La Sénia
Institut d'Educació Secundària
Paiporta

UP3-4. Seguridad: Roles y Tokens

Desarrollo Web en Entorno Servidor

Profesora: Silvia Vilar Pérez

curso 2025-2026

Contenidos

- Seguridad: usuarios, perfiles, roles.
- Autenticación de usuarios.
- Tokens de formulario

Seguridad: usuarios, perfiles, roles

- Para controlar el acceso de los usuarios, se implementan las ACL (Access Control List) en las que se determinan los permisos de acceso en base a **roles o perfiles** que son aplicables a usuarios y grupos de usuarios.
- Los pasos para implementar la seguridad con ACL son:

1) Crear la ACL que puede estar implementada en una BD, una estructura de datos, etc.

2) Usar la ACL en la página Web comprobando el rol al que pertenece el usuario para aplicarle los permisos correspondientes y visualizar contenido

Ejemplo roles

```
<?php      /* roles.php */
session_start();
     $_SESSION['usuario']=$_POST['usuario'];
     $_SESSION['pass']=$_POST['password'];
     $_SESSION['rol']=$_POST['rol'];

if (isset($_SESSION['usuario'])){
   switch ($_SESSION['rol']) {
      case 'Estudiante':
         $location = 'Location: indexEstudiante.php';
         break;
      case 'Profesor':
         $location = 'Location: indexProfesor.php';
         break;
      case 'Otro':
         $location = 'Location: indexDefault.php';
         break;
   }
}
header($location);
?>
```

Nota: No se han validado los datos, sólo la comprobación de hay un usuario en la sesión

Ejemplo roles

```
/* roles.php */
<html>
    <form  action="roles.php" method="post">
        Nombre: <input type="text" name="usuario"/>
        Password: <input type="password" name="password"/>
        <p>
        Estudiante <input type="radio" name="rol" value="Estudiante"/>
        Profesor <input type="radio" name="rol" value="Profesor"/>
        Otro <input type="radio" name="rol" value="Otro" />
        </p>
        <input type="submit" name="Acceder" value="Acceder"/>
    </form>
</html>
```

Ejemplo roles

/* En cada uno de los ficheros indexDefault.php, indexProfesor.php e indexEstudiante.php */

```
<?php
session_start();

print ("Bienvenido ".$_SESSION['usuario']."!\n");
print ("Tu rol es ".$_SESSION['rol']."\n");
print ("Tu password es ".$_SESSION['pass']);

?>
```

Autenticación de usuarios

1) Podemos crear nuestro propio formulario de autenticación de usuarios donde en base a las características del usuario (identificador y password) se pueda validar.
2) Por otra parte, la autenticación de usuarios puede realizarse en el propio servidor web. En el caso de Apache, los ficheros **.htaccess** permiten limitar el acceso a un determinado recurso del servidor. Basta con configurar las directivas adecuadas en el .htaccess en la carpeta de nuestro proyecto para que se apliquen en nuestro sitio web
3) Además, disponemos de la llamada ***autenticación universal*** (ejemplo: validarse con una cuenta de gmail, FB, Twitter, etc.) que nos facilita los siguientes métodos:
   - **OpenId**: Permite a un usuario entrar en una página web pudiendo ser verificado por otro servidor que soporte este protocolo (identificación)
   - **OAuth**: Dispone de una API segura de autorización.

Autenticación de usuarios - Passwords

- Para encriptar los passwords es recomendable usar la función **password_hash()** que incluye el algoritmo, el coste y el salt (obsoleto) del hash devuelto y simplifica su comprobación con **password_verify()**

Ejemplo de creación de password:

```
<?php
/**
 * Se prefiere el modificador PASSWORD_DEFAULT a PASSWORD_BCRYPT
 * Actualmente BCRYPT produce un truncamiento de 72 caracteres . Pero con
 * DEFAULT la longitud puede variar, se debería poder almacenar hasta 255
 * caracteres
 */
    echo password_hash("silviavilar", PASSWORD_DEFAULT);
?>
```

Resultado:

```
$2y$10$aGPT0rVPoBo9161i6.uKf.Pxw0.BLmYjc85lypdOoxPJsbzWa3GXy
```

Autenticación de usuarios - Passwords

Ejemplo de comprobación de un password:

```
<?php
// Ver el ejemplo de password_hash() para ver de dónde viene este hash.
$hash
='$2y$10$aGPT0rVPoBo9161i6.uKf.Pxw0.BLmYjc85lypdOoxPJsbzWa3GXy';

if (password_verify('silviavilar', $hash)) {
    echo '¡La contraseña es válida!';
} else {
    echo 'La contraseña no es válida.';
}
if (password_verify('otrouser', $hash)) {
    echo '¡La contraseña es válida!';
} else {
    echo 'La contraseña no es válida.';
}
?>
```

Resultado:
¡La contraseña es válida!
La contraseña no es válida.

