

The background of the slide is a dark blue, textured surface. In the upper left, there is a faint, glowing globe. To its right, a large, white padlock icon is centered on a rectangular, textured block. Further right, the text 'https://www' is visible in a light blue, glowing font. The overall theme is digital security and web application safety.

OWASP Top 10 + Web Pentesting



OWASP

The Open Web Application Security Project



OWASP

The Open Web Application Security Project

– Mario Robles Tencio

“ Profesional de seguridad

*+10 años de experiencia en tema de seguridad de redes ,
desarrollo de aplicaciones web, seguridad de aplicaciones web,
PenTesting (Ethical Hacking)*

Consultor para diversos clientes

*Equifax: Senior Web Application Security Engineer Responsable
del análisis de penetración por medio de Hacking ético para las
aplicaciones web a lo largo de Latinoamérica, el Reino Unido e
Iberia. “*

Mario.Robles@owasp.org

Miembro activo: 37849215



OWASP

The Open Web Application Security Project

Agenda

- Introducción al OWASP Top 10 2010
 - Descripción de cada ítem
- +
- Ejemplos para Web App Pentesting
- Próximos pasos
 - Preguntas



OWASP

The Open Web Application Security Project

- Diez **Riesgos** Más Críticos sobre Seguridad en Aplicaciones.
- Por cada ítem en el Top 10: riesgo, como verificar si usted posee problemas en esta área, como evitarlos, ejemplos y enlaces a mayor información.
- El objetivo principal del Top 10 es educar desarrolladores, diseñadores, arquitectos, gerentes, y organizaciones sobre las consecuencias de las vulnerabilidades de seguridad más importantes en aplicaciones web.
- Gracias a Aspect Security por iniciar, liderar, y actualizar el OWASP Top 10 desde su inicio en 2003, y a sus principales autores: Jeff Williams y Dave Wichers.



OWASP

The Open Web Application Security Project

A1 – Inyección

A2 – Secuencia de comandos en sitios cruzados (XSS)

A3 – Pérdida de Autenticación y Gestión de Sesiones

A4 – Referencia Directa Insegura a Objetos

A5 – Falsificación de Peticiones en Sitios Cruzados (CSRF)

A6 – Defectuosa configuración de seguridad

A7 – Almacenamiento Criptográfico Inseguro

A8 - Falla de Restricción de Acceso a URL

A9 – Protección Insuficiente en la capa de Transporte



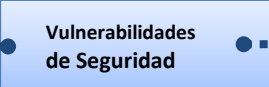


A10 – Redirecciones y Reenvíos no validados

Factores de Riesgo



OWASP

The Open Web Application Security Project

RIESGO	 Agentes De Amenaza	 Vectores de Ataque	 Vulnerabilidades de Seguridad		 Impactos Técnicos	 Impactos al Negocio
		Explotación	Prevalencia	Detección	Impacto	
A1-Inyeccion		FACIL	COMUN	MEDIA	SEVERO	
A2-XSS		MEDIA	MUY DIFUNDIRA	FACIL	MOERADO	
A3-Autent'n		MEDIA	COMUN	MEDIA	SEVERO	
A4-DOR		FACIL	COMUN	FACIL	MODERADO	
A5-CSRF		MEDIA	MUY COMUN	FACIL	MODERADO	
A6-Config		FACIL	COMUN	FACIL	MODERADO	
A7-Crypto		DIFICIL	POCO COMUN	DIFICIL	SEVERO	
A8-Accesso URL		FACIL	POCO COMUN	MEDIA	MODERADO	
A9-Transporte		DIFICIL	COMUN	FACIL	MODERADO	
A10-Redirects		MEDIA	POCO COMUN	FACIL	MODERADO	



A1 – Inyección

Las fallas de inyección, tales como SQL, OS, y LDAP, ocurren cuando datos no confiables son enviados a un interprete como parte de un comando o consulta. Los datos hostiles del atacante pueden engañar al interprete en ejecutar comandos no intencionados o acceder datos no autorizados.



SQL Injection (SQLi)

Inicio de sesión

Usuario

Contraseña

Ingresar

Código que genera el Query

```
sql = "SELECT * FROM usuarios  
WHERE usuario = '" + usuario +  
' and clave = '" + clave + "'"
```

Así se interpreta el query por el motor de BD

```
SELECT * FROM usuarios WHERE  
usuario = 'admin'--' and clave  
= 'no importa'
```

Si alguien se pregunta si esto tan grave puede ser real pues...

Este es un ejemplo de la vida real de un sitio que recibía pagos por tarjeta de crédito

Podría ser el escenario aún peor ??



