Tema 4 : Formularios

Ciclo Superior DAW

Asignatura: Desarrollo web en entorno servidor

Curso 20/21

Introducción

- En este capítulo veremos cómo crear formularios y programas que recogen los valores introducidos
- Veremos los diferentes controles de formularios
- Por último, veremos problemas potenciales que tienen los formularios y cómo sanitizarlos

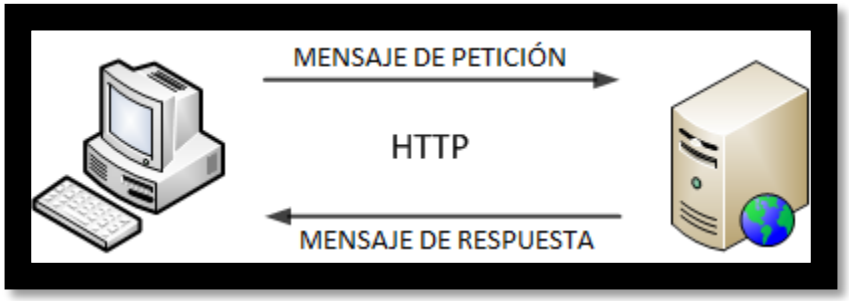
Protocolo HTTP

- Antes de comenzar a trabajar con formularios web, debemos conocer los mensajes que entrega el protocolo HTTP para enviar información del servidor al cliente y viceversa, y los mecanismos que podemos utilizar desde PHP para gestionar estos mensajes.

Protocolo HTTP. Mensajes

- El protocolo HTTP realiza el envío de mensajes de petición y de respuesta.
- El navegador manda un **mensaje de petición** para solicitar al servidor web un recurso
- El servidor devuelve el recurso solicitado en un **mensaje de respuesta**.

Protocolo HTTP. Mensajes



Protocolo HTTP. Mensajes

- Cada mensaje se compone de:
  - Línea de petición o de respuesta.
  - Encabezados.
  - Cuerpo del mensaje (opcional). Se emplea una línea en blanco para separarlo de los encabezados.

Protocolo HTTP. Mensajes de petición

- Son enviados por el cliente web (normalmente un navegador) al servidor web.
- Por ejemplo:

```
GET /images/logo.png HTTP/1.1
```

Protocolo HTTP. Mensajes de petición

- Los métodos más comunes son:
  - **GET**: Pide una representación del recurso especificado (el contenido del archivo correspondiente al recurso). Se emplea constantemente en el navegador web para obtener páginas, imágenes y cualquier otro documento almacenado en un servidor web.
  - **HEAD**: Pide una respuesta idéntica a la que correspondería a una petición GET, pero sin el cuerpo de la respuesta. Esto es útil por ejemplo para saber si una página web que el navegador tiene almacenada en caché, fue actualizada en el servidor.

Protocolo HTTP. Mensajes de petición

- Los métodos más comunes son:
  - **POST**: Envía datos para que sean procesados por el recurso identificado. Los datos se incluyen en el cuerpo del mensaje.
  - **PUT**: Sube un recurso especificado (archivo) al servidor.
  - **DELETE**: Borra el recurso especificado.

Protocolo HTTP. Mensajes de respuesta

- Son enviados por el servidor web al cliente o navegador web como respuesta a un mensaje anterior de petición
- Por ejemplo:

```
HTTP/1.1 200 OK
```

Protocolo HTTP. Mensajes de respuesta

- En un mensaje de respuesta, la primera línea (la línea de respuesta) contiene:
    - La versión del protocolo.
    - Un código de estado.
    - Una descripción.

Protocolo HTTP. Mensajes de respuesta

- El código de respuesta o código de estado es un número de tres cifras que sirve para indicar si una petición se recibió y atendió correctamente, o si se produjo cualquier problema. El primer dígito indica el tipo del código.

- A continuación, veremos los diferentes códigos de respuesta:

Protocolo HTTP. Mensajes de respuesta

Código de respuesta	Tipo	Significado
`1xx`	Respuesta informativa	Le indica al cliente que se recibió la petición y se está procesando. Se trata de una respuesta provisional y hay que realizar una nueva transacción HTTP para que se pueda obtener una respuesta definitiva.
`2xx`	Petición correcta	Indica que la petición recibida anteriormente fue recibida, aceptada y procesada correctamente en el servidor.
`3xx`	Redirección	Se le indica al cliente HTTP que tiene que realizar alguna acción adicional para que se pueda resolver completamente la petición que se realizó.
`4xx`	Errores del cliente	Se recibió una petición con una sintaxis errónea o no se pudo procesar la petición.
`5xx`	Errores del servidor	Se produjo un error en el servidor que le impidió atender y procesar la petición.

Protocolo HTTP. Mensajes de respuesta

- Los mensajes de respuesta más comunes son:
  - **200 (OK).** Es la respuesta estándar para peticiones correctas.
  - **403 (Prohibido).** El servidor rechaza responder a la solicitud por falta de privilegios (por ejemplo, el usuario que lo solicita no está autenticado o no tiene permiso).
  - **404 (Recurso no encontrado).** Se utiliza cuando el servidor web no encuentra la página o recurso solicitada.

Protocolo HTTP. Encabezados

- Se utilizan en función del método de la petición o respuesta, para dar más información.

- Hay **encabezados generales** (se utilizan tanto en mensajes de petición como en mensajes de respuesta), y **encabezados específicos** para mensajes de petición o de respuesta

Protocolo HTTP. Encabezados

- Tipos de encabezado

Encabezado	Tipo	Significado
`Host`	de petición	Equipo al que se envía a petición.
`User-Agent`	de petición	Nombre y versión del cliente (navegador) y del sistema operativo.
`Server`	de respuesta	Nombre y versión del software que se ejecuta en el servidor web.
`MIME-Version`	general	Versión de MIME que utiliza el cliente.
`Accept`	de petición	Tipo de contenido que acepta el navegador.
`Accept-Language`	de petición	Idiomas que espera el navegador en las páginas recibidas.
`Accept-Encoding`	de petición	Sistema de codificación que espera el navegador para el recurso a recibir.
`Accept-Charset`	de petición	Juego o conjunto de caracteres que espera recibir el navegador.
`Referer`	de petición	URL desde donde se originó la petición (URL de la página que contenía o enlace).
`Location`	de respuesta	Se utiliza en una mensaje de redirección para indicar al navegador la nueva URL.
`Set-Cookie`	de respuesta	Envía una cookie del servidor al cliente para que la almacene.
`Cookie`	de petición	Envía al servidor las cookies almacenadas para el dominio.

Protocolo HTTP. Encabezados

- Por ejemplo, para indicar que la petición va destinada al equipo **www.ciclosmontecastelo.com**, se añadiría el siguiente encabezado al mensaje de petición:

```
Host: www.ciclosmontecastelo.com
```

Protocolo HTTP. Ejemplo completo

- Una petición HTTP para obtener una página web HTML de un servidor podría tener la siguiente forma (el mensaje no contiene cuerpo, solamente la línea de petición y encabezados):

```
GET / index.html HTTP/1.1
Host: www.ciclosmontecastelo.com
User- Agent:  Mozilla/5.0 (Windows  NT 6.1;  WOW64)
[línea en blanco]
```

Protocolo HTTP. Ejemplo completo

- Y su respectiva respuesta:

```
HTTP/1.1 200  OK
Date:  Fri, 21  Oct 2021 14:21:29 GMT
Content- Type:  text/html
Content- Length: 2221
[línea en blanco]
<body>
 <h1>Encabezado</h1>
 (Contenido)
 …
</body>
```

Protocolo HTTP. Funciones

- Desde PHP existen funciones para gestionar los mensajes enviados por el protocolo HTTP. Las veremos a continuación:

Protocolo HTTP. Funciones. Header

Header

- Desde PHP podemos emplear la función "header" para añadir un encabezado específico a la página de respuesta generada.

