UNIDAD 6.- SESIONES

Este tema consta de tres partes:
En la primera parte del tema veremos el funcionamiento general de las sesiones en php.
En la segunda parte la configuración de php.ini
En la tercera parte, seguridad con sesiones.

Funcionamiento General de las Sesiones en PHP.

Introducción

Las sesiones son un mecanismo que utilizan los servidores web para guardar información en el servidor sobre el usuario y su actividad y recuperarla cada vez que el navegador les pide una página. Permite compartir información entre una página y otra de manera segura. De esta manera no hay

La diferencia con respecto a las cookies es que con cookies la información se guarda en el navegador del usuario y se envía en cada petición al servidor, mientras que mediante sesiones la información se guarda en el servidor. En el manual de PHP se ofrece un capítulo dedicado a las sesiones.

El trabajo con sesiones tiene tres partes:

- **Creación o apertura de la sesión**

  Cuando alguna página crea una sesión utilizando la función correspondiente, el servidor asocia al navegador del usuario un identificador de usuario único. El identificador se guarda en el usuario en forma de cookie o, si no se permite la creación de cookie, añadiendo el identificador en la dirección de la página, esta última forma es mucho menos segura.

- **Utilización de la sesión**

  Una vez creada la sesión, las páginas solicitadas por el mismo navegador pueden guardar y recuperar información en el servidor, información que se asocia al identificador de usuario, por lo que no es accesible a otros. La información se conserva hasta que el usuario o el servidor destruyan la sesión.

- **Destrucción o cierre de la sesión**

  Tanto el usuario como el servidor pueden cerrar la sesión. El usuario puede destruir la sesión cerrando el navegador. El servidor puede destruir la sesión cuando alguna página utilice la función correspondiente.

Crear la sesión

En PHP, las sesiones se crean mediante la función session_start(). Si la sesión no existía, esta función crea la sesión y le asocia un identificador de sesión único. Si la sesión ya existía, esta función permite que la página tenga acceso a la información vinculada a la sesión.

Tenemos que incluir sesión_start() en todas las páginas que vayan a hacer uso de las sesiones.

Por ejemplo:

```
<?php
session_start();
?>
```

Como en el caso de las cookies, hay que **tener la precaución de utilizar la función session_start() antes de empezar a escribir el contenido de la página**, **porque si no PHP producirá un aviso y no se creará la sesión**. El motivo es que el idetificador de sesión se utiliza en las cabeceras de respuesta HTTP y las cabeceras se envían antes del texto de la página. Es decir, cuando PHP encuentra una instrucción que escribe texto, cierra automáticamente la cabecera; si a continuación PHP encuentra en el programa la función session_start(), da un aviso porque ya se han enviado las cabeceras y no se crea la sesión.

Utilizar la sesión

Cuando una página ha creado una sesión o ha accedido a una sesión ya existente mediante session_start(), la página tiene acceso a la matriz $_SESSION que contiene las variables de esa sesión.

La matriz $_SESSION es una matriz asociativa en la que se pueden definir valores como en cualquier otra matriz. **La diferencia es que $_SESSION es accesible desde páginas diferentes (siempre que esas páginas tengan asociada la misma sesión), manteniéndose los valores de una página a otra.**

El ejemplo siguiente muestra dos páginas. La primera página guarda información en $_SESSION y la segunda la utiliza.

`<?php`
`session_start();`
`$_SESSION["nombre"] = "Pepe Pérez";`
`print "<p>Se ha guardado su nombre.</p>";`
`?>`
	Se ha guardado su nombre.

`<?php`
`session_start();`
`print "<p>Su nombre es $_SESSION[nombre].</p>";`
`?>`
	Su nombre es Pepe Pérez.

Los nombres de los índices de la matriz $_SESSION tienen que cumplir las mismas reglas que los nombres de las variables, es decir, que el primer carácter debe ser una letra o un guión bajo (_).

