PHP
“Seguro”

Inyección SQL

Uno de los aspectos que se pueden mejorar para aportar seguridad a nuestros proyectos es la forma de acceder a la base de datos. Una forma de realizar consultas es la siguiente:

```
mysql_query("SELECT * FROM usuario WHERE nombre = $usuario");
```

Sin embargo esta manera no es segura, ya que no impide que se pueda realizar inyección de código SQL. Para solucionar esto existen dos maneras de proceder, MySQLi y PDO.

MySQLI

Es una versión mejorada de mysql.

```
$mysqli = new mysqli("mihost", "usuario", "contraseña", "basedatos");
$sentencia = $mysqli->prepare("SELECT * FROM usuarios WHERE nombre = ?");
$sentencia->bind_param("i", $usuario );
$sentencia->execute();
```

Parametros: `i` para enteros y `s` para cadenas

PDO

```
$pdo = new PDO('mysql:host=mihost;dbname=basedatos', "usuario", "contraseña");
$sentencia = $pdo->prepare("SELECT * FROM usuarios WHERE nombre = :usuario");
$sentencia=$pdo->setAttribute(PDO::ATTR_EMULATE_PREPARES, false);
$sentencia->bindParam(":usuario", $usuario, PDO::PARAM_STR);
$sentencia->execute();
```

Parametrización:

PDO::PARAM_INT => int.

PDO::PARAM_STR => String

Otro punto a tener en cuenta en el inyectado de SQL son los buscadores. Es un aspecto en el que hay que tener cierto cuidado. Una solución es implementar buscadores de google. En cualquier caso tanto en los buscadores como en cualquier formulario es importante cuidar ciertos aspectos.

1. Usar captchas en los formularios para validar las peticiones
2. Crear limitaciones y reglas en los formularios, como por ejemplo la cantidad de caracteres
3. Enviar formularios siempre con POST
4. En los formularios usados para subir archivos como imágenes, fotos, etc. puede limitarse el tipo de archivo a subir, basado en su extensión, así como regular su tamaño

En general una buena estrategia para evitar inyección de código es utilizar la expresión:

```
mysqli_real_scape_string();
```

Cifrado de datos

La mayor manera de "garantizar" la seguridad de los datos de nuestra aplicación web es mediante el cifrado de datos. Ya que de esta manera si algún individuo consiguiera acceder a los datos no sería capaz de leerlos de manera inmediata. Existen diferentes librerías PHP que facilitan el cifrado y descifrado de datos. Un ejemplo es la extensión mcrypt, que es compatible con diferentes algoritmos criptográficos.

Para ver cuáles son compatibles con su plataforma, utiliza la función mcrypt_list_algorithms ():

```
<? Php
echo '<pre>. print_r (mcrypt_list_algorithms (), TRUE). '</ Pre> ";
?>
```

Para realizar el encriptado y desencriptado de datos se pueden utilizar dos funciones, mcrypt_encrypt() y mcrypt_decrypt(). Estas funciones aceptan cinco argumentos:

- El primero de ellos es acerca del algoritmo a usar
- El segundo es la clave
- El tercero incluye los datos para encriptar o desencriptar
- El cuarto indica el modo de encriptado o desencriptado
- El quinto argumento está dedicado al vector de inicialización

En la actualidad se recomienda dejar de utilizar estas funciones (deprecated).

Otra forma de realizar un cifrado es mediante encode-decode. Para cifrar un texto se realizaría de la siguiente manera:

```php
<?php
$str = 'Estos datos son una cadena codificada';
echo base64_encode($str);
?>
```

Dando lugar por ejemplo a la siguiente cadena:

```
VGhpcyBpcyBhbiBlbmNvZGVkIHN0cmluZw==
```

Para descifrarlo se realizaría así:

```php
<?php
$str = 'VGhpcyBpcyBhbiBlbmNvZGVkIHN0cmluZw==';
echo base64_decode($str);
?>
```

