Permisos de Archivos en Linux: Modo Simbólico vs. Modo Octal

Un error común pero peligroso reside en asignar permisos de ejecución a un *script* en sistemas tipo Unix (como Linux), y cómo el **Modo Octal** es la solución preferida sobre el **Modo Simbólico Relativo** para lograr resultados predecibles y seguros.

1. El Problema del Modo Simbólico Relativo (+x)

El comando chmod +x deploy.sh utiliza el **Modo Simbólico** con un operador **Relativo** (+).

- **¿Qué hace?** Le dice al sistema: "Añade el permiso de ejecución (x) para el **usuario (u)**, el **grupo (g)** y **otros (o), manteniendo intactos todos los demás permisos existentes**".
- **La Inseguridad:** El resultado final de los permisos depende por completo de los permisos iniciales del archivo.
    - **Permisos Iniciales (Ejemplo Común):** rw-rw-rw- (666)
    - **Acción:** chmod +x añade x a todos:
    - **Permisos Finales:** rwxrwxrwx (777).
- **Consecuencia (Riesgo):** Esto resulta en un archivo que cualquier usuario en el sistema puede **leer, escribir y ejecutar**. Si un atacante (o un *malware*) gana acceso de bajo privilegio al sistema, puede modificar (inyectar código) en el *script* de despliegue y comprometer toda la cadena de DevOps.

2. La Solución: El Modo Octal Determinista

La práctica recomendada, especialmente en entornos de despliegue (DevOps), es usar el **Modo Octal**.

- **¿Qué hace?** Define el **estado final absoluto** de los permisos, ignorando por completo cómo el archivo comenzó. Esto garantiza un resultado **determinista** (predecible).
- **Estructura Octal:** Los permisos se representan con un número de tres dígitos. Cada dígito es la suma de los valores binarios para el grupo de permisos:

- r (Lectura) = 4
- w (Escritura) = 2
- x (Ejecución) = 1

Dígito	Usuario (Dueño)	Grupo	Otros
Ejemplo	7	5	5
Permisos	rwx (4+2+1)	rx (4+1)	rx (4+1)

- **El Estándar Seguro:**
  - chmod 755 deploy.sh
    - **Dueño (7):** Lee, Escribe y Ejecuta (rwx)
    - **Grupo (5):** Lee y Ejecuta (r-x)
    - **Otros (5):** Lee y Ejecuta (r-x)
  - Resultado: El dueño puede modificarlo, pero otros solo pueden ejecutarlo (no modificarlo).

Ejercicio Completo: Seguridad con chmod

Este ejercicio práctico está diseñado para que **experimentes la diferencia** entre el modo simbólico relativo y el modo octal determinista, y comprendas el riesgo de seguridad.

Objetivo del Ejercicio

Demostrar cómo el chmod +x puede llevar a permisos inseguros (777) y cómo el chmod 755 garantiza la seguridad sin importar el estado inicial.

Configuración

1. **Crear el archivo de prueba:**

```
touch script_de_prueba.sh
```

2. **Verificar los permisos iniciales (666 o similar):**

```
ls -l script_de_prueba.sh
```

Resultado (ejemplo común): -rw-rw-r-- 1 usuario grupo ... (664) o -rw-rw-rw- (666)

Escenario 1: El Modo Simbólico Peligroso (+x)

Paso 1: Simular el inicio inseguro (Permisos 666)

- *Instrucción:* Vamos a simular que nuestro umask o proceso inicial dejó el archivo con permisos que incluyen escritura para todos.

```
chmod 666 script_de_prueba.sh
ls -l script_de_prueba.sh
```

Resultado: -rw-rw-rw- (666)

Paso 2: Aplicar el modo simbólico relativo

- *Instrucción:* Ejecuta el comando "lógico" para darle ejecución.

```
chmod +x script_de_prueba.sh
```

Paso 3: Analizar el resultado (¡La trampa!)

- *Pregunta:* ¿Cuáles son los permisos resultantes?

```
ls -l script_de_prueba.sh
```

Resultado (Inseguro): -rwxrwxrwx (777)

- **Discusión:** En este punto, cualquiera puede escribir y ejecutar el *script*. Resalta el peligro de la **Relatividad** del +x.

Escenario 2: El Modo Octal Determinista (La Solución)

Paso 1: Reiniciar con cualquier estado inicial

- *Instrucción:* No importa cómo esté el archivo ahora (incluso si está en 777), queremos que termine en 755.

(El archivo sigue en 777 de la prueba anterior)

ls -l script_de_prueba.sh

Resultado: -rwxrwxrwx (777)

Paso 2: Aplicar el modo octal determinista

- *Instrucción:* Define el estado final deseado.

chmod 755 script_de_prueba.sh

Paso 3: Analizar el resultado (¡La seguridad!)

- *Pregunta:* ¿Cuáles son los permisos resultantes?

ls -l script_de_prueba.sh

Resultado (Seguro): -rwxr-xr-x (755)

- **Conclusión:** El chmod 755 **sobrescribió** los permisos anteriores y garantizó que solo el dueño pueda modificar (escribir) el archivo, mientras que otros solo pueden ejecutarlo. El resultado es **predecible** y **seguro**, ideal para la automatización en DevOps.