

Normativa para el Desarrollo de Aplicaciones Web Seguras

IDENTIFICACIÓN

Proyecto	WEBconnector - OWASP
Nombre del Documento	NORowasp.odt
Autor	Juan Luis Serradilla Amarilla
Versión Actual	1.1
Fecha de la Versión	27 de octubre de 2010

RESUMEN

Este documento contiene la normativa para el desarrollo de aplicaciones web seguras en ATICA, basándose en la guía OWASP de revisión de código (<http://www.owasp.org>), incluyendo una serie de puntos de verificación de la seguridad de la aplicación/módulo.

Además, para cada punto de verificación de la seguridad de la aplicación/módulo, se incluyen anotaciones relativas a como FUNDEWEB () puede ayudar a validar el citado punto de verificación

Puedes descargar este documento desde la Web de MNCS

(<http://www.um.es/atika/documentos/NORowasp.pdf>), y también desde el repositorio SVN de MNCS (<https://svn.atika.um.es/svn/MNCS/NuevaMNCS/OWASP/4.construccion/NORowasp.pdf>).

VERSIONES

Versión	Fecha	Autor	Descripción
0.1	10/06/2008	Juan Luis Serradilla Amarilla	Borrador 1
0.2	03/07/2008	Juan Luis Serradilla Amarilla	Borrador 2
1.0	07/07/2008	Juan Luis Serradilla Amarilla	Versión 1
1.1	27/10/2010	Juan Miguel Bernal González	Versión 1.1

Índice de contenido

1.Introducción.....	3
1.1.Objetivos.....	3
1.2.Ámbito.....	3
1.3.Referencias.....	3
2.Revisión de código.....	4
2.1.Autenticación.....	4
2.2.Autorización.....	5
2.3.Gestión de Cookies.....	5
2.4.Validación de Entrada de Datos.....	6
2.5.Gestión de Errores / Fuga de Información.....	7
2.6.Log / Auditoría.....	7
2.7.Cifrado de Datos.....	7
2.8.Entorno de Código Seguro.....	8
2.9.Gestión de Sesiones (Login / Logout).....	8
3.Ejemplos de cada Vulnerabilidad.....	9
4.Herramientas de revisión del código.....	10

1. Introducción.

La Sección de Metodologías, Normalización y Calidad del Software (en adelante MNCS) es una sección del Servicio de Desarrollo, Aplicaciones y Metodologías que tiene entre sus cometidos el estudio y evaluación de nuevas metodologías y tecnologías, destinadas a mejorar el proceso de construcción del software en ATICA.

En lo referente a Seguridad en Aplicaciones Web, hasta ahora (véase <http://www.um.es/atica/mncs/owasp>), MNCS ha recomendado a los desarrolladores aplicar las guías OWASP (<http://www.owasp.org>). Esto ha sido algo voluntario hasta ahora, y queremos incorporarlo como un **requisito no funcional obligatorio para todas las aplicaciones que se construyan en ATICA**.

En este sentido, MNCS ha empezado a elaborar una normativa sobre Desarrollo de Aplicaciones Web Seguras, basada en las guías OWASP (Building Guide, Code Review Guide and Testing Guide), con indicaciones para la construcción de software seguro, la revisión del código, y las pruebas de seguridad.

1.1. Objetivos

El objetivo del presente documento es definir la Normativa de Desarrollo de Aplicaciones Web Seguras en ATICA.

1.2. Ámbito

El ámbito de este documento es el la definición de la “Normativa de Desarrollo de Aplicaciones Web Seguras”, centrándose de momento en la revisión de código. Este documento se basa en la “Guía de Revisión de Código OWASP” (ver Referencias), por lo que recomendamos la lectura de la citada guía.

