UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES ARAGÓN

CARACTERISTICAS DE SEGURIDAD EN EL SISTEMA DE VOTACIONES ELECTRONICAS, DESARROLLADO PARA ELECCIONES EN LA UNAM

T E S I N A

QUE PARA OBTENER EL TÍTULO DE
INGENIERO EN COMPUTACIÓN

PRESENTA:

JOSÉ OTHONIEL CHAMÚ ARIAS

DIRECTOR DE TESIS:

ING. ERNESTO PEÑALOZA ROMERO

INDICE

INTRODUCCIÓN

En este trabajo se van a revisar las “características de seguridad en el sistema de voto electrónico”, ya que en la forma de voto tradicional las personas que se encargan de vigilar las urnas pueden verse envueltas en algunos riesgos, y mediante la aplicación de un proceso electrónico puede ser acelerado, transparentado y hacerse más accesible, pero obviamente se requiere de un sistema de seguridad lo más fiable posible.

Por otra parte el análisis de seguridad de los sistemas WEB (Red), es un aspecto muy importante si se desea tener datos confiables, pero ha sido ignorado por la demanda creciente que exige crear sistemas rápidos y económicos. Es por eso que hemos decidido realizar un estudio minucioso sobre la seguridad de tales sistemas con la intención de generar recomendaciones útiles para desarrollos futuros, como lo es el presente caso: Un sistema de voto electrónico.

El marco teórico abarca de manera general la forma en que han evolucionado las aplicaciones WEB, ya que permiten el fácil acceso a la información; pero que, han experimentado un crecimiento en el nivel de complejidad. Se abordan también las consideraciones de seguridad que se implementaron en el sistema de voto electrónico de la UNAM, posteriormente en el capítulo 3 se describe el sistema tal como lo ve el usuario, por lo cual se va a desarrollar toda la información en forma de ventanas. En el capítulo 4 se muestra el esquema empleado para salvaguardar los datos de las urnas. Finalmente se presentan las conclusiones y algunas recomendaciones útiles para mejorar el sistema de seguridad. También se presenta una lista de las obras consultadas, cabe señalar que no se encontró literatura extensa sobre el tema, ni en medios impresos, ni en electrónicos debido en parte a que éste es un tópico relativamente nuevo que involucra aspectos diversos y además de tener un carácter secreto lo cual vuelve este tema algo muy complejo.

Para desarrollar el trabajo se simuló un sistema de votación que tiene como *objetivo* analizar la mayor cantidad de aspectos de seguridad que se encuentran relacionados con los sistemas de información orientados a la WEB, de tal manera que se puedan hacer recomendaciones específicas que aseguren la confiabilidad.

El sistema de voto electrónico surge debido a la necesidad de garantizar la integridad física de los votantes, agilizar, transparentar y hacer más confiables los resultados.

1.

APLICACIONES WEB

1.1 AMENZAS Y ERRORES EN LAS APLICACIONES WEB

Las aplicaciones WEB surgen por la necesidad de tener acceso rápido a la información, desde sitios lejanos, tales aplicaciones conforme han ido avanzando permiten tener a las empresas un mayor control sobre sus datos, lo que le facilita compartirlos y personalizarlos a detalle con sus usuarios y el público en general, además de permitirles dar a conocer sus productos y servicios hacia todo aquel que cuente con acceso a la Internet; tal facilidad en el manejo de información va de la mano con el aumento en la complejidad para el desarrollo, así como también de las herramientas y tecnologías que involucra.

Las tecnologías que una aplicación WEB involucra son muy variadas, entre ellas podemos encontrar:

- **Sistemas Manejadores de Base de Datos Relacionales** (RDBMS),-permiten almacenar y manejar grandes volúmenes de información-.
- **Lenguajes de programación** -permiten explotar la información de las bases de datos y a su vez pueden proporcionar una interfaz gráfica al usuario-.
- **Servidores WEB** (Apache, Internet Information Server, Tomcat, etc.) -sirven de intermediarios entre los usuarios, la aplicación y los RDBMS-.

La necesidad de involucrar tantas tecnologías, plantea también otros retos, los cuales tienen que ver con la seguridad de la aplicación, ya que las fallas de seguridad pueden provenir de las herramientas siguientes:

- Sistema Operativo (S.O).
- Servidores WEB.
- Servidores de Base de Datos (BD).
- Lenguajes de programación.
- Navegadores WEB.

Las aplicaciones WEB son acechadas también por problemas, que no tienen nada que ver con las herramientas y el desarrollo, por ejemplo:

- Suplantación del sitio.
- Configuración deficiente del sitio, al ponerlo en producción.
- Almacenamiento de las contraseñas.
- Contraseñas débiles de los usuarios.
- Intercepción y captura de la información que transfieren (recordando que la información en Internet suele ser transferida en claro).
- Fraudes mediante cuentas de correo robadas.
- Desplomes del sistema provocadas por un exceso de peticiones al sistema (Denegación de Servicio).
- Desarrolladores mal intencionados que dejen puertas traseras.
- Empleados descontentos que conozcan la aplicación.
- Etc.

Actualmente no se ha hecho el énfasis necesario en la seguridad que deben contemplar las aplicaciones tanto a nivel interno como externo, muchas de las fallas de seguridad se deben en parte a equipos de desarrolladores con muy poca experiencia y desconocimiento del área de seguridad; no se descarta que a los expertos se les escapen algunas consideraciones, además de la naturaleza misma de las aplicaciones, ya que incluyen problemas de tiempos y costos, mismos que afectan la codificación y las pruebas funcionales que deben contemplarse para liberar una aplicación.

Por lo anterior se puede decir que el reforzar la seguridad en las aplicaciones WEB, implica realizar un análisis más detallado que se ve reflejado en todas las etapas del desarrollo; debido a ello los puntos de seguridad que se deben considerar al desarrollar el sistema, deben incluir de manera primordial los aspectos siguientes:

- Manejo de Errores.
- Filtrado de Variables.
- Inyección de código.
- Redirección de datos desde equipos cliente.
- Validación de Datos.
- Revisión de galletas "Cookies".
- Protección de archivos y directorios.
- Privilegios en la BD.
- Revisión del código fuente (para evitar instrucciones malintencionadas).

Tales aspectos son mencionados por Scambriay y Shema; Horton y Mugge y en el sitio The Open Web Application Security Project (OWASP[1]), además que el estudio también sugiere lo siguiente:

- Subir archivos sin filtrar y validar.
- Dar de alta nuevas extensiones (PHP).
- Consideraciones de Concurrencia.

Las consecuencias que implica el no tomar en cuenta todos los puntos anteriores, son algunos de las siguientes:

- **Manejo de Errores**. -Proporciona información sobre aspectos tales como: la base de datos, cuentas del sistema operativo, lenguajes de programación, nombre de variables, tipo de servidor WEB, rutas de directorios y en algunos casos entrega al intruso información del código con las consultas SQL (Structured Query Language) utilizadas, así como la estructura de la BD, lo cual puede derivar en la obtención del código fuente-
  .

[1] En la siguiente dirección de internet http://www.owasp.org/documentation/topten se muestran los 10 errores más comunes cometidos en las aplicaciones WEB.

- **Validación de Datos**. -En el servidor, es importante para evitar modificaciones de la información procedente de los clientes, el hecho de no realizarse puede dar como resultado fraudes y alteración de la información, así como el robo de la misma; ya que el uso de un canal seguro de comunicación o el cifrado de la información no es suficiente, debido ha que el usuario tiene acceso a los datos en claro, una vez que se descargan en la máquina cliente-.

El no realizar el filtrado de variables que maneja la aplicación, la expone a un ataque de inyección de código, mejor conocido como "***SQL Injection***", que se agrava de acuerdo al tipo de manejador de BD utilizado.

De igual forma, al no realizar el filtrado de variables del mismo lenguaje, en el cual se desarrolló la aplicación, puede dar pie a la manipulación de variables internas del sistema y de la aplicación misma, esta situación sólo es peligrosa si el atacante conoce el código (por ejemplo un empleado descontento).

Se debe tomar en cuenta no incluir comentarios que proporcionen información del sistema en el código que es entregado al cliente, en dado caso sería recomendable que se codifique.

Otro aspecto importante en el cual se utilizan galletitas "cookies" es que no contengan usuarios y contraseñas, debido a que podrían comprometer el sitio WEB, en caso de ser muy necesario su uso, sería recomendable que estén protegidas.

Una vez mencionadas algunas consecuencias, se puede considerar que la seguridad es una labor de equipo ya que una falla en cualquier parte de la sección de *administradores, desarrolladores y/o analistas*, puede generar un peligro en potencia para la información y/o la aplicación, aspecto que puede ser aprovechado o explotado por algún usuario malicioso, ello tendría repercusiones en la imagen del equipo de trabajo, la pérdida de confianza en la empresa o institución o bien pérdidas económicas por mencionar sólo algunas.

Después de citar algunos de los problemas que suelen enfrentar las aplicaciones WEB, nos enfocaremos al objetivo de la presente tesina: ***Características de Seguridad en un Sistema de Voto Electrónico***, la cual surge debido a la necesidad de garantizar la integridad física de los votantes, agilizar los resultados, transparentarlos y hacerlos confiables.

Para desarrollar dicho objetivo, en el siguiente capítulo simularemos electrónicamente y de la manera más fiel posible el proceso de votación físico. Posteriormente se reunirá un equipo de auditores externos, quienes validaran la transparencia e imparcialidad de los resultados electorales obtenidos con el uso de la aplicación.

2.

SEGURIDAD PARA UN SISTEMA DE VOTACIÓN ELECTRÓNICA

2.1 OCULTAR ARCHIVOS, EXTENSIONES Y NOMBRE DE DIRECTORIOS

El proceso de ocultar los directorios se logra creando un arreglo general dentro del cual se almacenan las rutas y archivos, al combinarlo con variables de sesión de usuario y generación de claves aleatorias, permite enviar un código numérico al equipo cliente donde se despliega el contenido no muestra la estructura del directorio ni el nombre de los archivos que almacena. Lo anterior queda mostrado gráficamente en la Fig. 2.1.

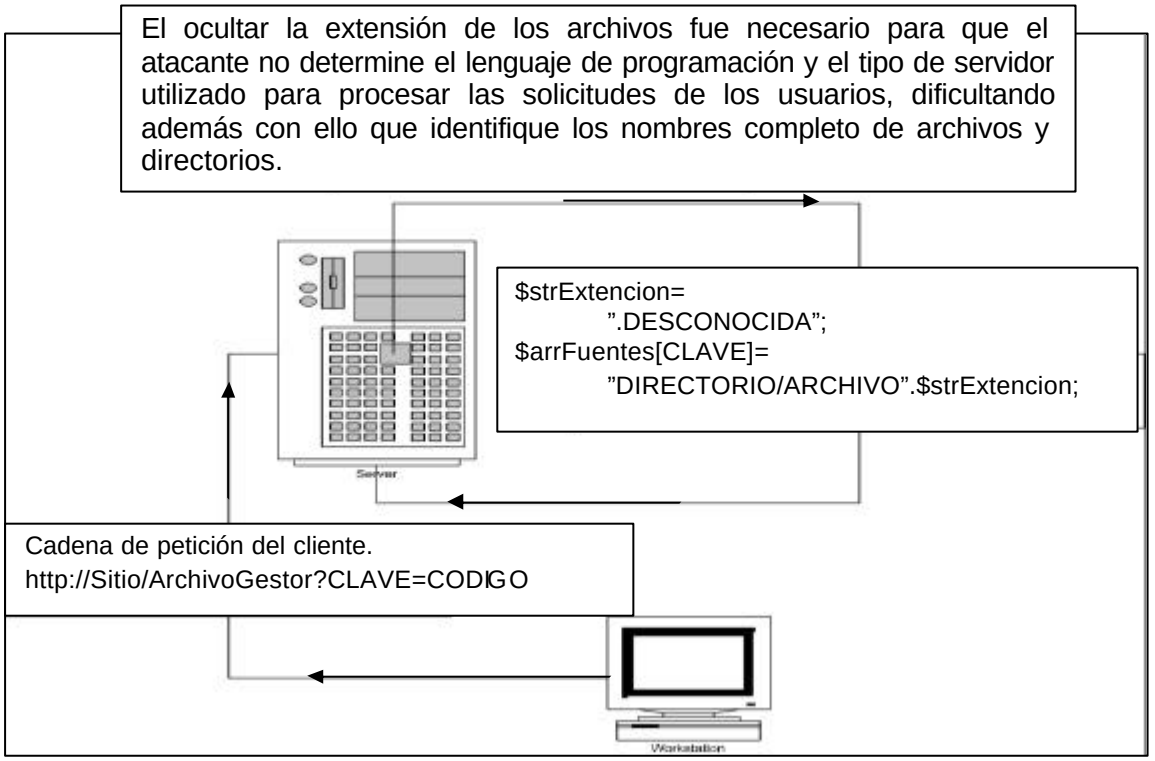

Fig. 2.1 Ejemplo de un arreglo de variables que almacena el nombre**.**

Es importante instrumentar el arreglo de variables, ya que de no hacerlo el atacante podría ver la estructura de directorios y los archivos sensibles para el proceso, dando como resultado la alteración del flujo del sistema con las consecuencias siguientes:

- Cambiar el valor de las variables.
- Establecer la estructura de directorios de la aplicación.

A continuación veamos la forma en que aparecería el código HTML (Hipertext Markup Language) **sin proteger** en comparación el que está **protegido**; posteriormente se verá la forma de implementarlo:

Código sin proteger

```
<FORM METHOD=GET ACTION= "gestor">
  <!-- Aquí se puede apreciar el directorio y el nombre del archivo -->
 <INPUT TYPE=SUBMIT NAME=Enviar>
 <INPUT TYPE=HIDDEN NAME=CLAVE VALUE=directorio/archivo>
</FORM>
```

Código protegido

```
<FORM METHOD=GET ACTION= "gestor">
 <--! Aquí se aprecia la clave que el servidor utilizará para acceder al archivo --
>
 <INPUT TYPE=SUBMIT NAME=Enviar>
 <INPUT TYPE=HIDDEN NAME=CLAVE VALUE=234234265SFS2342>
</FORM>
```

Ahora veamos la forma de implementar este método de protección:

Se requiere de un archivo principal que se encargue de gestionar las peticiones e invocar a su ves todo lo que requiera, este será el único nombre de archivo visible.

