

UD-05a: Seguridad y Control de Acceso

Desarrollo Web en Entorno Servidor

Curso 2020/2021

Autenticación de usuarios y control de acceso

- Existen métodos para identificar tanto al servidor que aloja el sitio web, como al usuario (navegador) que se encuentra en el otro extremo.
- Los sitios web que necesitan emplear identificación del servidor, como las tiendas o los bancos, utilizan el protocolo HTTPS. Este protocolo requiere de un certificado válido, firmado por una autoridad confiable, que es verificado por el navegador cuando se accede al sitio web. Además, HTTPS usa métodos de cifrado para crear un canal seguro entre el navegador y el servidor, de tal forma que sea mucho más complicado interceptar la información que se transmite.
- Para identificar a los usuarios que visitan un sitio web, se pueden usar distintos métodos como el DNI digital o certificados digitales de usuario, pero el más extendido es solicitar nombre de usuario y contraseña.

Autenticación de usuarios y control de acceso

- A un usuario identificado en un sitio web, se le puede limitar la información a la que puede acceder. Por ejemplo, una web de gestión de una empresa puede tener un grupo de usuarios que pueden visualizar la información y otro grupo de usuarios que, además, también la pueden modificar. Otro ejemplo son las webs de bancos, que permiten a sus clientes acceder únicamente a la información de sus cuentas.
- En esta unidad, vamos a enviar la información de autenticación (usuario y contraseña) en texto plano desde navegador hasta el servidor web. Esta práctica es altamente insegura y nunca debe usarse sin un protocolo como HTTPS (que permite cifrar las comunicaciones con el servidor web).
- La configuración de servidores web que permitan usar HTTPS no forma parte de los contenidos de este módulo. Por este motivo, durante esta unidad usaremos únicamente el protocolo HTTP.

Autenticación de usuarios y control de acceso

Autenticación por HTTP

- El servidor web debe proveer algún método para definir los usuarios que se usarán y cómo se pueden autenticar. Además, se tendrán que definir los recursos a los que se restringe el acceso y qué lista de control de acceso (**ACL**) se aplica a cada uno.
 - ✓ **ACL:** lista de permisos sobre un fichero, directorio, etc., que indica qué usuarios pueden usarlo y qué acciones pueden realizar con él (lectura, escritura, borrado).
- Cuando un usuario no autenticado intenta acceder a un recurso restringido, el servidor web responderá con un error de "Acceso no autorizado" (código 401). El navegador recibe el error y abre una ventana para solicitar al usuario que se autentique mediante su nombre y contraseña.
- La información de autenticación del usuario se envía al servidor, que la verifica y decide si permite o no el acceso al recurso solicitado. Esta información se mantiene en el navegador para usarse en posteriores peticiones a ese servidor.

Autenticación de usuarios y control de acceso

Autenticación por HTTP usando PHP

Desde PHP podemos acceder a la información de autenticación HTTP que ha introducido el usuario gracias al array superglobal `$_SERVER`.

Valor	Contenido
<code>\$_SERVER['PHP_AUTH_USER']</code>	Nombre de usuario que se ha introducido.
<code>\$_SERVER['PHP_AUTH_PW']</code>	Contraseña introducida.
<code>\$_SERVER['AUTH_TYPE']</code>	Método HTTP usado para autenticar. Puede ser Basic o Digest.

- **Basic** es un método diseñado para permitir a un cliente web, que solicita una página al servidor web, proveer credenciales de usuario y contraseña. No requiere el uso de cookies, identificadores de sesión o página de ingreso.
- **Digest** es usado para confirmar la identidad de un usuario antes de servir información sensible, como el historial de transacciones de un banco. Se aplica una función hash a la contraseña antes de ser enviada sobre la red, lo que resulta más seguro que enviarla en texto plano como en la autenticación *Basic*.

Autenticación de usuarios y control de acceso

Podemos usar la función `header` para forzar al servidor a enviar un error de "Acceso no autorizado" (código 401) y el navegador muestre un formulario de login.

Simplemente añadimos estas líneas al principio de nuestro código. Debe usarse antes de que se muestre nada por pantalla:

`<?php`

```
if (!isset($_SERVER['PHP_AUTH_USER']))
{
    header('WWW-Authenticate: Basic realm="Contenido restringido"');
    header('HTTP/1.0 401 Unauthorized');

    echo "Usuario no reconocido";
    exit; // exit = exit() = exit(0);
}
```

`?>`

Realm → ámbito de aplicación: las páginas del mismo ámbito comparten credenciales. Si las credenciales funcionan para una página con el dominio "Contenido restringido", la misma combinación de nombre de usuario y contraseña deben funcionar para cualquier página con el mismo dominio.