SQL Injection (SQLi)

Inicio de sesión

Usuario

Contraseña

Ingresar

```
'UNION/**/SELECT/**/  
CAST(usuario/**/as/**/int),  
1,1,1,1/**/  
FROM/**/usuarios/**/  
WHERE/**/usuario/**/></**/'a  
'--
```

Código que genera el Query

```
sql = "SELECT * FROM usuarios  
WHERE usuario = '" + usuario +  
' and clave = '" + clave + "'"
```

Así se interpreta el query por el motor de BD

```
SELECT * FROM usuarios WHERE  
usuario = ' UNION SELECT  
CAST(usuario as int),1,1,1,1  
FROM usuarios  
WHERE usuario > 'a'--' and  
clave = 'no importa'
```



SQL Injection (SQLi)

Error de SQL puede ser reflejado

```
Microsoft OLE DB Provider for SQL Server</font> <font  
face="Arial" size=2>error '80040e07'</font>  
<font face="Arial" size=2>Syntax error converting the  
varchar value 'admin' to a column of data type int.</font>
```

Así se interpreta el query por el motor de BD

```
SELECT * FROM usuarios WHERE usuario = '' UNION SELECT  
CAST(usuario as int),1,1,1,1  
FROM usuarios  
WHERE usuario > 'a'--' and clave = 'no importa'
```

Seleccionar todo de la tabla usuarios donde usuario es igual a nada uniendo el recordset devuelto con la selección del usuario convertido a Integer de la tabla usuarios donde Usuario sea mayor que "a"



OS Command Injection (OSi)

Caso de la vida real, código PHP vulnerable en un servidor Linux:

```
<?php
crearReporte('reporte.html' );
shell_exec('generarpdf reporte.html '.$_GET['nombre'].'.pdf');
EntregarReporte();
?>
```

1. Se crea el reporte en formato HTML generado como texto por una función interna
2. **Se ejecuta un comando para generar el PDF utilizando un programa instalado en el servidor**
3. Se entrega el PDF generado al usuario



OS Command Injection (OSi)

Ejecución de comandos con información brindada por el “usuario”

```
https://misitioinseguro.com/generarpdf.php?nombre=miinfo;  
cat /etc/passwd > /var/www/passwd
```

Así se interpreta el comando al ejecutarse

```
generarpdf reporte.html miinfo;  
cat /etc/passwd > /var/www/passwd.pdf
```

El comando generarpdf genera un archivo PDF a partir de una platilla html

Adicionalmente con el comando **cat** genera el archivo passwd.pdf en la raíz del sitio a partir del archivo passwd (Contraseñas)

Por fortuna para el atacante el archivo passwd.pdf puede ser descargado desde:

```
https://misitioinseguro.com/passwd.pdf
```



OWASP

The Open Web Application Security Project

Prevención y remediación

- SQLi
 - Consultas parametrizadas
 - Procedimientos almacenados
 - Validación de entradas
- OS Command Injection
 - Listas blancas
 - Validación de Entradas
- Consejos comunes para hardening
 - No utilizar el SA en su aplicación
 - Cuenta de sistema operativo con permisos limitados



A2 – Secuencia de comandos en sitios cruzados (XSS)

Las fallas XSS ocurren cada vez que una aplicación toma datos no confiables y los envía al navegador web sin una validación y codificación apropiada. XSS permite a los atacantes ejecutar secuencias de comandos en el navegador de la víctima los cuales pueden secuestrar las sesiones de usuario, destruir sitios web, o dirigir al usuario hacia un sitio malicioso.



Cross Site Scripting (XSS)