1. Almacenar el nombre del directorio dentro de un arreglo, que se almacenara en variables de sesión.
2. Generar una clave aleatoria con algún método el cual garantice que los datos proporcionados por el mismo no se repitan nunca.
3. Almacenar el directorio solicitado dentro del arreglo de sesión con la clave aleatoria como índice.
4. Entregar al cliente la clave generada de manera aleatoria, en el lugar que haga referencia a los archivos, por ejemplo:

    a. En las propiedades *value* de los botones *hidden*, las cuales permiten manipularlo con JavaScript (value y hidden, son propiedades de los elementos de HTML, tales como botones, cajas de texto, etc).

Una vez que ya se generó la clave aleatoria, lo que resta es obtener el archivo asociado a la clave que manda el cliente al servidor, lo que se realizaría de la siguiente manera:

1. Obtención de la clave recibida.

2. Obtener el nombre del directorio o archivo, del arreglo almacenado en las variables de sesión.
3. Eliminar el arreglo de las variables de sesión una ves que el usuario ha terminado de consultar la página WEB.

En seguida se presenta un script propuesto en PHP para generar y recuperar la clave:

Generación de la clave.

```
//Arreglo que almacena los directorios y archivos.
$arrDatos[1000]="directorio/cliActContrasena";

//Generar dígitos aleatorios
$lngRandom=random(200000);
$ruta_en_claro="$lngRandom";
$CLAVE = md5 ($ruta_en_claro);

//Almacenar de manera temporal.
$_SESSION[$CLAVE]= $arrDatos[1000];

//Generar el código HTML con la clave
 <INPUT TYPE=SUBMIT NAME=Enviar>
<INPUT TYPE = HIDDEN
NAME = dato VALUE=<?php echo "$CLAVE"?> >
…
```

La petición que realiza el formulario con el método get y que se aprecia en la barra de direcciones tendría un aspecto similar al siguiente.

http://servidor.com/Gestor.html?dato=avmalkdes

Recuperación de la ruta

La variable dato(es la que se encuentra después del signo de interrogación) trae el número aleatorio que servirá para extraer el directorio y archivo requerido.

```
//Se extrae la firma
$CLAVE = $_GET[ dato ];
//Con la firma se extrae el archivo.
$strArchivo = $_SESSION[$CLAVE];
```

Dando como resultado el valor "**directorio/cliActContrasena**", el cual es el archivo solicitado.

Otro aspecto a proteger es la extensión de los archivos la cual se logra configurando el servidor de aplicación WEB (en este caso Apache), para que busque archivos con instrucciones en extensiones diferentes, por ejemplo: los que

tienen extensión ".oculta" en lugar de ".php". Siendo está última la que esta configurada por default en el servidor de aplicación WEB.

El cambio de extensión crea la sensación de un falso ambiente de producción, dando la impresión de que el sistema está montado sobre un servidor Windows o Linux.

La configuración anterior tiene como objetivo que los ataques no sean dirigidos a una plataforma especifica y la generación de la clave aleatoria es para evitar que se acceda a un archivo en específico directamente, ya que se desconoce su extensión y nombre.

El archivo que se modificó para anexar las extensiones fue *httpd.conf*, dentro del cual se anexo lo siguiente:

```
#
# AddType allows you to add to or override the MIME configuration
# file mime.types for specific file types.
#
AddType application/x-httpd-php .php .html .oculta .classes
```

Este esquema de seguridad (ocultar extensiones de archivos) por oscuridad no funciona, ya que con solo ejecutar un comando en Linux, es fácil de identificar el S.O, el servidor WEB y las versiones de los módulos de seguridad con que está configurado; ejemplo:

```
bash$ nc -vv servidor.algo.com 80
servidor.algo.com [servidor.algo.com] 80 (http) open
GET directorio/archivo.oculto HTTP/1.1

HTTP/1.1 400 Bad Request
Date: Thu, 11 Aug 2005 17:51:35 GMT
Server: Apache/1.3.29 (Unix) PHP/4.3.4 mod_ssl/2.8.16 OpenSSL/0.9.7c
Connection: close
Transfer-Encoding: chunked
```

Como se muestra en el ejemplo anterior, se obtiene el sistema operativo, la versión del servidor de páginas WEB, la versión de PHP instalada, la versión de SSL(Secure Socket Layer) que se utiliza, así como la de OPENSSL.

2.2 FILTRAR CÓDIGO MALICIOSO

Para filtrar el código se desarrollo una clase que se encarga de eliminar palabras reservadas SQL(Structured Query Language), se debe tomar en cuenta que no debe eliminar frases de dos caracteres por que tienen más posibilidad de aparecer de manera aleatoria en las contraseñas de los usuarios, en la siguiente imagen Fig. 2.2 se ilustra una de las maneras comunes para mandar una instrucción SQL y la forma en que es filtrada.

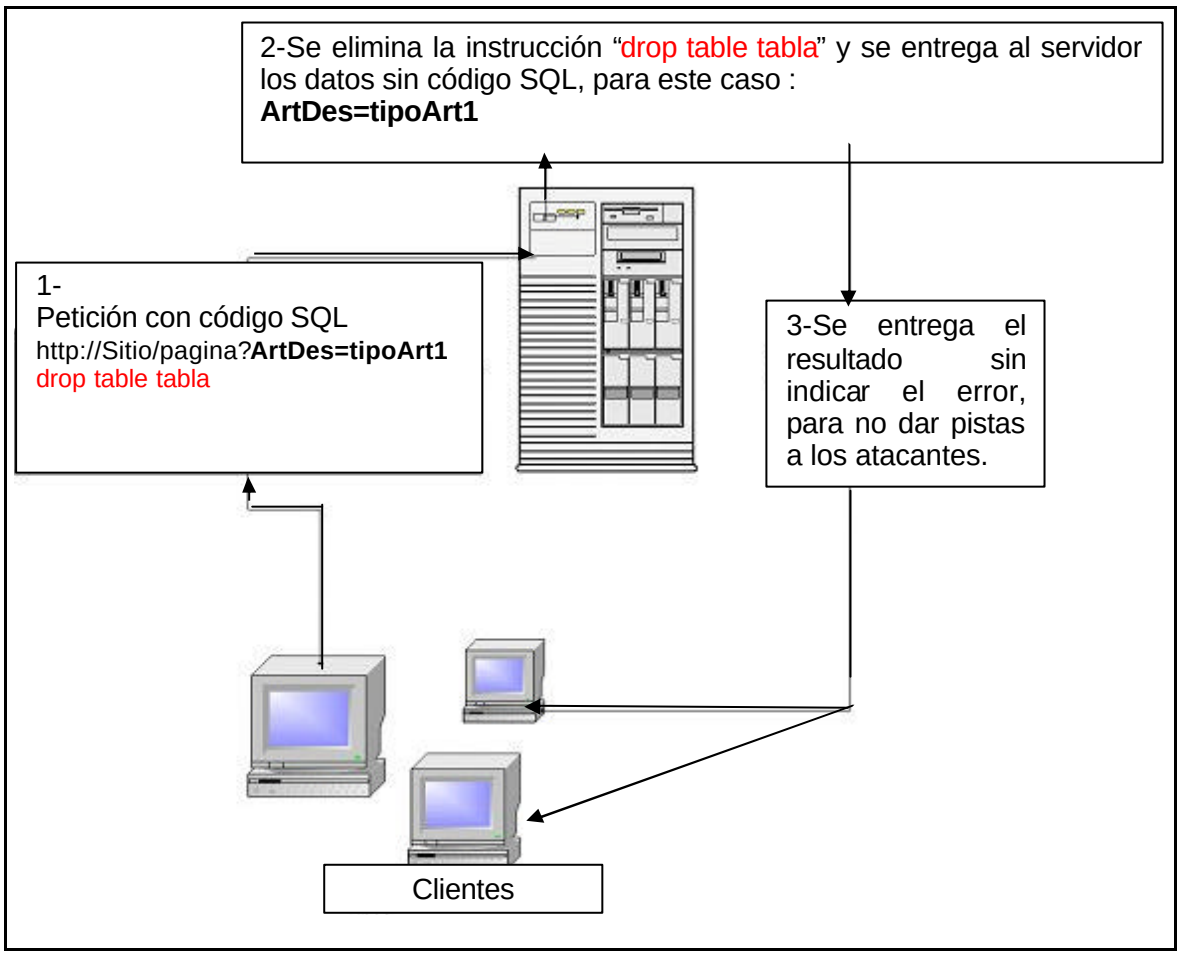

Fig. 2.2 Se ejemplifica como ingresa el atacante una instrucción SQL en la URL y la forma en que se filtra.

Ahora veremos cuales serian los pasos a seguir para poder ingresar una instrucción SQL:

1. Generar un error.

Se logra colocando algún carácter especial o una instrucción trunca de sql, ejemplo (esto se debe colocar en la barra de dirección del navegador).

http://algunSitio/archivo.html?artDetalle=tipoArtClave=1;drop table tabla

2. Revisar la salida, si no se tiene un buen manejo de errores se podrían ver en pantalla las instrucciones SQL, que pudiera afectar, ver Fig. 2.3.

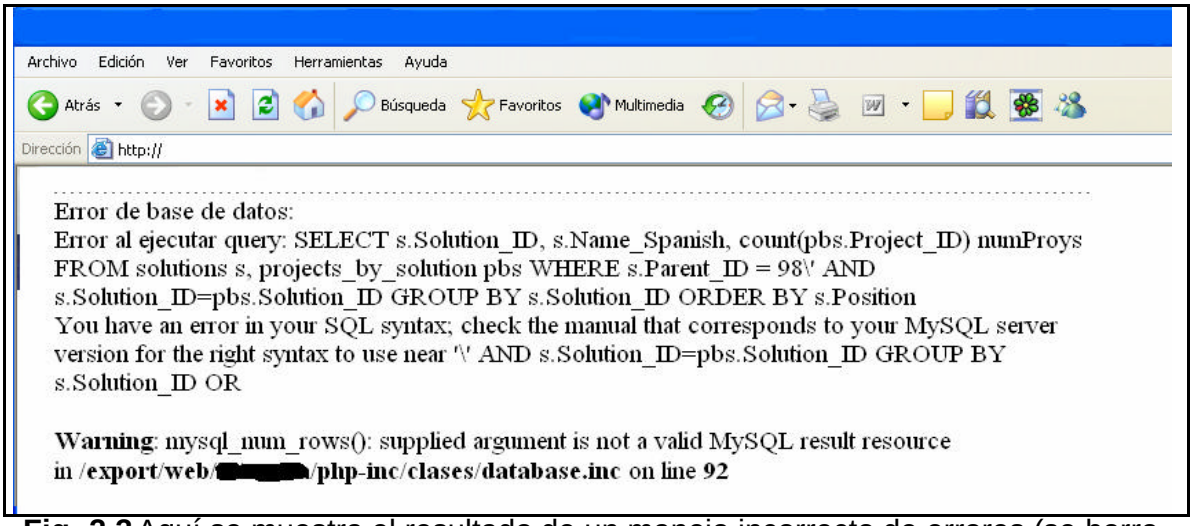

Fig. 2.3 Aquí se muestra el resultado de un manejo incorrecto de errores (se borro el nombre del sitio y el de un directorio para evitar que se haga mal uso del ejemplo).

3. Una vez que se ha visto el error producido, se procedería a realizar combinaciones de caracteres de SQL, hasta lograr que alguna sea ejecutada por la BD, y así obtener información, eliminar registros, provocar una negación de servicio, conocer la plataforma sobre la que se ejecuta, el lenguaje usado, cabe aclarar que se debe identificar el manejador de BD para realizar un ataque con mayor éxito.

   Por Ejemplo:

   Borrado de información, las instrucciones de este tipo pueden causar daños directos a la BD.
   http://algunSitio/archivo.html?artDetalle=tipoArtClave=1;drop table tabla

   Negación de servicio, este tipo de instrucciones hacen que la BD realice una tarea que consuma todos los recursos.
   http://algunSitio/archivo.html?artDetalle=tipo ArtClave=1;select * from t1,t2 where t1.a=t2.a

Como se puede observar si no se filtran las instrucciones SQL los daños mencionados en el punto 3 (página 11) serían de alto impacto; una vez

ejemplificada la manera de ingresar instrucciones maliciosas, veremos algunas contramedidas:

1. Restringir privilegios sobre los elementos de las BD, tales como tablas, esto es prohibir instrucciones de borrado o inserción de registros si el usuario no lo requiere, entre otros.

2. Se deben borrar las bases y usuarios creados por defecto cuando se instala la BD.

3. Se debe filtrar el código del lado del servidor para evitar que pase alguna instrucción de este tipo; un código (script) propuesto sería el siguiente:

```
class Filtro{
    //VARIABLES DE CLASE
        var $strLogin;
        var $strLoginResp;
        var $arrKeys;
        var $strDato1;
        var $strDato2;

        // ASIGANCION DE VALORES
        function Filtro($strLogin){
                $this->$strLogin        = $strLogin;
                $this->$strLoginResp    = $strLogin;
                $this-> strDato1        = 0;
                $this-> strDato2        = 0;

                //LAS PALABRAS A BUSCAR DEBEN IR EN MAYUSCULAS
                $this->arrKeys          = array("=","'\"","\'",

        "SELECT","UPDATE","INSERT","FROM",
        "WHERE","ORDER","GROUP","COMMAND",";","ABORT",
        "AGGREGATE","ALTER","USER","VACUUM");

        }

        //FUNCION QUE REMPLAZA LOS CARACTERES CONTENIDOS
        //EN UN ARREGLO DE UNA CADENA PROPORCIONADA
        function checkString (){

                $this->strLogin = strtoupper($this->strLoginResp);
                foreach ($this->arrKeys as $strKeySuspicius){
                        $this->strLogin = str_replace("$strKeySuspicius ", "",
$this->strLogin);
                }
        }
```

```
        //Regresa la Cadena Filtrada
        function getPhrase(){

                $this->checkString();
                $this->strDato1 = strlen($this->strLoginResp);
                $this->strDato2 = strlen($this->strLogin);
                if( strcmp($this->strDato1,$this->strDato2))
                        return $this->strLoginResp;
                else
                        return " ";
        }
}
```

La clase anterior se encarga de buscar y reemplazar las palabras reservadas de SQL, que pudieran ser ingresadas a través de la URL o bien por medio de alguna caja de texto de HTML.

