Práctica: Usuarios y Privilegios en Docker

Objetivo

1. Entender cómo Docker maneja los permisos de usuario por defecto.
2. Observar cómo los contenedores pueden ejecutar procesos con privilegios de root y las implicaciones de esto.
3. Mostrar cómo limitar privilegios usando usuarios específicos para evitar escaladas de privilegios.

Paso 1: Ejecutar un Contenedor con Privilegios

1. Ejecuta un contenedor de Alpine en modo interactivo para observar los permisos:

   ```
   docker run -it alpine sh
   ```

2. Dentro del contenedor, verifica el usuario con el que estás ejecutando el shell:

   ```
   whoami   # Esto debería mostrar "root"
   id       # Muestra detalles de UID y GID, debería ser UID=0
   ```

   Explicación: Por defecto, Docker ejecuta los contenedores como `root`, lo que significa que tienes permisos de administración dentro del contenedor. Sin embargo, este `root` está aislado en el contenedor y no debería tener acceso directo al sistema host, a menos que se utilicen opciones específicas como `--privileged`.

3. Mantén el contenedor activo durante 100 segundos y sal de la sesión:

   ```
   sleep 100
   ```

   Luego, presiona `CTRL + P + Q` para salir del contenedor sin detenerlo. Esto mantendrá el proceso `sleep` activo en segundo plano.

4. En el host, verifica el proceso de `sleep` en ejecución:

   ```
   ps -fc | grep sleep
   ```

   Explicación: Observa que el proceso de `sleep` tiene el UID=0, lo que indica que está ejecutándose como `root` dentro del contenedor y en el contexto del host. Esto demuestra que el contenedor tiene permisos elevados en su proceso.

> ⚠️ **Nota de Seguridad**: Mantener procesos activos con UID=0 puede permitir que, en ciertas configuraciones incorrectas, un contenedor afecte el host. Por lo tanto, es importante restringir los permisos.

Paso 2: Mostrar Diferencias en el Acceso a Archivos con y sin Privilegios

1. Ejecuta un contenedor en modo privilegiado para ver el archivo de contraseñas encriptadas (`/etc/shadow`), al que normalmente solo `root` tiene acceso:

   ```
   docker run alpine cat /etc/shadow
   ```

   Resultado Esperado: Dado que el contenedor está ejecutándose como `root` , debería mostrar el contenido del archivo `/etc/shadow` .

2. Ahora, ejecuta un contenedor como un usuario no privilegiado (por ejemplo, UID=1001) y trata de acceder al mismo archivo:

   ```
   docker run --user 1001 alpine cat /etc/shadow
   ```

   Resultado Esperado: Este comando debería fallar con un error de "Permiso denegado", ya que el usuario no tiene privilegios para leer `/etc/shadow` .

   Explicación: Esto muestra cómo ejecutar contenedores con un usuario sin privilegios mejora la seguridad y previene el acceso a archivos críticos.

Conclusión

Para mejorar la seguridad de los contenedores:

- **No uses** `root` en el contenedor a menos que sea absolutamente necesario.
- **Define un usuario no root** en el `Dockerfile` para que el contenedor siempre se ejecute con un usuario sin privilegios.
- **Evita el modo** `--privileged` a menos que sea imprescindible, ya que otorga permisos amplios que pueden representar un riesgo para el host.

Ejemplo de un Dockerfile Seguro

Para crear una imagen que use un usuario no root:

```
# Usar Alpine como base
FROM alpine:latest

# Crear un usuario no root (UID 1001)
RUN adduser -D -u 1001 appuser

# Cambiar al usuario sin privilegios
USER appuser

# Definir el comando por defecto
CMD ["sh"]
```

Este `Dockerfile` crea un usuario sin privilegios (`appuser`) que evita accesos innecesarios y protege contra escaladas accidentales de privilegios, mejorando la seguridad del contenedor.