Protocolo HTTP. Funciones. Header

- Por ejemplo, para redireccionar la petición a una nueva página, podemos hacer:

```php
<?php
    header(" Location: http://www.ciclosmontecastelo.com/");
?>
```

Protocolo HTTP. Funciones. Header

- Es posible emplear la función header para especificar la línea de respuesta.

  En este caso el texto deberá comenzar por " HTTP/". Por ejemplo:

```php
<?php
    header(" HTTP/1.1 404 Not Found");
?>
```

Protocolo HTTP. Funciones. Header

- Podemos emplear la función header para el navegador responde con una imagen.
- Lo veremos de dos formas distintas

Protocolo HTTP. Funciones. Header

- Empleando un mensaje de redirección con la URL de la imagen:

```
<?php
   header(' Location: imágenes/usuarios/anonymous.png');
?>
```

Protocolo HTTP. Funciones. Header

- Enviando el contenido de la imagen, extraído del archivo, e indicando el tipo de contenido adecuado:

```php
<?php
    $imagen = file_get_contents(RAIZ_WEB.'imágenes/usuarios/anonymous.png');
    header("Content-type: image/jpeg");
    echo $imagen;
?>
```

Protocolo HTTP. Funciones. Header_remove

Header_remove

- Para eliminar un encabezado añadido anteriormente podemos emplear la función " header_remove".

Protocolo HTTP. Funciones. Header_remove

- Por ejemplo, para eliminar un encabezado de PHP:

```
<?php
    header_remove('X- Powered- By');
?>
```

Protocolo HTTP. Funciones. Headers_list

Headers_list

- Devuelve un array con los encabezados que vamos a enviar en la respuesta al cliente

Protocolo HTTP. Funciones. Headers_list

- Por ejemplo:

```php
<?php
    var_dump(headers_list());
?>
```

- El resultado obtenido es:

```
array (size=1)
  0 => string 'X-Powered-By: PHP/5.4.7' (length=23)
```

Protocolo HTTP. Funciones. Headers_sent

Headers_sent

- La utilizamos para comprobar si aún no fueron enviados los encabezados (y por lo tanto aún estamos a tiempo de añadir alguno más o de eliminarlos)

Protocolo HTTP. Funciones. Headers_sent

- Por ejemplo:

```php
<?php
    if(!headers_sent()) {
        header('Location: http://www.ciclosmontecastelo.com/');
        exit;
    }
?>
```

Protocolo HTTP. Extensiones

- En PHP podemos emplear las siguientes extensiones para trabajar directamente con el protocolo HTTP:
  - **cURL**: Utiliza la librería libcurl para funcionar como cliente de varios protocolos
  - **HTTP**: Facilita entre otras el manejo de los mensajes HTTP

Formulario web

- Para hacer llegar a la aplicación web los datos del usuario desde un navegador utilizamos **formularios HTML**.
- Un formulario HTML es un conjunto de controles (botones, cajas de texto, casillas de verificación, etc.) que permiten al usuario introducir datos y enviarlos al servidor web para su procesamiento.

Formulario web

- Los formularios HTML van encerrados siempre dentro de un elemento "form", esto es, entre las etiquetas "**<form>**" y "**</form>**".
- Dentro de un formulario se incluyen los controles sobre los que puede actuar el usuario, principalmente usando las etiquetas "**<input>**", "**<select>**", "**<option>**", "**<textarea>**" y "**<button>**".

Formulario web

- Los siguientes atributos pueden aplicarse a los controles de un formulario:
    - **name**: identifica al control. El formulario envía al servidor los datos de los controles que tienen establecido el atributo "name", junto con los valores que introdujo en ellos el usuario.
    - **value**: permite establecer el valor inicial de un control. Cada control lo utiliza diferente.
    - **disabled**: permite deshabilitar el control. Una vez deshabilitado, el control no puede coger el foco.

Formulario web

- Los siguientes atributos pueden aplicarse a los controles de un formulario:
  - **readonly**: permite que el control no sea modificable, aunque puede coger el foco.
  - **tabindex**: controla el orden en la que el foco pasa de un elemento a otro usando tabulador.
  - Dentro de un formulario se incluyen los controles sobre los que puede actuar el usuario, principalmente usando las etiquetas "**<input>**", "**<select>**", "**<option>**", "**<textarea>**" y "**<button>**".

Formulario web

- El elemento "form" de un formulario HTML tiene dos atributos importantes:
    - El atributo "**action**" del elemento FORM indica la URL (relativa o absoluta) del recurso o la página a la que se le enviarán los datos del formulario para procesarlos.

Formulario web

- El elemento "form" de un formulario HTML tiene dos atributos importantes:
    - El atributo "**method**" especifica el método usado para enviar la información. Este atributo puede tener dos valores: **GET**
        - **indica que se empleará un mensaje de petición GET para enviar los datos del formulario**.
        - **Los datos del formulario se agregan al URI utilizando un signo de consulta "?" como separador.**

Formulario web

- Por ejemplo, si pasáramos un nombre de usuario y su contraseña mediante GET, podrían verse ambos en la dirección de la página:

```
http://localhost/BoletinesDWES/Boletin6/welcome.php
Host: localhost
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:69.0) Gecko/20100101 Firefox/69.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: es-MX,es;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 34
Connection: keep-alive
Referer: http://localhost/BoletinesDWES/Boletin6/newhtml.html
Cookie: has_js=1
Upgrade-Insecure-Requests: 1
name=Martin&email=Estanolaadivinas
POST: HTTP/1.1 404 Not Found
Date: Wed, 18 Sep 2019 09:24:30 GMT
Server: Apache/2.4.41 (Win64) OpenSSL/1.1.1c PHP/7.3.9
Vary: accept-language,accept-charset
Accept-Ranges: bytes
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: text/html
Content-Language: es
```

Formulario web

- El elemento "form" de un formulario HTML tiene dos atributos importantes:
    - El atributo "**method**" especifica el método usado para enviar la información. Este atributo puede tener dos valores: **POST**
        - **Indica que se empleará un mensaje de petición POST para enviar los datos del formulario.**
        - **En una petición POST los datos que se envían al servidor se incluyen como parte del cuerpo del mensaje.**

Formulario web

- Por ejemplo, empleando POST el mensaje de petición anterior sería:

```
http://localhost/BoletinesDWES/Boletin6/welcome.php
Host: localhost
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:69.0) Gecko/20100101 Firefox/69.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: es-MX,es;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 29
Connection: keep-alive
Referer: http://localhost/BoletinesDWES/Boletin6/newhtml.html
Cookie: has_js=1
Upgrade-Insecure-Requests: 1
name=Martin&email=Adivinaesta
POST: HTTP/1.1 404 Not Found
Date: Wed, 18 Sep 2019 09:23:06 GMT
Server: Apache/2.4.41 (Win64) OpenSSL/1.1.1c PHP/7.3.9
Vary: accept-language,accept-charset
Accept-Ranges: bytes
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: text/html
Content-Language: es
```

Formulario web

- Existen dos botones que tienen un comportamiento especial dentro de un formulario web:
    - El **botón de envío** de un formulario provoca que se envíen sus datos al servidor. Se crea con la etiqueta "<input>" indicando en su atributo "type" el valor "submit".
    - El **botón de reset** limpia los datos del formulario, dejándolo tal y como se mostró al cargar la página. Se crea con la etiqueta "<input>" indicando en su atributo "type" el valor "reset". Tanto este control como el anterior mostrarán el texto que figure en su atributo "value".

Formulario web

- Por ejemplo:

```
<form>
  <p><input type="submit" value="registrar">
  <input type="reset" value="borrar">p>
</form>
```

Actividad 1

Crea dos páginas web Actividad1 y Actividad1_Recibir.