Las inyecciones de código no serían muy exitosas con Mysql combinado con PHP, ya que la función **mysql_query**, no acepta más de una consulta a la vez, ni al mezclar las instrucciones con algún UNION o haciendo uso de consultas anidadas, aunque sigue siendo susceptible a un ataque de SQL numérico el cual puede provocar una negación de servicio; para cualquiera de los siguientes RDBMS tales como Postgres, Informix, Sybase, Oracle y Microsoft SQL Server, combinado con PHP resulta difícil anexar consultas que requieran algún campo de tipo carácter, ya que al agregar una comilla, PHP le agrega un símbolo (backslash) "\" lo cual provoca un error de sintaxis en el manejador, por lo que la instrucción SQL que se intenta ejecutar falla, pero aún así son susceptibles a instrucciones:

- UNION.
- Consultas anidadas.
- Instrucciones de borrado de campos, tablas, bases de datos.
- Procedimientos propios de cada manejador de Base Datos.
- Ejecución de órdenes del sistema operativo.

2.3 VALIDACIONES DE JAVASCRIPT

Las validaciones de JavaScript son necesarias para evitar que los usuarios se pierdan navegando dentro del sistema así como para quitarle un poco de carga al servidor al realizar validaciones, tales como la longitud del dato requerido, formato de fechas,etc, que no es conveniente debido a que son fáciles de evitar.

Dentro de la aplicación se llevaron a cabo algunas de las siguientes, restricciones:

- Prohibición de teclas tales como:
  - Alt.
  - f5 … f12.
  - Ctrl.
  - Windows.
  - Botones de ratón derecho y rueda central.

- Elementos de navegación:
  - Botones de avance y retroceso.
  - Barra de direcciones.
  - Menú de archivos.

- Despliegue en ventana completa, entre otros.

Las validaciones antes mencionadas tienen los objetivos siguientes:

- Evitar la recarga de la página.
- Prohibir que se guarden la página en disco.
- Que la navegación no este controlada por el navegador.
- Evitar la impresión de la página.
- Dificultar el análisis del código HTML.

Las restricciones indicadas previamente no se lograron en su totalidad debido a la gran variedad de navegadores Web y las diversas versiones de JavaScript que ellos soportan las cuales no son compatibles unas con otras, en la Fig. 2.4 se muestran las validaciones que no serían necesarias en el servidor si el lenguaje JavaScript no pudiese ser manipulado por los usuarios.

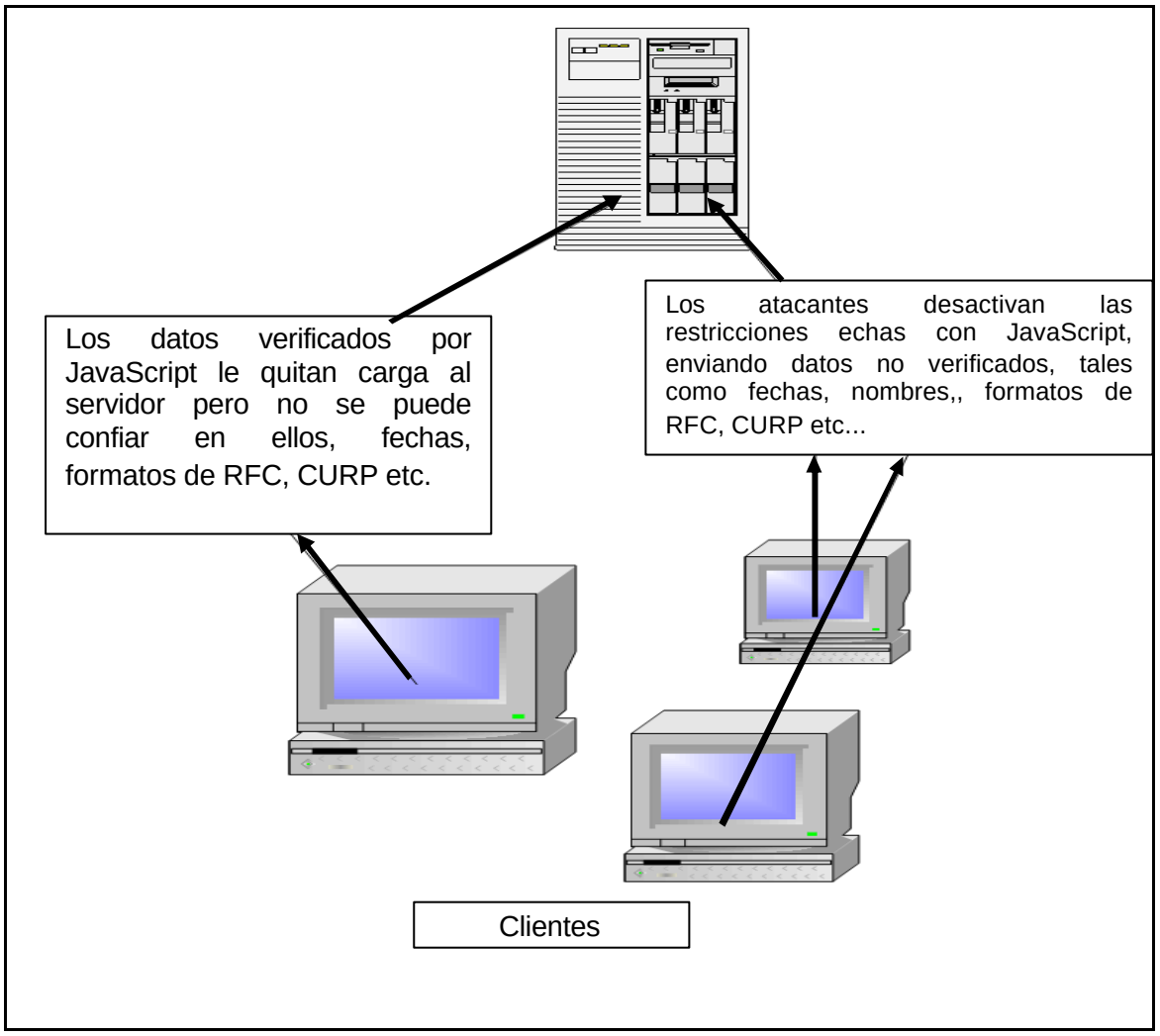

Fig. 2.4 En esta imagen se aprecian solo algunas de las tantas validaciones de JavaScript que pueden ser eludidas por más de un usuario.

La razón de poder eludir fácilmente las validaciones de JavaScript, representa un riesgo para la aplicación, **la solución en cuanto a la validación de datos es comprobándolos nuevamente del lado del servidor**, ya que nunca se debe confiar en la información proveniente de los clientes.

Las consecuencias que conlleva el no validar los datos procedentes de los clientes en el servidor, son algunas de las siguientes:

- El ingreso de datos no permitidos en la BD(Basura).
- Saturación del servidor WEB con peticiones de recarga.
- Saltarse pasos que solicita la aplicación para poder realizar un proceso (eludir los controles de navegación).
- Ver el código HTML para manipularlo y analizarlo.

Por ejemplo la manera de evitar las validaciones de JavaScript es descargando la página a nuestra computadora, esto se hace mediante el siguiente comando (en la línea de comandos de Linux o Windows):

bash$ ***wget*** ***http://algunSitio.com/paginaDeValidaciones.html***

Una vez que se tiene el código en la computadora se eliminan las validaciones de JavaScript y HTML, lo que permite mandar información no verificada al servidor.

Para eliminar las validaciones de JavaScript, se analiza el código para identificar la página que procesa los datos la instrucción más común es la etiqueta FORM en su campo ACTION, ya que dentro está se coloca el nombre de la página que se encarga de procesar la información, el código descargado puede tener un aspecto similar al siguiente:

Código descargado el cual está protegido con Html y JavaScript

```
<FORM METHOD=POST ACTION=archivoObjetivo.php >
<INPUT TYPE=TEXT NAME=LOGIN SIZE=40>
<INPUT TYPE=PASSWORD NAME=PASSWORD SIZE=40>
<INPUT TYPE=BUTTON NAME=ENVIAR VALUE=ACEPTAR
ONCLICK=VALIDAR( LOGIN , PASSWORD ) >
</FORM>
```

Como se observa en el código anterior las etiquetas *text* aceptan como máximo 40 caracteres, ambos son verificados con JavaScript y el archivo que las procesa es *archivoObjetivo.php*.

Código manipulado y desprotegido

```
<FORM METHOD=POST ACTION=
http://algunSitio.com/archivoObjetivo.php >
<INPUT TYPE=TEXT NAME=LOGIN >
<INPUT TYPE=PASSWORD NAME=PASSWORD >
<INPUT TYPE=SUBMIT NAME=ENVIAR VALUE=ACEPTAR >
</FORM>
```

Como se puede observar en el código superior, se eliminan las restricciones que limitan la cantidad de caracteres que pueden ser capturadas y las validaciones de formato de JavaScript, con ello se permite redireccionar el código del cliente al servidor, ingresar datos no formateados a la base y provocar errores.

CONTRAMEDIDAS

Una contramedida que solo nos ayuda a proteger el código de los principiantes es codificar el JavaScript con la función **scape**, lo cual se realiza de la siguiente forma:

(Ejemplo tomado de la página http://scriptasylum.com/tutorials/encdec/encode-decode.html):

```
<script language="javascript">

var encN=1;

// ENCODES, IN UNICODE FORMAT, ALL TEXT AND THEN ESCAPES THE OUTPUT
function encodeTxt(s){
        s=escape(s);
        var ta=new Array();
        for(i=0;i<s.length;i++)ta[i]=s.charCodeAt(i)+encN;
                return
""+escape(eval("String.fromCharCode("+ta+")"))+encN;
}


// WRITES THE DECODED OUTPUT, ALONG WITH THE ESCAPED DECODER FUNCTION TO THE DIV
function writeOut(){
//Esta es una sola línea.
document.forms["fa"].output.value="<"+"script language=javascript"+">
document.write(unescape('%3C%73%63%72%69%70%74%20%6C%61%6E%67%75%61
%67%65%3D%22%6A%61%76%61%73%63%72%69%70%74%22%3E%66%75%6E%63
%74%69%6F%6E%20%64%46%28%73%29%7B%76%61%72%20%73%31%3D%75%6E
%65%73%63%61%70%65%28%73%2E%73%75%62%73%74%72%28%30%2C%73%2E
%6C%65%6E%67%74%68%2D%31%29%29%3B%20%76%61%72%20%74%3D%27%27
%3B%66%6F%72%28%69%3D%30%3B%69%3C%73%31%2E%6C%65%6E%67%74%68
%3B%69%2B%2B%29%74%2B%3D%53%74%72%69%6E%67%2E%66%72%6F%6D%43
%68%61%72%43%6F%64%65%28%73%
31%2E%63%68%61%72%43%6F%64%65%41%74%28%69%29%2D%73%2E%73%
75%62%73%74%72%28%73%2E%6C%65%6E%67%74%68%2D%31%2C%31%29%
29%3B%64%6F%63%75%6D%65%6E%74%2E%77%72%69%74%65%28%75%6E%
65%73%63%61%70%65%28%74%29%29%3B%7D%3C%2F%73%63%72%69%70%74%3E'));

dF('"+encodeTxt(document.forms['fa'].f2.value)+"')
<"+"/script>";
}
```

Lo anterior pasa el código a formato unicode el cual es legible para la computadora pero no para el usuario, el resultado anterior se incrustaría de la forma siguiente:

```
<html>

<script language=javascript>
//Está es una sola línea.
document.write(unescape('%3C%73%63%72%69%70%74%20%6C%61%6E%67%75%61
%67%65%3D%22%6A%61%76%61%73%63%72%69%70%74%22%3E%66%75%6E%63
%74%69%6F%6E%20%64%46%28%73%29%7B%76%61%72%20%73%31%3D%75%6E
%65%73%63%61%70%65%28%73%2E%73%75%62%73%74%72%28%30%2C%73%2E
%6C%65%6E%67%74%68%2D%31%29%29%3B%20%76%61%72%20%74%3D%27%27
%3B%66%6F%72%28%69%3D%30%3B%69%3C%73%31%2E%6C%65%6E%67%74%68
%3B%69%2B%2B%29%74%2B%3D%53%74%72%69%6E%67%2E%66%72%6F%6D%43
%68%61%72%43%6F%64%65%28%73%31%2E%63%68%61%72%43%6F%64%65%41
%74%28%69%29%2D%73%2E%73%75%62%73%74%72%28%73%2E%6C%65%6E%67
%74%68%2D%31%2C%31%29%29%3B%64%6F%63%75%6D%65%6E%74%2E%77%72
%69%74%65%28%75%6E%65%73%63%61%70%65%28%74%29%29%3B%7D%3C%2F
%73%63%72%69%70%74%3E'));
//Esta es una sola línea.
dF('%264DTDSJQU%2631MBOHVBHF%264E%2633kbwbtdsjqu%2633%264F%261E
%261B00%2631TBNQMF%2631TDSJQU%2631%26342%261E%261Bbmfsu%2639
%2638Ifmmp%2631Xpsme%2638%263%3A%264C%261E%261B%264D0TDSJQU
%264F%261E%261B%261E%261B%264DTDSJQU%2631MBOHVBHF%264E
%2633kbwbtdsjqu%2633%2631TSD%264E%2633zpvs%60kt%60gjmf/kt%2633%264F
%261E%261B00%2631TBNQMF%2631TDSJQU%2631%26343%2631.%2631DBMMJOH
%2631BO%2631FYUFSOBM%2631KT%2631GJMF%261E%261B%264D0TDSJQU%264F
%261E%261B1');
</script>
<body>
                Hola :)
</body>
</html>
```

Lo anterior a ojos expertos es fácil de revertir, pero sirve para dificultar las cosas un poco, otro problema que tiene es el aumento de tamaño del archivo haciendo más lentas las descargas además de generar un dilema en la comunidad desarrolladora , donde se debate que el código sea libre y abierto.

2.4 OCULTACIÓN DE ARCHIVOS EN LA URL Y EL CÓDIGO HTML

Como se ha visto los navegadores WEB en su barra de dirección muestran la URL, la cual tiene el nombre de los archivos que procesan las peticiones de los clientes, es por ello que los programadores ocultan la barra de direcciones eliminando elementos del navegador, pero esto no funciona en todos los navegadores ni en todos los S.O, es por ello que tratan de ocultar el código, las rutas y los nombres de archivos a los usuarios que en determinado momento lograsen evadir los controles de seguridad.

La solución propuesta fue encapsular el código dentro de un FRAME (cuadro), su finalidad es no dar nombres de archivos y variables que pudieran proporcionar pistas al posible atacante, para infiltrarse en el flujo del sistema u obtener información que pudiese ser útil para establecer el perfil del sistema.

Está forma de protección (FRAME para encapsular el HTML) es fácil de evadir y además requiere de una archivo central, pero es una capa más que nos hace ganar tiempo; lo único que se tiene que hacer para eludirlo es descargar el código en la computadora, donde se pueden encontrar los nombres de los archivos principales.

