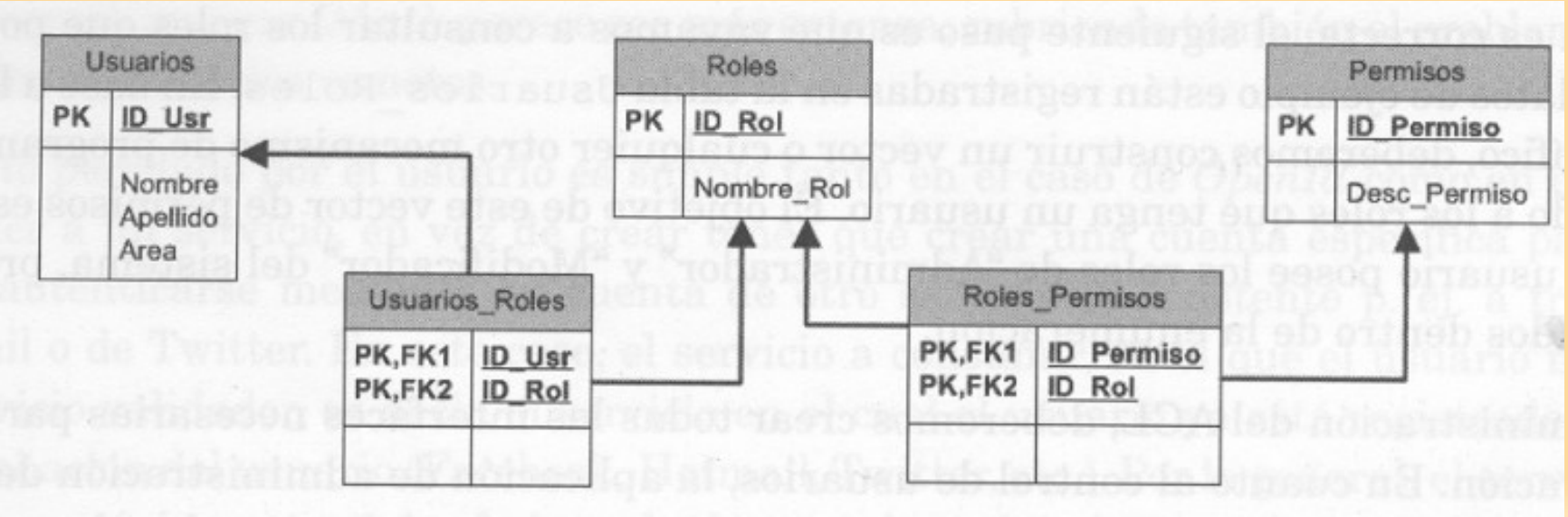
Seguridad en aplicaciones web con PHP

Autenticación mediante ACL

- Casi todas las aplicaciones web incluyen un subsistema de autenticación de usuarios (ACL = Access Control Login).
- Ese subsistema suele estar basado en este diseño de base de datos:



Cookies

- Las **cookies** son variables que se guardan en el ordenador del cliente.

- Sintaxis:

```
bool setcookie ( string $name [, string $value [, int $expire = 0 [, string $path [, string $domain [, bool $secure = false [, bool $httponly = false ]]]]] )
```

- Ejemplo:

```
setcookie("TestCookie", $value, time()+3600);
```

Cookies

- Para acceder al valor de una cookie:

```
$_COOKIE["NombreCookie"];
```

- Ejemplo:

```
echo "La cookie TestCookie vale ".
$_COOKIE["TestCookie"];
```

Sesiones

- Las **sesiones** sirven para guardar variables en el servidor.
- Esas variables sólo son accesibles para el cliente que creó esa sesión.

- Sintaxis:

```
session_start();
$_SESSION["variable"] = $valor;
```

Sesiones

- Algunas funciones para manejar sesiones:

```
// Abre una sesión o la retoma si ya estaba abierta
session_start();
// Cierra una sesión abierta y destruye sus variables
session_destroy();
// Devuelve el ID de la sesión
session_id();
// Destruye todas las variables de sesión
session_unset();
```

Sesiones, cookies y seguridad

- Cookies y variables de sesión se usan a menudo para **controlar la seguridad** de la aplicación web.
- Por ejemplo, tras el login, el ID del usuario puede almacenarse en:
  - Una cookie. Si existe esa cookie, significa que el login ha sido correcto y la aplicación puede continuar.
  - Una variable de sesión. Si existe tal variable, el login ha sido correcto.
- Cuando el usuario abandona la aplicación, el programa debe destruir la cookie o cerrar la sesión.

¿Mejor cookies o variables de sesión?

- **¡Ningún método es completamente seguro!**
  - Las cookies pueden rastrearse o modificarse en el ordenador del cliente. Además, algunos clientes las tienen desactivadas. ¡No te puedes fiar de ellas!
  - Las variables de sesión, en principio más seguras, pueden ser atacadas capturando el ID de sesión.
- El método más seguro, y el más complicado de programar, es el que combina:
  - Cookies y/o variables de sesión.
  - Variables guardadas en una tabla de la BD.

Técnicas de ataque frecuentes

- Captura de ID de sesión
- Inyección de SQL
- XSS (cross site scripting)
- CSRF (cross site request forgery)
- DT (directory transversal)
- RFI (remote file inclusion)

Técnicas de ataque frecuentes

- **Captura de ID de sesión**
  - El ID de sesión se pasa entre páginas de forma transparente a través de cookies o de la URL (con POST). Un atacante puede leer el ID de sesión en el paquete http y acceder a las variables de sesión.
- Solución:
  - Combinar las variables de sesión con cookies o con entradas en la base de datos.
  - No confiar en variables de sesión para información sensible.

Técnicas de ataque frecuentes

- **Inyección de SQL**
  - Se inserta código SQL en campos de formulario. Este código actúa sobre la BD, dando información al atacante sobre su estructura y contenido, o permitiéndole destruir datos.
- Soluciones:
  - Usar filtros de SQL (PDO y MySQLi ya los incluyen)
  - Utiliar usuarios de MySQL sin privilegios destructivos.
  - Filtrar los datos de entrada de los formularios.

Técnicas de ataque frecuentes

- **XSS** (cross site scripting)
  - Se inyecta código JavaScript a través de la URL, de un formulario o de algún otro elemento externo.
  - Ese código JS redirecciona a otra página o tiene algún otro efecto indeseado.
- Soluciones
  - Filtrar todos los datos externos.
  - Usar listas blancas de datos válidos.

Técnicas de ataque frecuentes

- **CSRF** (cross site request forgery)
  - Consiste en que un usuario accede a partes no permitidas de la aplicación insertando datos maliciosos en la URL o en un formulario.
- Soluciones:
  - Utilizar POST en lugar de GET para no dar pistas.
  - Generar tokens únicos para cada petición.
  - Filtrar los datos de entrada.

Técnicas de ataque frecuentes

- **DT** (directory transversal)
  - El atacante accede a ficheros fuera del directorio público (htdocs o public_html) mediante rutas relativas (../../ejemplo.php)
  - Sucede cuando la página que se va a cargar se envía como un parámetro en la URL (index.php?page=ejemplo.php)
- Soluciones:
  - Filtrar el formato de las páginas enviadas por la URL, o tener una lista de páginas válidas.

Técnicas de ataque frecuentes

- **RFI** (remote file inclusion)
  - Consiste en acceder al sistema de ficheros del servidor mediante inyección de código malicioso en la URL o en un formulario.
- Soluciones:
  - Filtrar datos de entrada.
  - Tener una lista de páginas válidas.