Crea un formulario web en el que envíes el usuario y contraseña que el usuario mande por pantalla. La contraseña no debe ir en claro y no se debe enviar por la URL

Formulario web. Controles

- Algunos de los controles son:
  - Caja de texto
  - Área de texto
  - Casilla de verificación
  - Botones de selección
  - Lista de selección

Formulario web. Controles. Caja de texto

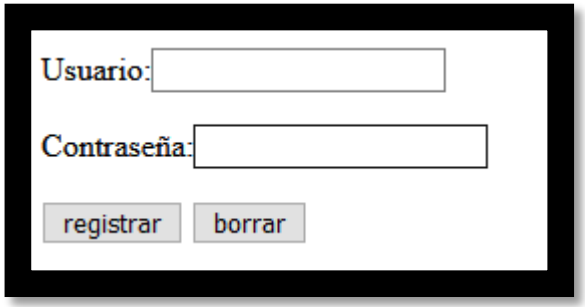
- Es un control que permite al usuario introducir una línea de texto.
- Se crea también con la etiqueta "<input>" indicando en su atributo "type" el valor "text", o el valor "password" si no queremos que se muestre en pantalla los caracteres que se teclean.

Formulario web. Controles. Caja de texto

- La sintaxis es:

```
<form>
  <p>Usuario:<input type="text"  name=" login"></p>
  <p>Contraseña:<input type="password" name="pword"></p>
 <p>
 <input type="submit" value="registrar">
 <input type="reset" value="borrar">
 </p>
</form>
```

- Y el resultado:



Formulario web. Controles. Área de texto

- Es un control semejante al anterior pero que puede extenderse la dos o más líneas.
- Se crea con la etiqueta "<textarea>" y se pueden emplear los atributos "rows" y "cols" para indicar el número de filas y columnas que ocupará.
- El único control sin atributo "value" es "<textarea>".

Formulario web. Controles. Área de texto

- La sintaxis es:

```
<form>
    <p>Observaciones:
        <textarea rows="2" cols="30" name="observs"></textarea>
    </p>
    <p><input type="submit" value="registrar">
    <input type="reset" value="borrar"></p>
</form>
```

- Y el resultado:

Formulario web. Controles. Casilla de verificación

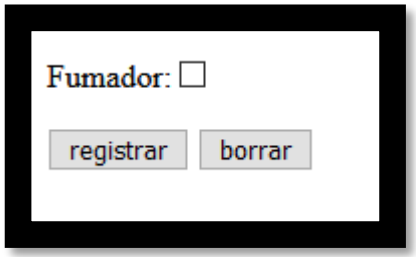
- Una casilla de verificación se define mediante una etiqueta "<input>" indicando en su atributo "type" el valor "checkbox".
- Inicialmente aparece sin marcar, y en este estado no se envía al servidor.
- Puede añadírsele el atributo "checked" para que aparezca marcada por defecto.

Formulario web. Controles. Casilla de verificación

- La sintaxis es:

```
<form>
    <p>Fumador:<input type="checkbox" name="Fumador"></p>
    <p>
        <input type="submit" value="registrar">
        <input type="reset" value="borrar">
    </p>
</form>
```

- Y el resultado:



Formulario web. Controles. Botones de selección

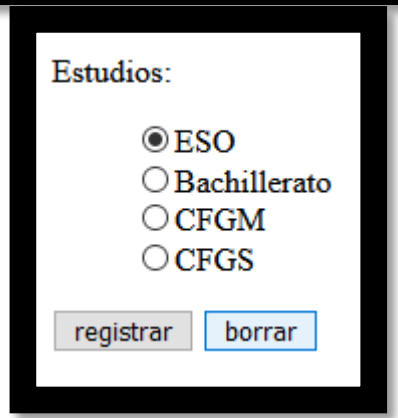
- Conjunto de opciones de las que solamente se puede escoger una.
- Cada una de las opciones del grupo se define mediante una etiqueta "<input>" indicando en su atributo "type" el valor "radio".
- El atributo "name" se emplea para agrupar las opciones de un mismo grupo y el atributo "checked" para indicar una opción marcada por defecto.

Formulario web. Controles. Botones de selección

- La sintaxis es:

```
<form>
    <p>Estudios:</p>
    <ul style="list-style: none;">
    <li><input type="radio"  name="est" value="eso" checked="checked">ESO</li>
    <li><input type="radio"  name="est"  value="bac">Bachillerato</li>
    <li><input type="radio"  name="est"  value="cfm">CFGM</li>
    <li><input type="radio"  name="est"  value="cfs">CFGS</li>
    </ul>
    <p>
        <input type="submit" value="registrar">
        <input type="reset" value="borrar">
    </p>
</form>
```

- Y el resultado:



Formulario web. Controles. Lista de selección

- Muestra varias opciones de las que se puede escoger una o varias.

  Comprende un elemento "select" para definir la lista y uno o varios elementos "option" por cada una de las opciones.

- El elemento "select" puede tener un atributo "size" para indicar el número de elementos que se muestran de manera simultánea, y otro atributo "multiple" cuando se desea que se pueda seleccionar y enviar más de un elemento.

Formulario web. Controles. Lista de selección

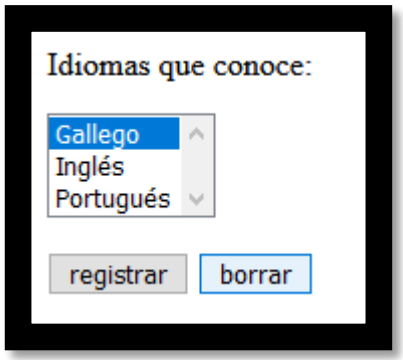
- Si queremos que alguna opción aparezca escogida por defecto, deberemos añadirle el atributo "selected".

Formulario web. Controles. Lista de selección

- La sintaxis es:

```
<form>
    <p>Idiomas que conoce:</p>
    <select name="idioma[]" size="3" multiple="multiple">
      <option selected="selected" value="g">Gallego</option>
      <option value="i">Inglés</option>
      <option value="P">Portugués</option>
    </select>
    <p><input type="submit" value="registrar">
    <input type="reset" value="borrar"></p>
</form>
```

- Y el resultado:



Formulario web. Recogida de datos

- Los datos se recogerán de distinta forma dependiendo de cómo se envíen:
  - **Si usamos el método POST:**
    - Se recogen empleando la variable **$_POST**, que es un array global asociativo compuesto por los datos recibidos del formulario web.
  - **Si usamos el método GET:**
    - Los datos se recogen en una variable de nombre **$_GET**.

Formulario web. Recogida de datos

- Por ejemplo, los datos del siguiente formulario:

```
<form method="post" action="_pruebas2.php">
  <p>Nombre: <input type="text" name="nombre"></p>
  <p>Módulos que estudia:</p>
  <select name="modulos[]" size="3" multiple="multiple">
    <option selected="selected" value="LMS">Lenguajes de marcas</option>
    <option value=" SRI">Servicios de red e Internet</option>
    <option value="BBDD">Bases de datos</option>
  </select>
  <p>
      <input type="submit" value="registrar">
      <input type="reset" value="borrar">
  </p>
</form>
```

Formulario web. Recogida de datos

- Se pueden procesar con el código:

```php
<?php
    $nombre = $_POST['nombre'];
    $modulos = $_POST['modulos'];
    print "Nombre: ".$nombre."<br />";
    foreach ($modulos as $modulo) {
        print "Modulo: ".$modulo."<br />";
    }
?>
```

Actividad 2

Repite el ejemplo anterior considerando que el método de envío es GET en vez de POST.

¿Cómo debería ser el formulario y el código que lo recoge?

Formulario web. Recogida de datos. Arrays

- Cuando un elemento del formulario contiene o puede contener más de un valor distinto, el nuevo miembro creado en el array asociativo **$_POST es un array.**

Formulario web. Recogida de datos. Arrays

- Si por el contrario hubiéramos usado el método GET, en el servidor **los datos se recogen en una variable de nombre $_ GET**. El código necesario para procesar los datos sería similar; simplemente haría falta cambiar el nombre de la variable $_POST por $_GET.

