ESCUELA POLITÉCNICA NACIONAL

CARRERA DE INGENIERÍA DE SOFTWARE

DESARROLLO DE SOFTWARE SEGURO ISWD853

Informe de análisis estático

Alumnos:

Alejandro Jiménez

Bryan Rosillo

Christian Hernández

Jorge Segovia

Mateo Dávalos

PROFESOR: Jhonattan Barriga

FECHA DE ENTREGA: 08/01/2025

Contenido

I. Código fuente

- https://github.com/BryanRosillo/Core_bank_ec

II. Resumen ejecutivo

En este apartado se resumirá los hallazgos que se han dado en el análisis de la aplicación analizada, en la siguiente imagen se puede observar de forma gráfica las vulnerabilidades halladas.

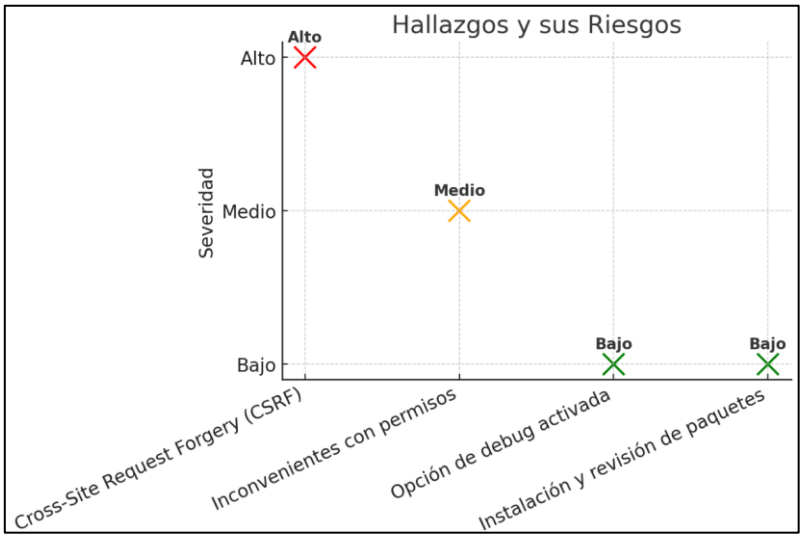


Figura 1. Gráfica de resumen de vulnerabilidades

Ahora se presenta a manera de tablas la gravedad y recomendaciones para combatir con dichas vulnerabilidades.

0	1	1	2	0
Crítico	Alto	Medio	Bajo	Informativo

Tabla 1. Vulnerabilidades descubiertas y su gravedad

Hallazgo	Severidad	Recomendación
Cross-Site Request Forgery (CSRF)	**Alto**	Para proteger contra ataques CSRF, se debe habilitar la protección por defecto en métodos HTTP inseguros, implementar tokens CSRF seguros y evitar el uso de métodos como GET para realizar operaciones sensibles. En aplicaciones Flask, es fundamental utilizar el módulo CSRFProtect y asegurarse de que no se deshabilite, evitando configurar WTF_CSRF_ENABLED = False. Además, la protección CSRF debe mantenerse activa en vistas y formularios (csrf = True). Estas prácticas refuerzan la seguridad de la aplicación y minimizan el riesgo de explotación de vulnerabilidades.

Inconvenientes con permisos.	**Medio**	En el Dockerfile, se recomienda crear un usuario no privilegiado con la instrucción USER y, si es posible, utilizar usuarios específicos de la imagen, como postgresql o zookeeper, en lugar de root. En contenedores Windows, se debe emplear ContainerUser. Al ejecutar el contenedor, es importante especificar un usuario con la opción --user en Docker o en docker-compose.yml, y en Linux, asignar capacidades específicas solo cuando sea necesario para reducir riesgos de seguridad y minimizar la exposición a ataques.
Opción de debug activada	**Bajo**	En aplicaciones Flask, es fundamental deshabilitar las funciones de depuración en entornos de producción para evitar la exposición de información sensible. Para ello, se debe configurar app.debug = False y asegurarse de ejecutar la aplicación con app.run(debug=False). Mantener la depuración activada en producción puede revelar rutas de archivos, configuraciones internas y mensajes de error detallados, lo que aumenta el riesgo de explotación de vulnerabilidades. Implementar esta práctica refuerza la seguridad y evita fugas de información crítica
Instalación y revisión de paquetes	**Bajo**	Para mejorar la seguridad en Docker, es fundamental evitar la instalación de paquetes innecesarios y verificar vulnerabilidades en los paquetes requeridos. Se recomienda utilizar: “RUN apt update && apt --no-install-recommends install -y build-essential && rm -rf /var/lib/apt/lists/*”