1.3. Referencias

Proyecto OWASP (“<http://www.owasp.org>”):

- Guía de Construcción de Aplicaciones y Servicios Web Seguros (“http://www.owasp.org/index.php/OWASP_Guide_Project”)
- Guía de Revisión de Código (“http://www.owasp.org/index.php/OWASP_Code_Review_Project”)
 - Libro en PDF: “<http://www.lulu.com/content/1415989>” ó “https://svn.atica.um.es/svn/MNCS/NuevaMNCS/OWASP/7.documentos/OWASP_Code_Review_2007__RC2.pdf”
- Guía de Pruebas (“http://www.owasp.org/index.php/OWASP_Testing_Project”)
- Preguntas Frecuentes sobre Seguridad en Aplicaciones Web (“http://www.owasp.org/index.php/OWASP_AppSec_FAQ”)
 - PDF en castellano del 2005 (“<http://www.um.es/atica/documentos/FAQSeguridadAplicacionesWebOWASP.pdf>”)
- OWASP Top Ten
(“http://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project”)

- OWASP Top Ten for J2EE
(“https://www.owasp.org/images/8/89/OWASP_Top_10_2007_for_JEE.pdf”)
- WebGoat (“http://www.owasp.org/index.php/OWASP_WebGoat_Project”)
- WebScarab (“http://www.owasp.org/index.php/OWASP_WebScarab_Project”)
- Portal FundeWeb – Guías Técnicas
(“https://fundeweb.um.es/prototipo//paginas/guias_tecnicas/principal.seam?actionMethod=paginas/home.xhtml:controlNavegacion.irIndice”)
- Portal FundeWeb – FAQs (“<https://fundeweb.um.es/prototipo/paginas/faqs/faqs.seam>”)
- Jboss Seam 2.2.0.GA – Documentación Infraestructura de Seguridad
(“<http://docs.jboss.org/seam/2.2.0.GA/en-US/html/security.html>”)

2. Revisión de código.

La revisión de código debe cubrir las siguientes áreas:

- Autenticación
- Autorización
- Gestión de Cookies
- Validación de Entrada de Datos
- Gestión de Errores / Fuga de Información
- Log / Auditoría
- Cifrado de Datos
- Entorno de Código Seguro
- Gestión de Sesiones (Login/Logout)

2.1. Autenticación

- Asegurar que todas las peticiones pasan por un formulario de autenticación, y que éste no se puede saltar (ojo con las URLs de acceso directo, que deberemos asegurar que pasan por una autenticación)
- Asegurar que todas las páginas cumplen el requisito de autenticación.
- Asegurar que siempre que se pasen credenciales de autenticación (o cualquier información sensible), sólo se aceptará la información vía HTTP POST y nunca con GET.

FundeWeb: Este requisito y los anteriores, se cumplen en FundeWeb siempre que la página o recurso, este dentro de la carpeta *paginas* del módulo web, o configuremos la página para ello. El generador siempre crea páginas aseguradas por autenticación. Más información en la [Guía Técnica de Autenticación](#) y en el punto de [Autenticación](#) de la documentación de JBoss Seam.

- Cualquier página para la que se descarte el mecanismo de autenticación debe ser revisada para asegurarse de que no tiene brechas de seguridad.

FundeWeb: este requisito debe de ser comprobado por el programador. Pero desde una página no protegida por autenticación, no se pueden acceder a recursos protegidos por

autenticación. Más información en la [Guía Técnica de Autenticación](#) y en el punto de [Autenticación](#) de la documentación de JBoss Seam.

- Asegurar que las credenciales de autenticación no van en claro.

FundeWeb: este requisito se cumple siempre que utilicemos el protocolo HTTPS. Por eso cuando rellenemos la plantilla de creación de aplicaciones, tenemos que indicar que queremos SSL.

- Asegurar que no hay “puertas traseras” en el código en producción.

FundeWeb: este requisito solo existe si se programa por algún programador o la hay en la librería de JBoss Seam.

2.2. Autorización

- Asegurar que tenemos mecanismos de autorización (control de acceso y gestión de roles).
- Asegurar que la aplicación tiene claramente definidos los tipos de usuario y sus privilegios.
- Asegurar que asignamos los mínimos privilegios necesarios.
- Asegurar que los mecanismos de autorización funcionan bien y no pueden saltarse.

FundeWeb: este requisito y los anteriores, se cumplen por FundeWeb, y no se pueden saltar, siempre que se siga la [Guía Técnica de Autorización](#), o al punto de [Autorización](#) de la documentación de JBoss Seam.

- Asegurarnos de chequear la autorización en todas las peticiones.

FundeWeb: este requisito debe de ser comprobado por el programador, ya en gran parte es su tarea indicar las comprobaciones de seguridad en cada elemento de la petición. Más información en la [Guía Técnica de Autorización](#), o al punto de [Autorización](#) de la documentación de JBoss Seam.