Formulario web. Recogida de datos. Arrays

```php
<?php
    $nombre = $_POST['nombre'];
    $modulos = $_POST['modulos'];
     print "Nombre: ".$nombre."<br/>";
     foreach ($modulos as $modulo) {
         print "Modulo: ".$modulo."<br/>";
    }
?>
```

Formulario web. Recogida de datos. Arrays

- En cualquiera de los dos casos, cualquier que sea el método empleado por el formulario web para enviar los datos, **podríamos haber usado la variable $_REQUEST** sustituyendo respectivamente a $_POST y $_GET.

- **$_REQUEST almacena el contenido de los arrays $_POST, $_GET y $_COOKIE.**

Formulario web. Recogida de datos. Arrays

```php
<?php
    $nombre = $_REQUEST['nombre'];
    $modulos = $_REQUEST['modulos'];
    print "Nombre: ".$nombre."<br/>";
    foreach ($modulos as $modulo) {
        print "Modulo: ".$modulo."<br/>";
    }
?>
```

Formulario web. Subir ficheros al servidor

- Usando el método POST se puede subir un fichero al servidor web
- Debemos:
  - En la etiqueta **<form>**:
    - El tipo de codificación para los datos que se envían esté especificado como "multipart/form-data"
  - En la etiqueta **<input>**:
    - Definir el atributo “type=file”

Formulario web. Subir ficheros al servidor

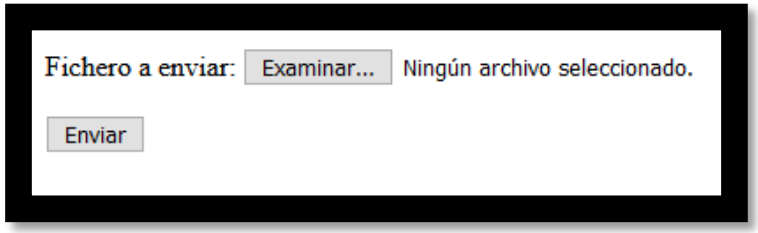
- Es posible limitar el tamaño máximo del archivo a enviar desde el formulario web.
- Se hace añadiendo al formulario un campo oculto de tipo "<input>" con el atributo "**name='MAX_ FILE_ SIZE**'".
- El tamaño máximo (en KB) se indica en el atributo "value".

Formulario web. Subir ficheros al servidor

- La sintaxis es:

```
<form enctype="multipart/form-data" action="envio.php" method="POST">
    <p>Fichero a enviar: <input name="imagen" type="file" /></p>
    <input type="hidden" name="MAX_FILE_SIZE" value="1024" />
    <input type="submit" value="Enviar" />
</form>
```

- Y el resultado:



Formulario web. Subir ficheros al servidor

- Las directivas de configuración para subir ficheros se pueden ver a continuación:

Directiva de configuración de PHP	Significado
`file_uploads`	Indica si se permiten o no las subidas de ficheros al servidor.
`post_max_size`	Tamaño máximo de la información transmitida con el método POST. Afecta a los ficheros y a cualquier otra información transmitida por ese método.
`upload_max_filesize`	Tamaño máximo de cada uno de los ficheros que se suben al servidor.
`upload_tmp_dir`	Directorio en que se almacenan de forma temporal los ficheros que se suben al servidor web.

Formulario web. Subir ficheros al servidor

- **El nombre del archivo enviado se recibe en el array global $_ FILES.**
- Si el fichero se sube correctamente al servidor, se creará un nuevo elemento en el array $_FILES con el nombre definido en el atributo "name" del elemento "<input>“.

Formulario web. Subir ficheros al servidor

- Los elementos del array **$_FILES** son:

Miembros de los elementos del array $_FILES	Contenido
`$_FILES['imagen']['name']`	Nombre original del fichero
`$_FILES['imagen']['size']`	Tamaño del fichero
`$_FILES['imagen']['type']`	Tipo MIME del fichero, tal y como lo proporciona el navegador (por ejemplo, "image/png").
`$_FILES['imagen']['tmp_name']`	Nombre temporal del fichero recibido en el servidor.
`$_FILES['imagen']['error']`	Código de error en el caso de un envío incorrecto

Formulario web. Subir ficheros al servidor

- Los errores se guardarán en constantes en el miembro

  $_FILES['imagen']['error']:

Constante de erro	Significado
`UPLOAD_ERR_OK`	Archivo subido correctamente.
`UPLOAD_ERR_INI_SIZE`	Tamaño del archivo superior al definido en la directiva upload_max_filesize.
`UPLOAD_ERR_FORM_SIZE`	Tamaño del archivo superior al definido en la directiva MAX_FILE_SIZE del formulario HTML.
`UPLOAD_ERR_PARTIAL`	Archivo parcialmente subido.
`UPLOAD_ERR_NO_FILE`	No se recibió el archivo
`UPLOAD_ERR_NO_TMP_DIR`	No existe la directiva upload_tmp_dir que define la carpeta de almacenamiento temporal.
`UPLOAD_ERR_CANT_WRITE`	No se pudo escribir el archivo en la carpeta de almacenamiento temporal.
`UPLOAD_ERR_EXTENSION`	Subida detenida por una extensión de PHP.

Formulario web. Subir ficheros al servidor

- Veamos un ejemplo completo:

```php
<?php
try {
  // Comprobamos si existe el elemento en el array $_FILES
  if (!isset($_FILES['imagen']['error']))
    throw new RuntimeException('Se produjo un error en el envío del fichero.');

  // Comprobamos que el código de error sea UPLOAD_ERR_OK
  switch ($_FILES['imagen']['error']) {
    case UPLOAD_ERR_OK:  // Todo correcto
        break;
    case UPLOAD_ERR_NO_FILE:
        throw new RuntimeException('No se recibió el archivo.');
    case UPLOAD_ERR_INI_SIZE:
    case UPLOAD_ERR_FORM_SIZE:
        throw new RuntimeException('Tamaño del archivo demasiado grande.');
    default:
        throw new RuntimeException('Error desconocido.');
  }

  // Comprobamos el tamaño de la imagen
  if ($_FILES['imaxe']['size'] > 1000000)
    throw new RuntimeException('Tamaño del archivo demasiado grande.');
```

Formulario web. Subir ficheros al servidor

- Veamos un ejemplo completo:

```
  // Usamos la extensión Fileinfo para comprobar que el tipo MIME
  //sea correcto (que sea una imagen)
  $finfo = finfo_open(FILEINFO_MIME_TYPE);
  $ext = array_search(
    finfo_file($finfo, $_FILES['imagen']['tmp_name']),
    array('jpg' => 'image/jpeg',
          'png' => 'image/png',
          'gif' => 'image/gif')
    );

  // Si no es una imagen, acabamos
  if ($ext === false) throw new RuntimeException('Imagen no reconocida.');

  // Renombramos y movemos la imagen recibida a su localización definitiva
  $res = move_uploaded_file($_FILES['imagen']['tmp_name'], 'dt/foto.'.$ext);
  if (!$res) throw new RuntimeException('La imagen no se pudo mover.');

  echo 'Imagen subida correctamente.';
}

catch (RuntimeException $e) {
  echo $e->getMessage();
}
?>
```

Actividad 3

Añade la funcionalidad de subir ficheros al servidor a las páginas de Actividad 1

Formulario web. Funciones

- A continuación, veremos algunas funciones para permitir el trabajo con ficheros:
  - **file_exists($fichero)**: Permite comprobar si el fichero existe
  - **mkdir($carpeta)**: Permite crear una carpeta para el usuario
  - **fopen($ruta, modo)**: Permite abrir un fichero en modo lectura (“r”) o escritura (“w”)