Tabla 2. Recomendaciones para las vulnerabilidades

III. Informe y análisis

a) Resultados del análisis estático

Para llevar a cabo el análisis estático, se ha elegido la herramienta SonarCloud, ya que permite un monitoreo continuo ante nuevos cambios o actualizaciones. Además, facilita la identificación de vulnerabilidades de seguridad, proporcionando una trazabilidad clara del origen del problema y las posibles soluciones.

- **Problema 1**

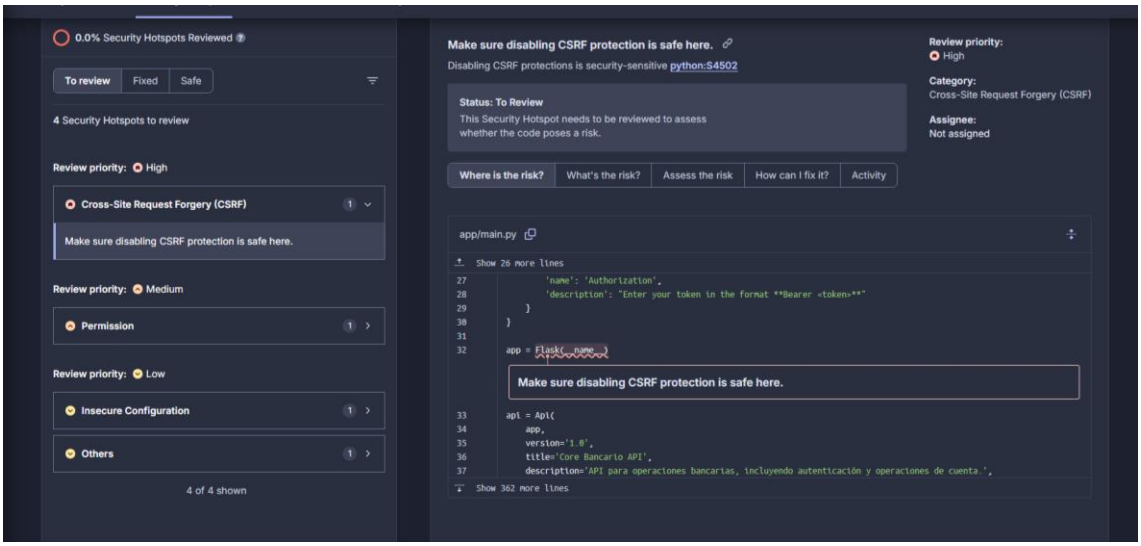
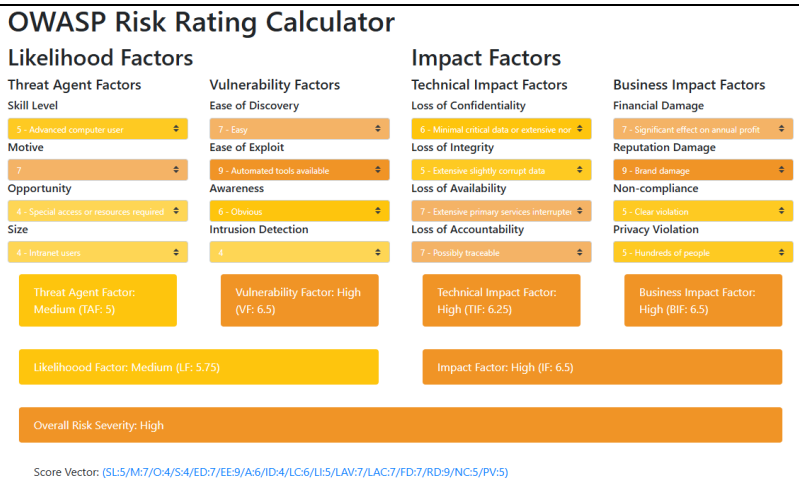