Por ejemplo, descargamos la página con el siguiente comando (Linux o Windows):

```
bash$ wget http://algunSitio.com/archivo.html
```

Una vez con el código en nuestro poder, podemos observar el nombre de los archivos que requiere para funcionar, como se aprecia en el siguiente segmento de código en la propiedad SRC de la etiqueta FRAME.

```
<FRAMESET cols="33%,33%,33%">
  <FRAMESET rows="*,200">
     <FRAME src="contenidos_del_marco1.html">
     <FRAME src="contenidos_del_marco2.gif">
  </FRAMESET>
  <FRAME src="contenidos_del_marco3.html">
  <FRAME src="contenidos_del_marco4.html">
</FRAMESET>
```

Una vez identificados los contenidos, se obtienen los demás códigos de igual manera que la vez anterior con "**wget http://algunSitio.com/contenidos_del_marco4.html**", luego entonces se continúa el análisis.

Está no es una medida de protección recomendada en lo más mínimo, ya que permite inyectar código de otros sitios WEB, además del robo del certificado digital, facilitando los ataques de tipo **Phishing** (método de engaño) con la suplantación de sitios y de certificados digitales.

2.5 EVITAR DESCARGAS DEL SITIO

La finalidad de evitar descargas es aumentar el tiempo necesario para que el sitio fuese duplicado; y las medidas adoptadas para ello son:

- Ocultar el código HTML en “frames” (inseguros).
- Eliminar la mayor parte de referencias que se encuentren dentro del código.
- Usar la clave aleatoria (vista anteriormente) para ocultar los archivos.
- Codificar el JavaScript (visto previamente).

La idea principal para evitar las descargas automatizadas, es haciendo que las ligas de HTML apunten siempre al mismo archivo, para que las herramientas no puedan realizar una búsqueda recursiva, lo cual derivada en la obtención de nombres de archivos, variables y directorios ver, la Fig. 2.5.

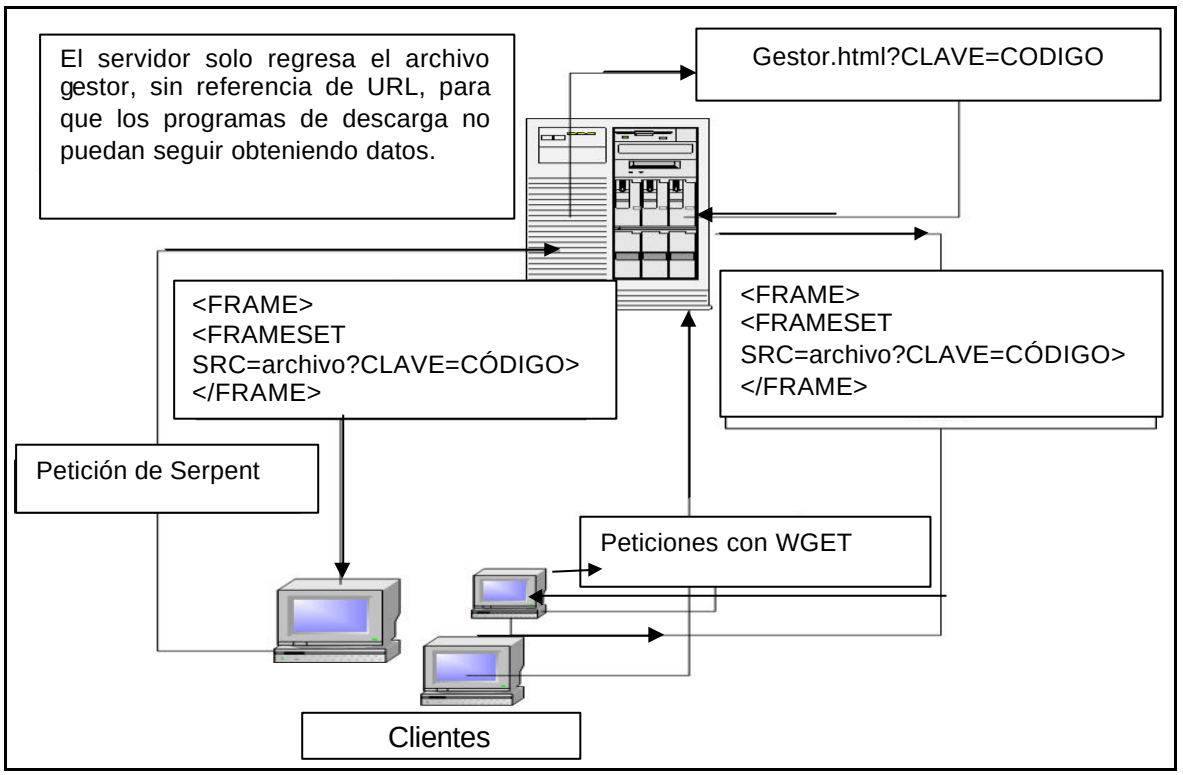

Fig. 2.5 Se muestra como las herramientas siempre obtienen el mismo archivo.

La asignación de nombres aleatorios para los directorios se logra almacenando los nombres dentro de un arreglo en un archivo de configuración principal, esto combinado con la generación de nombre aleatorios visto previamente, da un resultado como el siguiente:

```
<?php
        //archivoDirectoriosGlobales.php
```

```
        $arrDirectorio[asdasasda]  = ”directorio/Archivo1”;
        $arrDirectorio[adasdfhds]   = ”directorio/Archivo2”;
        $arrDirectorio[a235sdgsa]  = ”directorio/Archivo3”;
?>
```

Los nombres aleatorios representados por los caracteres ***asdasasda***, son los que se le asignarían a las ligas, para que no puedan ser interpretadas por los programas como Serpent, wget entre otros, el resultado visto el código HTML se vería de la siguiente manera:

```
. . .
 <a href=”archivoCentral.html?archivo= asdasasda;”>
. . .
```

Como el archivo que se invoca es siempre el mismo “**archivoCentral.html**” los programas no podrían escalar dentro el árbol de directorios; y la clave **asdasasda,** no proporciona una ruta con la cual pueda seguir escalando en el servidor.

Para evitar las descargas con TELNET, se dió de baja el servicio.

2.6 EVITAR ACCESOS Y DATOS NO VALIDOS EN EL FLUJO DEL SISTEMA

Se logra colocando instrucciones de validación en cada archivo para evitar su ejecución de manera directa, la idea principal de esto radica en que el sistema debe pasar por un archivo principal o realizar un paso previo dentro del cual se genera una cadena o huella que debe presentar en el paso siguiente, en la Fig. 2.6 se muestra las validaciones y pasos que realiza el servidor.

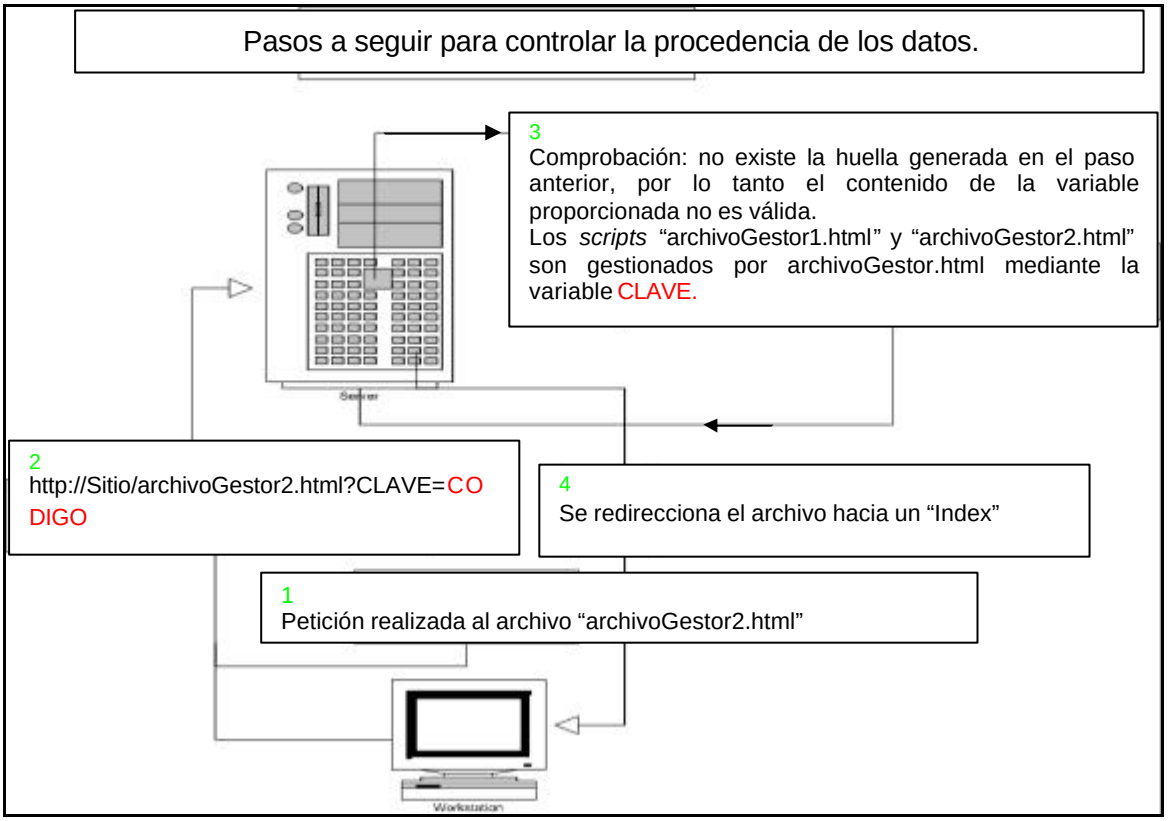

Fig. 2.6 Muestra como el servidor verifica que los pasos anteriores hayan sido realizados.

La finalidad es que los usuarios no realicen las siguientes actividades:

- Saltarse pasos necesarios en la aplicación.
- Manipular variables en archivos sensibles que no estén protegidos.
- Evitar una puerta trasera, si el usuario conoce el código y el comportamiento del mismo.

La idea principal radica en que para llegar al archivo "B", se tiene que pasar previamente por el archivo "A", dentro del cual se genera una variable o huella que se almacena en el servidor y existe sólo para la sesión de usuario que la solicita, y que además tiene las siguientes características, es temporal y de un tiempo de

vida corto, a continuación se muestra de manera resumida el funcionamiento del método de protección propuesto:

1- El archivo “A” genera la huella, en este caso una frase simple que se almacena en sesión.

```
<?php

    //GENERA LA VARIABLE
    $_SESSION[nombre_confuso]="PASO_EN_A";

?>
```

2- El archivo “B” verifica la existencia de la huella generada en el paso previo.

```
<?php

        If($_SESSION[nombre_confuso]=="PASO_EN_A")
            // SI EXISTE ENTONCES REALIZO LOS PASOS PREVIOS
            //Y EL FLUJO NO SE Ha CORROMPIDO.
?>
```

Un esquema difícil de implementar en las aplicaciones WEB, debido a los diferentes requerimientos que tienen, pero es necesario en los archivos de administración del sistema.

2.7 PROHIBIR EL ACCESO AL ÁRBOL DE DIRECTORIOS

La protección se logra colocando archivos de tipo index (índice) que interpreta directamente el servidor de aplicaciones WEB (Apache) y despliega, otra herramienta usada es **Apache Guardián** cuya función es notificar los intentos de intrusión a los árboles de directorios además de prohibirlos ver la Fig. 2.7.

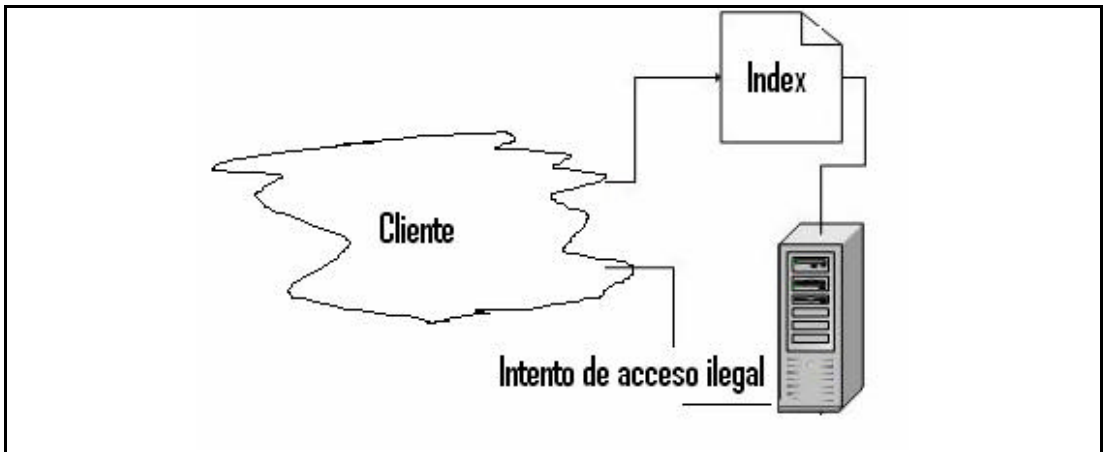

Fig. 2.7 El acceso al árbol de directorios es controlado y protegido mediante archivos index y herramientas como Apache Guardian.

Las directivas para interpretar los index se agregan en el archivo "httpd.conf" de Apache y las líneas que se tienen que anexar son las siguientes:

```
#
# DirectoryIndex: sets the file that Apache will serve if a directory
# is requested.
#
# The index.html.var file (a type-map) is used to deliver content-
# negotiated documents.  The MultiViews Option can be used for the
# same purpose, but it is much slower.
#
DirectoryIndex index.html index.html.var INDEX.EXTENCION
```

2.8 CIFRADO DE INFORMACIÓN

El cifrado se realiza mediante un algoritmo de llave pública y privada, conexión segura SSL, habilitando MODSSL en Apache, ver Fig. 2.8.

La finalidad de cifrar la información es evitar ataques como:

- Hombre en medio.
- Alteración de la información que viaja a través de la Internet.
- Captura de variables sensibles, entre otros.

