■ Mitigación de Fugas de Credenciales en Repositorios

1. Prevención (antes de subir)

• .gitignore – evita que archivos sensibles entren al repositorio.

Ejemplo:

```
# .gitignore
*.txt
*.env
config.json
```

• Uso de variables de entorno – en vez de guardar contraseñas en texto plano dentro del repo.

2. Detección (cuando ya está en el repo)

• Escaneo con Gitleaks:

```
.\gitleaks.exe detect --source . --no-git --report-format json --report-path report.json
```

• Reglas personalizadas en Gitleaks:

```
[[rules]]
description = "Secretos de prueba"
regex = '''PRUEBA_SECRETA_[0-9]+'''
tags = ["prueba", "credencial"]
```

3. Remediación (cuando ya se subió)

• Eliminar el archivo de la rama activa:

```
git rm prueba.txt
echo prueba.txt >> .gitignore
git add .gitignore
git commit -m "remediación: elimino prueba.txt y agrego a .gitignore"
```

• Eliminar la credencial del historial con BFG:

```
git clone --mirror repo-lab repo-lab-mirror.git
cd repo-lab-mirror.git
java -jar C:\tools\bfg.jar --replace-text ..\passwords-to-remove.txt
git reflog expire --expire=now --all
git gc --prune=now --aggressive
git push --force
```

4. Respuesta final

• Cambiar la contraseña expuesta (aunque sea de prueba).

• Notificar al equipo de seguridad.

• Documentar el incidente en mitigation.md con controles aplicados.

■ Con este flujo: Prevención → Detección → Remediación → Respuesta, se reduce al máximo el impacto de fugas de credenciales.