Los valores de $_SESSION se borran como en cualquier otra matriz mediante la función unset().

`<?php`
`session_start();`
 	
`if (isset($_SESSION["nombre"])) {`
`  print "<p>Su nombre es $_SESSION[nombre].</p>";`
`} else {`
`  print "<p>No sé su nombre.</p>";`
`}`
 	
`unset($_SESSION["nombre"]);`
 	
`if (isset($_SESSION["nombre"])) {`
`  print "<p>Su nombre es $_SESSION[nombre].</p>";`
`} else {`
	Su nombre es Pepe Pérez.
	
	 No sé su nombre.

```
    print "<p>No sé su nombre.</p>";
}

?>
```

Un detalle importante que distingue a las sesiones de las cookies es que:

- En el caso de las cookies, cuando una página pide al navegador que cree una cookie, el valor de la cookie no está disponible en $_COOKIE en esa página. Lo estará en páginas posteriores, cuando el navegador las pide otra página y envíe el valor en la petición.
- En el caso de las sesiones, cuando una página crea un valor en $_SESSION, ese valor está disponible en ese mismo página.

Es decir, que el siguiente programa siempre producirá el mismo resultado:

```
<?php
session_start();
$_SESSION["nombre"] = "Pepe Pérez";

if (isset($_SESSION["nombre"])) {
    print "<p>Su nombre es $_SESSION[nombre].</p>";
} else {
    print "<p>No sé su nombre.</p>";
}

unset($_SESSION["nombre"]);

if (isset($_SESSION["nombre"])) {
    print "<p>Su nombre es $_SESSION[nombre].</p>";
} else {
    print "<p>No sé su nombre.</p>";
}

?>
```

```
<p>Su nombre es Pepe Pérez.</p>
<p>No sé su nombre.</p>
```

Destruir la sesión

El usuario puede destruir la sesión cerrando el navegador, pero las sesiones también se pueden destruir por programa mediante la función session_destroy().

Cuando se destruye una sesión, **el script que ha destruido la sesión sigue teniendo acceso a los valores de $_SESSION creados antes de la destrucción de la sesión, pero las páginas siguientes no.** Si se ejecuta el primero de los ejemplos siguientes y después el segundo, se obtienen los resultados indicados:

```
<?php
session_start();
$_SESSION["nombre"] = "Pepe Pérez";
session_destroy();

if (isset($_SESSION["nombre"])) {
    print "<p>Su nombre es $_SESSION[nombre].</p>";
} else {
    print "<p>No sé su nombre.</p>";
}
?>
```

```
<p>Su nombre es Pepe Pérez.</p>
```

```php
<?php
session_start();

if (isset($_SESSION["nombre"])) {
    print "<p>Su nombre es $_SESSION[nombre].</p>";
} else {
    print "<p>No sé su nombre.</p>";
}
?>
```

Hay que tener en cuenta que **sesión_destroy** no destruye la cookie del navegador, para eso tendríamos que hacerlo específicamente.

```php
//Si se desea destruir la sesión completamente, borre también la cookie de sesión.
//Nota:¡Esto destruirá la sesión, y no la información de la sesión!

if (ini_get("session.use_cookies")) {
//session_get_cookie_params    Obtener los parámetros de la cookie de sesión
    $params = session_get_cookie_params();
    setcookie(session_name(), '', time() - 42000,
        $params["path"], $params["domain"],
        $params["secure"], $params["httponly"]
    );
}
```

Configuración Sesiones en php.ini

Referente a las sesiones vemos que:

1. Las sesiones tienen un identificador que se guarda, por defecto, en una **cookie** que recibe el navegador la primera vez que se crea la sesión y que viaja con cada página a partir de ese momento para identificar esa sesión.
2. Para acceder a la información almacenada en la sesión se utiliza el array superglobal $_SESSION.

