

# **VULNERABILIDADES DE SEGURIDAD EN APLICACIONES WEB**

# CONTENIDO



1. El problema
2. Casos reales de mala seguridad
3. Estadísticas, cifras, y datos para asustar
4. Proyecto OWASP Top 10
5. CWE/SANS Top 25 errores más peligrosos
6. Otras referencias sobre seguridad
7. Preguntas



# **1. EL PROBLEMA**

# EL PROBLEMA: Tecnologías Heredadas

- La web emplea tecnologías heredadas que hoy en día se usan para fines diferentes a los que fueron concebidas.
- Ejemplo: HTTP, un protocolo sin estado.
  - Necesario utilizar mecanismos externos para mantener el estado.
  - No hay negociación de conexión ni mantenimiento de sesión.
    - El atacante sólo necesita enviar una petición de página con las cabeceras apropiadas.

# EL PROBLEMA: Responsabilidades

---

Como consecuencia:

1. La aplicación debe protegerse frente a intentos de entrada de usuarios malintencionados.
2. La aplicación deben proteger a los usuarios legítimos de los usuarios malintencionados.

# EL PROBLEMA: Responsabilidades



El peso de la seguridad recae en el programador.

*(Aunque en la práctica haya soluciones que nos ayuden).*

# EL PROBLEMA: Exceso de confianza

---

## ¿Confiar en el usuario? ¡Jamás!

- La validación JavaScript no sirve como mecanismo de seguridad.
- Es necesario hacer validación en el servidor.
  - Y hay que validarlo TODO.

# EL PROBLEMA: Exceso de confianza

- Es muy habitual que los programadores confíen en:
  - Valores transmitidos en cookies.
  - Cabeceras HTTP.
  - Campos en formularios:
    - Desplegables
    - Radio buttons
    - Hidden fields



# EL PROBLEMA: Exceso de confianza

Un atacante puede:

- Manipular cualquier fragmento de información transmitido:
  - Parámetros
  - Cookies
  - Cabeceras
- Evitar fácilmente cualquier mecanismo de control implementado en lado cliente.
- Enviar peticiones en cualquier orden, enviar parámetros en diferente formato a lo esperado, o no enviarlos.
  - Cualquier suposición de uso que haga el desarrollador puede ser violada.



## **2. CASOS REALES DE MALA SEGURIDAD (y sus consecuencias)**

# CASOS REALES DE MALA SEGURIDAD

## Facebook:

- En 2008: se descubrió que era posible acceder a fotografías privadas de millones de usuarios.
  - El ID de las fotografías era predecible.
  - En 2011 se descubrió un error similar.
- En 2010: se descubrió un procedimiento para obtener datos confidenciales de un usuario a partir de su email.
  - Fue utilizado por spammers y scammers para enviar correo basura y realizar estafas.

# CASOS REALES DE MALA SEGURIDAD

## Twitter:

- Numerosos problemas de XSS a lo largo de su historia, todos fácilmente explotables.
- En mayo de 2012 se publicó en PasteBin una relación de 55.000 usuarios y contraseñas.

# CASOS REALES DE MALA SEGURIDAD

## LinkedIn:

- En junio de 2012: se publicaron más de 6 millones de contraseñas cifradas.
  - Muchas fueron descifradas al momento gracias a las tablas Rainbow.

# CASOS REALES DE MALA SEGURIDAD

';--have i been pwned?