Figura 2. Resultado de problema de seguridad nivel alto

Identificador	Problema de seguridad WT-1
Título	Cross-Site Request Forgery (CSRF)
Descripción	Un ataque de Cross-Site Request Forgery (CSRF) ocurre cuando un atacante logra que un usuario legítimo de una aplicación web ejecute acciones sensibles sin su consentimiento, como actualizar su perfil, enviar un mensaje o cualquier otra operación que modifique el estado de la aplicación. El atacante puede engañar a la víctima para que haga clic en un enlace que desencadena la acción privilegiada o visite un sitio web malicioso que contiene una solicitud web oculta. Dado que los navegadores incluyen automáticamente las cookies en las solicitudes, estas acciones pueden ser autenticadas y ejecutadas sin que el usuario lo advierta, lo que compromete la seguridad de la aplicación.
Referencias	• OWASP_2021_A07 • CWE-352 • CWE-20

Riesgo	

Recomendaciones:

Por la parte de prácticas se debe tener en cuenta los siguientes puntos:

- Habilitar la protección por defecto en todos los métodos HTTP inseguros.
- Implementar un token CSRF seguro y difícil de predecir para validar cada solicitud que modifique el estado de la aplicación.
- Evitar el uso de métodos HTTP seguros (como GET) para realizar operaciones sensibles, ya que estos deben emplearse exclusivamente para la recuperación de información.

En cuanto al código como se está usando flask se recomienda lo siguiente:

- Se debe utilizar el módulo **CSRFProtect** y asegurarse de que no se deshabilite, evitando configurar **WTF_CSRF_ENABLED** en **False**. Esto garantiza una protección adecuada contra ataques **CSRF** y refuerza la seguridad de las solicitudes en la aplicación.
- No se debe deshabilitar la protección CSRF en las vistas y formularios: **csrf = True # Compliant**

- **Problema 2**

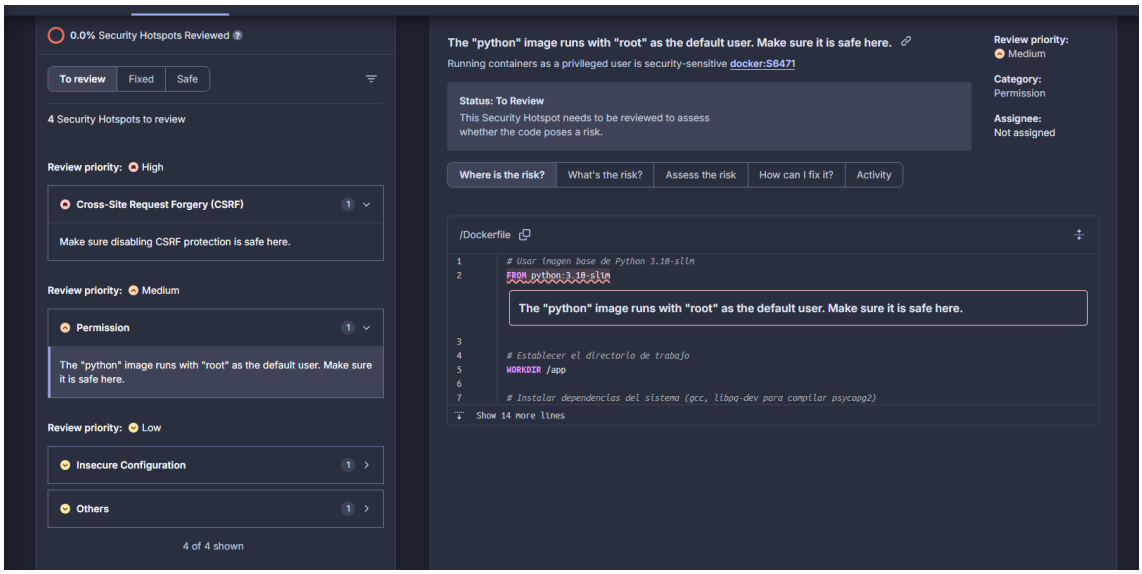
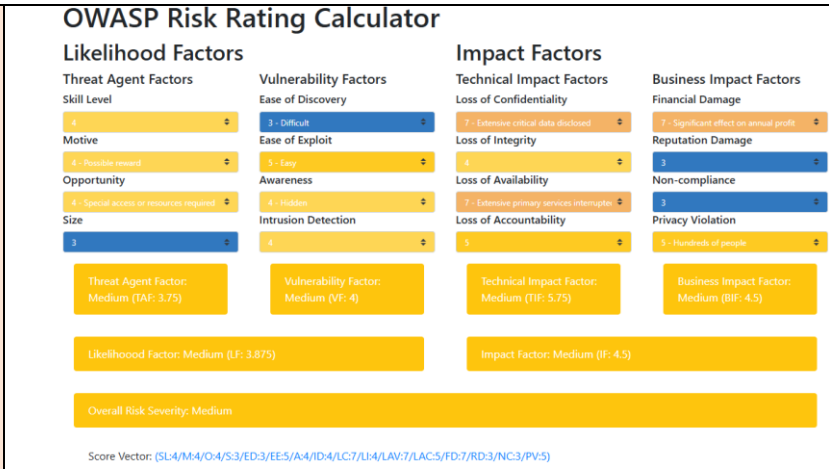


