OWASP top 10

Buscar 2 soluciones para cada vulnerabilidad

1. Tips para evitar SQL Injection:

- `Usar sentencias preparadas (con consultas parametrizadas)`: El uso de sentencias preparadas con variables vinculadas (también conocidas como consultas parametrizadas) es la forma en que todos los programadores deberían escribir sus consultas SQL. Parametrizar las consultas fuerza al desarrollador a primero definir todo el código SQL, y luego pasar cada parámetro a la consulta luego.

Hacer esto asegura que un atacante no pueda modificar la intención de una consulta, incluso si intenta introducir una sentencia SQL maliciosa.

Ejemplo inseguro:

```
String query = "SELECT account_balance FROM user_data WHERE user_name = " +
request.getParameter("customerName");
try {
    Statement stmt = conn.createStatement( ... );
    ResultSet rs = stmt.executeQuery(query);
}
```

Este ejemplo es inseguro porque el atacante puede modificar el valor de la variable `customerName` para que el programa intente ejecutar una consulta SQL maliciosa.

Ejemplo Java EE - PreparedStatement( ):

```
// Este parámetro debería ser validado antes de ser usado
String custname = request.getParameter("customerName");
// Consulta SQL
String query = "SELECT account_balance FROM user_data WHERE user_name = ?";
// Preparar la consulta
PreparedStatement pstmt = connection.prepareStatement(query);
pstmt.setString(1, custname);
ResultSet results = pstmt.executeQuery();
```

La consulta SQL se define en una variable, y se pasa como parámetro luego. Este ejemplo es seguro porque al ser definida en una variable, el atacante no puede modificar la intención de la consulta.

- `Procedimientos almacenados`: Los procedimientos almacenados tienen el mismo efecto que las consultas parametrizadas. La diferencia es que los procedimientos almacenados son definidos y almacenados en la base de datos, y son llamados desde la aplicación.

```
// Este parámetro debería ser validado antes de ser usado
String custname = request.getParameter("customerName");
try {
    // Consulta SQL
    CallableStatement cstmt = connection.prepareCall("{call
sp_getAccountBalance(?)}");
    cstmt.setString(1, custname);
    ResultSet results = cstmt.executeQuery();
} catch (SQLException e) {
    // error handling
}
```

2. Tips para Broken Authentication:

- Donde sea posible, implementar autenticación multi-factor para prevenir ataques automatizados, fuerza bruta, credenciales robadas, etc.
- Implementar un chequeo de contraseñas débiles, comprobar la seguridad de la contraseña en base al top 1000 peores contraseñas.

3. Tips para Sensitive Data Exposure:

- Clasificar la data procesada, almacenada o transmitida por la aplicación. Identificar cual data es sensible de acuerdo con las leyes de privacidad, requisitos reguladores, o necesidades del negocio.
- Asegurarse de encriptar la data sensible.

4. Tips para XML External Entities (XXE):

- Cuando sea posible, usar formatos menos complejos como JSON, y evitar serializar data sensible.
- Parchear o actualizar todos los procesadores XML y librerías que usa la aplicación o el sistema operativo. Usar *dependency checkers*. Actualizar SOAP a SOAP 1.2 o superior.

5. Tips para Broken Access Control:

- Con la excepción de recursos públicos, denegar por defecto todo acceso a los usuarios que no están autorizados.
- Implementar mecanismos de control de acceso una vez y luego reusarlos a través de la aplicación.

6. Tips para Security Misconfigurations:

- Tener una plataforma mínima sin ninguna funcionalidad innecesaria. Remover o desintalar cualquier componente innecesario.
- Los entornos de desarrollo, QA y producción deben tener la misma configuración, con las respectivas diferentes credenciales usadas en cada ambiente.

7. Tips para Cross-Site Scripting (XSS):

- Usar frameworks los cuales tengan protección contra XSS por defecto, como Ruby on Rails, React JS, etc.
- Descartar peticiones HTTP no confiables en el contexto del HTML output.

8. Tips para Insecure deserialization:

- Implementar controles en las entradas de datos, como firmas digitales o cualquier objeto serializado para prevenir la creación de objetos hostiles o la manipulación de datos.
- Monitoreo de las deserializaciones de datos, alertar si un usuario intenta hacerlo constantemente.

9. Tips para Using Components with Known Vulnerabilities:

- Remover dependencias sin usar, funcionalidades innecesarias, componentes, archivos y documentación innecesaria.
- Obtener componentes solo de fuentes oficiales mediante links seguros. Preferir paquetes firmados para reducir el chanse de incluir un componente malicioso modificado.

10. Tips para Insufficient Logging and Monitoring:

- Asegurar todos los loggins, las fallas de control de acceso, y verificar toda data procesada por el servidor.
- Asegurar que todos los logs sean registrados en un formato el cual pueda ser consultado fácilmente.