Obteniéndose el siguiente String:

```
Estos datos son una cadena codificada.
```

Una manera aún más completa de realizar el cifrado de datos es juntar métodos de cifrado. Por ejemplo en el siguiente código se utilizan los dos últimos métodos de cifrado

```
function encrypt ($string) {
   $key = "TIENE QUE SER LA MISMA EN LAS 2 FUNCIONES";
   return base64_encode(mcrypt_encrypt(MCRYPT_RIJNDAEL_256, md5($key), $string, MCRYPT_MODE_CBC, md5(md5($key))));
}
```

function decrypt ($string) {

$key = "TIENE QUE SER LA MISMA EN LAS 2 FUNCIONES";

return rtrim(mcrypt_decrypt(MCRYPT_RIJNDAEL_256, md5($key), base64_decode($string), MCRYPT_MODE_CBC, md5(md5($key))), "\0");

}

OpenSSL es otra forma de encriptar String en PHP. Para realizar las operaciones se podrían realizar de la siguiente manera:

```
$myText = 'This is my secure text';

$myText_encrypted = openCypher('encrypt',$myText);
echo $myText_encrypted;
// RESPUESTA: xrgFsPYDTxCBQbxbIteSmSJLaHlaGVmlV5oNIqvW9Sk=

$myText_decrypted = openCypher('decrypt',$myText_encrypted);
echo $myText_decrypted;
// RESPUESTA: This is my secure text
```

La función openCypher del ejemplo podría ser como la siguiente:

```
function openCypher ($action='encrypt',$string=false)
{
    $action = trim($action);
    $output = false;

    $myKey = 'oW%c76+jb2';
    $myIV = 'A)2!u467a^';
    $encrypt_method = 'AES-256-CBC';

    $secret_key = hash('sha256',$myKey);
    $secret_iv = substr(hash('sha256',$myIV),0,16);

    if ( $action && ($action == 'encrypt' || $action == 'decrypt') &&
$string )
    {
        $string = trim(strval($string));

        if ( $action == 'encrypt' )
```

```
        {
            $output = openssl_encrypt($string, $encrypt_method,
$secret_key, 0, $secret_iv);
        };

        if ( $action == 'decrypt' )
        {
            $output = openssl_decrypt($string, $encrypt_method,
$secret_key, 0, $secret_iv);
        };
    };

    return $output;
};
```

Fuente → https://junihh.com/encriptar-y-desencriptar-cadenas-de-texto-con-php.html

CONTRASEÑAS

La parte de encriptar las contraseñas es lo más básico y elemental con lo que debes empezar para tener una buena seguridad en tu aplicación. Lo ideal en este caso es utilizar las funciones hash, de manera que no se guarde la contraseña tal cual la escribiste. Una manera de poder encriptar un texto, es utilizando la función password_hash() usando el algoritmo Bcrypt. La función password_hash lleva dos parámetros: el primero el texto a encriptar y segundo el algoritmo.

```
require 'password.php';

$passwordHash = password_hash('password', PASSWORD_DEFAULT);
echo $passwordHash;
```

La contraseña se almacenará cifrada, pero cuando se quiera comprobar con en un login, será necesario verificar que la contraseña introducida por el usuario es la misma que la contraseña cifrada.

password_verify(‘dato a cifrar’, $passwordHash)

Donde ‘texto_original’ es el texto que ingresa el usuario y $passwordHash es lo que se obtiene desde la base de datos

Otra manera de encriptar datos clave es la utilización del algoritmo SHA3

```
echo hash('sha3-512' , 'Dato a cifrar');
```

Otra librería es crypt, que permite el encriptado de datos de diferentes formas. Se puede obtener más información de ella en el siguiente enlace.

http://php.net/manual/es/function.crypt.php

Filtrado de datos en formularios

La inserción de datos extraños es otro método que un atacante puede dañar nuestra aplicación, por esto es recomendable sanitar y validar todas las entradas de datos que podemos recibir. La validación se refiere a la comprobación y control de datos, es decir, los datos están de manera correcta. La sanitación se refiere a limpiar todos aquellos caracteres ilegales para evitar que un atacante inyecte código JavaScript.

