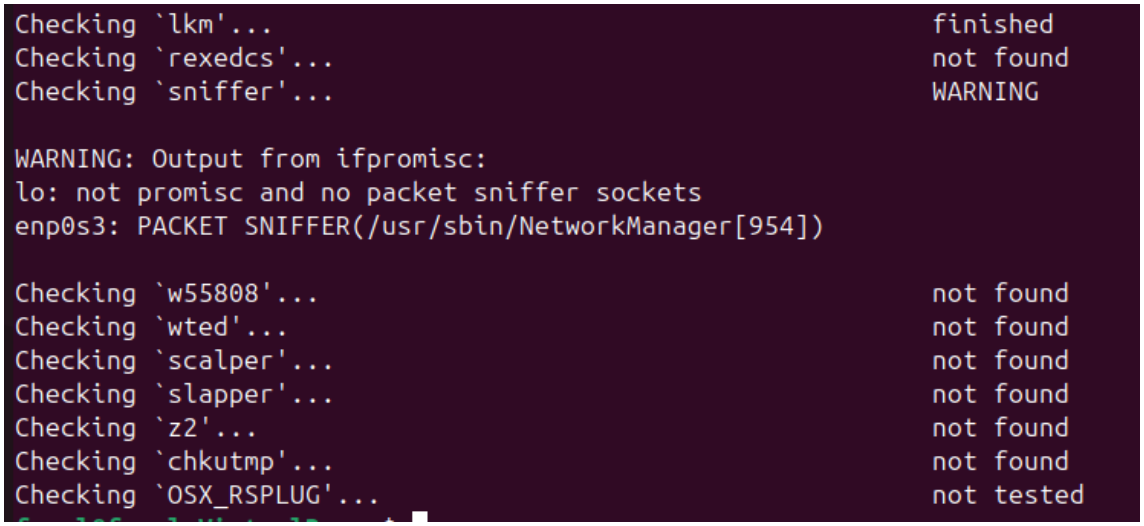
Chkrootkit

- **Chkrootkit:** Otra alternativa más ligera.
    - **Instalación:**

    sudo apt install chkrootkit

    - **Uso:**

    sudo chkrootkit



Las dos únicas advertencias que te ha dado chkrootkit son **falsos positivos comunes** en sistemas modernos basados en Debian/Ubuntu.

1. Advertencia de Archivos y Directorios Sospechosos

Advertencia: Searching for suspicious files and dirs... WARNING

Archivos reportados:

- /usr/lib/ruby/vendor_ruby/rubygems/ssl_certs/.document
- /usr/lib/ruby/vendor_ruby/rubygems/tsort/.document
- /usr/lib/ruby/vendor_ruby/rubygems/optparse/.document
- /usr/lib/modules/6.14.0-33-generic/vdso/.build-id
- /usr/lib/libreoffice/share/.registry
- /usr/lib/debug/.dwz
- /usr/lib/debug/.build-id

Análisis

Todos estos archivos son **falsos positivos**. El *script* de chkrootkit busca archivos ocultos (que empiezan con un punto .) en directorios de sistema sensibles, pero muchos paquetes legítimos de Linux, como **RubyGems**, **LibreOffice**, y los paquetes de **depuración del kernel**, utilizan archivos ocultos como marcadores, cachés o metadatos.

- Los archivos .document son marcadores de paquetes Ruby.
- El directorio .build-id es parte del sistema de paquetes de depuración.
- .registry pertenece a LibreOffice.

Acción Requerida

Ninguna acción de seguridad es necesaria. Puedes ignorar estos archivos con seguridad. chkrootkit no ofrece una forma sencilla de poner estos archivos en una lista blanca como rkhunter, por lo que simplemente debes recordar que son normales si vuelves a ejecutarlo.

2. Advertencia del "Sniffer" (Olfateador de Red)

Advertencia: Checking 'sniffer'... WARNING

Detalle: enp0s3: PACKET SNIFFER(/usr/sbin/NetworkManager[954])

Análisis

Esta advertencia es un **falso positivo muy común** en la mayoría de los sistemas con un entorno de escritorio (como Ubuntu).

chkrootkit reporta que el proceso **NetworkManager** (/usr/sbin/NetworkManager[954]) está escuchando en modo **"PACKET SNIFFER"** (modo promiscuo) en la interfaz de red enp0s3.

- **¿Qué es el modo promiscuo?** Es un modo en el que la interfaz de red captura todos los paquetes que pasan por la red, no solo los dirigidos a ella, algo que hacen los *sniffers* maliciosos.
- **La realidad:** El **NetworkManager** utiliza este modo de manera legítima para algunas funciones de red, especialmente al interactuar con **VirtualBox** (como indica tu *prompt* feval-VirtualBox) o para tareas de configuración de red, y **no indica una intrusión**.

Acción Requerida

Ninguna acción de seguridad es necesaria. Este es el comportamiento esperado del NetworkManager. Puedes ignorar esta advertencia también.

3. Escaneo de Antivirus (Opcional, Principalmente para Archivos Compartidos): ClamAV

Aunque los sistemas Linux son menos propensos a los virus de Windows, **ClamAV** es útil si tu máquina interactúa con archivos de otros sistemas operativos (como un servidor de correo o archivos compartidos) y quieres escanearlos.

- **Instalación:** sudo apt install clamav clamav-daemon
- **Actualizar la base de datos de virus:** sudo freshclam
- **Escanear un directorio, por ejemplo, el directorio personal:** clamscan -r /home/tu_usuario

4. Monitoreo de Integridad de Archivos: AIDE (Advanced Intrusion Detection Environment)

AIDE crea una base de datos de "instantánea" de la estructura, permisos y contenido de archivos importantes del sistema. Si alguno de estos archivos cambia (lo que podría indicar una intrusión), AIDE te avisará en las comprobaciones futuras.