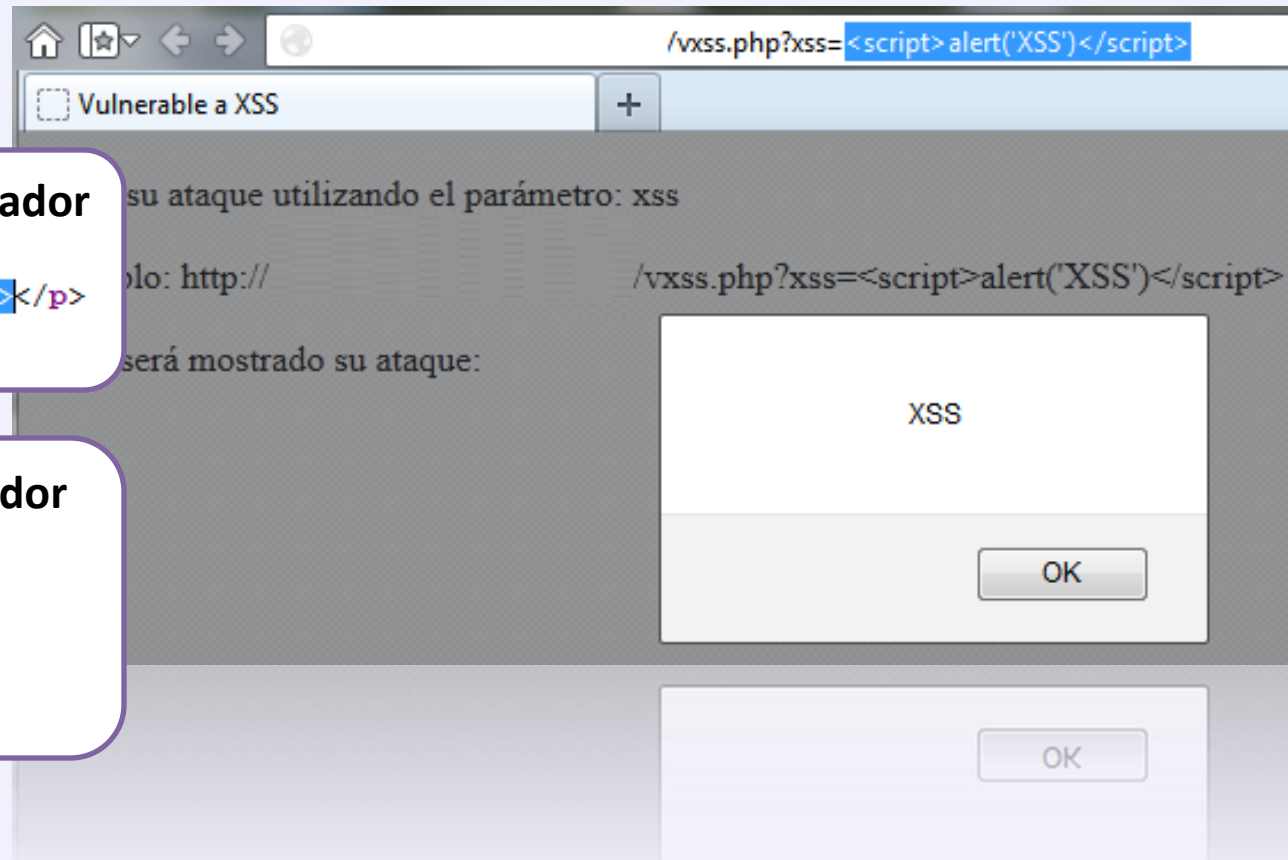
- Reflejado (Reflected)
 - Datos maliciosos son enviados al servidor y son reflejados de vuelta en la respuesta mostrada en el cliente.
- Almacenado (Stored XSS)
 - Es XSS reflejado pero que ha sido almacenado del lado del servidor siendo reflejado cada vez que los datos son enviados al cliente
- Basado en el DOM (DOM Based XSS)
 - La ejecución de secuencias de comandos ocurren en el cliente utilizando el DOM (Document Object Model) sin necesidad de que los datos fueran enviados al servidor



Reflected XSS

Solicitud enviada a un sitio vulnerable:

```
http://sitioconxss.com/pagina.asp?param=<script>alert('XSS')</script>
```



Código fuente en el navegador

```
<p>  
<script>alert('XSS')</script></p>  
</body>
```

Código fuente en el servidor

```
<p>  
<?php  
echo $_GET['xss'];  
</p>
```



Stored XSS

Envío de información por medio de un formulario vulnerable:

Actualizar su información de perfil

Nombre

Juan Perez

Dirección

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

Actualizar

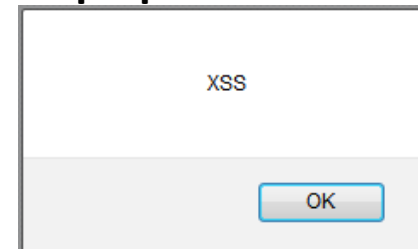
El contenido se envía y sin realizar validaciones se almacena en el servidor

Cada vez que la información que contiene el código malicioso es enviada al cliente se refleja el ataque si no se utiliza codificación

```
<p>Enviamos su paquete a la  
direcci&oacute;n:  
<script>alert('XSS')</script>  
</p>
```

Estimado Juan Perez

Enviamos su paquete a la dirección:





DOM based XSS

Javascript utilizado de forma insegura:

Brinde su PIN antes de continuar

PIN

Actualizar 

Contenido peligroso se aprovecha de la falta de validación para ejecutar contenido en el cliente

```
<input type="button" onclick="validarpin()"
value="Actualizar" />
```

La función de validación del PIN construye código Javascript que luego es ejecutado

```
function validarpin(){
  var pin = document.getElementById('pin');
  if (!IsNumber(pin.value)) {
    //Enviar formulario
  } else {
    eval("alert('El PIN: "+pin.value+" es incorrecto')");
  }
}
```

```
alert('El PIN: 123'); alert('XSS');
// es incorrecto')
```




Protección contra XSS

- Validación de Entradas
 - Filtrado
 - Listas blancas (Whitelisting)
- Codificación de salidas (Encoding)
 - HTML Encoding
 - URL Encoding
 - Javascript Encoding





Protección contra XSS

Input

Ataque:

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

```
<div>  
Defacement  
</div>
```

Server

Whitelisting:

Países, Colores

Input Validation / Sanitization:

RegEx es tu aliado

Encoding:

HTML, JS, URL

Output

Ataque?:

```
&lt;script&gt;  
alert(&#39;XSS&#39;)  
&lt;/script&gt;
```

```
&lt;div&gt;  
Defacement  
&lt;/div&gt;
```