Figura 3. Resultado de problema de seguridad nivel medio

Identificador	Problema de seguridad WT-2
Título	Inconvenientes con permisos.
Descripción	El contenedor de Python se ejecuta con "root" como usuario predeterminado, lo que puede representar un riesgo de seguridad. Ejecutar contenedores con un usuario privilegiado (root en Linux o ContainerAdministrator en Windows) compromete la seguridad, permitiendo que cualquier código en el contenedor realice acciones administrativas. Un atacante puede ejecutar código arbitrario, acceder a archivos sensibles, establecer conexiones maliciosas o incluso escapar del contenedor.
Referencias	• OWASP_2021_A05 • CWE-284
Riesgo	



Recomendaciones:

Para el dockerfile lo que se puede hacer es:

- Crear un usuario no privilegiado y establecerlo con la instrucción USER.
- Usar usuarios específicos proporcionados por la imagen, como postgresql o zookeeper, en lugar de root.
- En contenedores Windows, utilizar ContainerUser.

Al momento de ejecutar el contenedor pues:

- Especificar un usuario con la opción --user en Docker o en docker-compose.yml.
- Asignar capacidades específicas en Linux solo cuando se requieran privilegios elevados.

Para explicar mejor lo dicho se colocan imágenes con código de ejemplo de algunas de las situaciones:

Linux

```
FROM alpine

RUN addgroup -S nonroot \
    && adduser -S nonroot -G nonroot

USER nonroot

ENTRYPOINT ["id"]
```

Windows

```
FROM mcr.microsoft.com/windows/servercore:ltsc2019

RUN net user /add nonroot

USER nonroot
```

Builds

```
FROM alpine as builder
COPY Makefile ./src /
RUN make build

FROM alpine as runtime
RUN addgroup -S nonroot \
    && adduser -S nonroot -G nonroot
COPY --from=builder bin/production /app
USER nonroot
ENTRYPOINT ["/app/production"]
```

