El control de acceso es un mecanismo fundamental de seguridad en aplicaciones que asegura que solo usuarios autorizados pueden realizar acciones específicas o acceder a datos sensibles. El control de acceso roto (Broken Access Control) ocurre cuando una aplicación no restringe adecuadamente el acceso a funciones y recursos. Este es uno de los principales problemas de seguridad web y se encuentra en el top 10 de vulnerabilidades de OWASP.

**Tipos de Vulnerabilidades de Broken Access Control**

1. **Acceso Horizontal Roto**:
   * Ocurre cuando un usuario puede acceder a los recursos de otros usuarios con el mismo nivel de permiso.
   * Ejemplo: Un usuario puede acceder a las cuentas de otros usuarios modificando el ID en la URL.
2. **Acceso Vertical Roto**:
   * Sucede cuando un usuario con permisos limitados puede acceder a recursos o funciones destinadas a usuarios con mayores privilegios.
   * Ejemplo: Un usuario regular puede acceder a las funciones administrativas.
3. **Acceso a Datos Sin Autorización**:
   * Cuando los datos sensibles no están correctamente protegidos y cualquier usuario puede acceder a ellos.
   * Ejemplo: Archivos de configuración, datos de usuarios o informes.
4. **Ejecución de Funciones No Autorizadas**:
   * Ocurre cuando un usuario puede ejecutar funciones restringidas.
   * Ejemplo: Un usuario no administrador puede realizar acciones administrativas como eliminar registros.

**Ejemplos de Ataques de Broken Access Control**

1. **Manipulación de URLs**:
   * Un atacante modifica la URL para acceder a recursos no autorizados.
   * Ejemplo:

ruby

Copy code

https://example.com/user/profile?userId=123

Cambiado a

https://example.com/user/profile?userId=124

1. **Forzado de Navegación**:
   * Acceso a páginas restringidas mediante la manipulación directa de URLs.
   * Ejemplo:

arduino

Copy code

https://example.com/admin/dashboard

1. **Modificación de Tokens de Sesión**:
   * Alteración de tokens de sesión para suplantar la identidad de otros usuarios.
2. **Manipulación de Parámetros en Formularios**:
   * Cambiar parámetros en formularios para acceder a funciones restringidas.
   * Ejemplo: Cambiar un valor oculto en un formulario para elevar privilegios.

**Pruebas para Detectar Broken Access Control**

1. **Revisión Manual de Código**:
   * Inspeccionar el código fuente para asegurar que todas las rutas y funciones estén adecuadamente protegidas.
2. **Pruebas Automatizadas**:
   * Usar herramientas de escaneo de seguridad web como Burp Suite, OWASP ZAP para detectar vulnerabilidades de acceso.
3. **Fuzzing de URLs**:
   * Probar diferentes variaciones de URLs para ver si se pueden acceder a recursos no autorizados.
4. **Pruebas de Roles**:
   * Verificar que cada tipo de usuario solo tenga acceso a los recursos y funciones que les corresponden según sus permisos.

**Medidas de Mitigación**

1. **Implementación Correcta de Control de Acceso**:
   * Asegurarse de que todas las funciones y recursos están protegidos por controles de acceso adecuados.
2. **Principio de Privilegios Mínimos**:
   * Otorgar a los usuarios el mínimo nivel de acceso necesario para realizar sus tareas.
3. **Validación del Lado del Servidor**:
   * Realizar todas las validaciones de control de acceso en el lado del servidor, no en el lado del cliente.
4. **Uso de Frameworks Seguros**:
   * Utilizar frameworks que proporcionen mecanismos de control de acceso robustos y bien probados.
5. **Registro y Monitoreo**:
   * Implementar un sistema de registro y monitoreo para detectar y responder a intentos de acceso no autorizado.
6. **Revisión Regular de Permisos**:
   * Revisar periódicamente los permisos de los usuarios para asegurar que no tengan accesos innecesarios.

**Ejemplo de Implementación de Control de Acceso en Código**

**PHP con Middleware**

php

Copy code

// Middleware para verificar acceso

function checkAccess($requiredRole) {

if (!isset($\_SESSION['role']) || $\_SESSION['role'] !== $requiredRole) {

header('Location: /login.php');

exit();

}

}

// En una ruta específica

checkAccess('admin'); // Solo usuarios con rol 'admin' pueden acceder

**Python con Flask**

python

Copy code

from flask import Flask, request, redirect, url\_for, session

app = Flask(\_\_name\_\_)

def check\_access(role):

def wrapper(f):

def decorated\_function(\*args, \*\*kwargs):

if 'role' not in session or session['role'] != role:

return redirect(url\_for('login'))

return f(\*args, \*\*kwargs)

return decorated\_function

return wrapper

@app.route('/admin')

@check\_access('admin')

def admin\_dashboard():

return "Admin Dashboard"

if \_\_name\_\_ == '\_\_main\_\_':

app.run()

Implementar controles de acceso robustos y realizar pruebas de seguridad regulares son pasos fundamentales para proteger las aplicaciones contra el Broken Access Control y otras vulnerabilidades.