A3 – Pérdida de Autenticación y Gestión de Sesiones

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



Credenciales y Sesiones

- Credenciales fáciles de adivinar o de sobrescribir por medio de métodos débiles de administración
 - Creación de cuentas, cambio de contraseñas, restablecer contraseñas, IDs de sesión predecibles
- IDs de sesión expuestos en el URL (URL rewriting)
- IDs de sesión vulnerables a ataques de fijación de sesión (session fixation)
- Expiración de sesiones y su cierre
- Rotar los identificadores de sesiones después de una autenticación correcta



Credenciales y Sesiones

No enviar información sensible en el URL

```
http://example.com/accion;  
jsessionid=2P0OC2JDPXM0OQSNDLPSKHCJUN2JV?parametro=valor
```

No almacenar información sensible en Cookies

```
Cookie: User=Juanito  
Cookie: Password=123
```

Complejidad de contraseñas y nomenclatura de ID de usuarios

```
Administrador: User = admin | User = m4r  
Contraseña: 123, Mario | !3vdg$&*hd
```




Prevención

- Un único conjunto de controles de autenticación fuerte y gestión de sesiones:
 - Requisitos de gestión de sesiones y autenticación definidos en el Application Security Verification Standard (ASVS) de OWASP, secciones V2 (Autenticación) y V3 (Gestión de sesiones).
- Tener una interfaz simple para los desarrolladores.
- Se debe hacer especial hincapié en evitar vulnerabilidades de XSS que podrían ser utilizadas para robar identificadores de sesión.



A4 – Referencia Directa Insegura a Objetos

Una referencia directa a objetos ocurre cuando un desarrollador expone una referencia a un objeto de implementación interno, tal como un fichero, directorio, o base de datos. Sin un chequeo de control de acceso u otra protección, los atacantes pueden manipular estas referencias para acceder datos no autorizados.



Insecure Direct Object References (iDOR)

Verificar que todas las referencias a objetos tienen las protecciones apropiadas, considerar:

Referencias directas a recursos restringidos, verificar si el usuario está autorizado a acceder al recurso en concreto que solicita.

Objetos de implementaciones internas, como archivos, directorios, registros de bases de datos o llaves



Insecure Direct Object References (iDOR)

Ejemplo1: Consulta de transacción A

```
http://example.com/vertrans?idt=205683
```

Consulta de transacción X

```
http://example.com/vertrans?idt=205684
```

Decidir si la transacción X es un problema o no es difícil de comprobar por medio de análisis dinámicos de vulnerabilidades

Ejemplo2: Paso 1 de un asistente web

```
http://example.com/paso1?codigo=1234
```

Paso 2 del asistente web

```
http://example.com/paso2?siguientepaso=valor
```

Se puede llegar al paso dos sin pasar por validaciones del paso uno ?



A5 – Falsificación de Peticiones en Sitios Cruzados (CSRF)

Un ataque CSRF obliga al navegador de una víctima autenticada a enviar una petición HTTP falsificado, incluyendo la sesión del usuario y cualquier otra información de autenticación incluida automáticamente, a una aplicación web vulnerable. Esto permite al atacante forzar al navegador de la víctima para generar pedidos que la aplicación vulnerable piensa son peticiones legítimas provenientes de la víctima.