49

pwned websites

185,229,998

pwned accounts

24,804

pastes

15,536,528

paste accounts

## Top 10 breaches



152,445,165 Adobe accounts



4,821,262 mail.ru Dump accounts



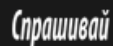
4,789,599 Bitcoin Security Forum  
Gmail Dump accounts



4,609,615 Snapchat accounts



3,867,997 Adult Friend Finder  
accounts



3,474,763 Спрашивай.ру accounts



1,247,574 Gawker accounts



1,194,597 NextGenUpdate accounts



1,186,564 Yandex Dump accounts



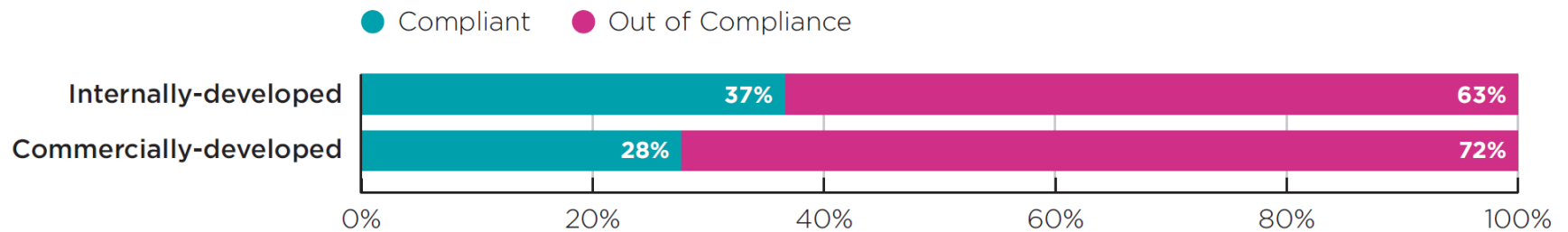
1,057,819 Forbes accounts

<https://haveibeenpwned.com/>



## **4. ESTADÍSTICAS, CIFRAS, Y DATOS PARA ASUSTAR**

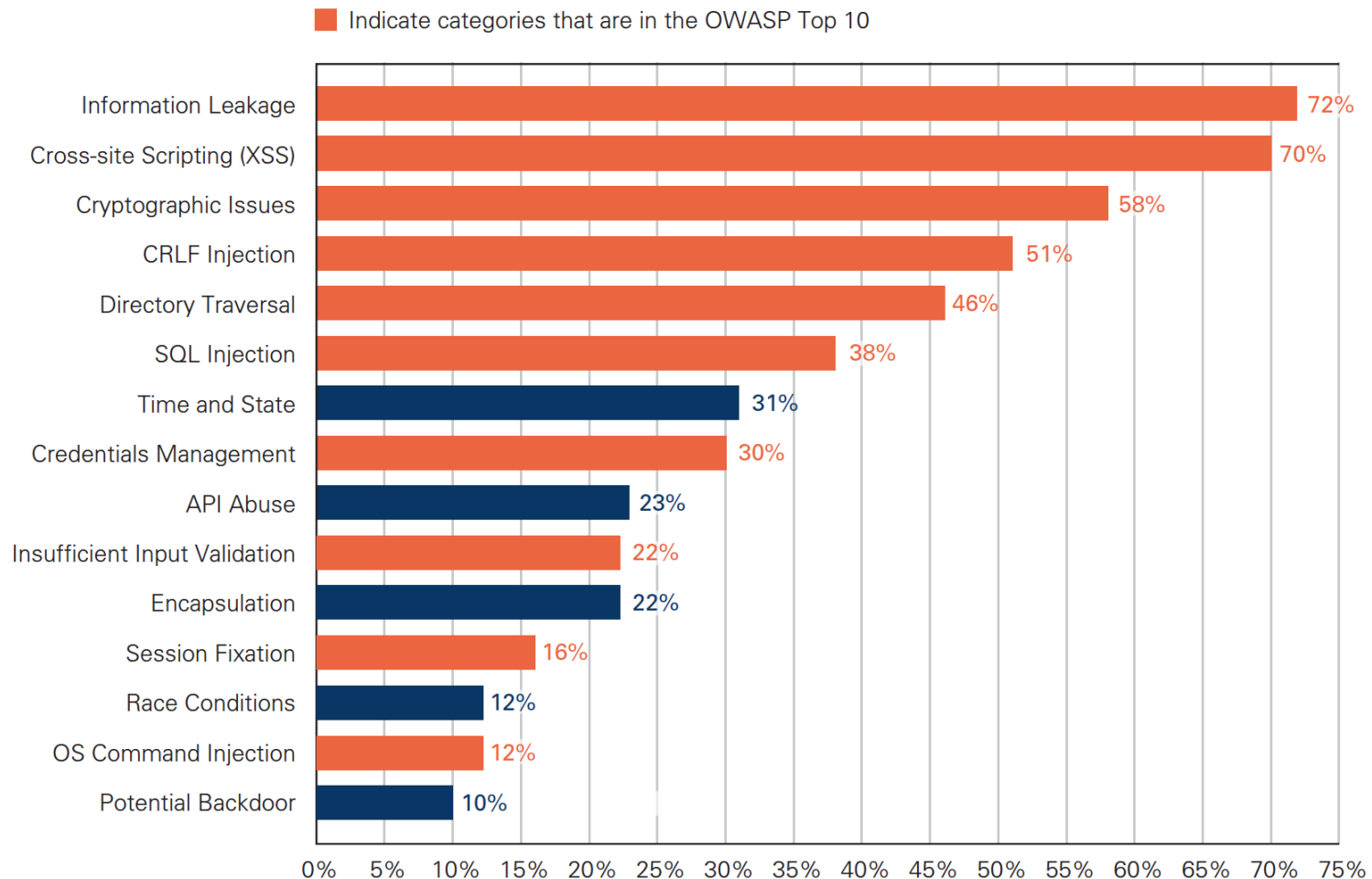
# ESTADÍSTICAS, CIFRAS, Y DATOS PARA ASUSTAR



Presencia de alguna vulnerabilidad Top 10 OWASP en primera entrega.  
(Veracode. *State of Software Security Report Volume 6*. 2015.)



# ESTADÍSTICAS, CIFRAS, Y DATOS PARA ASUSTAR



Probabilidad de encontrar una vulnerabilidad  
(Veracode. *State of Software Security Report Volume 3*. 2011.)



## **4. PROYECTO OWASP TOP 10**

# PROYECTO OWASP TOP 10

OWASP = Open Web Application Security Project 

Proyecto de código abierto para determinar y combatir las causas que hacen que el software sea inseguro.

Enfoque de seguridad en todas sus dimensiones:

- Personas
- Procesos
- Tecnologías

# PROYECTO OWASP TOP 10

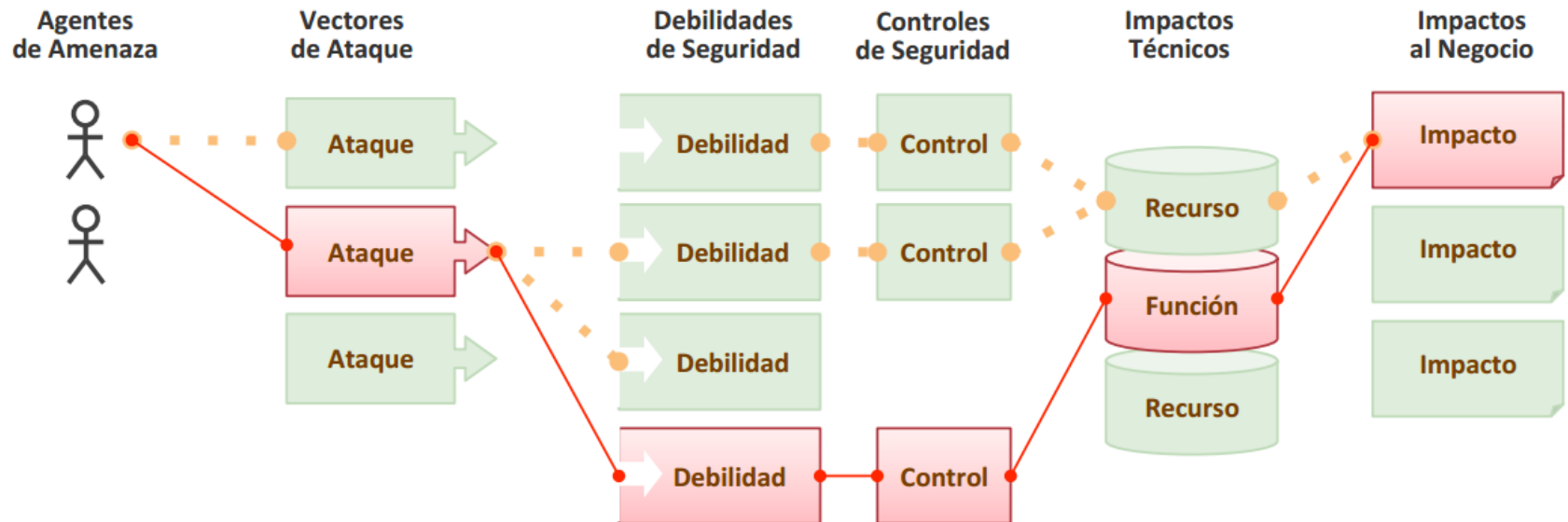
---

## OWASP Top 10:

- Establece una lista de debilidades ordenada por riesgo, y no por número de apariciones.
- Es mundialmente referenciada en libros, publicaciones, y organizaciones dedicadas a la seguridad (MITRE, Veracode, etc).
- Publicada en 2003, 2004, 2007, 2010 y 2013.

# PROYECTO OWASP TOP 10

## ¿Qué es un riesgo?



Un riesgo es cada una de las rutas que puede recorrer un atacante para hacerte daño.

# PROYECTO OWASP TOP 10

---

Para valorar un riesgo debemos preguntarnos:

- ¿Es frecuente que ocurra?
- ¿Es fácil descubrirlo?
- ¿Es fácil explotarlo?
- ¿Cuál será el impacto técnico?
- ¿Cuál será el impacto en el negocio?

# PROYECTO OWASP TOP 10



Tener una vulnerabilidad != Ser vulnerable

# PROYECTO OWASP 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 – Configuración de Seguridad Incorrecta
- A6 – Exposición de datos sensibles
- A7 – Ausencia de Control de Acceso a Funciones
- A8 – Falsificación de Peticiones en Sitios Cruzados (CSRF)
- A9 – Utilización de componentes con vulnerabilidades conocidas
- A10 – Redirecciones y reenvíos no validados



# PROYECTO OWASP TOP 10



## **A1 – Inyección**

# PROYECTO OWASP TOP 10: A1 - Inyección

## EL PROBLEMA

Construir comandos o consultas enviando al intérprete datos no confiables sin filtrar.

En general, componer el String del comando o consulta concatenando variables procedentes del usuario sin control alguno.

# PROYECTO OWASP TOP 10: A1 - Inyección

## EJEMPLO 1



A screenshot of a login form. It has two input fields: 'Correo' (Email) and 'Contraseña' (Password). To the right of the password field is a button labeled 'Entrar' (Login). Below the email field is a checkbox labeled 'No cerrar sesión' (Do not log out). Below the password field is a link that says '¿Has olvidado tu contraseña?' (Forgot your password?).

```
$query = "SELECT id FROM usuarios  
        WHERE email='$email' AND password='$password'";
```

Supongamos que introducimos el siguiente valor en la caja "Correo":

```
cualquiercosa' OR 1=1 LIMIT 1;--
```

Lo que enviamos al intérprete sería:

```
SELECT id FROM usuarios  
        WHERE email='cualquiercosa' OR 1=1 LIMIT 1;  
        --' AND password=''
```

# PROYECTO OWASP TOP 10: A1 - Inyección

## EJEMPLO 2

Si enviamos:

```
' OR id='1';--
```

Obtenemos:

```
SELECT id FROM usuarios  
WHERE email='' OR id='1';  
--' AND password=''
```