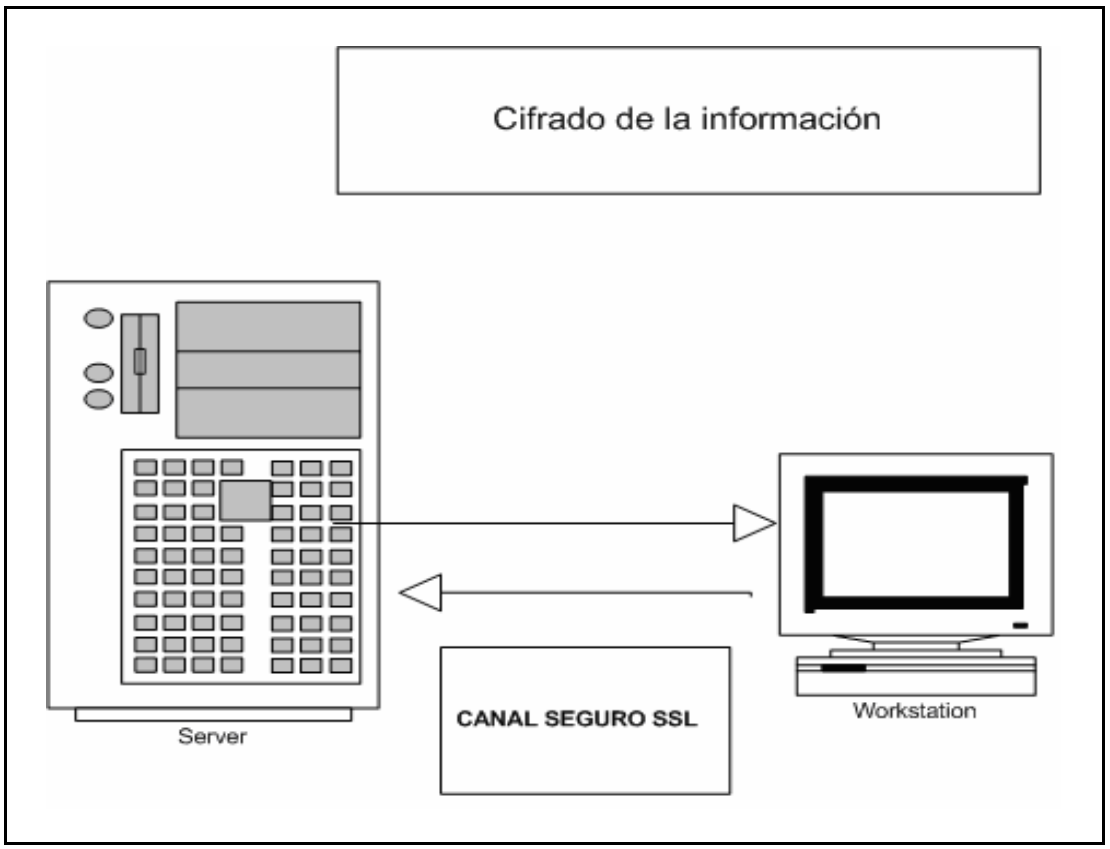

Fig. 2.8 La información entre el cliente y el servidor viaja cifrada, para evitar que está pueda ser interpretada por personas ajenas al proceso.

Cada uno de los ataques mencionados en la parte superior tiene consecuencias diversas en la información transferida y en la aplicación, es por ello que los datos deben ser cifrados.

2.9 CONCURRENCIA

El problema de la concurrencia en la BD se soluciona mediante un algoritmo que verifica la existencia del registro antes y después de realizar una transacción, ya que Postgres mezclado con PHP no la maneja de manera correcta y se tiene que programar, haciéndola poco fiable debido a que se involucra el error humano, ver la Fig. 2.9.

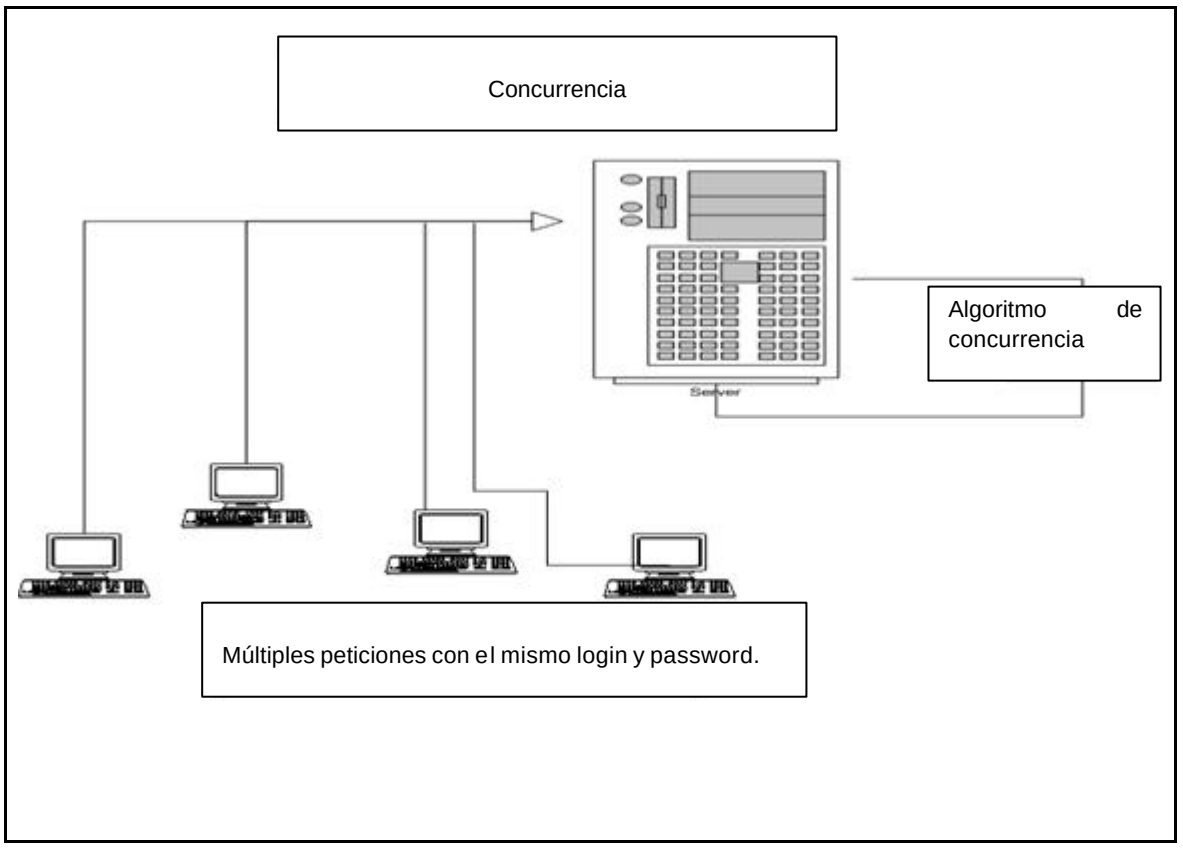

Fig. 2.9 Votar con la misma clave es muy malo para el sistema, sin un control correcto de la concurrencia, se altera el número de votos**.**

Las consecuencias de no controlar la concurrencia en la BD son de alto impacto en los resultados, ya que si no se implementan se pueden ingresar una cantidad elevada de datos con el mismo usuario, dando un resultado poco coherente y alterado, una condición equivalente al de una urna embarazada.

2.10 SIMULACIÓN DE ATAQUES

La simulación de ataques es necesaria para verificar que las medidas implementadas al servidor respondan de manera adecuada, entre las pruebas que se deben realizar están las siguientes:

- Ingresar consultas SQL en el código HTML que está al descubierto, para verificar los filtros.
- Inyectar errores para verificar el correcto manejo de los mismos.
- Intentar descargas de la aplicación con comandos y herramientas, para corroborar el funcionamiento del algoritmo que genera claves aleatorias.
- Ingreso masivo de datos con el mismo usuario, para verificar el manejo de la concurrencia.
- Revisión de las galletas "cookies", para evitar que almacenen información sensible, tal como usuarios y claves.
- Intentar acceder directamente a los archivos del sistema, tecleando su nombre en la URL, para corroborar el funcionamiento del algoritmo de huella.
- Intentar acceder al árbol de directorios, para comprobar el funcionamiento de la herramienta Apache Guardian y la implementación correcta de los índices.
- Verificar que las extensiones de los archivos estén dadas de alta en Apache, para que no se pueda obtener el código fuente.
- Verificar que los FRAMES utilizados no sean susceptibles de ser manipulados para inyectar código de páginas externas.
- Revisar que el HTML y JavaScript entregado a los clientes no tenga comentarios ni datos que puedan ser útiles para los atacantes.
- Eliminar todos los elementos colocados por default en el servidor.
- Redireccionar las peticiones desde una máquina local al servidor, para verificar el manejo correcto de variables.
- Checar el filtro de variables propias del lenguaje PHP algunos ejemplos son:
    - $_GET, $_POST, $_SESSION, $HTTP_POST_VARS, $HTTP_GET_VARS.

Para realizar este tipo de pruebas es necesario que el equipo domine:

- El lenguaje de programación PHP, para manipular variables propias del sistema.
- El lenguaje de programación JavaScript, para evadir las validaciones.
- El Lenguaje de marcas HTML, para analizar los archivos que procesan los datos y las variables que maneja.
- El lenguaje SQL, para explotar las vulnerabilidades y manipular las consultas.
- Experiencia en desarrollo de sistemas WEB, para conocer la estructura de la aplicación y la forma de evadir sus validaciones.

- Administración de Linux, para interpretar los mensajes de error y aprovecharlos, así como conocer los comandos con los cuales se puede interrogar al firewall, servidor y las instrucciones para capturar lo que sale de la máquina.
- Manejo de cookies, para analizar el contenido que aloja.
- Conocimientos de herramientas de pruebas, algunas de ellas son Brutus, Serpent, OpenStat.
- Conocimientos de configuración de Red, para manipular los DNS en redes pequeñas y lograr así el robo de usuarios y contraseñas.

En la Fig. 2.10 se ilustra la manera en que se llevan a cabo los ataques desde varias máquinas.

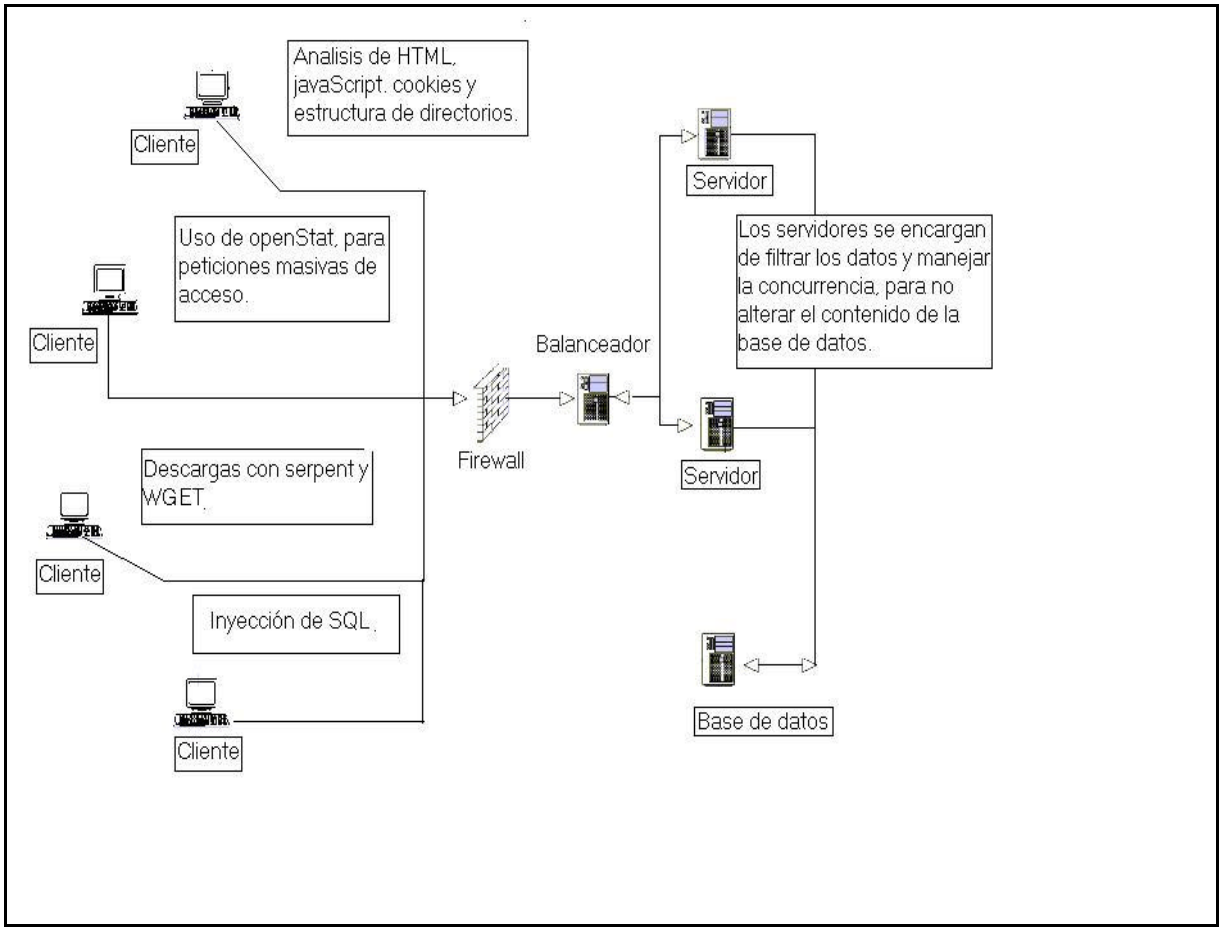

Fig. 2.10 Simulación de ataques.

Las pruebas listadas anteriormente son necesarias para comprobar la fortaleza del sistema, cabe mencionar que todas las medidas anteriores solo protegen a la aplicación WEB y en menor medida al Servidor de aplicación (Apache).

3.

DESCRIPCIÓN DEL SISTEMA

3.1 AVISO DE APERTURA

Aviso que indica el día y la hora de inicio del proceso, así mismo se indica la hora de término Fig. 3.1.