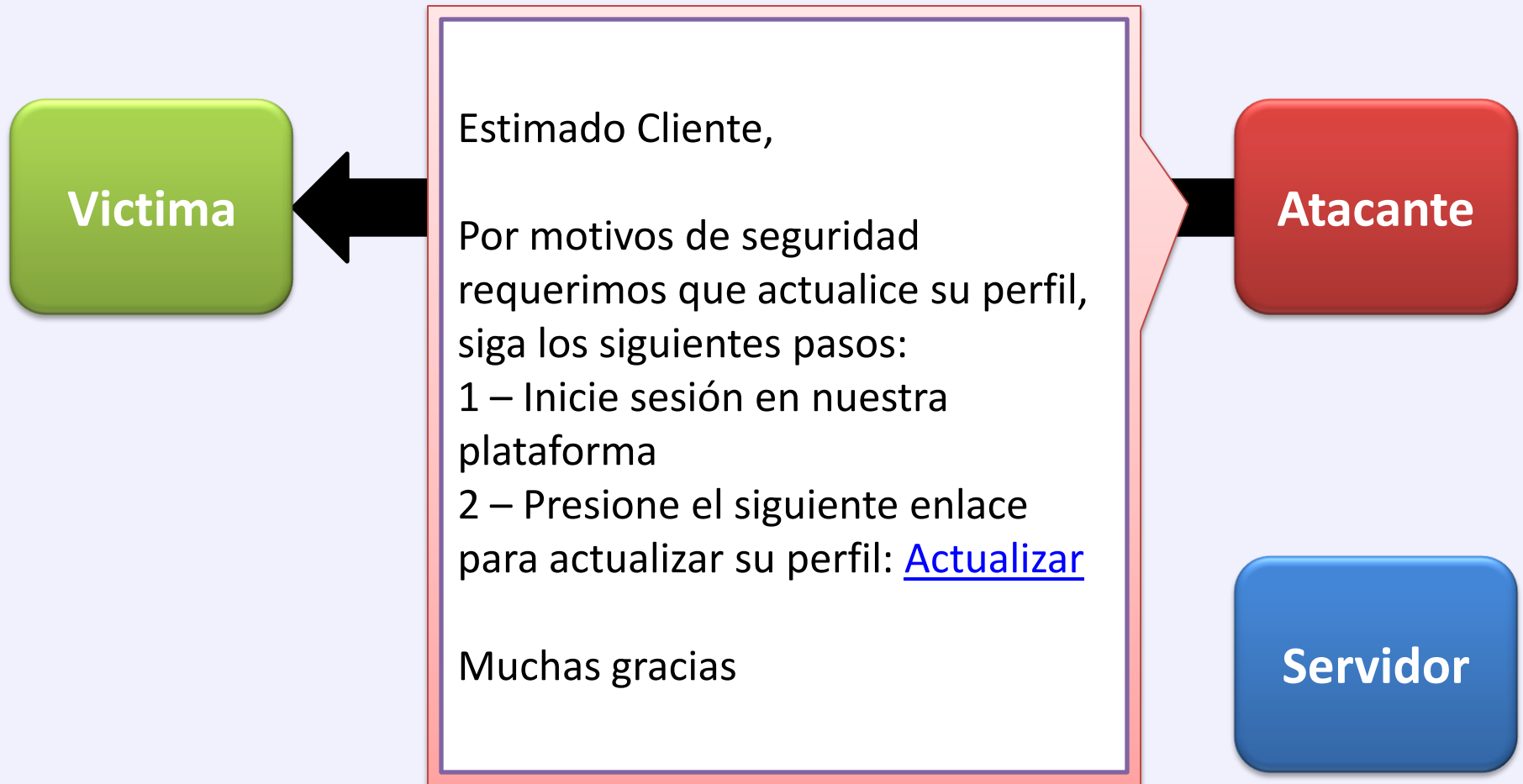
Cross Site Request Forgery (CSRF o XSRF)

Verificar si cada enlace, y formulario, contiene un token no predecible para cada usuario. Si no se tiene dicho token, los atacantes pueden falsificar peticiones.

Se debe concentrar el análisis en enlaces y formularios que invoquen funciones que permitan cambiar estados. Tales funciones son los objetivos más importantes que persiguen los ataques CSRF.