- **Instalación:**

```
sudo apt install aide
```

- **Inicializar la base de datos por primera vez (¡haz esto inmediatamente después de la instalación!):**

```
sudo aide --init --config /etc/aide/aide.conf
```

Cuidado, en una máquina virtual o en un disco lento o lleno tarda mucho en generar la base de datos de integridad del sistema

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

(para activar la base de datos)

- **Verificación (para ejecutar regularmente):**

```
sudo aide --check
```

Buenas Prácticas de Seguridad Adicionales

Además de las herramientas, estas configuraciones simples son cruciales:

- **Mantén tu Sistema Actualizado:** Esta es la medida de seguridad más básica y efectiva. Siempre aplica las últimas actualizaciones de seguridad.

  ```
  sudo apt update
  sudo apt upgrade
  ```

- **Deshabilita Servicios Innecesarios:** Cuantos menos servicios estén corriendo y escuchando conexiones, menor será la superficie de ataque. Revisa qué servicios tienes habilitados.

- **Configura una Contraseña Robusta:** Usa una contraseña fuerte y, si es posible, habilita la **autenticación de dos factores (2FA)** si accedes a tu sistema a través de SSH.

El proceso de **fortalecer un sistema Linux** (o *hardening*) se puede automatizar y guiar a través de múltiples proyectos de código abierto disponibles en GitHub. Estos proyectos suelen ser *scripts* o guías que aplican de forma sistemática las mejores prácticas de seguridad, a menudo siguiendo estándares como los de **CIS (Center for Internet Security)** o **DISA STIG**.

Herramientas y Scripts de Hardening

Estos proyectos ofrecen *scripts* o herramientas para automatizar la configuración de seguridad.

1. Lynis

Aunque no es un proyecto de *scripting* de *hardening* directo, **Lynis** es una herramienta de auditoría de seguridad y *hardening* muy popular en Linux. Escanea a fondo tu sistema en busca de debilidades y ofrece sugerencias detalladas sobre cómo corregirlas.

- **¿Qué hace?** Escanea el sistema operativo, configuración del kernel, paquetes instalados, información de red, firewall, etc., y te da una "puntuación" de seguridad con recomendaciones.
- **Proyecto GitHub:** Puedes encontrar el repositorio oficial y la documentación en GitHub. Su valor está en la **guía de las mejoras** que debes aplicar.

2. The Practical Linux Hardening Guide

Este repositorio no es una herramienta ejecutable, sino una **guía muy completa y práctica** sobre cómo proteger manualmente (o con scripts) un sistema GNU/Linux de producción. Es excelente para entender las bases del *hardening*.

- **¿Qué contiene?** Instrucciones detalladas sobre la configuración del kernel (sysctl), el sistema de archivos, el firewall, SSH, servicios, cifrado, y más.

- **Proyecto GitHub:** trimstray/the-practical-linux-hardening-guide

3. Scripts de Hardening Específicos (Ejemplo para Ubuntu)

Existen múltiples repositorios que ofrecen scripts Bash semi-automatizados para aplicar la seguridad en distribuciones como Ubuntu o Debian, a menudo enfocándose en cumplir con estándares.

- **Ejemplo:** Proyectos como **Z-A-P-P-I-T/Ubuntu-Security-Hardening-Script** o **captainzero93/security_harden_linux** suelen incluir módulos para:
    - Configurar **UFW** (firewall).
    - Instalar y configurar **Fail2Ban** (prevención de intrusiones).
    - Fortalecer la configuración de **SSH**.
    - Aplicar políticas de **contraseñas robustas**.
    - Instalar herramientas de detección como **rkhunter**.

Automatización y Gestión de Configuración (Ansible/SaltStack)

Para entornos más grandes o si quieres una gestión de configuración repetible, los proyectos basados en herramientas de automatización son clave.

4. Roles de Ansible para Hardening

Ansible es una herramienta de automatización muy utilizada. En GitHub encontrarás **"roles"** de Ansible diseñados para aplicar configuraciones de *hardening*.

- **Ejemplo:** El proyecto **ansible-lockdown** ofrece roles que te permiten aplicar configuraciones de seguridad basadas en las guías **CIS** o **DISA STIG** para distintas versiones de Linux (incluyendo Ubuntu). Esto garantiza que tu configuración cumple con un estándar reconocido de la industria.

Detección de Intrusiones y Auditoría

5. Auditoría con OpenSCAP/SCAP Workbench

El proyecto **OpenSCAP** proporciona una implementación de las guías de seguridad **SCAP (Security Content Automation Protocol).** Permite escanear y aplicar perfiles de *hardening* que cumplen con rigurosos estándares gubernamentales o industriales.

- **¿Qué contiene?** Herramientas para auditar tu sistema contra políticas de seguridad (como las reglas CIS o STIG) y generar informes de cumplimiento.

Advertencia Importante

Antes de ejecutar cualquier *script* de *hardening* de GitHub (o de cualquier otra fuente):

1. **Revisa el Código:** Asegúrate de entender exactamente qué comandos ejecuta el *script*. Un *script* de *hardening* mal configurado puede **romper tu sistema** o bloquear tu acceso.
2. **Haz Copias de Seguridad: Siempre** haz una copia de seguridad o prueba el script en un entorno de desarrollo/prueba antes de aplicarlo a un sistema de producción.
3. **Elige Proyectos Activos:** Prioriza repositorios que tengan actividad de *commits* reciente y un buen número de estrellas/forks, ya que es más probable que estén mantenidos y sean compatibles con las versiones actuales de Linux (como Ubuntu 24.04).