Seguridad en PHP

Área de Ingeniería Telemática

Seguridad

- La seguridad es muy importante
- Conocer riesgos ➔ enfrentarnos a ellos:
  - evitándolos
  - poniéndoles remedio
- No existe la seguridad completa ➔ siempre hay riesgo
- Objetivo: aumentar la seguridad para disuadir a la gente de que lo intente
- Posibles fallos de seguridad:
  - Software usado: S.O., servidor, FTP, …
  - Errores o lagunas de programación: SQL injection, no comprobación de datos, …
  - Humanos: login/password debajo del teclado, …

SQL Injection

- Cuando se puede "inyectar" código SQL "invasor" dentro del código SQL fuente para alterar su funcionamiento normal
- Origen: filtrado incorrecto de las variables utilizadas
- Ejemplo:

```
$user = $_POST['user'];
$pass = $_POST['password'];
SELECT * FROM login WHERE field_user='$user' AND
   field_pass='$pass';

Si en $_POST['user'] y $_POST['password'] ponemos: 666' OR '1'='1

SELECT * FROM login WHERE field_user='666' OR '1'='1' AND
   field_pass='666' OR '1'='1';
```

SQL Injection

- Soluciones:
  - Comprobación de los datos
  - Escapar los datos externos: `mysql_real_escape_string()`

```
SELECT * FROM login WHERE field_user='
  mysql_real_escape_string($user)' AND
  field_pass='mysql_real_escape_string($pass)';
```

  - Programación orientada a la comprobación de resultados

```
SELECT * FROM login WHERE field_user='user';
Si devuelve algo comprobar que el password es igual que el
  pasado por formulario
```

Datos de usuario

- Cuando se usan directamente los datos de los usuarios confiando en que meten lo que deben meter
- Ejemplo:

```
$fichero = $_GET['fichero'];
readfile($fichero);
```

- Uso normal del script:
  - Hacemos un listado de ficheros que el usuario puede ver y que al pinchar en ellos se muestren mediante el script
  - Confiamos en que el usuario use correctamente el sistema, pero...

Datos de usuario

- Uso malicioso:
  - Modificar la llamada para pasar como "fichero" el fichero de passwords de la máquina (/etc/passwd)
  - Logra fichero de passwords y puede entrar en la maquina

- ¿Porqué posible?
  - La dirección física del fichero se pasa al script desde el formulario ➔ el usuario lo puede ver ➔ puede modificarlo
  - El script usa directamente la dirección pasada sin tomar ningún tipo de protección

Datos de usuario

- Soluciones:
  - Sólo permitir direcciones relativas dentro de un directorio en concreto:

```
$fichero = basename($_GET['fichero']);
$relative_path = "files/";
readfile($relative_path.$fichero);
```

  - Comprobar la veracidad de los datos:

```
$array_ficheros = [...];
if (in_array($fichero, $array_ficheros)) {
      readfile($relative_path.$fichero);
}
```

Datos de usuario

- Otro ejemplo:

```
$user = $_POST['user'];
$edad = $_POST['edad'];
INSERT INTO registro (user, edad) VALUES ('$user', $edad);
```

- Problema: si en vez de pasar un número pasa una cadena de texto o esta vacío ➔ error
- Problema de seguridad: al mostrar el error del sistema el usuario obtiene información extra:
  - Como está hecha nuestra programación
  - Rutas de ficheros
  - Morfología de la base de datos
  - ...

Datos de usuario

- Soluciones:
  - Siempre limpiar los datos externos:
    - Comprobar que existen: `isset()`
    - Comprobar que son del formato que necesitas: `is_int()`
    - Convertir al formato que deberían ser
      - si es una clave de 4 números (tipo cajero) convertir a un número de 4 dígitos antes de usarlo
  - Configurar gestión de errores:
    - Mensajes de error del sistema solo visibles durante desarrollo
    - En producción mensajes propios asépticos

Resumen

- No hay seguridad total ➔ objetivo dificultarlo hasta el punto que la recompensa no merezca el esfuerzo
- Nunca fiarse del usuario:
  - Comprobar, validar y adaptar los datos enviados por usuario
  - Programar orientado a comprobar lo que sucede con los datos ➔ siempre pensar en el peor caso
- Gestión de errores adecuada