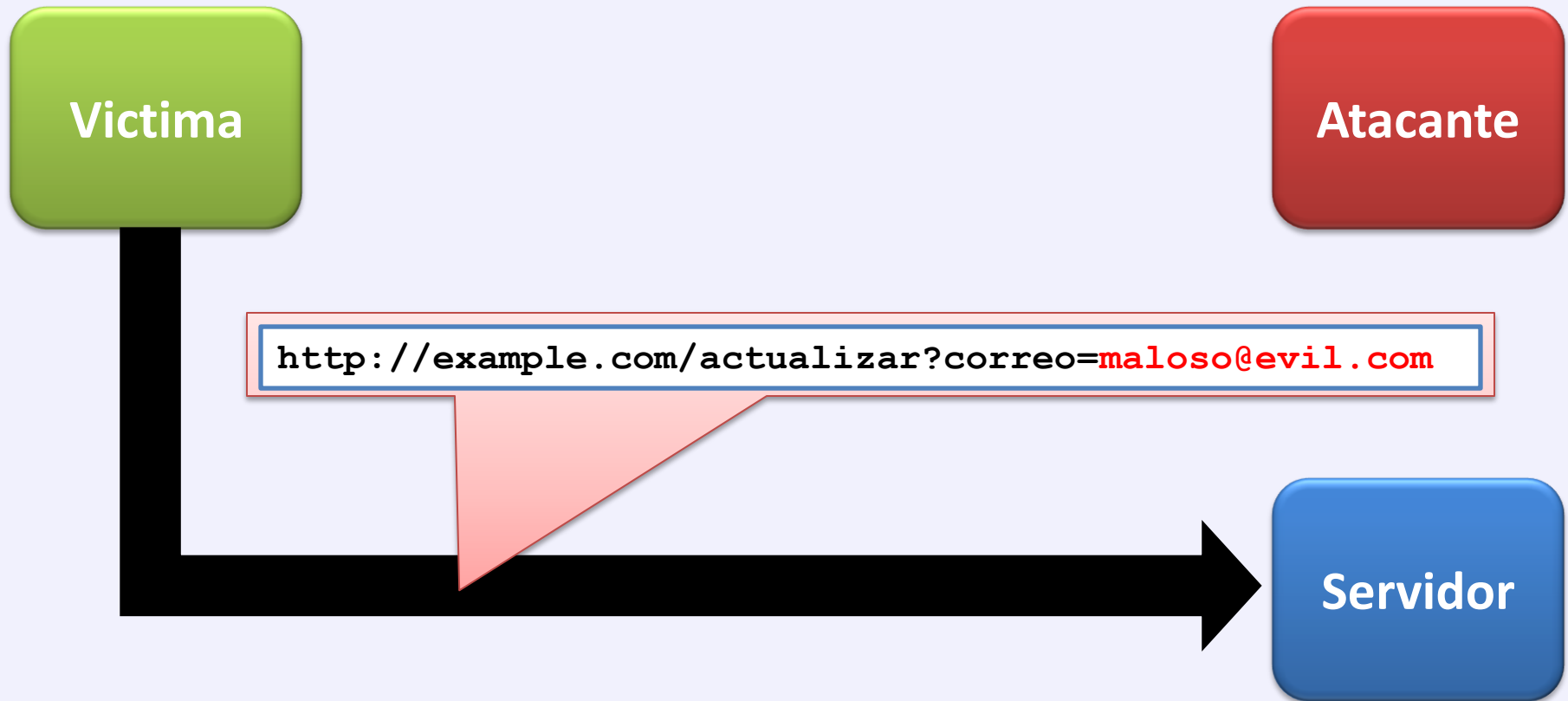


Cross Site Request Forgery (CSRF o XSRF)





Cross Site Request Forgery (CSRF o XSRF)





Cross Site Request Forgery (CSRF o XSRF)

Victima

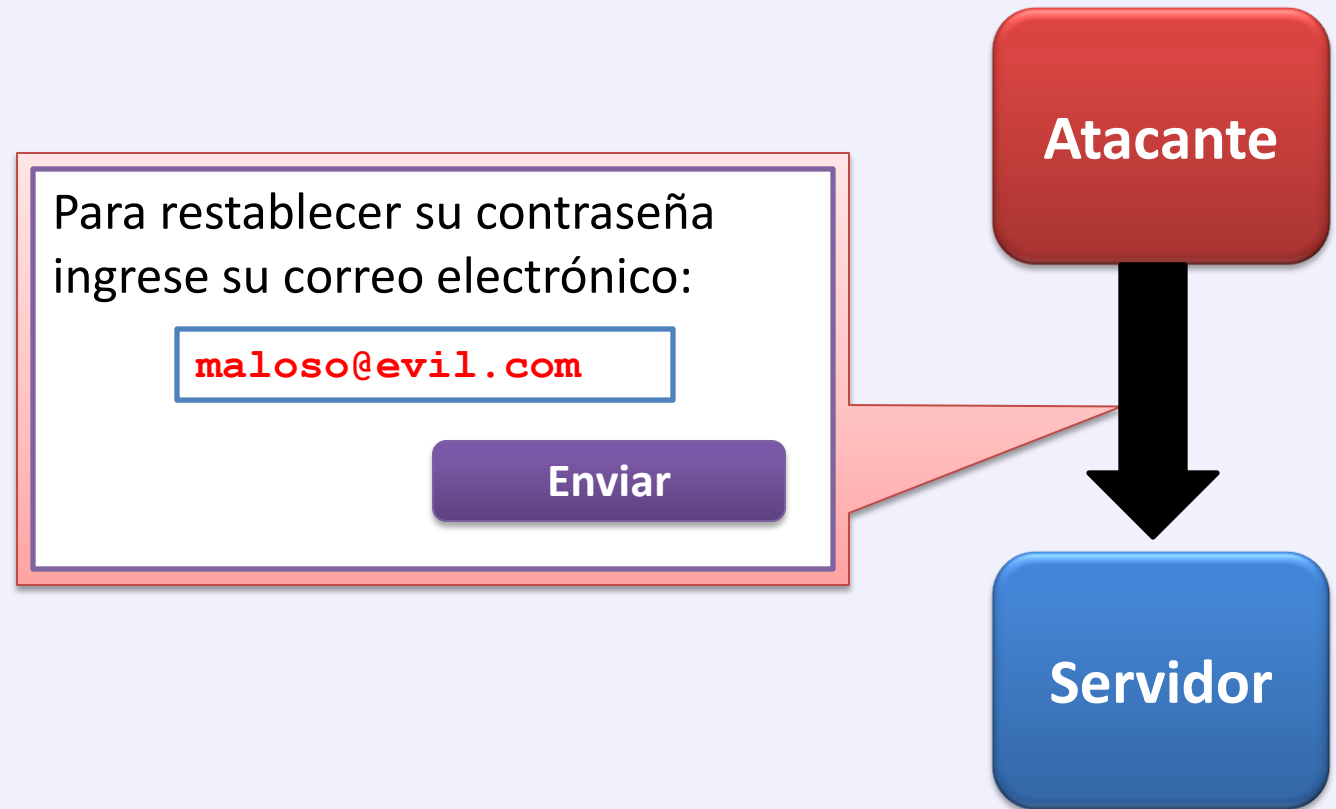
Para restablecer su contraseña
ingrese su correo electrónico:

