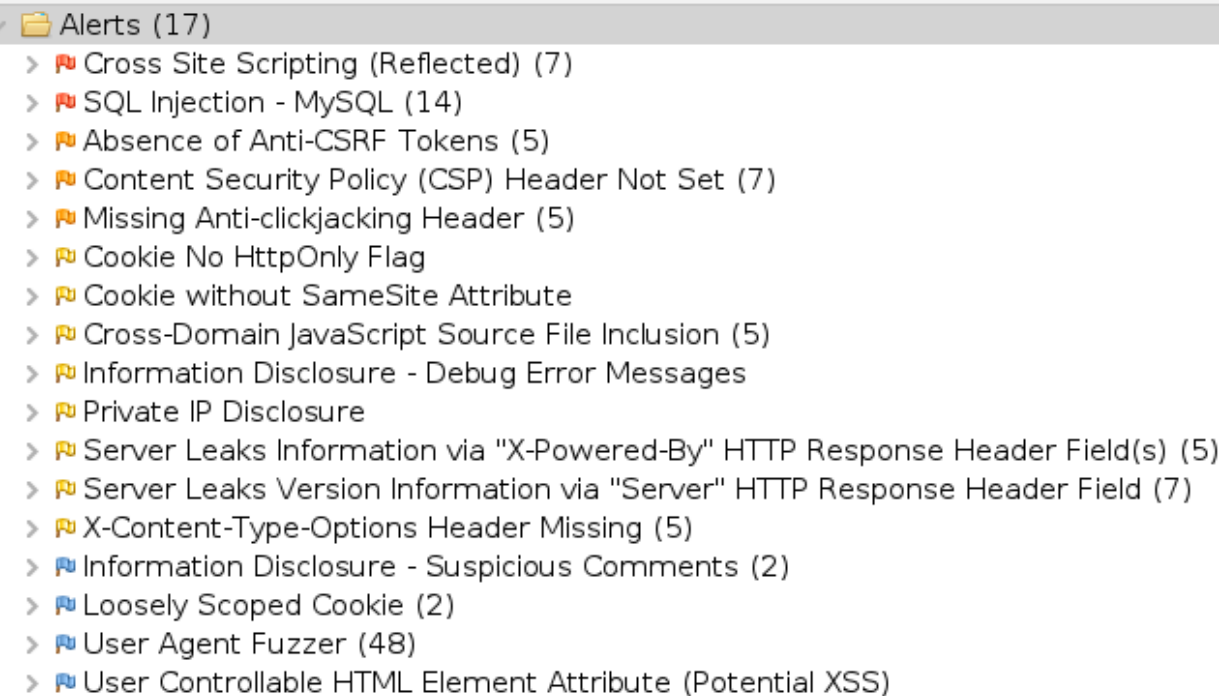


Sistemas de Gestión de Seguridad de Sistemas de Información


















Daniela Waldeck
Pedro Inciarte
20/11/2022

Link a repositorio en GitHub:
https://github.com/danielaWaldeck95/entrega_seguridad.git

Resultado del primer análisis con OWASP ZAP



The screenshot shows the 'Alerts' section of the OWASP ZAP interface, displaying a list of 17 alerts. The alerts are categorized by severity: High (red), Medium (orange), and Low (blue). The list includes various security issues such as Cross Site Scripting, SQL Injection, and missing security headers.

- >  Cross Site Scripting (Reflected) (7)
- >  SQL Injection - MySQL (14)
- >  Absence of Anti-CSRF Tokens (5)
- >  Content Security Policy (CSP) Header Not Set (7)
- >  Missing Anti-clickjacking Header (5)
- >  Cookie No HttpOnly Flag
- >  Cookie without SameSite Attribute
- >  Cross-Domain JavaScript Source File Inclusion (5)
- >  Information Disclosure - Debug Error Messages
- >  Private IP Disclosure
- >  Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s) (5)
- >  Server Leaks Version Information via "Server" HTTP Response Header Field (7)
- >  X-Content-Type-Options Header Missing (5)
- >  Information Disclosure - Suspicious Comments (2)
- >  Loosely Scoped Cookie (2)
- >  User Agent Fuzzer (48)
- >  User Controllable HTML Element Attribute (Potential XSS)

Vulnerabilidades a solucionar

Rotura de control de acceso

Modificando el parámetro id en el url de la vista create-or-update-product es posible acceder y modificar prendas de otros usuarios.