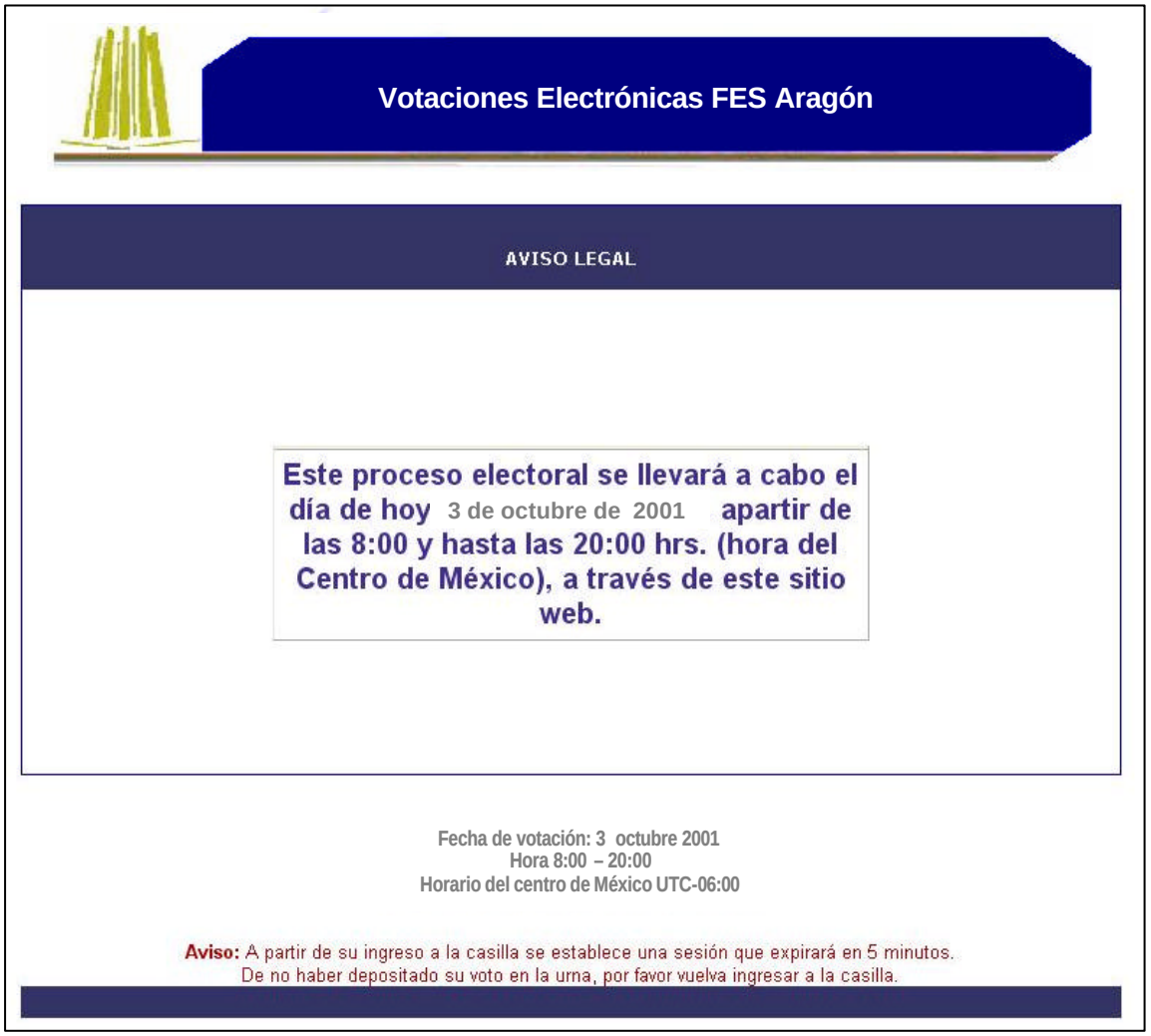

Fig. 3.1 Mensaje previo al inicio de las votaciones.

3.2 AVISO LEGAL

Reglamentos que rigen el proceso en curso, se muestran antes de realizar cualquier actividad en el sistema, no se puede utilizar si no se aceptan los términos y para ello, hay que presionar el botón que dice *ENTERADO* Fig. 3.2.

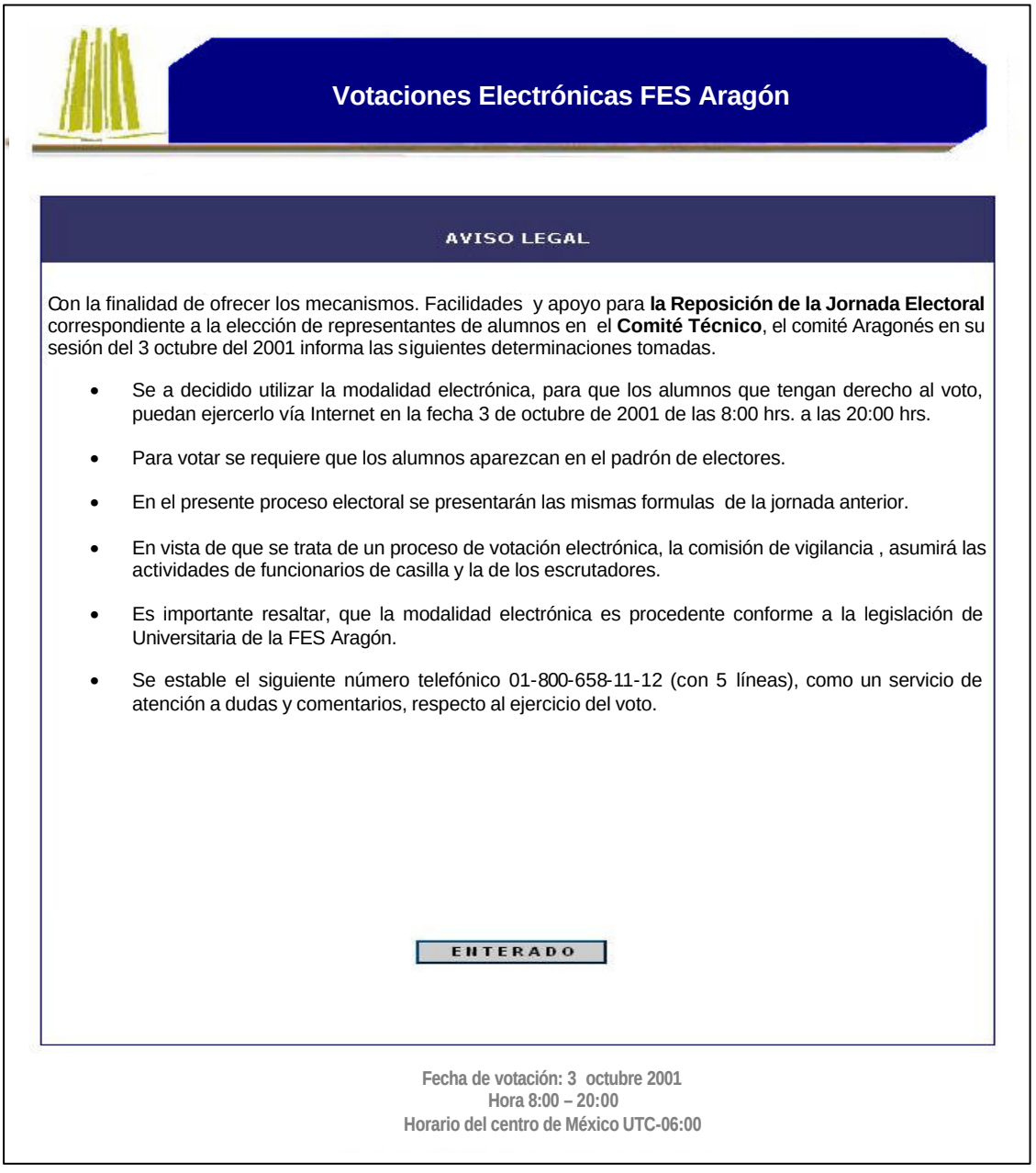

Fig. 3.2 Presentación de las reglas, que rigen el proceso de voto.

3.3 AUTENTICACIÓN

En esta parte del sistema se muestran dos cajas de texto en las cuales se ingresan el número de cuenta del usuario y su contraseña respectiva, datos necesarios para poder ejercer su derecho al voto Fig. 3.3.

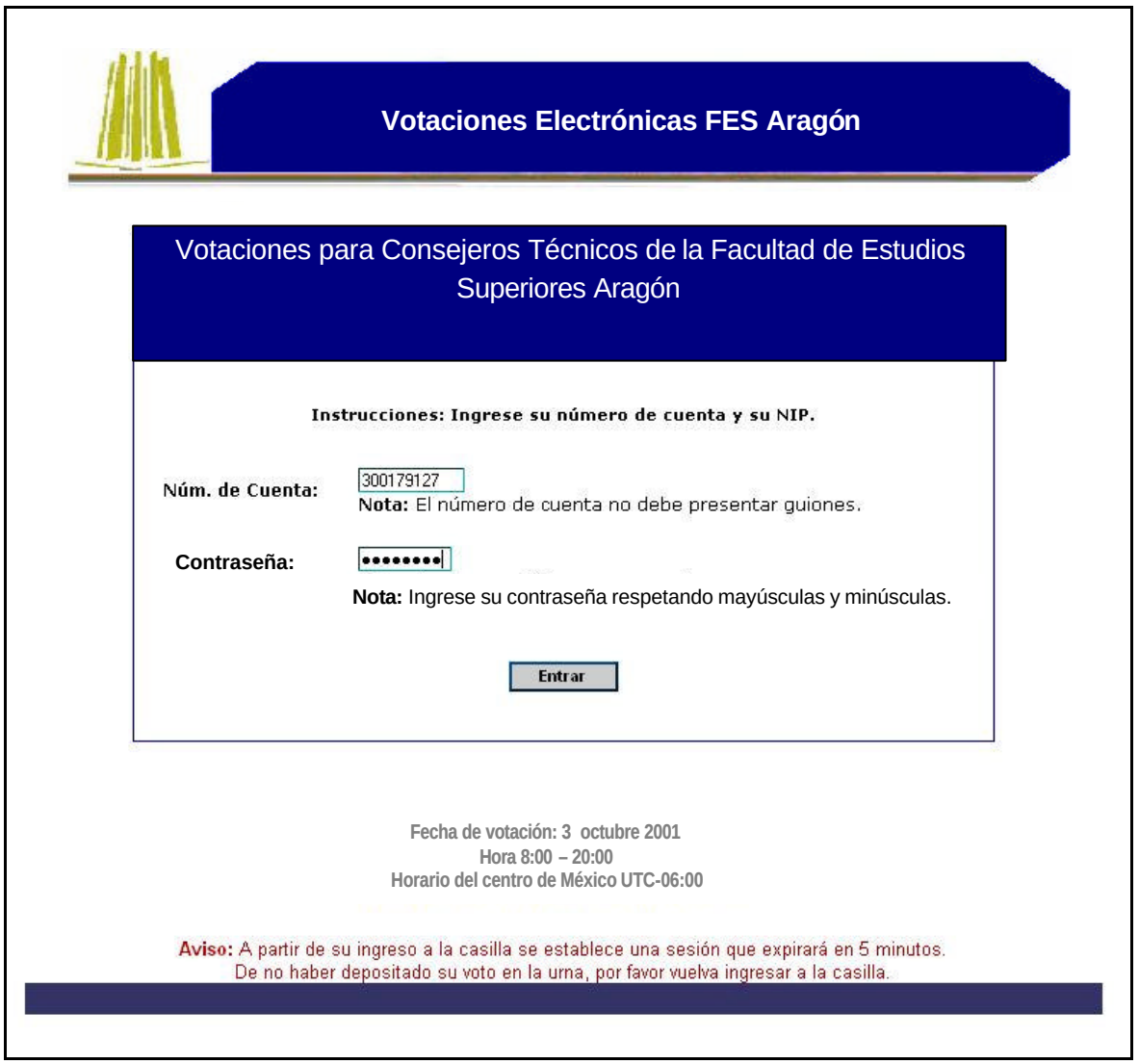

Fig. 3.3 Pantalla de seguridad para identificar a los votantes.

3.4 BOLETA

La boleta presenta información, necesaria para seleccionar su formula, candidato u representante elegido, solo tienen que seleccionar los candidatos haciendo click sobre la caja que aparece al final del renglón de cada propietario y suplente, la anulación del voto se hace seleccionando ambos, para dejar el voto en blanco no se debe seleccionar ninguna opción y para que sea valido se tendrá que seleccionar solamente una opción de las que se muestran Fig.3.4.

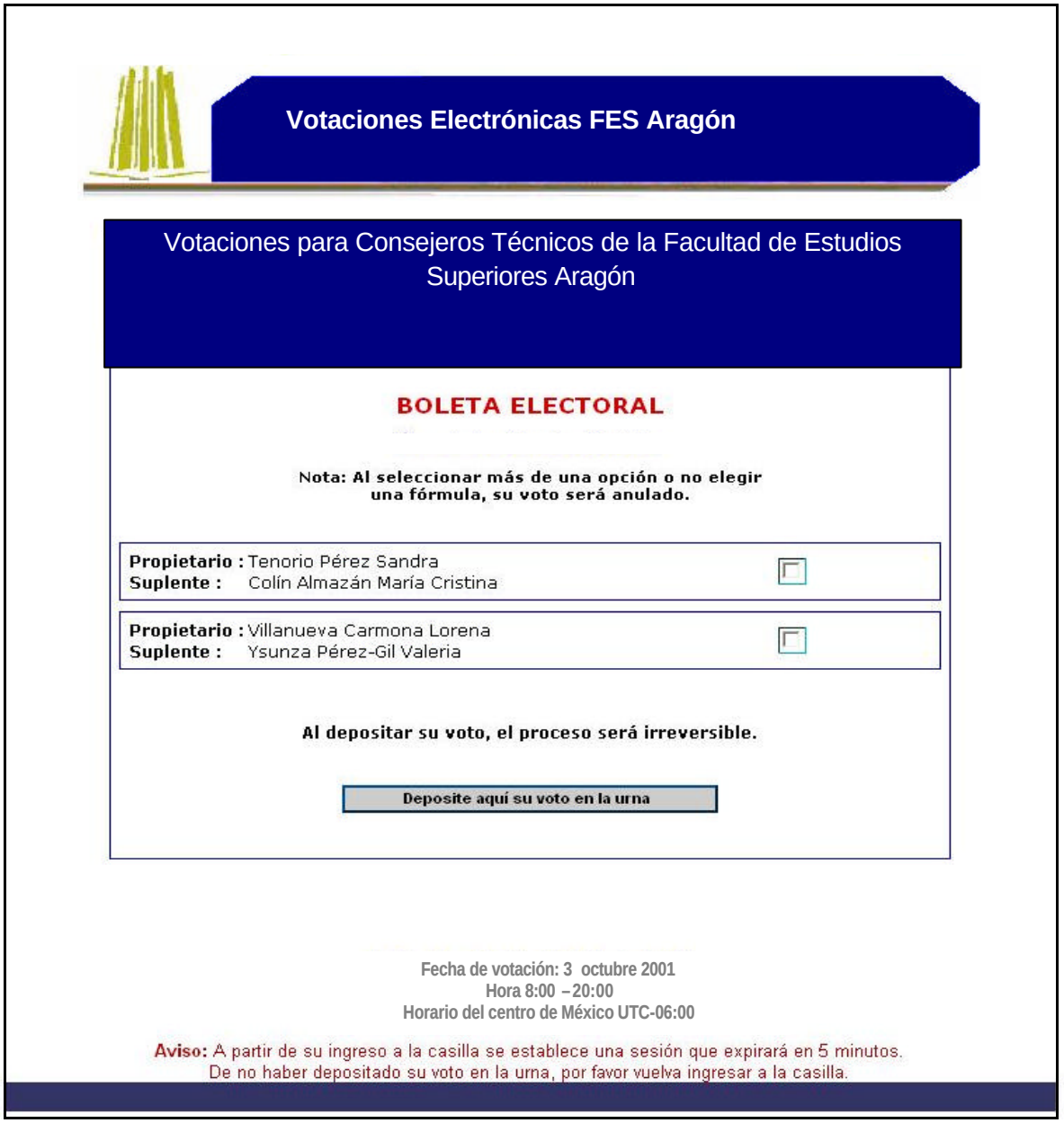

Fig. 3.4 Boleta Electoral.