# PROYECTO OWASP TOP 10: A1 - Inyección

## RIESGOS

- Acceder a datos confidenciales
- Manipular información
- Suplantar identidad
- Vía de acceso a otro tipo de ataques (ej: XSS)

# PROYECTO OWASP TOP 10



## **A2 – Pérdida de autenticación y gestión de sesiones**

# PROYECTO OWASP TOP 10: A2 – Autenticación

## EL PROBLEMA

Errores de implementación que permiten:

- Obtener hashes de contraseñas
- Obtener contraseñas en claro
- Manipular el token de la sesión

El objetivo: asumir la identidad de otros usuarios.

– Habitualmente la del administrador.

# PROYECTO OWASP TOP 10: A2 – Autenticación

## EL PROBLEMA

Errores habituales:

- No cifrar en BBDD las contraseñas.
- Se pueden adivinar o sobrescribir las contraseñas a través de funciones débiles de gestión de la sesión.
- Los IDs de sesión son expuestos en la URL.
- Los IDs de sesión son predecibles.
- Los IDs de sesión no cambian al hacer login (ataques de fijación de la sesión).
- Los IDs de sesión no expiran al hacer logout.



# PROYECTO OWASP TOP 10: A2 – Autenticación


## EJEMPLO 1


### IDs de sesión expuestos en la URL

<http://www.tiendaguay.com/producto1?sessionid=4U1OU3OUA9FFMJMF>

Mira qué buen precio tiene este artículo:

<http://www.tiendaguay.com/producto1?sessionid=4U1OU3OUA9FFMJMF>

 Añadir archivos

 Añadir fotos

Pulsa "E..." ☐

Responder

# PROYECTO OWASP TOP 10: A2 – Autenticación

## LA SOLUCIÓN

Revisar y cumplir con los requisitos definidos en el Application Security Verification Standard (ASVS) de OWASP, secciones V2 y V3.

# PROYECTO OWASP TOP 10: A2 – Autenticación

## LA SOLUCIÓN

Algunos puntos son:

- Comprobar que la sesión caduca al de cierto tiempo.
- Comprobar que la sesión caduca al hacer logout.
- Comprobar que las páginas que requieren autenticación no son accesibles de forma anónima.
- Comprobar que la sesión cambia al autenticar.
- Aceptar sólo tokens de sesión generados por la aplicación.
- Comprobar que los flags de las cookies de sesión sean correctos (httponly y desactivar método TRACE).
- Comprobar que la aplicación no permite acceso concurrente con un mismo usuario desde dos orígenes distintos.

# PROYECTO OWASP TOP 10



## **A4 – Referencia directa insegura a objetos**

# PROYECTO OWASP TOP 10: A4 – Referencias

## EL PROBLEMA

Acceder a información en BBDD, objetos o archivos sin comprobar que el usuario tenga permisos sobre ellos.

# PROYECTO OWASP TOP 10: A4 – Referencias

## EJEMPLO 1

### **Insuficiente control de acceso a referencias**

<http://victima.com/verfactura?fact=918433>

El atacante prueba códigos aleatorios de facturas para tratar de acceder a documentos de otras personas.

# PROYECTO OWASP TOP 10



## **A6 – Exposición de datos sensibles**

# PROYECTO OWASP TOP 10: A6 – Datos Sensibles

## EL PROBLEMA

- No cifrar en BBDD la información sensible.
- No utilizar SSL.
- Utilizar algún algoritmo criptográfico débil.
- Incorrecta rotación o mala gestión de claves criptográficas.
- Ficheros subidos al servidor que no deberían estar ahí.



# PROYECTO OWASP TOP 10: A6 – Datos Sensibles

## EJEMPLO 1

### **Datos sensibles sin cifrar**

Escenario: La aplicación almacena en BBDD las contraseñas de los usuarios sin cifrar.