`http://localhost:81/views/create-or-update-product.php?id=3`

No se valida que el usuario logueado sea el poseedor de esa prenda para visualizarla y editarla.

Solución: para evitar que un usuario visualice los datos de una prenda que no le pertenece agregamos la siguiente condición:

```
if ($product['user_id'] != $user_id) {  
    header("Location: /views/access-denied.php");  
}
```

Si el user_id del producto no coincide con el id del usuario logueado el atacante es redirigido a una pestaña de acceso denegado.

Para evitar que un usuario edite los datos de una prenda que no le pertenece antes de editar se debe verificar que el id del usuario de la sesión coincida con el `user_id` del producto. Antes de aplicar el `UPDATE` nos aseguramos que el `user_id` del producto que se desea modificar coincida con el `user_id` de la sesión.

```
$sql = "UPDATE products
SET name = '$name', brand = '$brand', size = '$size', color = '$color',
category_id = $category_id
WHERE id = $id
";

if ($product['user_id'] == $user_id && mysqli_query($conn, $sql))
```

*Estos cambios fueron implementados antes de resolver SQL Injection, puede ser que el código final tenga algunas modificaciones respecto a lo que se muestra aquí.

Fallos criptográficos

Problemas detectados:

- Datos que se transmiten como texto plano
- Almacenaje de contraseña sin hash ni sal

Solución:

Para solucionar esto utilizamos la siguiente función de PHP: `password_hash()`. Esta función crea un nuevo hash de contraseña usando un fuerte algoritmo de hash unidireccional. Nosotros utilizamos `PASSWORD_BCRYPT`. Crea un hash de 60 dígitos. `password_hash()` generará sal aleatoria para cada contraseña con hash. En la documentación oficial de PHP establecen lo siguiente: "Se recomienda fuertemente que no genere su propia sal para esta función. Creará una sal segura automáticamente para ti.". Para poder verificar la contraseña para que el usuario pueda realizar el log in utilizamos la función `password_verify()`. En ninguna parte del código aparece la contraseña real del usuario.

Inyección


Los datos que provee el usuario no se validan, filtran, o limpian. El intérprete usa directamente consultas dinámicas o llamadas no parametrizadas sin "escaping". Los datos hostiles introducidos por el usuario son usados directamente.

SQL Injection - MySQL

ZAP para la fecha de nacimiento ingreso lo siguiente y pudo conseguir todos los datos de nuestras tablas.

```
full_name=ZAP&birth_date=2022-11-20%27%29+UNION+ALL+select+NULL+--+&dni=ZAP&phone=9999999999&email=foo-bar%40example.com&
user_name=ZAP&password=ZAP
```

SQL Injection - MySQL

URL: http://localhost:81/views/register.php
Risk:  High
Confidence: Medium
Parameter: birth_date
Attack: 2022-11-20') UNION ALL select NULL --
Evidence: The used SELECT statements have a different number of columns
CWE ID: 89
WASC ID: 19
Source: Active (40018 - SQL Injection)
Input Vector: Form Query

Solución: Consultas parametrizadas

Preparamos las consultas y utilizamos `bind_param`

```
$stmt = mysqli_stmt_init($conn);  
mysqli_stmt_prepare($stmt, "INSERT INTO users (full_name, dni, birth_date, phone, email, password, user_name) VALUES (?, ?, ?, ?, ?, ?, ?)");  
mysqli_stmt_bind_param($stmt, "sssisss", $full_name, $dni, $birth_date, $phone, $email, $password, $user_name);  
$full_name = $_POST['full_name'];  
$user_name = $_POST['user_name'];  
$email = $_POST['email'];  
$phone = $_POST['phone'];  
$birth_date = $_POST['birth_date'];  
$password = $_POST['password'];  
$dni = $_POST['dni'];  
  
$result = mysqli_stmt_execute($stmt);  
  
if ($result) {  
    header("Location: /views/login.php");  
} else {  
    $_SESSION["Register.Error"] = 'Hubo un inconveniente al guardar sus cambios, inténtalo nuevamente';  
}  
mysqli_stmt_close($stmt);  
mysqli_close($conn);
```

Al utilizar `bind_params` no es necesario escapar los valores.

Repetimos esto en todas las sql query ver [commit](#).

Diseño inseguro

La gestión de los errores desvela información, los errores son demasiado informativos.

