Título
Informe ejecutivo de vulnerabilidad: Inyección SQL (SQLi) en DVWA

Introducción (por qué importa)
La inyección SQL es una vulnerabilidad grave: permite que entradas de usuario no controladas modifiquen consultas a la base de datos. En este caso de laboratorio, se demostró que al introducir un texto en el campo "User ID" la aplicación devolvió múltiples usuarios. Si esto ocurriera en producción, podría exponer contraseñas, datos personales y permitir ataques mayores contra la organización.

Descripción del incidente (qué pasó)
- **Dónde:** Módulo *SQL Injection* de la aplicación DVWA (entorno de pruebas, 127.0.0.1).
- **Qué se hizo:** Se envió el valor 1' OR '1'='1 en el parámetro id.
- **Qué ocurrió:** La aplicación devolvió múltiples registros (ej. admin/admin, Gordon Brown, Pablo Picasso, etc.), lo que confirma que la entrada se ejecutó dentro de una consulta SQL sin validación.
- **Conclusión:** La entrada de usuario no se filtró ni parametrizó; la consulta fue manipulada para devolver toda la tabla de usuarios.

Proceso de reproducción (pasos, explicado simple)
1. Abrir la página vulnerable de DVWA.
2. Ir al formulario "SQL Injection".
3. Introducir en *User ID*: 1' OR '1'='1.
4. Pulsar *Submit*.
5. La página muestra múltiples cuentas extraídas de la base de datos.

Impacto del incidente (qué significa para la organización)
- **Confidencialidad:** Alto — datos de usuarios y contraseñas pueden ser expuestos.
- **Integridad:** Medio-Alto — un atacante podría modificar o borrar datos.
- **Disponibilidad:** Medio — posible degradación o interrupción por manipulación de consultas.
- **Reputación y cumplimiento:** Alto — leak de datos personales puede conllevar sanciones y pérdida de confianza.
- **Escenario real:** En un entorno productivo la explotación permitiría acceso no autorizado a la base de datos y posibilitaría movimientos posteriores del atacante dentro de la infraestructura.

Recomendaciones (clara, priorizada, accionable)
Sigue estas acciones en orden (1 = mayor impacto / acción urgente):
1. **Bloqueo inmediato (contención — corto plazo, ≤24 h)**
    - Poner el módulo afectado fuera de servicio o activar un control que bloquee entradas malformadas hasta aplicar correcciones.
2. **Corregir el código (remediación — corto plazo, ≤1 semana)**

- Usar *consultas parametrizadas* (prepared statements) para cualquiera entrada que llegue a la base de datos.
- Evitar concatenar texto del usuario en sentencias SQL.

3. **Principio de menor privilegio (configuración — corto plazo)**
   - Cambiar el usuario DB usado por la aplicación para que tenga solo permisos mínimos (lectura/escritura limitados, no admin).
4. **Validación y saneamiento en la aplicación (mejora de calidad — medio plazo)**
   - Implementar validación por *whitelist* (aceptar solo formatos esperados) y sanitizar entradas.
5. **Detectar e impedir (defensa adicional — medio plazo)**
   - Activar/ajustar un WAF o reglas en firewall de aplicaciones para bloquear patrones de inyección comunes.
   - Configurar alertas en logs/siem para consultas anómalas.
6. **Pruebas y verificación (aseguramiento — medio plazo)**
   - Ejecutar pruebas de seguridad (scans y pruebas manuales) tras las correcciones; incluir pruebas de regresión.
   - Automatizar pruebas de inyección en pipeline CI/CD.
7. **Formación y gobernanza (preventivo — largo plazo)**
   - Capacitar a desarrolladores en prácticas de codificación segura y OWASP Top 10.
   - Actualizar políticas de desarrollo seguro y revisiones de código obligatorias.

Conclusión (mensaje ejecutivo final)
La evidencia muestra una falla típica y crítica: la aplicación acepta datos de usuario que alteran consultas SQL. En entornos reales esa debilidad puede resultar en exposición masiva de información y control sobre la base de datos. Recomendamos **acciones inmediatas** para contener el módulo, corregir el código usando consultas parametrizadas y aplicar controles de defensa (WAF, mínimos privilegios). Adoptando las recomendaciones anteriores reducimos rápidamente el riesgo y evitamos impactos mayores.