EJEMPLO

Forzamos al navegador a solicitar credenciales y que siempre otorgue acceso, es decir, no comprobaremos que el nombre de usuario y contraseña son válidos.

```
<!doctype html>
```

```
<?php
```

```
if (!isset($_SERVER['PHP_AUTH_USER']))
{
    header('WWW-Authenticate: Basic Realm="Contenido restringido"');
    header('HTTP/1.0 401 Unauthorized');

    echo "Usuario no reconocido";
    exit; // exit = exit() = exit(0);
}
```

```
?>
```

```
<html lang="es">
```

```
<head>
```

```
<meta charset="utf-8">
```

```
<title>Desarrollo web en entorno servidor - ud 5</title>
```

```
</head>
```

```
<body>
```

```
<?php
```

```
echo "Nombre de usuario: ".$_SERVER['PHP_AUTH_USER']."<br />";
```

```
echo "Contraseña: ".$_SERVER['PHP_AUTH_PW']."<br />";
```

```
?>
```

```
</body>
```

```
</html>
```

Se requiere una autenticación

http://localhost

Nombre de usuario

Contraseña

Nombre de usuario: pepe
Contraseña: melainvento

Autenticación de usuarios y control de acceso

Incorporación de métodos de autenticación

El método más simple para comprobar que las credenciales que ha introducido el usuario son correctas es comparar los datos introducidos con datos fijos.

Por ejemplo, permitir el acceso a un usuario '**dwes**' con contraseña '**abc123.**':

```
<?php
if($_SERVER['PHP_AUTH_USER'] != 'dwes' || $_SERVER['PHP_AUTH_PW'] != 'abc123.')
{
    header('WWW-Authenticate: Basic Realm="Contenido restringido"');
    header('HTTP/1.0 401 Unauthorized');
    echo "Usuario no reconocido.";
    exit; // exit = exit() = exit(0);
}
?>
```

- **Problema:** cambiar nombre de usuario/contraseña, implica modificar el código. Además, el usuario no podría introducir su propia contraseña.
- **Solución:** utilizar almacenamiento externo para nombres de usuario y contraseñas. Para esto podrías emplear un fichero de texto, o mejor aún, una base de datos.

Autenticación de usuarios y control de acceso

Para almacenar la información de los usuarios en la BD llamada *bdprueba*, vamos a crear una nueva tabla llamada **usuarios**. Se podría hacer desde SQL así:

```
-- Seleccionamos la base de datos
USE bdprueba;
-- Creamos la tabla
CREATE TABLE usuarios (
    usuario VARCHAR(20) NOT NULL PRIMARY KEY,
    contrasena VARCHAR(32) NOT NULL
) ENGINE = INNODB;
-- Creamos el usuario dwes
INSERT INTO usuarios (usuario, contrasena)
VALUES ('dwes', 'e8dc8ccd5e5f9e3a54f07350ce8a2d3d');
```

Aunque se podrían almacenar las contraseñas en texto plano, es mejor hacerlo encriptadas.

En este ejemplo, para el usuario "dwes" se almacena el hash MD5 de la contraseña "abc123".

Podemos usar la función md5 para calcular el hash MD5 de una cadena de texto.


<http://es.php.net/manual/es/function.md5.php>

<?php

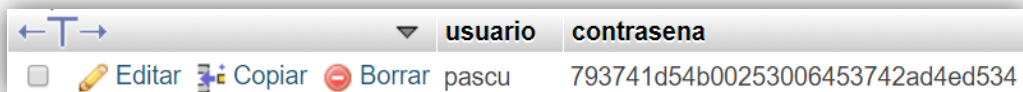
```
// Si el usuario aún no se ha autenticado, pedimos las credenciales
if(!isset($_SERVER['PHP_AUTH_USER'])){
    header('WWW-Authenticate: Basic realm="Contenido restringido"');
    header("HTTP/1.0 401 Unauthorized");
    exit;
}else{
    // Conectamos a la base de datos
    @$conexion = new mysqli("localhost", "alumno", "12345", "bdprueba");
    // Si se estableció la conexión
    if ($conexion->connect_errno == null)
    {
        // Comprobar si existe esa combinación de usuario y contraseña
        $sql = "SELECT usuario FROM usuarios ".
            "WHERE usuario = '" . $_SERVER['PHP_AUTH_USER'] . "' ".
            "AND contraseña = md5('" . $_SERVER['PHP_AUTH_PW'] . "')";

        $consulta = $conexion->query($sql);
        // Si no existe, se vuelven a pedir las credenciales
        if($consulta->fetch_assoc() == null){
            header('WWW-Authenticate: Basic realm="Contenido restringido"');
            header("HTTP/1.0 401 Unauthorized");
            exit;
        }
        $consulta->close();
        $conexion->close();
    }
}
```