Autenticación con Sesiones - Ejemplo

```
<?php                          /*autentica.php*/
//creamos la variable de sesión de usuario autenticado para consultarla después
session_start(); //iniciamos la sesión

if (!isset($_SESSION["autenticado"])){
    if (isset($_POST["user"]) && isset($_POST["pass"])){
        if ($_POST["user"]==="silvia" && $_POST["pass"]==="123"){
            $_SESSION["autenticado"]="SI";
            header("Location: aplicacionsegura.php");
        } else // Credenciales erróneas, mostramos de nuevo index
            header("Location: index.php”);
    } else //No ha rellenado el formulario de autenticación
        header("Location: index.php");
} else // Ya está acreditado y no ha cerrado la sesión aún
    header("Location: aplicacionsegura.php");
?>
```

Problemas de seguridad - XSS

La principal motivación de los ataques es el robo de cookies y de sesiones, modificar el sitio web, redireccionar a otras páginas, etc. **Ataques XSS (Cross-Site Scripting)**: se basan en explotar la confianza que tiene un usuario en un determinado sitio web o aplicación. ***Se ejecutan en el navegador (cliente)***.La forma más común de ataque es introduciendo scripts en los controles de formularios o mediante subida de archivos con código malicioso.

La protección frente a este ataque es:

1) **Validación** de datos: que sean correctos y esperados
2) **Sanitización** de los datos: garantizar que son seguros eliminando parte indeseable y normalizándolos en la forma correcta. Ejemplo: evitar marcas de HTML en un string introducido por el usuario
3) Aplicar output **scaping**, esto es, evitar caracteres especiales al devolver el dato al cliente. Por ejemplo, usando funciones como htmlespecialchars() y htmlentities()

Ver: https://es.wikipedia.org/wiki/Cross-site_scripting

Problemas de seguridad - XSRF/CSRF

Ataques XSRF / CSRF (Cross-Site Request Forgery): se basan en explotar la confianza que un sitio web o aplicación tiene en un usuario en particular.

- La forma más común de ataque es el robo de una sesión iniciada del usuario y éste visita una página generada por el atacante durante la vigencia de la sesión. Entonces el atacante interactúa con el servidor con la sesión del usuario legítimo.
- La protección frente a este ataque es el uso de **token de formulario** de modo que se generará un token conteniendo un identificador único para la sesión del usuario usando funciones seguras como **bin2hex()** y **openssl_random_pseudo_bytes()**
- Cuando el procesador del formulario recibe datos, comprueba si el token recibido del formulario coincide con el de la sesión. Si no coincide descarta la petición y si coincide procesa la petición y borra la variable de sesión que contenía el valor del token.
- Evitan que se pueda reenviar un formulario y ataques CSRF/XSRF

Ver: https://es.wikipedia.org/wiki/Cross-site_request_forgery

Tokens de Formulario

Un token de formulario es un campo oculto que incluye un valor único y, a la vez, ese mismo valor se guarda en una variable de la sesión del usuario para después comprobar que sean el mismo.

```
// Ejemplo generación
session_start();
$_SESSION["token"] = bin2hex(openssl_random_pseudo_bytes(24));
// Inclusión en el formulario en un control oculto
<form action="process.php" method="post">
    <input type="hidden" name="token" value="<?php echo $_SESSION['token']; ?>">
    <input type="text" name="email" placeholder="Your email address..."><br>
    <input type="submit" name="submit_form">
</form>
// También podemos pasarlo como parámetro en la url cuando cierre sesión, por ejemplo
<!--Protege la URL de cierre de sesión de ataques CSRF-->
<a href="logout.php?token=<?php echo $_SESSION['token'];?>"> Logout </a>
```

Tokens de Formulario

```
//Procesamos el formulario en process.php
//Debemos recordar siempre iniciar la sesión para recuperarla.
session_start();

//Nos aseguramos de que el token se ha guardado en la variable $_POST.
if (!isset($_POST['token'])) {
    print('No se ha encontrado token!');
} else {

    //Si existe, debemos comprobar que el token recibido en  $_POST es
    //el que hemos almacenado en la variable de la sesión $_SESSION
    if (hash_equals($_POST['token'], $_SESSION['token']) === false) {
        print('El token no coincide!');
    } else {

        //El token es correcto y continúa el procesamiento con seguridad
        print('El token es correcto y podemos ejecutar acciones');


    }
}
```