Para ello se puede utilizar por ejemplo la función: filter_var($str, FILTER_SANITIZE_STRING);

Esta función recibe 2 parámetros el primero la cadena a sanitar y segundo una constante que le indica que filtro debe usar. De esta manera se elimina cualquier carácter ilegal o especial y que no sea considerado como parte de una cadena.

Ejemplos de utilización:

Validación de Strings:

```
<?php
$str = "<h1>Hola mundo!</h1>";
$newstr = filter_var($str, FILTER_SANITIZE_STRING);
echo $newstr;
?>
```

Validación de int:

```
$int = 100;

if (!filter_var($int, FILTER_VALIDATE_INT) === false) {
    echo("Válido");
} else {
    echo("No válido");
}
?>
```

Validación de mail:

```
<?php
$email = "elargor@gmail.com";

// primero se quita los carácteres no válidos
$email = filter_var($email, FILTER_SANITIZE_EMAIL);

// luego se válida el formato del email
if (!filter_var($email, FILTER_VALIDATE_EMAIL) === false) {
    echo("$email es una dirección de correo");
} else {
    echo("$email no es una dirección de correo");
}
?>
```

PHP también permite hacer validaciones de tipos, de tal manera que nos podamos asegurar de que el tipo introducido por el usuario es el correcto. Un ejemplo de ello es la función is_numeric(), que permite validar si el dato introducido es un dato numérico. Ejemplo:

```
<?php if (is_numeric($telefono)) {
            XXXXX
      } else {
            XXXXX
      } ?>
```

Otras funciones similares son:

is_null()

is_float()

is_int()

is_string()

is_object()

is_array()

is_bool()

Otras funciones interesantes a utilizar en formularios pueden ser la función strip_tags() y la función trim(). La primera de ellas evita que se pongan etiquetas HTML o PHP, mientras que la segunda evita que se introduzcan espacios al principio o al final del texto.

```
if ( isset( $_POST[ ‘nombre’ ] ) )
$nombre = strip_tags( trim( $_POST[ ‘nombre’] ) );
```

Sesiones

Se debe de priorizar el uso de sesiones frente a la utilización de cookies. La utilización de cookies es una práctica muy insegura, suelen transmitirse en texto plano, y además se almacenan en ordenador del usuario, por lo que pueden ser leídas por virus o troyanos. Las variables de sesión por el contrario son almacenadas en el servidor.

Un problema de la utilización de sesiones es que se puede hacer un ataque en el que se suplante un cliente. A esto se le suele llamar session hijacking. Para ello puede ser conveniente almacenar una serie de variables del usuario en cada visita, como por ejemplo dirección IP o la información de su navegador…

Si estos cambian significaría que podría ser un intento de hijacking, y no permitiríamos el acceso en la aplicación.

$_SERVER[‘REMOTE_ADDR’]

$_SERVER[‘HTTP_USER_AGENT’]