- **Problema 3**

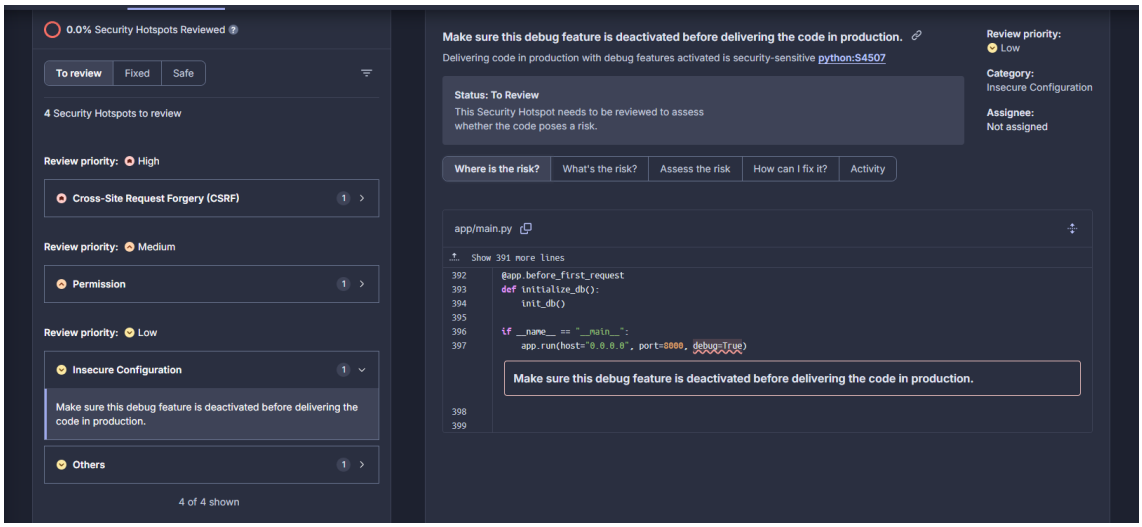
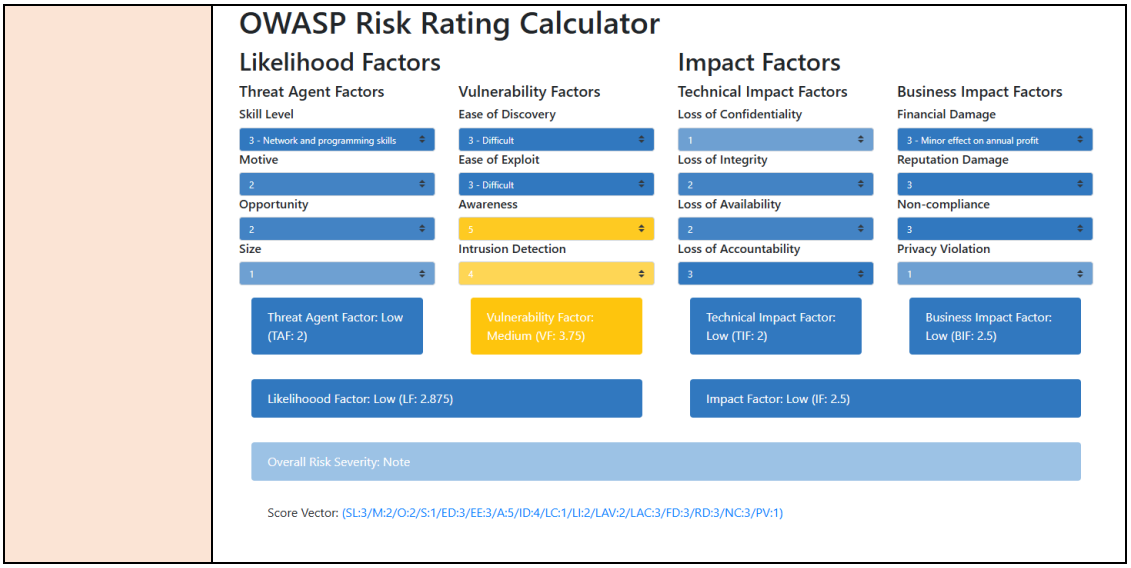


Figura 4. Resultado de problema de seguridad nivel bajo

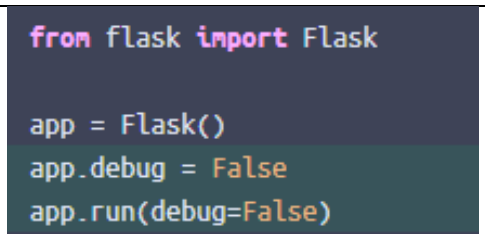
Identificador	Problema de seguridad WT-3
Título	Opción de debug activada
Descripción	Las herramientas y frameworks de desarrollo suelen incluir opciones para facilitar la depuración. Aunque son útiles en entornos de desarrollo, nunca deben estar habilitadas en producción, ya que pueden exponer información sensible del sistema, como rutas de la aplicación o nombres de archivos, a través de mensajes de error o instrucciones de depuración.
Referencias	• OWASP_2021_A5 • CWE-489 • CWE-215
Riesgo	



Recomendaciones:

- No activar las funciones de depuración en servidores de producción ni en aplicaciones destinadas a los usuarios finales.