Vamos a ver más detalles referentes a la forma de identificar las sesiones. Por ejemplo, veamos el significado de algunas directivas de configuración que aparecen en el archivo php.ini respecto a la cookie de sesión:

- **session.save_path**: aquí se especifica la ubicación donde se almacenará la información de sesión. Después veremos que, inicialmente, esa información se almacena como un archivo de texto por sesión, pues aquí se indicaría dónde se guardan esos archivos.

Es importante que la ubicación indicada en esta directiva esté fuera del espacio de documentos web del sitio, de forma que ningún usuario pueda acceder a él a través de su navegador.

- session.name: aquí indicaremos el nombre de la cookie que se utiliza para identificar las sesiones. Por defecto, el nombre es PHPSESSID y la verdad es que no tiene sentido cambiarlo excepto en el caso de que tengamos más de un sitio web en el mismo servidor y necesitemos diferenciar las sesiones de cada uno de ellos.

- session.autostart: si establecemos la directiva session.autostart al valor 1, entonces se iniciará una sesión automáticamente al solicitar cualquier página php del sitio web. Es como si incluyéramos una llamada a la función session_start al principio de cada página. No es recomendable.

- **session.cookie*httponly* = *On*.** Rechaza el acceso a la cookie de sesión vía JavaScript. Esto previene el robo de cookies a través de JavaScript injection.

- **session.cookie_lifetime**: aquí indicamos el período de vigencia de la cookie que identifica la sesión. Si se establece al valor 0, entonces se tratará de una cookie que sólo se almacena en la memoria del navegador; mientras que un valor distinto haría que se almacenara en el disco duro del usuario durante su vigencia. Así pues, si se cierra el navegador, la session ID se borra inmediatamente. Se desaconseja variar este valor.

Recuerde que si la cookie se almacena en la memoria del navegador, se eliminará al cerrarlo.

- **session.cookie_path**: aquí indicamos para qué directorios será válida la cookie que identifica la sesión. Por ejemplo, podríamos desear utilizar la información de sesión sólo en un subdirectorio y no en todas las páginas del sitio web, como es el valor predeterminado (/).

_session.hash*function* = *"sha256"*. Una función hash más fuerte generará una session ID más fuerte. Cuanto más complejo sea el cifrado mejor: sha384, sha512...

- **session.use_cookies**: aquí se indica si se podrán utilizar cookies para identificar la sesión. El valor 1 indica que sí, mientras que el valor 0 impediría que se utilizaran cookies para este propósito.

Pero, ¿qué ocurre si el usuario ha configurado su navegador para no permitir las cookies?

Esto depende de la configuración de la directiva **session.use_trans_sid.** Si se establece a 1, entonces PHP busca otra forma de identificar la sesión, que es pasándola a través de la URL de la página o como un campo oculto de un formulario.

Por defecto la configuración de esta directiva es que está desactivada, pero si la activamos, PHP pasará mediante Get o Post el identificador de la sesión:

En los hipervínculos añade el identificador de la sesión, de forma que éste se envía junto a la URL de la página de destino.

Si lo que hacemos es enviar la información mediante un formulario utilizando el método Post, utilizará un campo oculto para pasar el identificador de sesión.

Recuerda que PHP utilizará estos métodos alternativos para identificar la sesión sólo cuando no se permitan las cookies y si hemos habilitado la directiva **session.use_trans_sid**.

Primero siempre intenta utilizar cookies, ya que es un método mucho más seguro que pasar el identificador de la sesión bien mediante la URL de la página (Get) o mediante un campo oculto en un formulario (Post).

Sopesar muy seriamente habilitar o no la directiva session.use_trans_sid:

Si no la habilita, será necesario que los usuarios acepten sus cookies.

Si la habilitas, podrás utilizar la información de la sesión incluso aunque el usuario no acepte cookies, pero representa un riesgo adicional en cuanto a la seguridad porque el identificador de la sesión está visible.

Dónde se almacena la información

Dónde se almacena físicamente la información de las distintas sesiones que se establecen.