- Asegurar que no hay “puertas traseras” en el código en producción.

FundeWeb: este requisito solo existe si se programa por algún programador o la hay en la librería de JBoss Seam.

2.3. Gestión de Cookies

- Asegurarnos de no comprometer información sensible.

FundeWeb: la cookie utilizada por FundeWeb, solo almacena el identificador de la sesión y no almacena información sensible. Si el desarrollador utiliza cookies, entonces si se tienen que revisar.

- Asegurar que no se puedan hacer operaciones no autorizadas manipulando cookies.

FundeWeb: la cookie utilizada por FundeWeb, solo almacena el identificador de la sesión. Si el desarrollador utiliza cookies, entonces si se tienen que revisar.

- Asegurarnos de usar cifrado.

FundeWeb: este requisito se cumple siempre que utilicemos el protocolo HTTPS. Por eso cuando rellenemos la plantilla de creación de aplicaciones, tenemos que indicar que queremos SSL. Si el desarrollador utiliza cookies para almacenar datos, entonces si se tienen

que revisar, y usar algoritmos públicos fuertes (como AES) y no débiles (como MD5 o SHA-1).

- Determinar si todas las transiciones de estados en el código de la aplicación, verifican el uso seguro de cookies.

FundeWeb: la cookie utilizada por FundeWeb, solo almacena el identificador de la sesión, no hacen falta revisiones. Si el desarrollador utiliza cookies, entonces si se tienen que revisar.

- Asegurarnos de validar los datos de la sesión.

FundeWeb: .

- Asegurarnos que las cookies contienen la mínima información privada posible.

FundeWeb: la cookie utilizada por FundeWeb, solo almacena el identificador de la sesión. Si el desarrollador utiliza cookies, entonces si se tienen que revisar.

- Asegurarnos de cifrar una cookie completa si contiene información sensible.

FundeWeb: este requisito se cumple siempre que utilicemos el protocolo HTTPS. Por eso cuando rellenemos la plantilla de creación de aplicaciones, tenemos que indicar que queremos SSL. Si el desarrollador utiliza cookies para almacenar datos, entonces si se tienen que revisar, y usar algoritmos públicos fuertes (como AES) y no débiles (como MD5 o SHA-1).

- Definir todas las cookies que usa la aplicación, sus nombres y para qué son necesarias.

FundeWeb: la cookie utilizada por FundeWeb, solo almacena el identificador de la sesión. Si el desarrollador utiliza cookies, entonces si se tienen que revisar.

2.4. Validación de Entrada de Datos

- Asegurarnos de tener mecanismos de validación de datos.
- Asegurarnos de validar todas las entradas que pueden ser modificadas por un usuario malicioso: cabeceras HTTP, Input fields, hidden fields, drop down lists, etc.
- Asegurarnos de comprobar las longitudes de todas las entradas.
- Asegurarnos de validar todos los campos, cookies, http headers/bodies y form fields.
- Asegurarnos de formatear los datos convenientemente y que sólo contienen caracteres conocidos como buenos.
- Asegurarnos de validar los datos en el servidor.
- Asegurar que no hay “puertas traseras” en el modelo de validación.
- REGLA DE ORO: Cualquier entrada externa, sea cual sea, será examinada y validada.

FundeWeb: Se cumplen todos los requisitos, gracias a JSF y a la fase de Validación en el ciclo de vida de esta, se validan todos los datos de entrada en servidor. Además podemos crear nuevos validadores y usar los validadores de Hibernate. Incluso se pueden validar los parámetros de la URL. Más información en la [Guía Técnica de Validación](#), o al punto de [Validación](#) de la documentación de JBoss Seam.

2.5. Gestión de Errores / Fuga de Información

- Asegurar que todas las llamadas a métodos/funciones que devuelven un valor tienen su control de errores y además se comprueba el valor devuelto.

FundeWeb: este requisito depende del programador.

- Asegurarnos de gestionar adecuadamente las excepciones y los errores.
- Asegurar que al usuario no le devolvemos errores del sistema.
- Asegurar que la aplicación falla de un modo seguro.