Formulario web. Funciones

- A continuación, veremos algunas funciones para permitir el trabajo con ficheros:
    - **fgets($fichero)**: Permite obtener una línea del fichero
    - **finfo_open(CONSTANTE_FILEINFO)**: Crea un nuevo recurso fileinfo
    - **finfo_file($carpeta)**: Devuelve información sobre el fichero

Formulario web. Funciones

- A continuación, veremos algunas funciones para permitir el trabajo con ficheros:
    - **fwrite($fichero, $cadena)**: Permite escribir en un fichero
    - **fclose($fichero)**: Cierra un fichero

Formulario web. Funciones

- Algunos ejemplos de uso de estas funciones:

```php
<?php
$nombre_fichero = '/ruta/pruebaMontecastelo.txt';

if (file_exists($nombre_fichero)) {
    echo "El fichero $nombre_fichero existe";
} else {
    echo "El fichero $nombre_fichero no existe";
}
?>
```

Formulario web. Funciones

- Algunos ejemplos de uso de estas funciones:

```php
<?php
if (mkdir("/miCarpeta") === false) {
    echo "La carpeta no se ha creado";
} else {
    echo "La carpeta se ha creado correctamente";
}
?>
```

Formulario web. Funciones

- Algunos ejemplos de uso de estas funciones:

```
<?php
$fichero = fopen("/ruta/fichero.txt", "r");
$fichero = fopen("ruta/fichero.txt", "w");
?>
```

Formulario web. Funciones

- Algunos ejemplos de uso de estas funciones:

```php
<?php
$fichero = @fopen("/ruta/fichero.txt", "r");
if ($fichero) {
    $linea = fgets($fichero);
}
?>
```

Formulario web. Funciones

- Algunos ejemplos de uso de estas funciones:

```
<?php
$finfo = finfo_open(FILEINFO_MIME_TYPE);
$ext = array_search(
        finfo_file($finfo, $tmp_name),
        array('jpg' => 'image/jpeg',
              'png' => 'image/png',
              'gif' => 'image/gif'));
?>
```

Formulario web. Funciones

- Algunos ejemplos de uso de estas funciones:

```php
<?php

$fp = fopen('fichero.txt', 'w');
fwrite($fp, 'hola');
fwrite($fp, ', ¿como estás?');
fclose($fp);

?>
```

Formulario web. Validación de formularios

- Se deben validar los datos introducidos en un formulario.
- La mejor manera de validar los datos es empleando código Javascript que se ejecute en el navegador del usuario.
- De esta forma los datos se comprueban antes de enviarse al servidor.

Formulario web. Validación de formularios

- El procedimiento de validación de la información recibida de un formulario web consta normalmente de dos pasos:
  - Comprobar si se recibió toda la información
  - Comprobar los tipos de datos de la información recibida

Formulario web. Validación de formularios. Paso 1

Paso 1: “Comprobar si se recibió toda la información”

- **Comprobar si un campo no se recibió o está vacío:**
    - **Podemos emplear la función empty**, que comprueba tanto que una variable exista como que no esté vacía (considera vacía una variable si contiene alguno de los valores 0, NULL, FALSE o la cadena vacía).

Formulario web. Validación de formularios. Paso 1

- **La primera comprobación debería ser si existe algo en la variable $_POST (o $_ GET**, dependiendo del método empleado en el formulario web).

- **Si está vacía significa que no se recibió ningún formulario** (por ejemplo, porque el usuario tecleara la URL de la página directamente en su navegador).

Formulario web. Validación de formularios. Paso 1

- **Comprobación:**

```
if (empty($_POST))  exit('Información del formulario no recibida');
```

Formulario web. Validación de formularios. Paso 1

- **A continuación, deberemos comprobar que los campos obligatorios no se dejaron vacíos. Por ejemplo:**

```
// Campos obligatorios
$oblig = array('Nombre', 'contrasenha', 'email');

// Comprobamos que ningún campo obligatorio está vacío
$erro = false;
foreach($oblig as $campo) {
  if (empty($_POST[$campo])) $error = true;
}

if ($error) {
  // No existe alguno de los campos obligatorios
}
```

Actividad 4

Comprueba que se controla que se envíen los campos de la Actividad 3.

El envío del fichero no es obligatorio

Formulario web. Validación de formularios. Paso 2

Paso 2: “Comprobar los tipos de datos de la información recibida”

- Por ejemplo, los nombres deberían ser cadenas de texto, y la edad de una persona un número entero.

- Para estas comprobaciones podemos usar algunas de las funciones de comprobación de tipo, que comienzan con **'is_':**

Formulario web. Validación de formularios. Paso 2

Función de comprobación de tipo	
`is_array`	Comprueba que a variable sea un array.
`is_bool`	Comprueba que a variable sea de tipo booleano.
`is_float, is_double, is_real`	Comprueban que a variable sea un número real.
`is_int, is_integer, is_long`	Comprueban que a variable sea un número entero.
`is_numeric`	Comprueba que a variable sea un número o una cadena numérica.
`is_scalar`	Comprueba que a variable sea un escalar (entero, real, booleano o cadena de texto).
`is_string`	Comprueba que a variable sex una cadena de texto.

Formulario web. Validación de formularios. Paso 2

- **Comprobación:**

```
if (!is_string($_POST['nombre'])) $error = true;
```

Seguridad de información. Problemas potenciales

- Se debe comprobar que el contenido de la información recibida no va a producir errores o comportamientos indeseados cuando sea procesado o almacenado en una base de datos:
- Principales problemas cuando se trabaja con información no verificada:
  - Inyección SQL
  - Cross-Site Scripting(XSS)

Seguridad de información. Inyección SQL

- Consiste en introducir junto con los datos (por ejemplo, en los formularios) código que tiene por objeto modificar las consultas que se realizan sobre la base de datos.

- A continuación veremos un ejemplo completo:

Seguridad de información. Inyección SQL

- El siguiente código PHP recibe el nombre y la contraseña introducidos en un formulario HTML de login

```
$consulta = " SELECT * FROM usuarios WHERE nombre='".$_POST["nombre"]."'
AND contraseña='" . $_POST["contraseña"] . "'";
$resultado = $ db-> query($consulta);
if($resultado) {
    $usuario = $resultado-> fetch_array(); // Obtenemos el primer registro
    $nombre = $usuario['nombre'];
    $contraseña = $usuario['contraseña'];
    ...
}
```

Seguridad de información. Inyección SQL

- El objetivo del código anterior es comprobar si existe en la base de datos un usuario con el mismo nombre y contraseña introducidos por el usuario. Si existe, se suponen que es único y coge los datos del primer registro devuelto por la consulta.

Seguridad de información. Inyección SQL

PROBLEMA

- Si el usuario introdujera el siguiente texto en el campo "contraseña" del formulario anterior:

```
"'OR 1;"
```

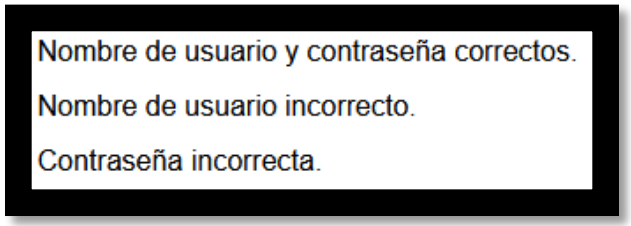
- La consulta que ejecutaría PHP sería la siguiente y devolvería la lista completa de usuarios:

```
SELECT * FROM usuarios WHERE nombre='xxx' AND contraseña=''OR 1;'
```

Seguridad de información. Inyección SQL. Ejemplos

Ejemplo 1: Acceder a la aplicación sin tener nombre de usuario ni contraseña

- Cuando el usuario escriba un nombre de usuario y contraseña, la aplicación responde uno de los siguientes mensajes de error.