Try Hack Me > OWASP TOP 10 > Broken Access Control (IDOR Challenge)?

ChatGPT

El desafío IDOR (Insecure Direct Object References) en la plataforma TryHackMe se encuentra dentro del módulo de OWASP Top 10, específicamente en el tema de "Broken Access Control". Aquí te proporcionaré una guía para abordar y comprender mejor el desafío de IDOR.

**Insecure Direct Object References (IDOR)**

IDOR es una vulnerabilidad de control de acceso roto en la que un atacante puede acceder directamente a objetos (como archivos, bases de datos o URL) sin la autorización adecuada. Esta vulnerabilidad ocurre cuando la aplicación utiliza identificadores directos (como IDs) en la URL o en los parámetros de solicitud, permitiendo a un atacante manipular esos valores para acceder a recursos no autorizados.

**Guía para el Desafío de IDOR en TryHackMe**

1. **Accede al Entorno de TryHackMe**:
   * Inicia sesión en tu cuenta de TryHackMe.
   * Dirígete al módulo de OWASP Top 10.
   * Selecciona el desafío de Broken Access Control, específicamente el IDOR Challenge.
2. **Comprende el Escenario**:
   * Lee la descripción del desafío para entender el contexto.
   * Normalmente, se te proporciona una aplicación web vulnerable que maneja recursos identificados por un ID en la URL.
3. **Identifica la Vulnerabilidad**:
   * Navega por la aplicación y observa cómo se gestionan los recursos.
   * Por ejemplo, puede que veas una URL como http://example.com/resource?id=123.
4. **Prueba la Manipulación de Parámetros**:
   * Cambia el valor del parámetro id en la URL para ver si puedes acceder a otros recursos.
   * Ejemplo: Cambia id=123 por id=124 y observa si obtienes acceso a recursos que no deberías.
5. **Valida la Explotación**:
   * Si al manipular el id puedes acceder a recursos de otros usuarios, has encontrado una vulnerabilidad IDOR.
   * Asegúrate de documentar los pasos y los resultados obtenidos.
6. **Ejemplo Práctico**:

markdown

Copy code

1. Accede a la URL inicial proporcionada en el desafío.

2. Observa una URL como `http://example.com/resource?id=1`.

3. Cambia el `id` de `1` a `2` (`http://example.com/resource?id=2`).

4. Si obtienes acceso a un recurso diferente que no te pertenece, has identificado un IDOR.

**Medidas de Mitigación**

1. **Validación de Autorización**:
   * Implementa controles de acceso en el servidor para validar que el usuario tiene permisos para acceder al recurso solicitado.
2. **Uso de Identificadores Indirectos**:
   * En lugar de utilizar IDs directos, usa identificadores indirectos o tokens que no puedan ser fácilmente adivinados o manipulados.
3. **Registro y Monitoreo**:
   * Implementa registro y monitoreo para detectar accesos inusuales o intentos de manipulación de parámetros.
4. **Principio de Privilegios Mínimos**:
   * Asegúrate de que los usuarios solo tienen acceso a los recursos que necesitan para sus tareas específicas.

**Ejemplo de Código Seguro**

**PHP**

php

Copy code

// Supongamos que tienes una función para obtener el recurso por ID

function getResourceById($id, $userId) {

// Consulta para obtener el recurso donde el ID y el UserId coinciden

$stmt = $pdo->prepare('SELECT \* FROM resources WHERE id = ? AND user\_id = ?');

$stmt->execute([$id, $userId]);

return $stmt->fetch();

}

// Validación de autorización

session\_start();

$userId = $\_SESSION['user\_id'];

$resourceId = $\_GET['id'];

$resource = getResourceById($resourceId, $userId);

if ($resource) {

// Mostrar el recurso

} else {

// Acceso denegado

header('HTTP/1.0 403 Forbidden');

echo 'No tienes permisos para acceder a este recurso.';

exit;

}

**Python con Flask**

python

Copy code

from flask import Flask, request, session, abort

app = Flask(\_\_name\_\_)

@app.route('/resource')

def get\_resource():

if 'user\_id' not in session:

abort(403) # No autenticado

user\_id = session['user\_id']

resource\_id = request.args.get('id')

# Consulta para obtener el recurso

resource = query\_db('SELECT \* FROM resources WHERE id = ? AND user\_id = ?', [resource\_id, user\_id])

if resource:

return render\_template('resource.html', resource=resource)

else:

abort(403) # No autorizado

if \_\_name\_\_ == '\_\_main\_\_':

app.run()

Con esta información y estos ejemplos, deberías estar bien preparado para enfrentar el desafío de IDOR en TryHackMe y comprender mejor cómo identificar y mitigar esta vulnerabilidad. ¡Buena suerte!