



TEMA 4.4

Desinfección y gestión
de errores

Inyección de HTML y de SQL

- Ya vimos que un usuario puede inyectar HTML a través de los campos de un formulario, y que podíamos prevenirlo utilizando la función `htmlspecialchars` de PHP
- Un usuario avanzado, también podría inyectar código SQL. En este caso, es posible atacar directamente a la base de datos.
- Es posible, con PDO, ejecutar directamente SQL sin una preparación previa de la sentencia. **Esto funciona**, y no es raro encontrarse este tipo de código. Veamos un ejemplo:

Código perezoso

```
if (isset($_POST["email"]) && isset($_POST["password"])) {  
    $email = $_POST["email"];  
    $password = $_POST["password"];  
    // ¡¡ NI SE OS OCURRA HACER ESTO !! X NO X NO X X X X  
    $sql = "SELECT nombre FROM usuarios"  
        . "WHERE email='$email'"  
        . "AND password='$password'";  
    $resultado = $pdo->query($sql);  
}
```



En este ejemplo, concatenamos el valor de las variables directamente a `$sql`:

1. Nos ahorramos el uso de placeholders
2. Nos ahorramos la sentencia prepare

Ejercicio 1:

Inyección de SQL

Prueba de login

Email :

Password:

1. Haz un programa que pida email y password (ambos campos tipo texto)
2. A efectos didácticos no vamos a encriptar la password en este ejercicio. Utilizarás tu tabla de usuarios de MySQL.
3. Este programa, comprobará que existe en la tabla de usuarios el email y la password, y si es así mostrará el mensaje de bienvenida.
4. Si no existe en la base de datos un usuario con esa password, mostrará el mensaje "Usuario o password incorrectos".
5. Pruébalo con datos correctos e incorrectos
6. Vas a inyectar SQL: Pondrás como password del usuario la cadena **"pirata' OR '1'=='1'"**

PISTA: Puedes utilizar el método rowCount para averiguar cuantas filas se han recuperado.

Qué puede pasar

- Que entren en nuestro sistema no es el único peligro.
- Un usuario malicioso podría cargarse nuestra información
- ¡¡ Cuidado con probar el chiste con tu programa del Ejercicio 1 !!



- Hola, le llamo del colegio de su hijo. Tenemos un problema con los ordenadores.
- Madre mía, ¿ ha roto algo ?
- Algo así. ¿ Es cierto que su hijo se llama **Roberto')**; **DROP TABLE Estudiantes;** ?
- Ah. Sí, le llamamos Robertito tablas.
- Bueno, pues hemos perdido los registros de todo el año de los estudiantes. Espero que esté contenta.
- Y yo espero que hayan aprendido a desinfectar las entradas a su Base de Datos.

Solución a la inyección de SQL: método prepare

- PHP proporciona, mediante el método **prepare** de **PDO**, una prevención de la inyección de sql.
- Cuando concatenemos nuestras sentencias SQL, utilizaremos **placeholders**, que son unos parámetros que PDO sustituirá por un valor que informaremos posteriormente.
- Todos los valores que provengan de entrada de usuario deberán ir siempre como parámetros en nuestras sentencias SQL
- El método **prepare** se encargará de evitar la inyección de SQL
- El método **execute** nos permitirá sustituir los parámetros por el valor que nos proporcionó el usuario
- Una ventaja adicional de utilizar prepare es que optimiza el acceso, sobre todo cuando tenemos muchas acciones repetitivas con distinto valor de los parámetros.

<https://www.php.net/manual/es/pdo.prepare.php>

Gestión de errores con PHP: ¿ Qué puede ir mal ?

- Como ya vimos, al hacer una conexión conviene configurar el entorno para la detección y gestión de errores.
- Para ello, configuramos la constante **PDO::ATTR_ERRMODE**
- Los posibles valores de esta constante son:
 - **PDO::ERRMODE_SILENT**: Modo silencioso. Sólo establece los códigos de error.
 - **PDO::ERRMODE_WARNING**: Modo aviso, avisa de los errores.
 - **PDO::ERRMODE_EXCEPTION**: Genera un objeto de tipo **PDOException** (Clase) que podremos capturar con **try..catch**

<https://www.php.net/manual/en/class.pdoexception.php>

<https://www.php.net/manual/en/pdo.setattribute.php>

Código para la gestión de excepciones. Configuración.