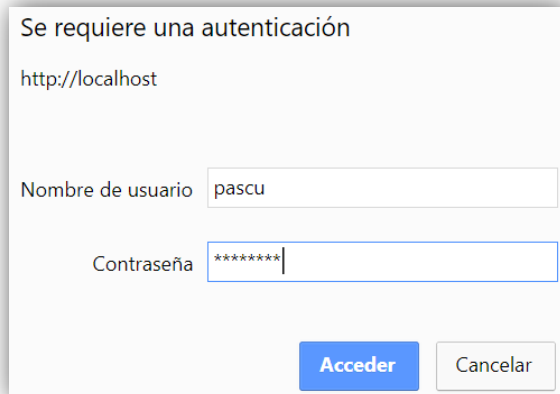
?>



```
<!DOCTYPE html>
<head>
    <meta charset="utf-8"> <title>Desarrollo web en entorno servidor - ud 5</title>
</head>
<body>
    <?php
        echo "Nombre de usuario: ". $_SERVER['PHP_AUTH_USER'] . "<br />";
        echo "Contraseña: ". $_SERVER['PHP_AUTH_PW'] . "<br />";
        echo "Hash de la contraseña: " . md5($_SERVER['PHP_AUTH_PW']) . "<br />";
    ?>
</body>
</html>
```




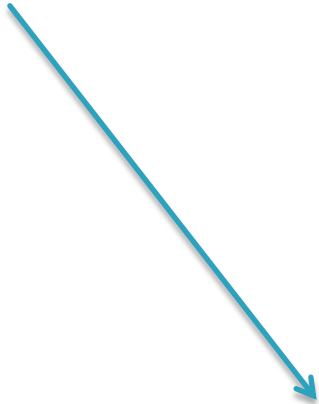
	usuario	contrasena
	pascu	793741d54b00253006453742ad4ed534



Se requiere una autenticación
http://localhost

Nombre de usuario:

Contraseña:



Nombre de usuario: pascu
Contraseña: profesor
Hash de la contraseña: 793741d54b00253006453742ad4ed534

Autenticación de usuarios y control de acceso

Otro algoritmo similar a MD5 es **SHA1**. Era más seguro que el anterior pero, hoy en día, **también es fácilmente descifrable**. Para usar SHA1, tenemos la función **sha1** para cifrar texto. El único requisito es que el tipo de dato del campo en la BD que guardará la password debe ser **binary(40)**.

El mismo manual de PHP advierte de que ya no son algoritmos seguros:

sha1

Advertencia No se recomienda utilizar esta función para contraseñas seguras debido a la naturaleza rápida de este algoritmo de «hashing». Véase las [Preguntas más frecuentes de «hash» de contraseñas](#) para más detalles y el empleo de mejores prácticas.




Existen webs como <https://hashkiller.co.uk/sha1-decrypter.aspx> que en un momento descifran un código de este tipo. No obstante, en esta unidad nos conformaremos con **SHA1**.



EJERCICIO

Crea una página llamada **ud05ejer01.php** con título 'UD 5 Ejercicio 1'.
Crea una tabla que se llame **usuarios** con los siguientes campos:

Nombre	Tipo
usuario 	varchar(20)
contrasena	binary(40)

Asegúrate de que
todo sea así.

El tipo **Binary** es como el tipo **Varchar** pero almacenando bytes en lugar de caracteres.

Crea un formulario para añadir usuarios a la BD encriptando su contraseña.
Si alguno de los dos campos está vacío cuando le demos al botón de submit, debe aparecer un aviso y el que ya estaba relleno, se conserva:

Registrarme

Usuario:

Contraseña:

Registrarme



Registrarme

Usuario: ¡Debe introducir un nombre de usuario!

Contraseña:

Registrarme

Cuando ambos estén rellenos y se pulse el botón, se hará un **INSERT** en la tabla, teniendo en cuenta que la contraseña se encripta en **sha1**.

Si todo ha ido bien, se vaciarán los campos y surgirá un *alert* de éxito en javascript. En caso contrario, muestra un *alert* avisando del error pero no borres los campos.

Por ejemplo, si se introduce *alumno* y *12345*, al clicar en **Registrarme** se vaciará el formulario y aparecerá el mensaje de éxito por la inserción del usuario. Sin embargo, si el nombre de **usuario (primary key)** ya existe, no se introducirá y se avisará del error.

Comprueba de un vistazo la inserción del usuario, y que su contraseña esté encriptada.

		usuario	contrasena
<input type="checkbox"/>	Editar	alumno	8cb2237d0679ca88db6464eac60da96345513964