Nuestra aplicación muestra al usuario los errores tal cual son capturados.

```
$_SESSION["UpdateUser.Error"] = mysqli_error($conn);
```

Ejemplo de mensaje recibido por el usuario donde se detalla información sobre tablas y restricciones.

```
Cannot add or update a child row: a foreign key constraint fails
(`database`.`products`, CONSTRAINT `products_ibfk_2` FOREIGN KEY
(`category_id`) REFERENCES `categories` (`id`))
```

Solución: En lugar de devolver como error `mysqli_error` devolveremos un string: ‘Hubo un inconveniente al procesar sus cambios, inténtalo nuevamente’.

`mysqli_error` solo debería ser utilizado únicamente para debugging

Configuración de seguridad insuficiente

En relación a la configuración de seguridad insuficiente detectamos que la clave de acceso a phpMyAdmin seguía siendo user: test password: admin.

Solución: Modificar el usuario y contraseña de nuestros servicios db y phpmyadmin. Escribir las contraseñas en un archivo `.env` local y referenciar en el archivo `docker-compose.yml`. El archivo `.env` existirá únicamente a nivel local de modo que las contraseñas no queden expuestas en el código.

`docker-compose.yml`

```
MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
MYSQL_USER: ${MYSQL_USER}
MYSQL_PASSWORD: ${MYSQL_PASSWORD}
```

`.env`

```
MYSQL_USER = new_user
MYSQL_PASSWORD = new_password
MYSQL_ROOT_PASSWORD = new_root
```

Por alguna razón lo anterior no anduvo. Se probó de muchas maneras y no pudimos encontrar la manera de que funcione. Si bien en el `.yml` original decía `MYSQL_USER = admin` y `MYSQL_PASSWORD = test`, en realidad no lo estaba seteando realmente. La gestión de los errores desvela información / los errores son demasiado informativos. Esto fue solucionado en diseño inseguro pero aplica a ambas clasificaciones de vulnerabilidades.

El servidor no responde con cabeceras HTTP de seguridad

Intentamos utilizar ssl para conectar php con phpmyadmin, pero no pudimos solucionarlo. Intentamos varias maneras para poder resolver esto, pero ninguna nos sirvió. Por ejemplo una fue creando un certificado autofirmado y luego forzar a phpmyadmin que utilice un https, pero no lo logramos.

Servidor de base de datos

- Servidor: db via TCP/IP
- Tipo de servidor: MariaDB
- Conexión del servidor: No se está utilizando SSL ⓘ
- Versión del servidor: 10.8.2-MariaDB-1:10.8.2+maria~focal - mariadb.org binary distribution
- Versión del protocolo: 10
- Usuario: admin@172.29.0.3
- Conjunto de caracteres del servidor: UTF-8 Unicode (utf8mb4)

Componentes vulnerables y obsoletos

Nuestra app podría considerarse insegura en relación a este punto ya que no se saben exactamente todas las versiones de los componentes utilizados.

En el caso de phpMyAdmin estamos usando utilizando siempre la última versión disponible del contenedor:

```
image: phpmyadmin/phpmyadmin:latest
```

Solución:

```
image: phpmyadmin
```

Fallos de identificación y autenticación

Atacamos Credential Stuffing utilizando un buen hash y sal para almacenar las contraseñas. De esta forma, si un atacante accede a datos de usuarios de otros servicios y desea probar si la persona utiliza su mismo nombre de usuario y contraseña en nuestra aplicación, el sal haría que dicha contraseña no coincida con la almacenada por nuestro sistema. Ver en la solución de fallos criptográficos la implementación de esta solución.

Implementamos una política de tamaño de contraseñas, complejidad y rotación. Los usuarios no tienen permitido definir contraseñas débiles y se les solicita una nueva contraseña luego de pasado determinado tiempo.

Solución:

Rotación de contraseñas

Se le pedirá al usuario cambiar la contraseña cada un mes. Una vez que el usuario inicie sesión, si la última vez que modificó la contraseña fue hace más de un mes se le redireccionará a una ventana en la que obligatoriamente tendrá que cambiar la contraseña para poder acceder a su armario. Si el usuario desea cambiar el url para acceder a otra ventana será redireccionado a la ventana que le solicita el cambio de contraseña.

Se le pide ingresar su contraseña actual, su nueva contraseña y la confirmación de la misma. La contraseña nueva no puede ser igual a la anterior. La contraseña debe ser lo

suficientemente segura. Para poder probar esto dejamos un usuario creado cuyo nombre de usuario es daniwal y la contraseña es Sgssi12345678-. (Sabemos que está mal dejar un usuario precargado, pero es la única forma de que lo puedas probar, sino habría que esperar un mes o modificar código).