FundeWeb: este requisito y los anteriores, dependen del programador. Pero en FundeWeb hay información al respecto para mejorar su gestión en los puntos [Gestión de Excepciones](#) y [Gestión de Excepciones de Seguridad](#) la documentación de JBoss Seam.

- Asegurarnos de liberar los recursos en caso de error.

FundeWeb: este requisito es tarea del programador.

2.6. Log / Auditoría

- Asegurar que no registramos información sensible en el log en caso de error.
- Asegurarnos de definir y controlar la longitud máxima de una entrada de log.
- Asegurar que no registramos datos sensibles en el log: cookies, método HTTP “GET”, credenciales de autenticación.
- Determinar si la aplicación auditará las operaciones lanzadas desde el cliente, sobre todo la manipulación de datos: Create, Update, Delete (operaciones CRUD).

FundeWeb: este requisito y los anteriores, dependen completamente del programador. FundeWeb indica mediante la [Guía Técnica de Log/Auditoría](#), como se tiene que utilizar el log/auditoría y que datos se deben mostrar.

- Asegurarnos de registrar en el log las operaciones de autenticación (fallidas o exitosas).

FundeWeb: este requisito se cumple por FundeWeb, si se sigue la [Guía Técnica de Autenticación](#). Si se utiliza otro tipo de autenticación de los especificados en la guía técnica, depende del programador su revisión. FundeWeb indica mediante la [Guía Técnica de Log/Auditoría](#), como se tiene que utilizar el log/auditoría y que datos se deben mostrar.

- Asegurarnos de registrar en el log los errores de la aplicación.
- Determinar si al hacer *debug* estamos registrando en el log datos sensibles.

FundeWeb: este requisito y el anterior, depende completamente del programador. FundeWeb indica mediante la [Guía Técnica de Log/Auditoría](#), como se tiene que utilizar el log/auditoría y que datos se deben mostrar.

2.7. Cifrado de Datos

- Asegurar que no transmitimos datos sensibles en claro, interna o externamente.

FundeWeb: este requisito se cumple siempre que utilicemos el protocolo HTTPS. Por eso cuando rellenemos la plantilla de creación de aplicaciones, tenemos que indicar que queremos SSL. Si se utiliza el protocolo HTTP, entonces es el desarrollador el que tiene que revisar el código. Para cifrar los datos es recomendable utilizar algoritmos públicos fuertes

(como AES) y no débiles (como MD5 o SHA-1).

- Asegurar que la aplicación implementa buenos y conocidos métodos criptográficos.

FundeWeb: este requisito depende completamente del programador. La recomendación de FundeWeb es utilizar algoritmos públicos fuertes (como AES) y no débiles (como MD5 o SHA-1).

2.8. Entorno de Código Seguro

- Examinar en la estructura de ficheros si hay algún componente que no debe estar directamente accesible para los usuarios.

FundeWeb: este requisito depende completamente del programador.

- Comprobar la gestión de memoria (reservar/liberar).

FundeWeb: este requisito es controlado por la máquina virtual de Java, que gestiona la memoria automáticamente.

- Comprobar si la aplicación usa SQL dinámico y determinar si es vulnerable a inyecciones de código.

FundeWeb: este requisito se cumple por FundeWeb, si se siguen las Guías Técnicas de [Creación de Consultas JPA en los Beans de Entidad](#), [Ejecutar Procedimientos y Consultas Almacenados en Base de Datos](#) y el punto [Using EL in EJB-QL/HQL](#) de la documentación de JBoss Seam.

- Comprobar si la aplicación tiene funciones “main()” ejecutables y depurar “puertas traseras”.
- Buscar código comentado (aunque sea para pruebas) que pueda contener información sensible.
- Asegurar que todas las bifurcaciones de código tengan su cláusula default (if-else, switch-default, etc).

FundeWeb: este requisito y los anteriores, dependen completamente del programador.

- Asegurar que no hay “development environment kits” en los directorios en explotación.

FundeWeb: este requisito se cumple por FundeWeb si solo se utilizan las tareas de sincronización para los entornos de desarrollo y explotación. Por lo que si los programadores no los suben, no los habrá. Si el desarrollador sube ficheros manualmente, tendra que ser revisada por él.

- Buscar llamadas al sistema operativo así como aperturas de ficheros, y comprobar las posibilidades de error.