3.5 CONFIRMAR ACCIÓN REALIZADA

Esta pantalla muestra un mensaje de confirmación indicando que la opción seleccionada ha sido depositada con éxito. A partir de este momento no podrá votar nuevamente, ya que su voto ha sido depositado y su contraseña ha sido eliminada, para que no pueda ser utilizada nuevamente Fig. 3.5.

Fig. 3.5 Confirmación del voto.

3.6 INTENTO DE VOTO DUPLICADO

La pantalla siguiente sirve para indicar cuando algún usuario intenta ingresar nuevamente, una vez que ya ha emitido su opinión, voto o registro, esto es para evitar duplicidades de votos en la urna electrónica Fig. 3.6.

Fig. 3.6 Advertencia del sistema indicando que no es posible votar nuevamente.

3.7 USUARIO O CONTRASEÑA INCORRECTOS

Esta pantalla indica cuando el usuario ha ingresado una contraseña o número de cuenta incorrecto, cuando esto sucede se le pide que lo haga nuevamente.

Si el usuario después de varios intentos no puede ingresar al sistema, tiene un teléfono a su disposición el cual se muestra en el aviso legal, donde se le informarán los pasos que tiene que seguir para poder votar o bien, la razón por la cual no puede ingresar al sistema Fig. 3.7.

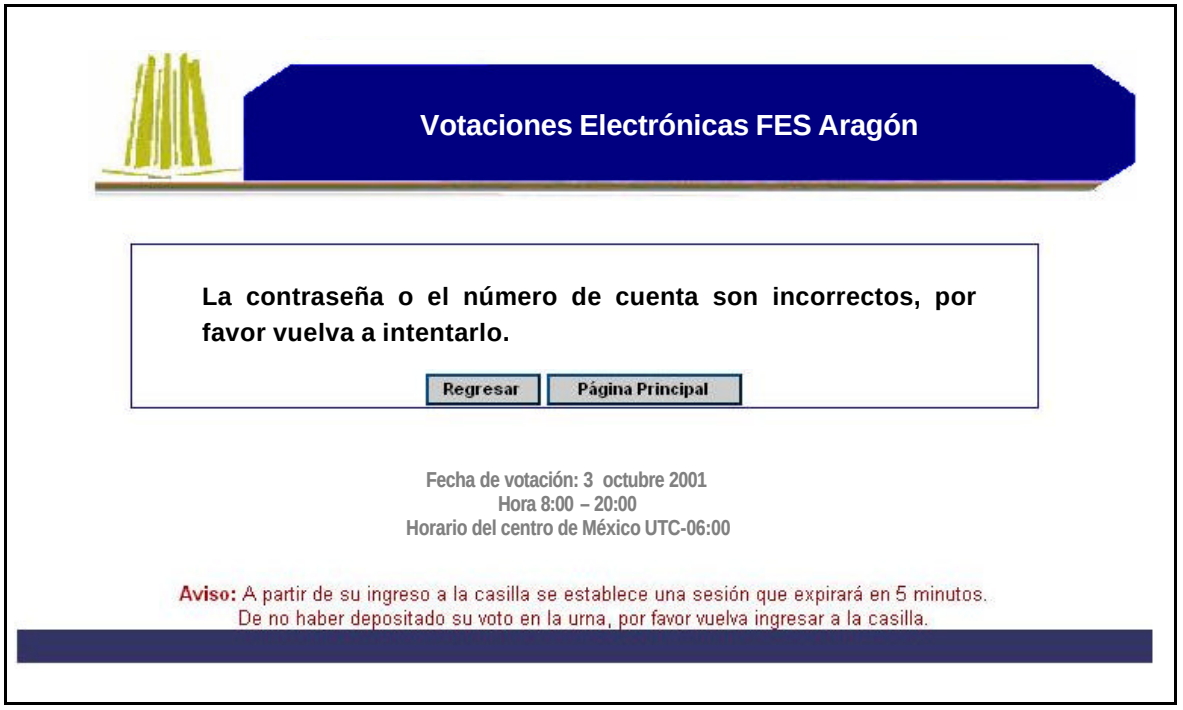

Fig. 3.7 Esta imagen indica que algo está incorrecto en su contraseña o número de cuenta.

3.8 FIN DEL PROCESO

Este mensaje es desplegado para indicar que el proceso en curso ha terminado, es muy necesario para que las personas estén informadas del momento en que finalizaron las votaciones, se activa a las 20:00 hrs., impidiendo el acceso a quienes intentan accesar, pero permitiendo terminar a los que se encuentren aún dentro del sistema, y los cuales tienen un tiempo límite de 5 minutos para llevar ha cabo su voto Fig. 3.8.

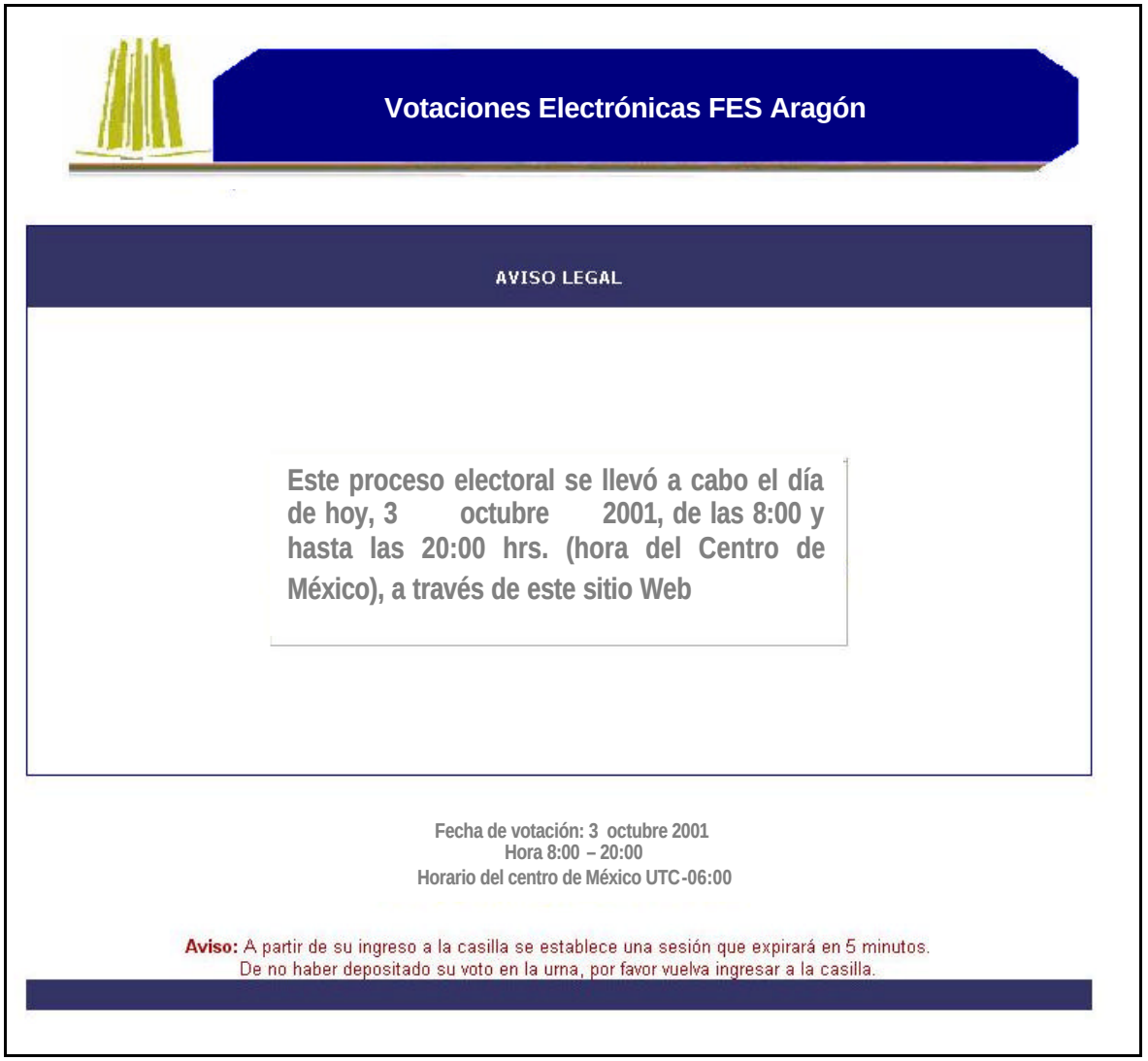

Fig. 3.8 Mensaje que cierra el proceso de votación.

3.9 ACCESO INCORRECTO

Esta pantalla se despliega cuando el usuario intenta un acceso inválido por medio de la URL, además de que también es parte del manejo de errores, así como de la protección de archivos y el árbol de directorios, su finalidad es evitar que usuarios maliciosos obtengan información sobre el sistema Fig. 3.9.

Fig. 3.9 Mensaje de protección de directorios.

4.

BASE DE DATOS

4.1 REPLICACIÓN

Dentro del siguiente tema solamente se describe el esquema de redundancia para proteger la información y la herramienta involucrada en la replicación, en este caso DBMIRROR.

La replicación nos permite transferir información de una BD a otra por lo que es un factor crítico, debido a la información que se maneja, ya que si una BD falla se pone en riesgo la disponibilidad del sitio, para resolver esta parte se implemento la herramienta antes mencionada la cual gestiona la transferencia de información entre bases de datos.

La herramienta maneja un esquema de replicación maestro – esclavo, esto es, que los cambios que se realicen sobre la base maestra serán reflejados después de un tiempo determinado (segundos o minutos) en la base de datos esclava, la cual debe tener exactamente la misma estructura que la principal, en lo referente a tablas.

La forma de operar del esquema de disponibilidad es el siguiente: La información que llega a las Bases de Datos , primero tiene que pasar por un "Firewall", cuya función es filtrar la información que a su vez entrega a un balanceador de carga etapa necesaria para distribuir las peticiones entre los diferentes servidores de la aplicación WEB, así mismo, los servidores de aplicación concentran toda la información en una BD (Maestra) y es en este punto donde se replican los datos a otra BD (esclava), está última entra en funcionamiento si la BD principal colapsa (el proceso de cambiar la BD de esclava a maestra se realiza de forma manual), en la siguiente Fig.4.1 se ilustra la manera en que se implementa dicho esquema.

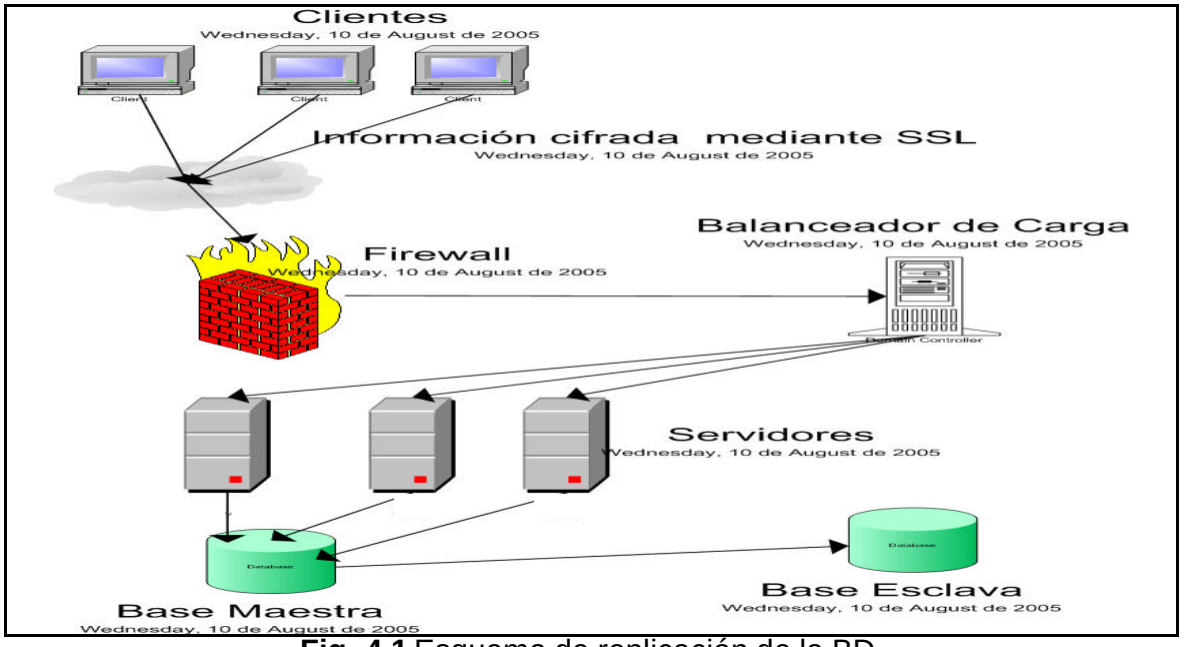

Fig. 4.1 Esquema de replicación de la BD.

5.

CONCLUSIONES FINALES

5.1 CONCLUSIONES EN APLICIONES WEB

Respecto a las aplicaciones WEB se puede concluir que los errores más comunes que provocan los fallos de seguridad son:

- No filtrar las instrucciones SQL en el servidor.
- Manejar los errores de manera inadecuada.
- Ejecutar la aplicación con demasiados permisos.
- Confiar en las restricciones que proporciona JavaScript.
- Desconocer el comportamiento de que forma se gestiona la BD, con el lenguaje que la utiliza.
- Subir archivos sin validación o bien no colocarlos dentro del directorio de la aplicación WEB, donde los interpreta el servidor y los entrega a los usuarios.
- Enviar las galletas "cookies" con información sensible del usuario.
- Incluir FRAMES de HTML, que reciben como parámetros ligas de manera dinámica.
- Dejar sin protección los directorios, es decir no colocar index.
- Falta de protección el flujo del sistema.
- No dar de alta las nuevas extensiones donde se busca código PHP para interpretar.
- Dejar la instalación del servidor WEB por default.