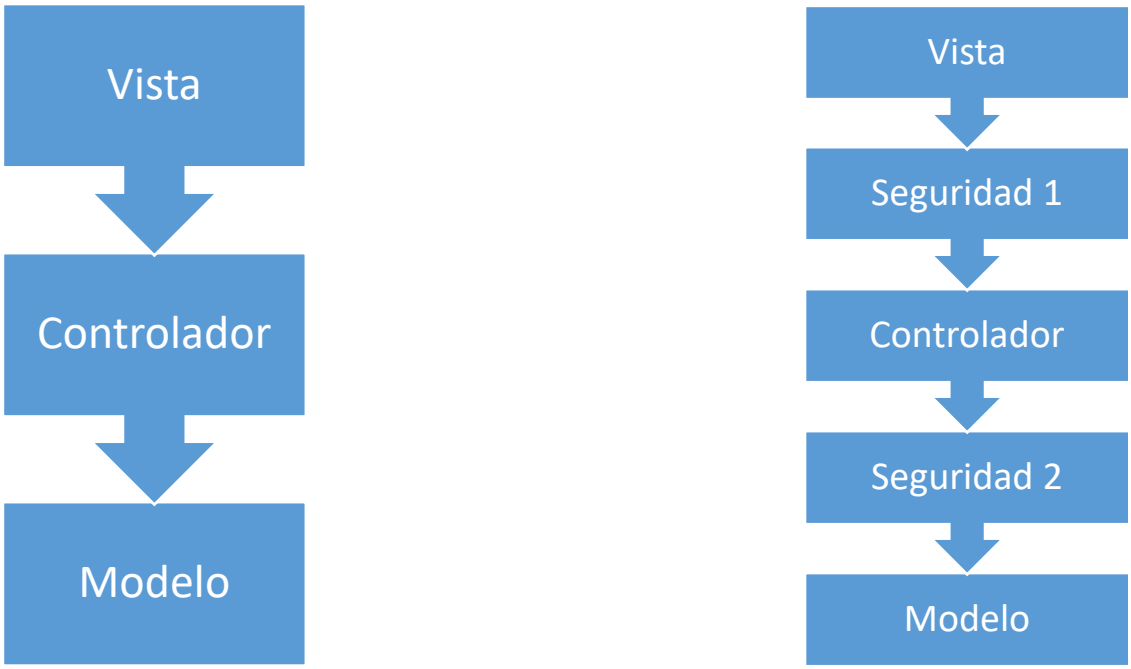
Envío de parámetros

El envío de parámetros del usuario solo se realizará por POST, nunca por GET. GET puede ser útil para informar a los usuarios de errores en la aplicación.

Además en cada página de la aplicación se deberá de comprobar si el usuario está logueado y si tiene permisos para acceder a dicha sección.

Capa de seguridad

Cuando se trabaja con Frameworks es habitual trabajar con tres capas: Vista -> Controlador -> Modelo. Otra alternativa a la hora de trabajar de manera más segura es añadir otra capa, que se encargue de evaluar la seguridad de la aplicación.



Estas capas de seguridad se encargarían de realizar diferentes funciones, como puede ser la comprobación de privilegios, de inicio de sesión, de sesión activa, cifrado y descifrado de datos…

Rendimiento

Funciones _once

En PHP existen un conjunto de librerías que evitan que los ficheros, clases o librerías cargados puedan cargarse de nuevo. Son las llamada _once. Un ejemplo de ello es la librería include_once(), que realiza las mismas funciones que la librería include, pero asegurándose de que solo se carga una única vez. El problema de utilizar este tipo de librerías es que producen un aumento considerable del rendimiento, por lo que no es aconsejable abusar de este tipo de librerías.

Switch vs if

Si en nuestra aplicación Web disponemos de muchos if encadenados puede hacer que nuestra aplicación no sea eficiente. Utilizar Switch en estos casos es mucho más eficiente que concatenar if.

Resumen

La seguridad total no existe, siempre va a existir la posibilidad de que entren en nuestros sistemas. Sin embargo cuantos más impedimentos pongamos, será menos rentable intentar acceder a nuestros sistemas. A modo de resumen para aumentar la seguridad de nestras aplicaciones se deben de tener en cuenta los siguientes aspectos:

Validación: asegurar que nos van a introducir los datos que esperamos recibir.

Sanitar: limpiar todos aquellos caracteres ilegales para evitar que un atacante inyecte código JavaScript.

Escapar: escapar de expresiones JS, caracteres especiales, expresiones regulares….

Cifrar: cifrar aquellos datos personales, para que en caso de obtenerlos no sean legibles.

Por ultimo puede ser importante seguir las alertas, manuales y consejos que ofrece el Instituto Nacional de Ciberseguridad (INCIBE):

https://www.incibe.es/

Y a la oficina de seguridad del Internauta (OSI):

https://www.osi.es/es