

 **Monster Mitigations for OWASP Top 10**

OWASP
Education Project

Lic. Cristian Borghello, CISSP
Director Segu-Info
www.segu-info.com.ar
info@segu-info.com.ar
@seguinfo

Copyright 2007 © The OWASP Foundation
Permission is granted to copy, distribute and/or modify this document
under the terms of the OWASP License.

The OWASP Foundation
<http://www.owasp.org>

```
*****  
IDENTIFICATION DIVISION.  
PROGRAM-ID. HELLO.  
ENVIRONMENT DIVISION.  
DATA DIVISION.  
PROCEDURE DIVISION.  
MAIN SECTION.  
DISPLAY "Hola mundo"  
STOP RUN.  
*****  
http://www.roesler-ac.de/wolfram/hello.htm
```

OWASP  2

Licencia de uso Creative Commons 2.5

Ud. puede:

- Copiar, distribuir, exhibir, y ejecutar la obra
- Hacer obras derivadas

Bajo las siguientes condiciones:

-  **Atribución.** Debe atribuir la obra en la forma especificada por el autor
-  **No Comercial.** No puede usar esta obra con fines comerciales
-  **Compartir Obras Derivadas Igual.** Si altera, transforma, o crea sobre esta obra, sólo podrá distribuir la obra derivada resultante bajo una licencia idéntica a ésta

<http://creativecommons.org/licenses/by-nc-sa/2.5/ar/>

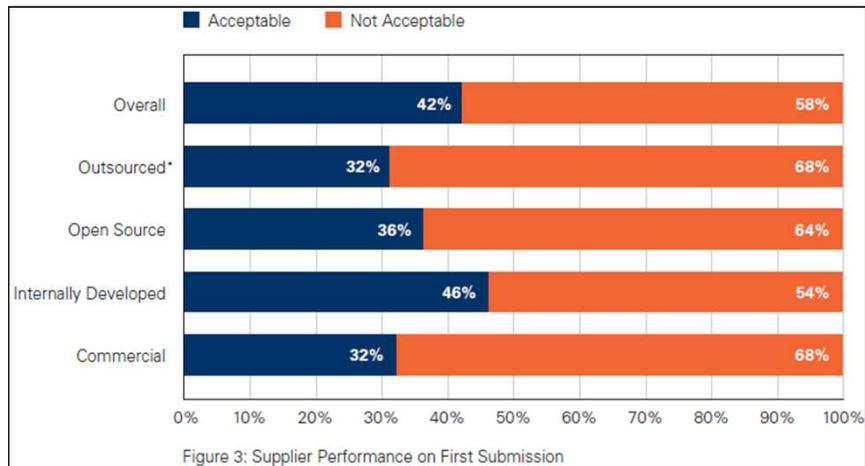
OWASP 

Agenda

- Qué son las Monster Mitigation
- Tipos de Monster Mitigation
- Clasificación y detalle
- Conclusiones

OWASP  4

Siempre un gráfico para empezar



Veracode - State of Software Security Report - Volume 3 – Abril 2011

OWASP  5

Monster Mitigations

Mitigaciones aplicables y efectivas para prevenir y solucionar las vulnerabilidades del OWASP Top 10 y del SANS Top 25

- Adaptando estas mitigaciones a cada aplicación, la misma será más segura
- Se dividen en mitigaciones específicas para cada error y, en generales, aplicables a todas las vulnerabilidades

Más información:

<http://cwe.mitre.org/top25/mitigations.html>

OWASP  6

Monster Mitigations

Según su efectividad se clasifican en:

- **High:** son debilidades con mitigaciones bien conocidas y entendidas que permiten su solución
- **Moderate:** la mitigación previene los errores pero no se cubren todos los aspectos
- **Limited:** la mitigación cubre algunos aspectos y se requiere configuración y entrenamiento adicional
- **Defense in Depth (DiD):** la mitigación sirve para minimizar el impacto cuando el atacante explota la vulnerabilidad
- Son aplicables a líderes de proyecto (L), analistas (A), diseñadores (D) y programadores (P)

M1 – Establecer y controlar las entradas

P

- Conocido como el “*Software’s Killer*”
- No se valida la entrada de variables
- No se inicializan variables ni se validan sus tipos
- Los atacantes pueden modificar sus entradas y producir valores inesperados
- **Secreto: validar, validar, validar...**

```
Protected Sub comprar_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles comprar.Click
    Try
        factura.Text = "<p>Gracias por su compra</p>"
        factura.Text = factura.Text & "<p>El to</p>"
    Catch ex As Exception
        factura.Text = "Por favor ingrese una c
        factura.ForeColor = Drawing.Color.Red
    End Try
End Sub

<?php
    $nombre = $_GET["nombre"];
    $nacimientto = $_GET["nacimientto"];

    echo "Bienvenido " . $nombre ;
    echo "Tu cumpleaños es el " . $nacimientto ;

?>
```