`maloso@evil.com`

Enviar

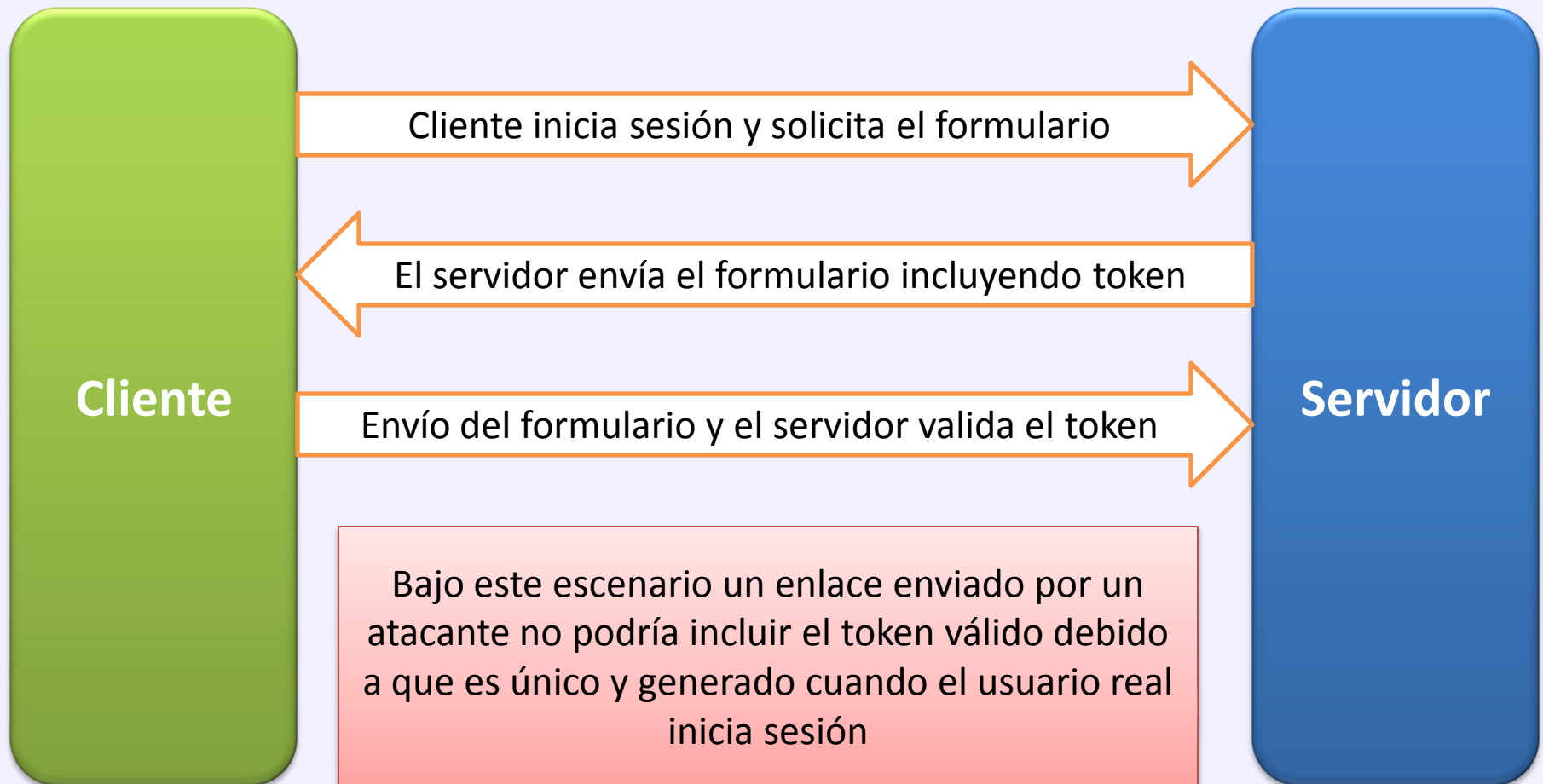
Atacante

Servidor





Protección contra CSRF





A6 – Defectuosa configuración de seguridad

Una buena seguridad requiere tener definida e implementada una configuración segura para la aplicación, marcos de trabajo, servidor de aplicación, servidor web, base de datos, y plataforma. Todas estas configuraciones deben ser definidas, implementadas, y mantenidas ya que por lo general no son seguras por defecto. Esto incluye mantener todo el software actualizado, incluidas las librerías de código utilizadas por la aplicación.



Configuraciones inseguras

Se requiere un proceso concertado, repetible y replicable; para desarrollar y mantener una correcta configuración de seguridad de la aplicación.