Tener en cuenta que un sitio web concurrido manejará muchas sesiones simultáneamente, por lo que es importante poder identificarlas de forma que el usuario reciba el contenido web a partir de la información particular que para su sesión se ha guardado en el servidor.

La forma en que el servidor web guarda la información de sesión viene dada por el valor de la directiva **session.save_handler** del archivo php.ini. Esta directiva puede tomar tres valores:

- files: es el valor predeterminado e indica que la información de sesión se almacenará en forma de archivos.

- mm: indica que se guardará en la memoria RAM del servidor.
- user: permite establecer una forma de almacenar la información “personalizada”. Por ejemplo, en una base de datos.

Adicionalmente, la directiva **session.save_path** indica la ubicación donde se almacenarán los archivos si utilizamos el método predeterminado. El paquete XAMPP configura la ubicación c:\apachefriends\xampp\tmp.

Podemos ver esos archivos de sesión. Se trata de archivos de texto planos que se irán eliminando de forma más o menos rápida en función de si utilizamos la función **session_destroy** o esperamos a que PHP los elimine para las sesiones caducadas.

Para configurar la eliminación de sesiones caducadas podemos configurar varios parámetros de php.ini.

- La directiva **session.gc_maxlifetime configura a partir de que momento (en segundos después de cerrar el navegador, si cerramos sesión con session_destroy,** el fichero desaparecerá en el momento de cerrar sesión**) el archivo creado por la sesión es considerado “basura” para poder eliminarlo. Este fichero de texto lo eliminará el servidor la siguiente vez que se abra una sesión pero hay que configurar cada cuantas sesiones se elimina basura (ficheros de sesiones caducadas), aquí intervienen los siguientes parámetros.**
- Las directivas **session.gc_probability** y **session.gc_dividend**, del archivo php.ini, establecen la probabilidad (en porcentaje) de eliminar los archivos de sesiones caducadas o finalizadas cada vez que se crea una nueva sesión. Por ejemplo **session.gc_probability=1 y session.gc_dividend=1000** quiere decir que se eliminará con una probabilidad de 1 sobre 1000 inicios de sesión, esta configuración dependerá del uso del servidor intentando no sobrecargarlo.

Como puedes ver, es importante que esta ubicación esté debidamente protegida en el servidor web porque aquí es donde se almacenará la información manejada en las distintas sesiones de los usuarios que acceden a nuestro sitio web.

En ocasiones, los usuarios acceden a Internet compartiendo una conexión.

Un proxy permite compartir una conexión por varios usuarios, que no se dan cuenta de esta circunstancia (excepto, tal vez por la velocidad de acceso), ya que acceden a Internet normalmente.

El proxy puede almacenar páginas visitadas en su propia memoria caché y proporcionarlas a otros usuarios que las soliciten posteriormente, de forma que el acceso a una misma página sea más rápido, sin necesidad de volver a cargarla ni de pedirla al servidor web original. Aquí es donde puede producirse un problema: que un usuario obtenga una página dinámica correspondiente a la petición (sesión) de otro usuario.

Para evitar esta circunstancia, podemos utilizar la directiva **session.cache_limiter** en el archivo php.ini o la función session_cache_limiter en las páginas que así lo nece- sitemos.

En cualquier caso, deberíamos establecer este detalle al valor nocache para que las páginas no se almacenaran en la caché del proxy:

session.cache_limiter = nocache //archivo php.ini o session_cache_limiter ("nocache"); //Utilícela antes //de session_start().

Seguridad con Sesiones en PHP.

Session Hijacking (secuestro o robo de sesión)

S usuario, de forma que puede hacerse pasar por este y acceder a su cuenta en esa página web.

El robo de la sesión puede conseguirse de varias formas, aunque nos centraremos en las que tienen que ver con las vulnerabilidades de las sesiones y en algunas técnicas para mitigarlas.