FundeWeb: este requisito depende completamente del programador.

2.9. Gestión de Sesiones (Login / Logout)

- Comprobar cómo y cuándo se crean las sesiones de usuario, ya sean autenticadas o no.

FundeWeb: este requisito se cumple por FundeWeb. La sesión se crea automáticamente por el servidor al realizar la petición de una página.

- Comprobar el ID de sesión y verificar que tiene la complejidad necesaria para “ser fuerte”.

FundeWeb: este requisito se cumple por FundeWeb. La creación de las sesiones, las realiza el servidor APACHE (contenido en el OAS) genera identificadores de sesión fuertes.

- Comprobar cómo se almacenan las sesiones: en base de datos, en memoria, etc.
- Comprobar cómo la aplicación hace el seguimiento de las sesiones (track sessions).

FundeWeb: este requisito y el anterior, se cumple por FundeWeb. La gestión de las sesiones las realiza el servidor APACHE (contenido en el OAS).

- Determinar qué hace la aplicación en caso de encontrar un ID de sesión inválido.

FundeWeb: este requisito se cumple por FundeWeb. FundeWeb redirige a una página de error, informando del error y proporcionando un enlace a la página de autenticación. Más información en la [Guía Técnica de Autenticación](#) y en el punto de [Autenticación](#) de la documentación de JBoss Seam.

- Comprobar la invalidación de sesiones.

FundeWeb: este requisito se cumple por FundeWeb. FundeWeb invalida la sesión automáticamente al ejecutar el método *logout* del componente *Identity*. Más información en la [Guía Técnica de Autenticación](#) y en el punto de [Autenticación](#) de la documentación de JBoss Seam.

- Determinar cómo se gestionan las sesiones multithreaded/multi-user.

FundeWeb: este requisito se cumple por FundeWeb. La gestión multi-hilo, las realiza el OC4J (contenido en el OAS).

- Determinar el timeout de inactividad de la sesión HTTP.

FundeWeb: este requisito se cumple por FundeWeb. En FundeWeb se puede establecer un timeout de sesión, sino se establece, se configura por defecto a 30 segundos.

- Determinar cómo funciona el log-out.

FundeWeb: este requisito se cumple por FundeWeb. Más información en la [Guía Técnica de Autenticación](#) y en el punto de [Autenticación](#) de la documentación de JBoss Seam.

3. Ejemplos de cada Vulnerabilidad

En la “Guía de Revisión de Código OWASP”, se pueden encontrar ejemplos de cómo revisar el código para prevenir las siguientes vulnerabilidades ([“http://www.owasp.org/index.php/OWASP_Code_Review_Guide_Table_of_Contents”](http://www.owasp.org/index.php/OWASP_Code_Review_Guide_Table_of_Contents)):

- Buffer Overruns and Overflows
- OS Injection
- SQL Injection
- Data Validation
- XSS issues
- Cross-Site Request Forgery issues
- Error Handling
- The Secure Code Environment
- Authorization Issues

- Authentication
- Session Integrity issues
- Cryptographic Code
- Race Conditions

4. Herramientas de revisión del código

Además, para ayudarnos a aprender a revisar la seguridad de una aplicación web, OWASP proporciona:

- Preguntas Frecuentes Sobre Seguridad en Aplicaciones Web (OWASP FAQ) (<http://www.um.es/atika/documentos/FAQSeguridadAplicacionesWebOWASP.pdf>)
- Guía para un primer barrido del código: http://www.owasp.org/index.php/First_sweep_of_the_code_base#Searching_for_code_in_J2EE.2FJava.
- WebGoat (http://www.owasp.org/index.php/OWASP_WebGoat_Project): entorno de enseñanza realista en base a una aplicación J2EE que deliberadamente NO es segura. Por ejemplo, en la primera lección el usuario tiene que usar SQL-injection para “robar” números de tarjetas de crédito.
- WebScarab (http://www.owasp.org/index.php/OWASP_WebScarab_Project): herramienta para analizar la seguridad de las aplicaciones web. Actúa como un proxy HTTP/HTTPS y está escrito en Java.
- RatproxyDoc (<http://code.google.com/p/ratproxy/wiki/RatproxyDoc>): herramienta de Google para analizar la seguridad de las aplicaciones web, basada en OWASP.