Para el código que se relacione con flask se debe colocar lo siguiente:



- **Problema 4**

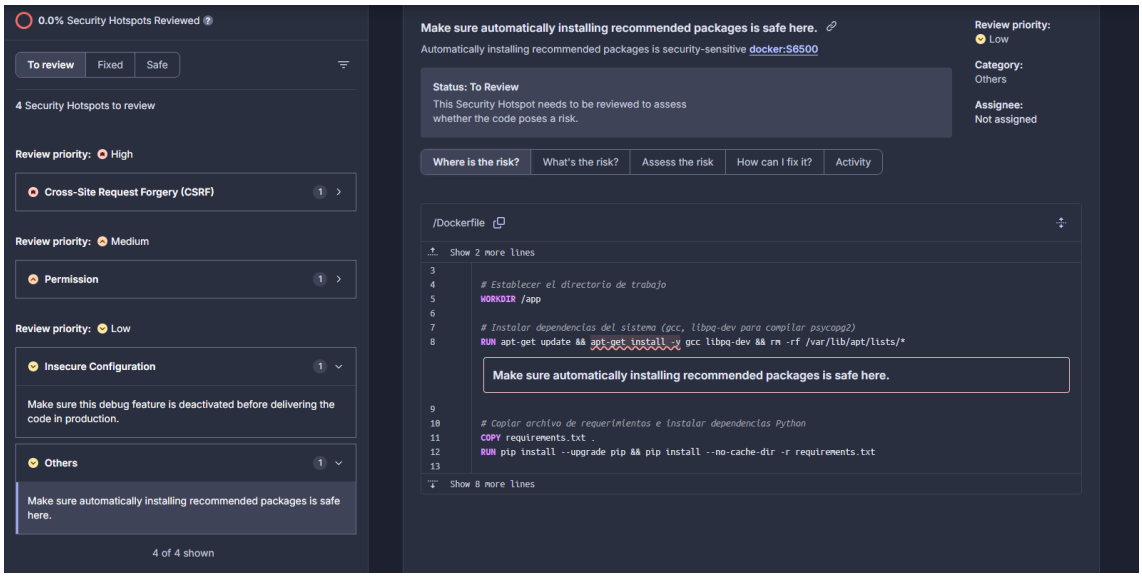
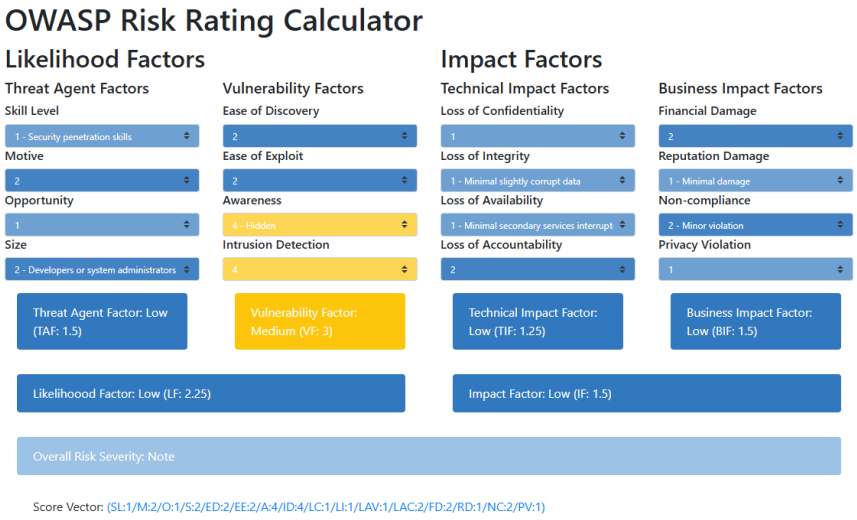


Figura 5. Resultado de problema de seguridad nivel bajo en asuntos de "otros"

Identificador	Problema de seguridad WT-4
Título	Instalación y revisión de paquetes
Descripción	Instalar paquetes innecesarios en imágenes Docker aumenta la superficie de ataque, pudiendo contener vulnerabilidades o código malicioso. Esto puede facilitar ataques a la cadena de suministro o escaladas de privilegios.
Referencias	• OWASP_2021_A06 • CWE-937
Riesgo	

Recomendaciones:

- Evitar la instalación de paquetes innecesarios.
- Verificar si existen vulnerabilidades en los paquetes requeridos.

Si nos vamos por la parte de comandos pues lo recomendable sería usar lo siguiente:

RUN apt update && apt --no-install-recommends install -y build-essential && rm -rf /var/lib/apt/lists/*

- **apt update** → Evita instalar paquetes obsoletos o vulnerables.
- **--no-install-recommends** → Reduce dependencias innecesarias, limitando la superficie de ataque.
- **rm -rf /var/lib/apt/lists/*** → Elimina caché de apt, reduciendo el tamaño y ocultando información sensible.