Un atacante consigue robar la tabla de usuarios mediante una vulnerabilidad de inyección SQL, y se lleva las contraseñas en claro.

# PROYECTO OWASP TOP 10: A6 – Datos sensibles

## LA SOLUCIÓN

- Cifra todos los datos sensibles que sean almacenados o transmitidos.
  - Utiliza hash con salt.
- No almacenes información de más.
- Utiliza algoritmos de cifrado fuerte.
  - MD5 está desaconsejado desde 2004 por problemas de colisiones.
- Deshabilita "autocompletar" en formularios que contengan datos sensibles.
- Deshabilita el cacheado de páginas que contengan datos sensibles.

# PROYECTO OWASP TOP 10



**A7 – Ausencia de control de acceso a funciones**

# PROYECTO OWASP TOP 10: A7 – Acceso

## EL PROBLEMA

- La aplicación no controla adecuadamente el acceso a las páginas privadas.
  - ¿La interfaz de usuario muestra links hacia funcionalidades no autorizadas?
  - ¿Se comprueba la autorización en todas las páginas y acciones de formularios?

# PROYECTO OWASP TOP 10: A7 – Acceso

## EJEMPLO 1

El atacante envía una petición POST a un formulario privado para intentar una escalada de privilegios:

<http://ejemplo.com/admin/createuser.php>

# PROYECTO OWASP TOP 10: A7 – Acceso

## LA SOLUCIÓN

- Diseñar cuidadosamente el sistema de control de accesos.
  - Debería ser fácilmente auditable y parametrizable.
- Realizar pruebas intensivas para garantizar la calidad de la implementación.
  - Hay que probar con usuarios no autenticados, y con usuarios con permisos limitados.

# PROYECTO OWASP TOP 10



## **A8 – Falsificación de Peticiones en Sitios Cruzados (CSRF)**

# PROYECTO OWASP TOP 10: A8 – CSRF

---

## EL PROBLEMA

Formularios que permiten a un atacante falsificar peticiones en nombre de la víctima.



# PROYECTO OWASP TOP 10: A8 – CSRF

## EL PROBLEMA

Cuando la víctima visita un sitio malicioso controlado por el atacante, éste último puede:

- Publicar mensajes en nombre de la víctima.
- Transferir dinero.
- Enviar emails.
- Y en general, actuar sobre cualquier formulario de los sitios web donde la víctima mantenga sesión abierta.

# PROYECTO OWASP TOP 10: A8 – CSRF

## EJEMPLO 1

La víctima deja abierta la sesión con su banco online, y visita la web del atacante.

La web maliciosa contiene una imagen con la siguiente definición:

```

```

# PROYECTO OWASP TOP 10: A8 – CSRF

## EJEMPLO 1

Si la víctima está autenticada en su banco online, las peticiones falsificadas incluirán automáticamente la información de la sesión, autorizando la operación.

# PROYECTO OWASP TOP 10: A8 – CSRF

## EJEMPLO 2

### **Espionaje en cuentas de Gmail (2007)**

1. El usuario inicia sesión en Gmail.
2. El usuario visita una página que contiene código malicioso.
3. El usuario envía (sin saberlo) una petición a Gmail, creando un filtro de correo para redirigir todo el correo entrante al atacante.

# PROYECTO OWASP TOP 10: A8 – CSRF

## LA SOLUCIÓN

1. Cuando el usuario solicita el formulario, generar un token único no predecible, y enviarlo en un campo oculto.
2. Cuando el usuario hace submit del formulario, comprobar que el token devuelto es el esperado.
  - Rechazar la petición si no coincide, pertenece a otro usuario, es vacío, o no está presente.

# PROYECTO OWASP TOP 10: A8 – CSRF

## OTRAS SOLUCIONES

1. Utilizar CAPTCHAs.
2. Solicitar re-autenticación para ciertas operaciones.



*That's all Folks!*