Ejemplos:

Plataformas vulnerables sin los últimos parches de seguridad
Configuraciones iniciales o funciones innecesarias activadas
Usuarios de sistema operativo o bases de datos con permisos excesivos



Prevenir configuraciones inseguras

Los programas de auditoría de seguridad son por lo general buenos para abarcar grandes cantidades de pruebas para detectar la falta de parches de seguridad así como la existencia de configuraciones por defecto, la verificación manual posteriormente será una responsabilidad muy importante por parte del profesional de seguridad.



A7 – Almacenamiento Criptográfico Inseguro

Muchas aplicaciones web no protegen adecuadamente los datos sensibles, tales como tarjetas de crédito, SSNs, y credenciales de autenticación con mecanismos de cifrado o hashing. Atacantes pueden modificar o robar tales datos protegidos inadecuadamente para conducir robos de identidad, fraudes de tarjeta de crédito u otros crímenes.



Cifrado de datos almacenados

- Identificar si los datos son sensibles y requieren cifrado. (contraseñas, tarjetas de crédito, datos médicos e información personal)
- Está cifrado en todos aquellos lugares donde es almacenado durante periodos largos.
- Sólo usuarios autorizados tienen acceso a los datos descifrados
- Utilice un algoritmo estándar seguro.
- Genere una clave segura, protéjala ante accesos no autorizados y elabore un plan para el cambio de claves

ASVS requirements on Cryptography (V7)



A8 - Falla de Restricción de Acceso a URL

Muchas aplicaciones web verifican los privilegios de acceso a URLs antes de generar enlaces o botones protegidos. Sin embargo, las aplicaciones necesitan realizar controles similares cada vez que estas páginas son accedidas, o los atacantes podrán falsificar URLs para acceder a estas páginas igualmente.



Falla de Restricción de Acceso a URL

- Verificar que el acceso a cada página con o sin sesiones autenticadas es el correcto
- Controles de autorización deben funcionar adecuadamente
- Un error común es suponer que al desconocerse una ubicación es suficiente para asumir que es segura

Ejemplo:

```
http://example.com/admin.php
```

Aunque en ningún sitio de la aplicación exista un enlace a una ubicación no implica que no pueda ser adivinada



A9 – Protección Insuficiente en la capa de Transporte

Las aplicaciones frecuentemente fallan al autenticar, cifrar, y proteger la confidencialidad e integridad de tráfico de red sensible. Cuando esto ocurre, es debido a la utilización de algoritmos débiles, certificados expirados, inválidos, o sencillamente no utilizados correctamente.



OWASP

The Open Web Application Security Project

Protección Insuficiente en la capa de Transporte (TLS)

- Se utiliza SSL para proteger todo el tráfico relacionado con la autenticación? (Contra ataques MiTM)
- Se utiliza SSL para todos los recursos de páginas y servicios privados?
- Se debe evitar el acceso SSL únicamente a determinados recursos de una página ya que esto provoca advertencias en el navegador y puede exponer el identificador de sesión de los usuarios.
- Sólo se soportan algoritmos considerados fuertes.
- Todas las cookies de sesión tienen el atributo “secure” activado.
- El certificado debe ser legítimo y estar configurado correctamente para este servidor. Un usuario acostumbrado a lidiar con mensajes de error en su sitio perderá la desconfianza a un sitio falso con las mismas características



A10 – Redirecciones y Reenvíos no validados

Las aplicaciones web frecuentemente redirigen y reenvían a los usuarios hacia otras páginas o sitios web, y utilizan datos no confiables para determinar la página de destino. Sin una validación apropiada, los atacantes pueden redirigir a las víctimas hacia sitios de phishing o malware, o utilizar reenvíos para acceder páginas no autorizadas.



Redirecciones o reenvíos (Open Redirects)

- Bajo que circunstancias puede ser importante utilizar datos provenientes del cliente para ser utilizados para una redirección?
- Es una práctica común disponer de una página para redireccionar al usuario en caso de error o de que algo adicional sea requerido:

```
http://www.example.com/redirect.jsp?url=evil.com
```

- Para ese caso se debe considerar si la validación se produce en el cliente o en el servidor, si es en el servidor porque es enviada al cliente ?
- Porque guardar la ubicación en un **input hidden** y no en una variable de sesión en el servidor ?
- De ser absolutamente requerido utilizar datos del cliente para la redirección, se debe realizar la validación correcta de hacia donde se enviará al usuario



OWASP

The Open Web Application Security Project

Próximos pasos

Requisitos de seguridad de la aplicación

Arquitectura de seguridad de la aplicación

Estándares de controles de seguridad

Ciclo de vida de desarrollo seguro (SSDLC)

Formación sobre seguridad en aplicaciones

Adoptar metodologías de revisión (ej: ASVS):

Revisión de Código, Scanners, PenTesting

No basta con el OWASP Top Ten



OWASP

The Open Web Application Security Project





Muchas Gracias!

mario.robles@owasp.org

https://www.owasp.org/index.php/Costa_Rica



OWASP

The Open Web Application Security Project