Resetear Contraseña

Contraseña actual

Ingrese su contraseña actual

Nueva contraseña

Ingrese su nueva contraseña

Repetir Contraseña

Vuelva a escribir su contraseña

Guardar

Validación de la seguridad de la contraseña

Consideramos 3 niveles de seguridad bajo, medio y alto en base a los siguientes requerimientos:

- al menos una minúscula
- al menos una mayúscula
- al menos un carácter especial
- al menos un dígito
- al menos 8 caracteres

Alto: si la contraseña cumple todos los requerimientos.

Medio: si la contraseña tiene al menos seis caracteres y cumple con todos los otros requerimientos, o no tiene un dígito pero cumple el resto de los requerimientos.

Baja: si no cumple los requerimientos para media o alta.

Validamos los requerimientos con expresiones regex.

```
function checkPasswordStrength() {
  let password = document.getElementById("password").value;
  let strengthBadge = document.getElementById('StrengthDispValue');
  document.getElementById('StrengthDisp').style.display = 'block';
  let strongPassword = new
RegExp('(?=.*[a-z])(?=.*[A-Z])(?=.*[0-9])(?=.*[^\A-Za-z0-9])(?=.{8,})')
  let mediumPassword = new
RegExp('((?=.*[a-z])(?=.*[A-Z])(?=.*[0-9])(?=.*[^\A-Za-z0-9])(?=.{6,}))|((?=.*[a-z])
(?=.*[A-Z])(?=.*[^\A-Za-z0-9])(?=.{8,}))')
}
```

```

if(strongPassword.test(password)) {
    strengthBadge.style.color = "green";
    strengthBadge.textContent = 'Alta';
    weakPassword = false;
} else if(mediumPassword.test(password)) {
    strengthBadge.style.color = 'yellow';
    strengthBadge.textContent = 'Media';
    weakPassword = false;
} else {
    strengthBadge.style.color = 'red';
    strengthBadge.textContent = 'Baja';
    weakPassword = true;
}
}

```

Si la seguridad es baja no permitimos completar el registro.

Fallos en la integridad de datos y software

Los fallos en la integridad de datos y software se dan cuando hay código e infraestructura que no protege contra violaciones de su integridad. En el caso de nuestra aplicación, se utiliza TailwindCss a través de un CDN (Content Delivery Network). Al analizar con ZAP esta vulnerabilidad se ve reflejada como **Cross-Domain JavaScript Source File Inclusion**.

Cross-Domain JavaScript Source File Inclusion	
URL:	http://localhost:81
Risk:	Low
Confidence:	Medium
Parameter:	https://cdn.tailwindcss.com
Attack:	
Evidence:	<script src="https://cdn.tailwindcss.com"></script>
CWE ID:	829
WASC ID:	15
Source:	Passive (10017 - Cross-Domain JavaScript Source File Inclusion)
Description:	The page includes one or more script files from a third-party domain.

Solución: instalar una dependencia de tailwind en el proyecto en lugar de utilizar el CND. Para esto fue necesario instalar npm en nuestro ordenador y seguir la guía de instalación que provee Tailwindcss en su documentación oficial: <https://tailwindcss.com/docs/installation>

Fallos en la monitorización de la seguridad

Los login fallidos no son registrados en un log. Los errores y advertencias generan mensajes inadecuados en los logs (O no los generan)

Solución

En el config.php agregamos lo siguiente:

```
ini_set('display_errors', 1);
```



```
error_reporting(E_ERROR | E_WARNING | E_PARSE);
error_reporting(E_ALL & ~E_NOTICE);
ini_set('error_log', 'php_error.log');
```

Agregamos lo siguiente en login.php:

```
error_log($error,0);
```

Los fallos de login son registrados en un log llamado php-error.log. En este registro guardamos la fecha, hora, IP, usuario al que se quiso ingresar y la contraseña ingresada.

Resultado del segundo análisis con OWASP ZAP

- ▼ Alerts (8)
 - > Content Security Policy (CSP) Header Not Set (3)
 - > Missing Anti-clickjacking Header
 - > Information Disclosure - Debug Error Messages
 - > Private IP Disclosure
 - > Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)
 - > Server Leaks Version Information via "Server" HTTP Response Header Field (3)
 - > X-Content-Type-Options Header Missing
 - > User Agent Fuzzer (12)