Práctica: Riesgos de Configuración en Docker

Objetivo

Explorar los riesgos de ciertas configuraciones en Docker que pueden exponer al host a problemas de seguridad o permitir accesos no deseados. La práctica cubrirá:

1. El riesgo de usar `--pid=host`.
2. El uso de contenedores de solo lectura.
3. Los riesgos de manejo de secretos en `Dockerfile`.
4. El riesgo de usar el modo `--privileged`.

---

1. Riesgo de Usar `--pid=host`

La opción `--pid=host` permite compartir el espacio de nombres de procesos (PID) entre el contenedor y el host. Esto significa que los procesos dentro del contenedor pueden acceder y visualizar los procesos en ejecución en el host, lo que puede exponer información sensible.

Instrucciones:

1. Ejecuta el siguiente comando para iniciar un contenedor Debian con `--pid=host` y privilegios completos:

   ```
   docker run -it --privileged --pid=host debian nsenter -t 1 -n -u -i -m sh
   ```

   - **`--privileged`**: Otorga al contenedor acceso a las capacidades completas del kernel, incluidos los dispositivos.
   - **`--pid=host`**: Permite al contenedor compartir el espacio de nombres de procesos con el host.
   - **`nsenter`**: Accede a diferentes namespaces del host.

2. Dentro del contenedor, prueba los siguientes comandos para ver la información del host:

   ```
   ps aux              # Muestra todos los procesos del host.
   lsns                # Lista todos los espacios de nombres activos en el host.
   cat /etc/os-release # Muestra la información del sistema operativo del host.
   ```

   Explicación: Usar `--pid=host` junto con `nsenter` permite al contenedor visualizar y potencialmente manipular procesos del host, lo que puede suponer un riesgo de seguridad si el contenedor es comprometido o mal utilizado.

2. Contenedores en Modo de Solo Lectura

La opción `--read-only` de Docker permite iniciar un contenedor en modo solo lectura, lo que limita las modificaciones en el sistema de archivos y ayuda a proteger el contenedor contra cambios no deseados. Sin embargo, esta opción bloquea la escritura en todas las ubicaciones excepto en los puntos de montaje explícitos.

Instrucciones:

1. Ejecuta un contenedor en modo solo lectura:

   ```
   docker run -it --read-only ubuntu bash
   ```

   - Intenta crear un archivo en el sistema:

     ```
     touch /nuevo_archivo.txt
     ```

     Resultado esperado: Debería fallar con un mensaje de "Read-only file system", ya que el contenedor está en modo solo lectura.

2. Ejecuta el contenedor en modo solo lectura pero permite escribir en `/tmp` usando `--tmpfs`:

   ```
   docker run -it --read-only --tmpfs /tmp ubuntu bash
   ```

   - Intenta crear un archivo en `/tmp`:

     ```
     touch /tmp/archivo_tmp.txt
     ```

     Resultado esperado: Este comando debería funcionar, ya que `/tmp` está montado como un sistema de archivos temporal con permisos de escritura.

   Explicación: El modo solo lectura es útil para evitar cambios en el contenedor, pero debe combinarse con montajes temporales (`tmpfs`) para permitir ciertos procesos que requieren escritura temporal.

3. Cuidado con los Secretos en el `Dockerfile`

Agregar secretos (como claves API o contraseñas) directamente en un `Dockerfile` es una mala práctica, ya que quedan registrados en las capas de la imagen y pueden ser visibles a través del historial de la imagen.

Instrucciones:

1. Crea un archivo `Dockerfile` que añada un secreto:

```
# Dockerfile
FROM ubuntu:latest
RUN echo "clave_secreta=supersecreto123" > /root/secreto.txt
CMD ["cat", "/root/secreto.txt"]
```

2. Construye la imagen:

```
docker build -t secreto-demo .
```

3. Ejecuta el comando `docker history` para ver el historial de capas y observa que el secreto puede estar expuesto:

```
docker history secreto-demo
```

Resultado esperado: El historial muestra el comando que incluye el secreto. Cualquier persona con acceso a la imagen podría ver esta información.

Explicación: Nunca guardes secretos en el `Dockerfile` o en el sistema de archivos de la imagen; en su lugar, usa herramientas como `docker secrets` o pasa variables de entorno en tiempo de ejecución para evitar exponer información sensible.

---

4. Riesgo de Usar el Modo `--privileged`

El modo `--privileged` permite al contenedor acceder a todas las capacidades del kernel del host, lo cual es útil en ciertos casos, pero es una configuración peligrosa porque elimina la mayoría de las restricciones de seguridad.

Instrucciones:

1. Ejecuta un contenedor con el modo `--privileged` y realiza cambios en la red:

```
docker run --rm -it --privileged alpine /bin/sh
```

Dentro del contenedor, prueba ejecutar los siguientes comandos:

```
ifconfig                # Verifica la configuración de red.
ip route add 10.10.20.0/24 via 0.0.0.0 dev eth0
```

Resultado esperado: Ambos comandos deben ejecutarse sin problemas, ya que el contenedor tiene permisos privilegiados.

2. Ejecuta el mismo contenedor pero sin `--privileged` y repite los comandos de red:

   ```
   docker run --rm -it ubuntu /bin/bash
   ```

   Resultado esperado: Los comandos de red deberían fallar sin `--privileged`, ya que el contenedor no tiene permisos elevados.

   Explicación: Usar `--privileged` otorga al contenedor acceso sin restricciones, lo que es peligroso si el contenedor se ve comprometido, ya que podría permitir la manipulación de la red del host y otras configuraciones críticas.

Conclusión de la Práctica

1. **Evita usar `--pid=host`** salvo que sea necesario y entiendas los riesgos, ya que permite al contenedor compartir el espacio de nombres de procesos con el host.
2. **Usa contenedores de solo lectura** para evitar cambios en el sistema de archivos del contenedor. Para necesidades de escritura temporal, usa `--tmpfs`.
3. **Nunca guardes secretos en el `Dockerfile`**, ya que pueden quedar expuestos en el historial de la imagen.
4. **Evita el modo `--privileged`** siempre que sea posible, ya que otorga acceso completo al sistema, eliminando restricciones de seguridad.

Este conjunto de prácticas te ayudará a identificar y mitigar algunos de los principales riesgos de configuración en Docker.