Y las actividades que no deben dejar pasar por alto los desarrolladores son:

- Restringir eventos en JavaScript.
- Cifrar la información depositada en la BD.
- Sincronizar los accesos a la BD (Si el manejador no los realiza de manera correcta).
- Proteger el código HTML generado.
- Implementar las funciones de cifrado necesarias.
- Modificar las extensiones de archivos y notificar a los administradores de las nuevas extensiones creadas.
- La simulación de ataques.
- Pruebas de sincronización en sistemas de muy alto tráfico.
- Que el algoritmo de cifrado esté integrado con la aplicación y no con el Sistema Operativo, para evitar problemas de concurrencia.

- Analizar a detalle el dinamismo de la aplicación para que no comprometa la seguridad del sistema.

- Realizar pruebas en navegadores diferentes (cross-server), para verificar el funcionamiento correcto de las funciones de JavaScript.

- Ocultar y proteger el nombre de directorios y archivos de la aplicación, con números aleatorios, utilizar herramientas tales como Apache Guardián, para el caso de PHP, ya que un patrón indicaría de que forma está organizada la estructura de directorios.

- Proteger el flujo del sistema estructurando la aplicación, de tal manera que deje una huella de los archivos por los cuales pasa, para poder rastrear y controlar accesos no válidos.

- Fortalecer e Implantar métodos de autenticación dirigidos al usuario, con métodos como el HumanCheck para eliminar ataques de fuerza bruta a las claves de acceso, implementar algoritmos de contraseña fragmentada, es decir pedir al usuario solamente 3 letras aleatorias de su contraseña para evitar que un posible sitio malicioso capture la contra seña completa.

- Realizar validación de variables dentro del servidor, debido a que JavaScript no es un lenguaje orientado a dar seguridad y, a que nunca se debe confiar en la información procedente del cliente ya que las validaciones del lado del cliente son fáciles de evadir.

- Evitar inyección de código, implementando el filtrado de variables en los datos utilizados para consultar la Base de Datos o bien separar correctamente las variables numéricas de las alfanuméricas y controlar la longitud de las variables alfanuméricas.

- Extraer en la medida de lo posible las variables del arreglo al que correspondan, por ejemplo $var= HTTP_POST_VARS[‘var’], y evitar que variables sensibles del flujo viajen a través de la URL para evitar una posible substitución, se recomienda almacenarlas en $_SESSION y extraerlas del mismo arreglo para evitar que sean suplantadas.

- Filtrar las variables propias del lenguaje, en este caso PHP, tales como HTTP_POST_VARS, HTTP_GET_VARS, _SESSION, GET_VARS, POST_VARS, para evitar su eliminación o alteración, mediante los datos que son recibidos desde el cliente.

- Si se realiza la creación de variables dinámicas comprobar que se generen antes de asignar valores a las variables sensitivas, por ejemplo: Usuarios, contraseñas, nombre de base de datos entre otros, para evitar que sean

sustituidas con el envió de variables, ya que representa un riesgo solo si el atacante conoce el código fuente.

- Verificar que las variables procedan del archivo que las manda o genera, realizando pruebas descargando el código fuente (HTML), para redireccionar los datos y comprobar que no sean válidos al ser recibidos dentro del servidor, esto es que se modifique el código dentro del cliente y que la aplicación los tome como válidos.

- Auditar el código para evitar instrucciones maliciosas por parte de los desarrolladores.

5.2 CONCLUSIONES EN LA BASE DE DATOS

Respecto a las bases de datos podemos decir que:

- El esquema de base no fue el más óptimo, la razón es que la estructura presentada tiene punto único de falla, por que si la BD principal cae, se pueden tener problemas de pérdida de datos y no se levanta de manera automática, pero se dió resultados.

- Para ese esquema se tiene que realizar el cambio de manera manual si un servidor se cae, (ya que no es maestro – maestro) fue debido a que no se cuenta con presupuesto para manejadores comerciales y Postgres no lo soporta.

- Es una opción libre.

- Demostró bastante estabilidad durante la implantación.

- Es de fácil aprendizaje e implementación, lo cual reduce el tiempo necesario para dominarlo.

- No importa que el manejador de BD sea el más seguro y tenga todas las actualizaciones, si no se cuidan los aspectos de protección desde la aplicación WEB, ya que es un elemento que interacciona con las BD y los servidores.

Glosario

PHP: (acrónimo de "PHP: Hypertext Preprocessor"), Es un lenguaje interpretado de alto nivel embebido en páginas HTML y ejecutado en el servidor.

HTML: Acrónimo de HiperText Markup Language (Lenguaje de Marcas de Hipertexto). El lenguaje de código que se usa para crear documentos de hipertexto para usar en las páginas de Internet.

JAVASCRIPT: Lenguaje de script desarrollado por Netscape, basado en la sintaxis del lenguaje Java. Se utiliza en páginas WEB HTML, para realizar tareas y operaciones en el marco de la aplicación cliente.

SISTEMA OPERATIVO: Un Sistema Operativo (SO) es el conjunto de programas básicos y utilidades que hacen que una máquina funcione y resulte útil a los usuarios.

SERVIDOR WEB: Programa que ofrece servicios de portales y páginas en Internet.

BASE DE DATOS: Explicado de forma sencilla, una base de datos es un conjunto de información almacenada sistemáticamente para su uso posterior

RDBMS: Un RDBMS es un Sistema Administrador de Bases de Datos Relacionales; RDBMS viene del acrónimo en inglés Relational Data Base Manager System.

SESION: En PHP es una instrucción que permite identificar y controlar los datos del usuario, se auxilia de cookies y almacena la información en el servidor principal, evitando está viaje.

ARREGLO: En PHP es una característica del lenguaje que permite almacenar una colección de datos bajo un mismo nombre, pero permite diferenciarlos mediante un número o nombre asociado a cada uno de los datos almacenados.

SCRIPT: Lenguaje de programación cuyo código no necesita ser compilado para ser ejecutado. Se suele llamar script a un programa o fragmento de código escrito en algún lenguaje.

ALEATORIO: Algo que carece de todo patrón regular debido a su inmunidad a cualquier ley que la forzaría a una secuencia aunque sea levemente repetitiva que podría ser considerada como patrón.

GET: Método de transferencia de datos de los formularios, cuya característica es encadenar los datos en una cadena dentro del encabezado HTTP/1.0 URL, la longitud máxima hoy día es de aproximadamente 4000 caracteres.

POST: Método de transferencia de datos de los formularios, hace uso del protocolo HTTP/1.0, no encadena los mensajes de petición al encabezado HTTP y por lo mismo soporta transferencias de información mucho más grandes.

FORMULARIO: Conjunto de etiquetas de HTML que permiten ingresar datos a los usuarios.

FRAME: Instrucción de HTML que permite agrupar código de otras páginas asignándoles orden en la pantalla.

WGET: Comando de Linux que permite descargar paginas y hasta sitios enteros de Internet.

APACHE: Herramienta que presta servicios a los usuarios, se encarga de responder las peticiones de echas por los usuarios, hace uso principalmente del lenguaje PHP.

IIS: Internet Information Server, herramienta que presta servicio a los usuarios, se encarga de responder las peticiones echas por los usuarios, hace uso principalmente del lenguaje ASP (Active Server Page).

TOMCAT: Herramienta que presta servicios a los usuarios, se encarga de responder las peticiones de páginas de Internet echas por los usuarios, hace uso principalmente del lenguaje JAVA.

NAVEGADOR: Programa que interpreta y visualiza los documentos de los diferentes sitios de Internet.

EQUIPO CLIENTE: En este documento se usa para hacer referencia a las computadoras personales de los usuarios.

BARRA DE DIRECCIONES: Elemento del navegador que presenta la dirección del sitio de Internet que se esta visitando.

WINDOWS: Sistema Operativo que se instala para administrar los recursos de la computadora personal del usuario, requiere licencia.

LINUX: Sistema Operativo que se instala para administrar los recursos de la computadora personal del usuario, no requiere licencia, excepto Red Hat.

PLATAFORMA: En la jerga que manejan los programadores, sirve para hacer referencia al sistema operativo sobre el cual

EQUIPOS DE DESARROLLO: Conjunto de personas que se encargan de crear los programas de computo.

URL: Uniform Resource Locator, es el mecanismo con el cual el World Wide Web asigna una dirección única a cada uno de los recursos de información de cualquier lugar de Internet.

MYSQL: Es una de las bases de datos más populares desarrolladas bajo la filosofía de código abierto.

POSTGRES: Es una de las bases de datos más populares desarrolladas bajo la filosofía de código abierto

INFORMIX: Empresa estadounidense, posteriormente adquirida por IBM, cuyo principal producto era un motor de base de datos relacionales del mismo nombre.

ORACLE: Sistema Manejador de base de datos relacionales, propiedad de la empresa que lleva el mismo nombre.

SYBASE: Sistema Manejador de base de datos relacionales, propiedad de la empresa que lleva el mismo nombre.

MICROSOFT SQL SERVER: Sistema Manejador de base de datos relacionales, propiedad de la empresa que lleva el nombre de Microsoft

UNION: Instrucción SQL que sirve para unir el resultado de dos consultas.

CONSULTAS ANIDADAS: Indica cuando una instrucción SQL lleva incluida otra dentro de la misma instrucción.

CONCURRENCIA: Acaecimiento de varios sucesos o cosas de manera simultanea.

LÍNEA DE COMANDOS: Se refiere a las pantallas donde se ingresan las instrucciones de los usuarios, en modo texto.

PHISHING: Método de engañar a la gente mediante correos, páginas de Internet, entre otros para el usuario proporcione información confidencial.

SITIO: Frase que hace referencia a los portales de Internet.

FLUJO DEL SISTEMA: Se hace referencia a la serie de pasos a seguir por los usuarios para utilizar algún determinado programa de manera normal.

ÁRBOL DE DIRECTORIOS: Lista de carpetas y archivos que los usuarios pueden ver.

INDEX: Se refiere a una página de presentación que muestran los servidores WEB para que no se puedan ver los directorios que este almacena.

TRANSACCIÓN: Una transacción en SQL es un conjunto de ordenes hacia el DBMS que se ejecutan en una unidad de trabajo, es decir en forma atómica.

DEFAULT: Se hace uso de esta palabra para indicar que las instalaciones se dejan con las opciones que trae recomendadas por el fabricante.

BRUTUS: Programa que se encarga de romper las contraseñas con fuerza bruta.

SERPENT: Programa que se encarga de descargas sitios enteros o por partes de Internet.

OPENSTAT: Programa que sirve para realizar pruebas de carga y estrés sobre servidores de aplicaciones WEB.

REPLICACIÓN: Método para copiar la información de una base de datos a otra cada cierto tiempo.

DBMIRROR: herramienta auxiliar de Postgresql que ayuda a replicar información de una base de datos a otra.

FIREWALL: Herramienta de hardware o software que permite aislar y filtrar la información que viaja en la red donde se instale.

BIBLIOGRAFÍA CONSULTADA

LIBROS:

Eco, Humberto. Cómo se hace una tesis.
Horton, Mike y Clinton Mugge. Claves Hackers. Mc Graw Hill.
Scambray, Joel y Mike Shema. Hackers de sitios WEB. Mc Graw Hill.

SITIOS WEB:

CODIFICACIÓN DE JAVASCRIPT Y HTML (ejemplos detallados para su implementación).
http://scriptasylum.com/tutorials/encdec/encode-decode.html

INYECCIÓN DE SQL.
http://php.mirrors.ilisys.com.au/manual/es/security.database.sql-injection.php
http://www.nextgenss.com/papers/HackproofingMySQL.pdf (análisis muy completo de "SQL INJECTION").

http://www.spidynamics.com/papers/SQLInjectionWhitePaper.pdf (Muchos ejemplos de "SQL INJECTION").

CONFIGURACIÓN DE PERMISOS DE MYSQL.
http://www.devshed.com/c/a/MySQL/The-MySQL-Grant-Tables/2/

DISEÑO DE BASE DE DATOS SEGURO EN MYSQL
http://www.securityfocus.com/infocus/1667

INSTALACIÓN Y CONFIGURACIÓN DE DBMIRROR
http://www.infocopter.com/know-how/postgresql/replication/installation.html

REEMPLAZO DE URL
http://www.hispasec.com/directorio/laboratorio/Software/

CONCEPTOS
http://es.wikipedia.org/wiki/Portada (Acceso al portal para búsqueda de conceptos por categorías)

CONCEPTOS
http://es.wikipedia.org/wiki/Categor%C3%ADa:Internet

CONCEPTOS
http://perseo.cs.buap.mx/~danguer/projects/papers/final/php/php01.html
(Diferencia entre GET y POST)

GUÍAS DE SEGURIDAD DE APACHE
http://www.cgisecurity.com/webservers/apache/

PÁGINA PARA DECODIFICAR SITIOS EN INTERNET
http://gooby.ca/decrypt/

NOTICIAS DE HISPASEC
http://www.hispasec.com/directorio/laboratorio/Software/

COMO DISEÑAR APLICACIONES WEB SEGURAS
http://www.adobe.com/devnet/server_archive/articles/design_secure_webapps.html

MARCADO DE DATOS HTML Y JAVASCRIPT
http://www.iec.csic.es/criptonomicon/tainting.html

PHISING BASADO EN TROYANOS
http://www.hispasec.com/unaaldia/2508

INYECCIÓN DE SQL EN PHP
http://php.mirrors.ilisys.com.au/manual/es/security.database.sql-injection.php

DISEÑO DE BASE DE DATOS SEGURA EN MYSQL
http://www.securityfocus.com/infocus/1667

LISTADO DE HERRAMIENTAS DE SEGURIDAD
http://boran.linuxsecuri ty.com/security/sp/toolsdigest/2001/tools20001229.html

CINCO VULNERABILIDADES COMUNES EN LAS APLICACIONES WEB
http://www.securityfocus.com/infocus/1864

LISTA DE HERRAMIENTAS QUE APROVECHAN VULNERABILIDADES
http://packetstormsecurity.nl/Crackers/wordlists/

DESCRIPCIÓN PHISING
http://www.directivoscede.com/conocimiento/detail.php?id=730&int_theme=44