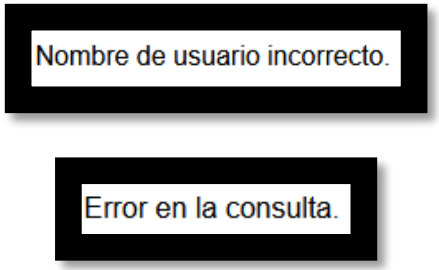


Seguridad de información. Inyección SQL. Ejemplos

Ejemplo 1: Acceder a la aplicación sin tener nombre de usuario ni contraseña

- Para comprobar si la aplicación incluye los datos enviados por el usuario sin ningún tratamiento previo, podemos enviar una comilla (simple o doble) como dato.

- El resultado puede ser:



Seguridad de información. Inyección SQL. Ejemplos

Ejemplo 1: Acceder a la aplicación sin tener nombre de usuario ni contraseña

- Este mensaje ("Error en la consulta") significa que en la aplicación no se tratan los datos y que además las consultas están delimitadas por dobles comillas.

- Veremos a continuación el código de la aplicación:

Seguridad de información. Inyección SQL. Ejemplos

Ejemplo 1: Acceder a la aplicación sin tener nombre de usuario ni contraseña

```
$usuario    = $_REQUEST["usuario"];
$contraseña = $_REQUEST["contraseña"];

$consulta = "SELECT COUNT(*) FROM $dbTabla
    WHERE campo1='$usuario'
    AND campo2='$contraseña'";
$result = sqlite_exec($db, $consulta);
if (!$result) {
    print "<p>Error en la consulta.</p>\n";
} elseif ($result[0][0] > 0) {
    print "<p>Nombre de usuario y contraseña correctos.</p>\n";
} else {
 ...
```

Seguridad de información. Inyección SQL. Ejemplos

Ejemplo 1: Acceder a la aplicación sin tener nombre de usuario ni contraseña

- Al escribir una comilla doble al principio del nombre de usuario, la consulta se convierte en…

```
SELECT COUNT(*) FROM $dbTabla
WHERE campo1='"hola'
AND campo2='hola'
```

- La consulta es correcta y el resultado el 0

Seguridad de información. Inyección SQL. Ejemplos

Ejemplo 1: Acceder a la aplicación sin tener nombre de usuario ni contraseña

- ¡OJO! Si escribimos una comilla simple al principio del nombre de usuario, la consulta se convierte en

```
SELECT COUNT(*) FROM $dbTabla
WHERE campo1=''hola'
AND campo2='hola'
```

- Esta consulta no es correcta y, cuando se ejecuta, la base de datos da error.

Seguridad de información. Inyección SQL. Ejemplos

Ejemplo 1: Acceder a la aplicación sin tener nombre de usuario ni contraseña

- Podemos modificar la consulta para que la aplicación crea que hemos introducido datos de un usuario registrado.
- El resultado será:

```
SELECT COUNT(*) FROM tabla
WHERE campo1='hola'
AND campo2='hola' OR '1'='1'
```

- Devuelve el número total de registros en la tabla

Actividad 5

Pon varios ejemplos de donde consideres que un ataque de inyección SQL podría ser una brecha de seguridad muy grave

Seguridad de información. Inyección SQL. Ejemplos

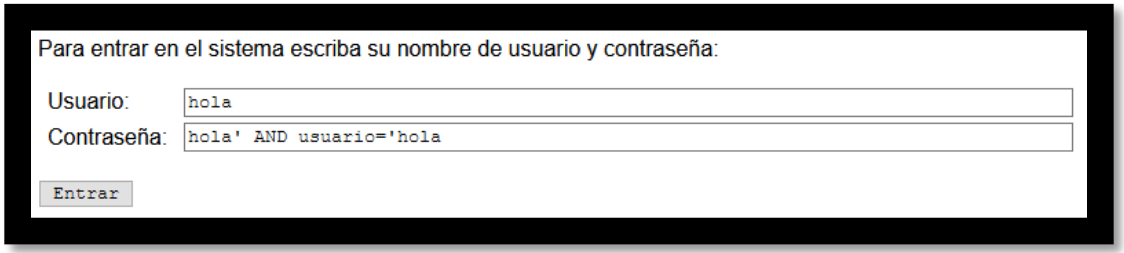
Ejemplo 2: Averiguar el nombre de los campos

- Los nombres de campos se pueden averiguar mediante prueba y error.
- Introducimos datos que construyan consultas en las que aparezcan posibles nombres de los campos.
- Si da error, el campo es incorrecto; si no, hemos acertado!

Seguridad de información. Inyección SQL. Ejemplos

Ejemplo 2: Averiguar el nombre de los campos

- Por ejemplo, vamos a probar si el nombre de uno de los campos es "usuario".



Seguridad de información. Inyección SQL. Ejemplos

Ejemplo 2: Averiguar el nombre de los campos

- La respuesta de la aplicación es "Error en la consulta", lo que nos indica que no hay un campo que se llame "usuario".

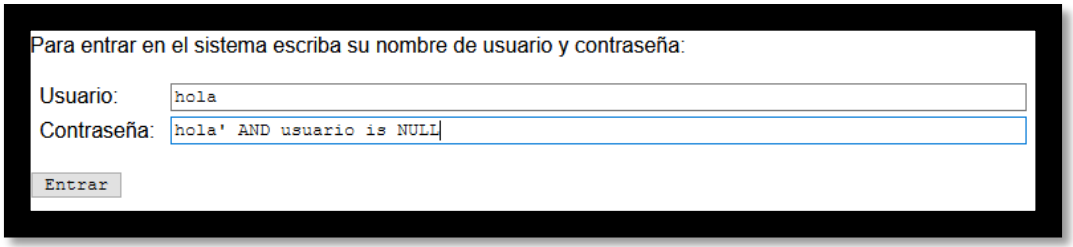
- La consulta a la base de datos habrá sido algo parecido a esto:

```
SELECT COUNT(*) FROM tabla
WHERE campo1='hola'
AND campo2='hola' AND usuario='hola'
```

Seguridad de información. Inyección SQL. Ejemplos

Ejemplo 2: Averiguar el nombre de los campos

- Podríamos probar:



Seguridad de información. Inyección SQL. Ejemplos

Ejemplo 2: Averiguar el nombre de los campos

- La respuesta de la aplicación es "Error en la consulta", lo que nos indica que no hay un campo que se llame "usuario".

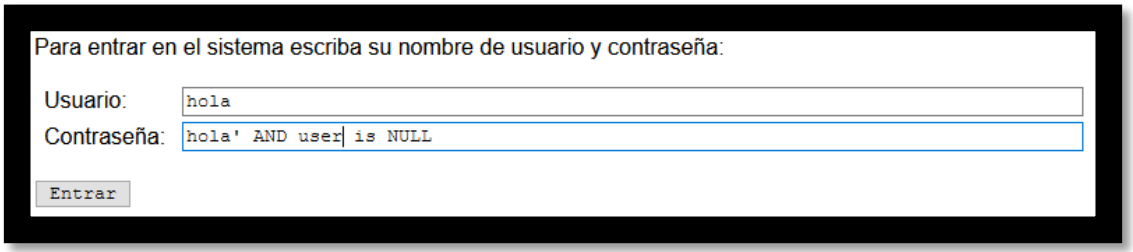
- La consulta a la base de datos habrá sido algo parecido a esto:

```
SELECT COUNT(*) FROM tabla
WHERE campo1='hola'
AND campo2='hola' AND usuario is NULL; --'
```

Seguridad de información. Inyección SQL. Ejemplos

Ejemplo 2: Averiguar el nombre de los campos

- Haremos un tercer intento, con el campo “user”



Seguridad de información. Inyección SQL. Ejemplos

Ejemplo 2: Averiguar el nombre de los campos

- La respuesta de la aplicación es:

```
Nombre de usuario incorrecto.
```

- Sabemos que uno de los campos se llama **user**

- La consulta a la base de datos habrá sido algo parecido a esto:

```
SELECT COUNT(*) FROM tabla
WHERE campo1='hola'
AND campo2='hola' AND user is NULL; --'
```

Actividad 6

Si nos ponemos en la piel de un hacker:

¿Es posible averiguar el nombre de todos los campos de una tabla?