M2 – Controlar las salidas

PD

- Cuando los datos se visualizan (salida), se puede afectar a quien los accede o ejecutar comandos
- Se debe forzar la separación de datos y código mediante la validación apropiada
- **Secreto: validar, validar, validar...**

id	nombre	email	comentario
10	aaa@aaa.com	aaa@aaa.com	Por favor ingrese su...
11	<script>alert(/XSS 2/);</scri...	aaa@aaa.com	Por favor ingrese su...
21	prueba		comentario...
26	<script>alert(/XSS/)</script>		/owned/

```
.252.1 - - [12/Feb/2011:14:56:20 -0200] "GET /1fi/1fi1.php%3C;%20passthru(\\$_GET[cmd])%20%3E HTTP/1.1" 404
.252.1 - - [12/Feb/2011:14:56:25 -0200] "GET /1fi/1fi1.php%3C;%20passthru(\\$_GET[cmd])%20%3E HTTP/1.1" 200
.252.1 - - [12/Feb/2011:14:58:11 -0200] "GET /1fi/1fi1.php?recurso=/var/log/apache2/access.log&cmd=ls%20/etc H
.252.1 - - [12/Feb/2011:15:03:37 -0200] "<? passthru($_GET[cmd]) ?>" 400 333 "-" "-"
.252.1 - - [12/Feb/2011:15:04:14 -0200] "<? passthru($_GET[cmd]) ?>" 400 333 "-" "-"
```

OWASP  9

M3 – Permisos y privilegios

LD

- No se analizan los permisos necesarios para ejecutar la aplicación (*"sin Admin no funciona"*)
- No se controla el flujo de errores y se entrega información excesiva a todos los usuarios por igual
- No se tienen en cuenta los **mínimos privilegios** ni la **separación de tareas**
- Al encontrar una vulnerabilidad, es aplicable a todo el entorno
- **Secreto: Defensa en Profundidad**

OWASP  10

M4 – Todos pueden ver el código



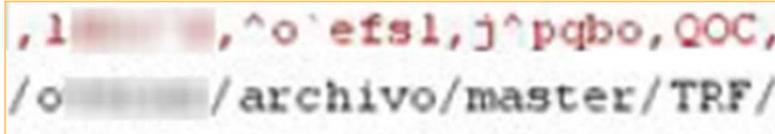
- Existen miles de formas de atacar una aplicación
- Aún el código compilado y/o ofuscado se puede “leer”
- Alguien podría acceder al código y a los datos
- Teniendo el código debería ser imposible acceder a los datos (**cifrado**)
- **El secreto: minimizar el riesgo con análisis de vulnerabilidades propios**


 OWASP 11

M5 – No reinventar la rueda



- Inventar algoritmos es... una tontería. **¡Existen los estándares!**
- Criptografía, autorización, manejo de sesión, procesos de aleatorización, etc.
- Un algoritmo secreto **no** es un algoritmo seguro
- La seguridad por oscuridad es... una tontería
- **Secreto: estudia y respeta los estándares**



```
,l[redacted],^o`efsl,j^pqbo,QOC,  
/o[redacted]/archivo/master/TRF/
```


 OWASP 12

G1 – Usar librerías y *frameworks*

LD

- Existen librerías y *frameworks* libres, pagos, gratuitos y para todos los gustos
- Si el código puede ser visto y analizado por muchos, es

- Pequeños cambios en la configuración de la librería o framework pueden causar problemas de seguridad
- Se debe ser cuidadoso al usar librerías y frameworks



P 13

G2 – Integrar seguridad en todo el SLDC

L

- El diseño y desarrollo de software son procesos de ingeniería
 - Sólo se puede escribir código seguro si se incorpora seguridad a **todo el proceso de desarrollo de software**
- El modelado de amenazas para el diseño y escribir aplicaciones



“codear” mucho no significa desarrollar seguro

OWASP 14

G3 – Analizar debilidades por muchos caminos L

- No existe una sola forma de desarrollar y tampoco una sola de desarrollar seguro
- Cada acercamiento puede tener debilidades y por ende es recomendable utilizar más de uno
- Realizar análisis estático, dinámico, automático y manual
- Probar en *Black Box* y *White Box*
- Utilizar herramientas de bloqueo (WAF, Proxies, etc.)
- Analizadores estáticos: Fortify, FindBugs, Pixy, etc.
- **Secreto: piensa como un atacante**

G4 – Los usuarios siempre deberían poder interactuar con la aplicación

- Si un usuario no puede ingresar a una aplicación porque su antivirus lo bloquea ¿qué hace?
→ **Deshabilita el antivirus** 😊
- Si la aplicación tiene funcionalidades que pueden ser

En estos momentos estás utilizando la vista Gmail en modo HTML básico. [Cambiar a la vista estándar](#) | [Establecer](#)



Conclusiones

- No existen soluciones mágicas
- Todos deben estar involucrados en el desarrollo de aplicaciones seguras
- No existen soluciones mágicas, solo la evaluación apropiada de riesgos
- Las mitigaciones son sólo eso: mitigaciones, pero si se utilizan, las aplicaciones serán más seguras

Referencias

- MITRE Mitigations: <http://segu.info/mit1>
- Guía de pruebas OWASP v3: <http://segu.info/guia>
- Safe Code: <http://segu.info/mit2>
- Securing Code: <http://segu.info/mit3>

Curso “Desarrollo Seguro” de Segu-Info ☺

<http://educacion.segu-info.com.ar>

```
body:before {  
    content: "¡Gracias!";  
}
```

<http://www.roesler-ac.de/wolfram/hello.htm>