- La configuración se establecerá en el fichero **pdo.php**, donde realizamos la conexión a nuestra base de datos:

```
<?php
// Conexión a la BD martabd
$pdo=new PDO("mysql:host=localhost;port=3306;dbname=martabd",
    'marta','marta');
// Control de errores
$pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
?>
```


Código para la gestión de excepciones. Gestión.

```
<?php
require_once("pdo.php");
$mensaje = "";
if (isset($_GET["id"])) {
    try {
        $sql = "SELECT * FROM usuarios WHERE usuario_id = :id";
        $sentencia = $pdo->prepare($sql);
        $sentencia->execute(array( ":id--" => $_GET["id"] ));
        $fila = $sentencia->fetch(PDO::FETCH_ASSOC);
        // fetch retorna false si no encuentra filas
        if ( $fila === false ) {
            echo "<p>No se encuentra usuario_id</p>";
        } else {
            echo "<p>usuario_id encontrado</p>";
        }
    } catch (Exception $ex) {
        error_log("Mensaje de la excepción: <br>" . $ex->getMessage());
        $mensaje = "Ha habido un problema técnico,
        llame a mantenimiento al 111 222 333";
    }
}
?>
```

- Introducimos el código de acceso a BD dentro de un **try..catch**
- Si se produce un error dentro del try..catch, PHP detiene la ejecución del programa, y continúa en el catch
- El catch recupera el error en un objeto de la clase **Exception**
- Dentro del catch, utilizo **error_log** para guardar el mensaje de la excepción en el log de errores de PHP.
- Obtengo el mensaje del error producido mediante un método de la clase Exception: **getMessage**
- Para el usuario, prepararé un mensaje más comprensible.

Código para la gestión de excepciones. Resultado.

navegador



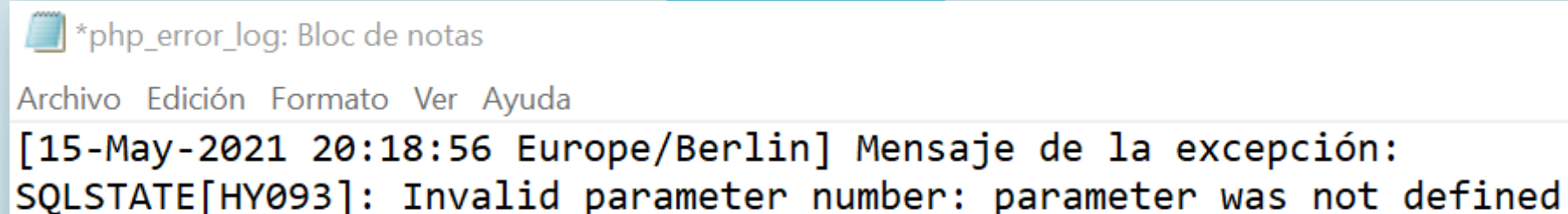
← → ↻ ⓘ localhost/marta/Transp-44-2-error.php?id=

Ha habido un problema técnico, llame a mantenimiento al 111 222 333

Prueba de errores

Id de usuario

php_error.log



*php_error_log: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
[15-May-2021 20:18:56 Europe/Berlin] Mensaje de la excepción:  
SQLSTATE[HY093]: Invalid parameter number: parameter was not defined
```

- He provocado un error en el código, escribiendo mal el nombre del parámetro del placeholder en el execute
- Al hacer submit, el programa falla en la zona interior al try..catch, con lo que no mostrará errores, sino que los recuperará en un objeto Exception.
- Mi programa informa del error de forma controlada, mostrando un mensaje al usuario y otro en el php_error.log para el programador.

Función `error_log()` y `php_error.log`

- Durante el desarrollo, nos interesa que los mensajes de error aparezcan en el navegador para poder detectarlos y corregirlos. Por eso hemos configurado el servidor para que aparezcan.
- En producción, no nos interesa que le salgan al usuario, y por eso los controlamos mediante excepciones.
- La función **`error_log`** nos permite hacer una gestión silenciosa de los mensajes. Por defecto, recibe como parámetro un string, y lo incluye en un fichero llamado **`php_error.log`**
- La localización de dicho fichero podemos verla en **PHPinfo**, en la directiva **`error_log`**

Tipos de registro de `error_log()`

- | | |
|---|---|
| 0 | <code>message</code> es enviado al registro del sistema de PHP, usando el mecanismo de registro del Sistema Operativo o un fichero, dependiendo de qué directiva de configuración esté establecida en <code>error_log</code> . Esta opción es la predeterminada. |
| 1 | <code>message</code> es enviado por email a la dirección del parámetro <code>destination</code> . Este es el único tipo de mensaje donde se usa el cuarto parámetro <code>extra_headers</code> . |
| 2 | Ya no es una opción. |
| 3 | <code>message</code> es añadido al final del fichero <code>destination</code> . No se añade automáticamente una nueva línea al final del string <code>message</code> . |
| 4 | <code>message</code> es enviado directamente al gestor de registro de la SAPI. |

<code>error_log</code>	C:\xampp\php\logs\php_error.log
-------------------------------	---------------------------------

Ejercicio 2:

Gestión de excepciones

Prueba de login

Email :

Password:

1. Partiendo del programa del ejercicio 1, vas a realizar los siguientes cambios:
2. Modifica el acceso a base de datos de forma que se realice con un prepare, y comprueba que no se consigue inyección SQL
3. Modifica el programa para que pueda capturar excepciones. Habrá un mensaje de error amigable al usuario, y otro, más técnico, que se guardará en `php_error.log`.
4. Cambia el nombre de la tabla, poniendo uno que no exista, y comprueba que se gestionan los errores correctamente

the elePHPant



<https://www.php.net/docs.php>

<https://www.php.net/manual/es/book.pdo.php>