¿Cuánto tardaríamos?

¿Sería eficiente?

Seguridad de información. Inyección SQL. Ejemplos

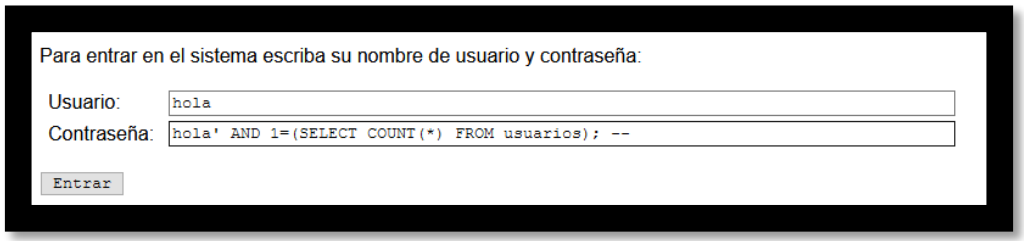
Ejemplo 3: Averiguar los nombres de las tablas

- Los nombres de las tablas se pueden averiguar mediante prueba y error.
- La idea es introducir datos que construyan consultas en las que aparezcan posibles nombres de las tablas

Seguridad de información. Inyección SQL. Ejemplos

Ejemplo 3: Averiguar los nombres de las tablas

- Por ejemplo, vamos a probar si el nombre de la tabla es "usuarios".



Seguridad de información. Inyección SQL. Ejemplos

Ejemplo 3: Averiguar los nombres de las tablas

- La respuesta de la aplicación es "Error en la consulta", lo que nos indica que no hay una tabla que se llame "usuarios".

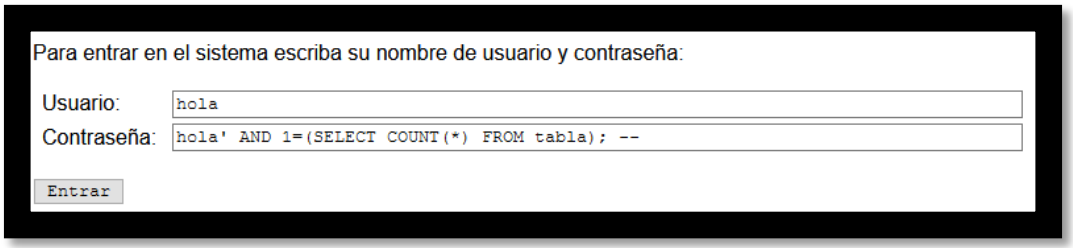
- La consulta a la base de datos habrá sido algo parecido a esto:

```
SELECT COUNT(*) FROM tabla
WHERE campo1='hola'
AND campo2='hola' AND 1=(SELECT COUNT(*) FROM usuarios); --'
```

Seguridad de información. Inyección SQL. Ejemplos

Ejemplo 3: Averiguar los nombres de las tablas

- Hacemos ahora un segundo intento, con el nombre "tabla“:



Seguridad de información. Inyección SQL. Ejemplos

Ejemplo 3: Averiguar los nombres de las tablas

- La respuesta de la aplicación es:

```
Nombre de usuario incorrecto.
```

- Sabemos que una de las tablas se llama **tabla**

- La consulta a la base de datos habrá sido algo parecido a esto:

```
SELECT COUNT(*) FROM tabla
WHERE campo1='hola'
AND campo2='hola' AND 1=(SELECT COUNT(*) FROM tabla);--
```

Seguridad de información. Inyección SQL. Ejemplos

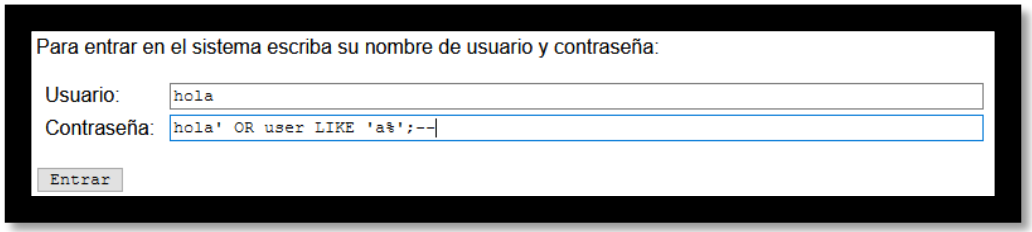
Ejemplo 4: Averiguar el contenido de los registros

- Conociendo la tabla de usuarios y los campos, se pueden intentar sacar valores concretos.
- La idea es introducir datos que construyan consultas en las que aparezcan posibles contenidos de los campos.

Seguridad de información. Inyección SQL. Ejemplos

Ejemplo 4: Averiguar el contenido de los registros

- Por ejemplo, vamos a buscar nombres de usuarios



Seguridad de información. Inyección SQL. Ejemplos

Ejemplo 4: Averiguar el contenido de los registros

- La respuesta es "Nombre de usuario y contraseña correctos.", lo que nos indica que hay un usuario cuyo nombre empieza por "a".

- La consulta a la base de datos habrá sido algo parecido a esto:

```
SELECT COUNT(*) FROM tabla
WHERE campo1='hola'
AND campo2='hola' OR user LIKE 'a%';--
```

Seguridad de información. Inyección SQL. Ejemplos

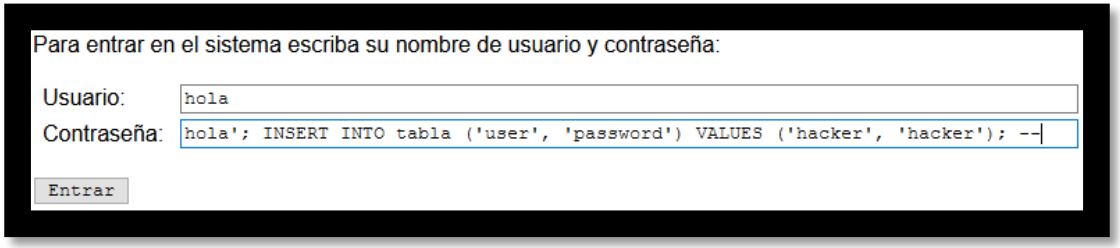
Ejemplo 5: Añadir un nuevo usuario

- Una vez se conoce el nombre de la tabla de usuarios y los nombres de los campos se puede intentar editar la base de datos, por ejemplo, añadiendo un usuario.

Seguridad de información. Inyección SQL. Ejemplos

Ejemplo 5: Añadir un nuevo usuario

- La técnica consiste en incluir una sentencia SQL que inserte un registro.



Seguridad de información. Inyección SQL. Ejemplos

Ejemplo 5: Añadir un nuevo usuario

- La consulta a la base de datos habrá sido algo parecido a esto:

```
SELECT COUNT(*) FROM tabla
WHERE campo1='hola'
AND campo2='hola'; INSERT INTO tabla VALUES (NULL, 'hacker', 'hacker'); --
```

- Para comprobar si el ataque ha tenido éxito, habría que probar a entrar como usuario "hacker" con contraseña "hacker".