La única forma que tiene la página web de reconocer a un usuario es por medio de su identificador de sesión. Si un atacante consigue el identificador de sesión de un usuario que ya está autenticado, puede hacerse pasar por él y entrar en su cuenta sólo con hacer que su navegador envíe el identificador a la página web, ya sea a través de la URL o de una cookie.

A continuación veremos algunas de las formas en las que un atacante puede robar este identificador y técnicas para intentar prevenirlo.

Ataque por fuerza bruta

El ataque por fuerza bruta significa probar identificadores aleatoriamente hasta encontrar uno que esté siendo usado. Es como intentar abrir una caja fuerte sin saber la combinación, poniendo números al azar.

Como en el caso de la caja fuerte, cuantos más números tenga la combinación (en este caso el identificador de sesión), más difícil será de adivinar. También ayuda el hecho de que el número o identificador sea aleatorio, y no algo que se pueda predecir. El sistema de identificadores de sesión de PHP es aceptable en este sentido.

Robo por sniffing

Este tipo de ataque se da cuando el atacante tiene un programa de sniffing en la red del usuario y puede interceptar el tráfico destinado al mismo, incluido su identificador de sesión.

La única forma de prevenir estos ataques es utilizando cifrado HTTPS en toda la página web.

Propagación en URL

Si el identificador de sesión se propaga utilizando la URL en lugar de las cookies, cualquier atacante puede robarlo desde muchos sitios:

- Un enlace que el propio usuario ponga en un lugar público. Los usuarios típicos no saben para que sirve ese identificador y no le dan importancia.
- El historial del navegador.
- El referrer, que es un encabezado que envían muchos navegadores a las páginas web en el que les indican la URL de la que vienen.

La forma de prevenir esto es no utilizar la URL para el identificador de sesión; utilizar únicamente las cookies. En PHP esto se consigue con la instrucción:

```
ini_set('session.use_only_cookies', 1);
```

Robo en servidor compartido

Si tenemos nuestra página web alojada en un servidor compartido, los archivos físicos de las sesiones se guardan, por defecto, en un directorio común para todas las páginas web del servidor. Esto quiere decir que todas las personas que tengan su página web en ese mismo servidor, tienen acceso a todos los archivos de sesiones. Dado que el nombre de los archivos es "sess_" más el

identificador de sesión, cualquier atacante tendrá una lista de identificadores de sesión válidos con sólo leer la lista de archivos del directorio común.

Esta vulnerabilidad se puede combatir:

- Usando la función `session_save_path` para guardar los archivos de sesión de la página web en un directorio dentro de su cuenta al que sólo pueda acceder PHP (ya sea por estar fuera del directorio web o con un `.htaccess` con la instrucción `deny from all`). Este método no es demasiado fiable, ya que el resto de usuarios seguirá pudiendo leer en ese directorio, sólo tienen que averiguar su localización.

Robo por Cross-Site Scripting

Si la página web es vulnerable a XSS el atacante puede insertar un código javascript que envíe las cookies de un usuario a su cuenta.

Este tipo de ataque se puede prevenir (además de evitando los ataques XSS) haciendo que las cookies de sesión tengan el atributo `HttpOnly`, que evita que puedan ser manejadas por javascript en la mayoría de navegadores. En PHP esto se consigue con la instrucción:

```
ini_set('session.cookie_httponly', 1);
```

Métodos de prevención generales

Además de los métodos de prevención concretos vistos para cada tipo de ataque, vamos a ver algunas técnicas de prevención que ayudan a evitar el robo de las sesiones:

- **Limitar tiempo de inactividad:** eliminar la sesión si está cierto tiempo sin ser usada (de 5 a 30 minutos, según el nivel de seguridad de la página web).
- **Cambiar el identificador de sesión:** cada cierto tiempo o después de cada acción, cambiar el identificador de la sesión por otro distinto y eliminar la sesión antigua.
- **Sistema de logout:** dar a los usuarios una forma de salir de su cuenta y destruir la sesión.
- **Cambiar el nombre de la cookie de sesión.**