Seguridad de información. Inyección SQL. Ejemplos

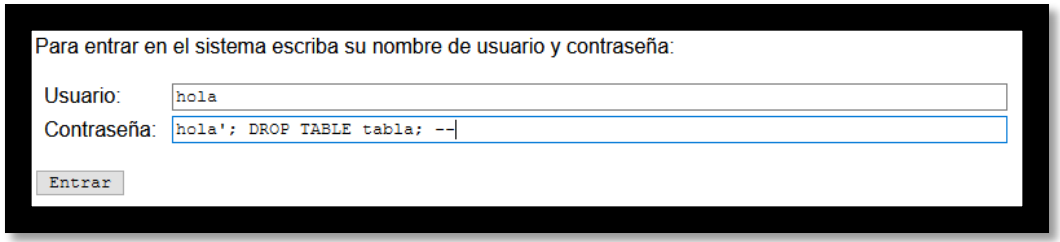
Ejemplo 6: Borrar una tabla

- Una vez se conoce el nombre de la tabla de usuarios vamos a realizar una acción destructiva, como por ejemplo borrar la tabla de usuarios.

Seguridad de información. Inyección SQL. Ejemplos

Ejemplo 6: Borrar una tabla

- La técnica consiste en incluir una sentencia SQL como esta.



Seguridad de información. Inyección SQL. Ejemplos

Ejemplo 6: Borrar una tabla

- La consulta a la base de datos habrá sido algo parecido a esto:

```
SELECT COUNT(*) FROM tabla
WHERE campo1='hola'
AND campo2='hola'; DROP TABLE tabla; --
```

- Si el ataque ha tenido éxito, la aplicación seguramente dejará de funcionar, puesto que ha desaparecido una de las tablas.

Seguridad de información. Inyección SQL. Ejemplos

Vídeo demostración

https://www.youtube.com/watch?v=qzE07kZ6CHk

Actividad 7

De todos los posibles ataques que hemos visto, ¿cuál os parece el más peligroso?

Seguridad de información. Cross-Site Scripting (XSS)

- Permite al atacante introducir y ejecutar en la página su propio código en lenguaje de script (generalmente Javascript).

Seguridad de información. Cross-Site Scripting (XSS)

- Por ejemplo, si en una página web dejamos a los usuarios introducir comentarios, y no comprobamos su contenido, alguno podría introducir la siguiente cadena de texto como comentario:

```
<script> window.location = "http://www.google.es"; </script>
```

Seguridad de información. Cross-Site Scripting (XSS)

- El problema viene cuando mostramos esa información en pantalla.
- Se publicará el comentario malicioso, se ejecutará el código Javascript y el navegador redireccionará al usuario a la otra página.

Seguridad de información. Cross-Site Scripting (XSS)

- Esta vulnerabilidad puede darse al mostrar contenido no verificado:
    - Como parte del cuerpo de la página web.
    - Como valor de un atributo de un elemento HTML.
    - Como parte de una hoja de estilos.

Seguridad de información. Cross-Site Scripting (XSS)

- Esta vulnerabilidad puede darse al mostrar contenido no verificado:
    - Como parte de una URL (en los enlaces o en las redirecciones).
    - Como parte de un código en lenguaje de script generado dinámicamente.

Seguridad de información. Cross-Site Scripting (XSS). Ejemplo

- Supongamos por ejemplo el siguiente formulario:

```
<form action="ejercicio_ejemplo.php" method="post">
    <input type="text" name="comment" value="">
    <input type="submit" name="submit" value="Submit">
<form>
```

- Supongamos que lo único que se hará con los datos en el destino será mostrarlos con un echo:

```
echo $_POST['comment'];
```

Seguridad de información. Cross-Site Scripting (XSS). Ejemplo

- Sin ningún tipo de filtrado, el atacante puede enviar el siguiente script a través del formulario, lo que generará un popup en el navegador con el mensaje "Hackeado":

```
<script>alert("hacked")</script>
```

Seguridad de información. Cross-Site Scripting (XSS). Ejemplo para evitar ataques XSS

```php
// Validar el comentario
$comentario = trim($_POST["comentario"]);
if(empty($comentario)){
    exit("Debes proporcionar un comentario");
}
// Sanitizar comentario
$comentario = strip_tags($comentario);
// El comentario ya se puede guardar de forma segura
file_put_contents("comentarios.txt", $comentario, FILE_APPEND);
// Escapar comentarios antes de mostrarlos
$comentarios = file_get_contents("comentarios.txt");
echo htmlspecialchars($comentarios);
```

Seguridad de información. Cross-Site Scripting (XSS). Ejemplo para evitar ataques XSS

- El procedimiento a seguir será:
    - Primero nos aseguramos de que no se guardan comentarios vacíos.
    - Después se sanitizan los datos eliminando cualquier posible etiqueta HTML
    - Los comentarios se devuelven filtrados. La función strip_tags hace que no sea posible insertar enlaces en los comentarios, ya que éstos utilizan una etiqueta que será eliminada. Para que puedan insertarse se puede utilizar htmlentities o htmlspecialchars en su lugar.

Actividad 8

Aplica las medidas que hemos visto para evitar un posible ataque XSS en nuestra Actividad 4

Saneamiento de la información en una base de datos

- El objetivo en este caso es impedir ataques de inyección SQL.

- Los caracteres que no pueden ser admitidos en las consultas como parte de los valores introducidos por los usuarios son, principalmente:
  - Comilla simple (')
  - Comilla doble (")
  - Punto y coma (;)
  - La barra inversa (\)

Saneamiento de la información de una página web

- Dependiendo de la parte de la página en la que vayamos a introducir el contenido, tendremos que sanear la información de una forma o de otra:
    - Contenido publicado en el cuerpo de una página web.
    - Contenido publicado como parte de una URL
    - Contenido pasado como valor a una variable de un lenguaje de script

Saneamiento de la información de una página web.

- Por el contenido publicado en el cuerpo de una página web:
    - Tenemos que vigilar la presencia en el texto de etiquetas HTML como por ejemplo "<script>".

      Tenemos varias alternativas en función de lo que busquemos:
        - Eliminar completamente las etiquetas HTML presentes en el texto. Función "**strip_tags**".

```
$salida = strip_tags($texto);
```

Saneamiento de la información de una página web.

- Por el contenido publicado en el cuerpo de una página web:
  - Tenemos que vigilar la presencia en el texto de etiquetas HTML como por ejemplo "<script>".

    Tenemos varias alternativas en función de lo que busquemos:
    - Convertir las etiquetas presentes en el texto a sus respectivas entidades HTML, para que puedan visualizarse correctamente pero no formen parte de la estructura de la página.

```
$salida = htmlentities($texto);
```

Saneamiento de la información de una página web.

- Por el contenido publicado en el cuerpo de una página web:
  - Contenido publicado como parte de una URL.
    - A veces es necesario emplear contenido de origen ajeno como parte de una URL. Por ejemplo, si queremos hacer una búsqueda en Google de un término introducido por el usuario, podemos hacerlo con:

```
$url = "http://www.google.es/search?q=".$_POST["término_búsqueda"];
```

Saneamiento de la información de una página web.

- Por el contenido publicado en el cuerpo de una página web:
  - Contenido publicado como parte de una URL.
    - Para asegurar que la URL es válida y se ajusta a nuestros requerimientos, debemos filtrar la cadena. En PHP la función "rawurlencode" se encarga de la dicha transformación::

```
$url = "http://www.google.es/search? q=".rawurlencode($_POST["término"]);
```

Saneamiento de la información de una página web.

- Por el contenido publicado en el cuerpo de una página web:
  - Contenido pasado como valor a una variable de un lenguaje de script.
    - En ocasiones necesitamos insertar código en la página web que se envía al navegador.
    - Por ejemplo, si queremos añadir algún tipo de validación en el lado cliente antes de enviar un formulario, podría añadir un atributo a la etiqueta "<form>".

```
<form onsubmit="return validar(this);">
```

Actividad 9

Veremos este vídeo acerca de la Inyección SQL

https://www.youtube.com/watch?v=6TXP90OlNRA

Tema 4 : Formularios

Ciclo Superior DAW

Asignatura: Desarrollo web en entorno servidor

Curso 20/21