Técnicas de programación segura para mitigar vulnerabilidades en aplicaciones web

Joffre Monar[a], Danilo Pástor[b], Gloria Arcos-Medina[b], Alejandra Oñate[b]
[a] Unidad de Admisión y Nivelación - sede Macas - Escuela Superior Politécnica de Chimborazo
[b] Facultad de Informática y Electrónica, Escuela Superior Politécnica de Chimborazo
jmonar@espoch.edu.ec, danilo.pastor@espoch.edu.ec, gloria.arcos@espoch.edu.ec, mayra.onate@espoch.edu.ec

***Resumen*—Actualmente, la gran mayoría de aplicaciones web contienen vulnerabilidades de seguridad. Probablemente, se deba a falta de cultura de los desarrolladores o a la ausencia de técnicas de codificación específicas. Se analizaron ciertos trabajos relacionados al tema, pero consideramos que no definen técnicas de programación precisos, ni se enfocan a un lenguaje de programación específico. El presente trabajo propone un conjunto de técnicas de programación segura para reducir las vulnerabilidades en las aplicaciones web utilizando el entorno de desarrollo PHP. Para esto se determinaron diez vulnerabilidades usando las recomendaciones OWASP TOP-10. Luego, se plantean las siete técnicas y su respectiva forma de implementarlas. Se valida las técnicas y se mide las vulnerabilidades de una aplicación web en dos escenarios; con y sin la implementación de las técnicas propuestas. Los resultados muestran que el uso de las técnicas propuestas se relaciona significativamente con la cantidad de vulnerabilidades encontradas y por lo tanto mejora el nivel de seguridad de las aplicaciones web.**

***Palabras Claves*—Vulnerabilidades, OWASP, ataques web, PHP, programación segura, seguridad web.**

***Abstract*—Currently, most web applications contain security vulnerabilities. Probably, it is due to lack of culture of the developers or the absence of specific coding techniques. We analyzed certain works related, but we consider that they do not define detailed programming techniques, nor do they focus on a specific programming language. This work proposes a set of secure programming techniques to reduce vulnerabilities in web applications using the PHP. For this, ten vulnerabilities were determined using the OWASP TOP-10 recommendations. Then, the seven techniques are presented and their respective way of implementing them. The techniques are validated; the vulnerabilities of a web application are measured in two scenarios; with and without the implementation of the proposed techniques. The results show that the use of the proposed techniques is significantly related to the number of vulnerabilities found and therefore improves the level of security of web applications.**

***Keywords*—Vulnerabilities, OWASP, web attacks, PHP, secure programming, web security.**

I. Introducción

Actualmente las empresas tanto públicas como privadas están incrementado el despliegue de sus aplicaciones web a través de redes globales en Internet. Esto implica que los sistemas basados en la web deberían ser revisados de manera minuciosa. En este contexto la seguridad sigue siendo un obstáculo importante para la aceptación mundial de la web, especialmente desde el reciente aumento de las vulnerabilidades explotables que han sido atribuidas a errores de las aplicaciones web [1].

A pesar que existir un nivel de madurez bastante aceptable de la seguridad informática en aspectos relacionados con la infraestructura tecnológica, sistemas operativos y bases de datos, se evidencia que aún existen vulnerabilidades en la web, por lo que siguen siendo un gran problema, ya que de acuerdo al reporte de Symantec de abril del 2017 se detectó un promedio de más de 229,000 ataques web todos los días en el 2016, también el 76% de los sitios web escaneados contenían vulnerabilidades, 9% de los cuales se consideraron críticos [2]. Esto denota que el nivel de seguridad de las aplicaciones web debe ser mitigado de manera prioritaria por parte de los desarrolladores web.

La programación de aplicaciones web seguras no es una tarea fácil, ya que el desarrollador debe centrar la atención en requerimientos funcionales y también cumplir recomendaciones de codificación segura como el manejo de peticiones, filtrado de datos, limpieza de variables entre otras [3].

Se realizó una revisión de trabajos enfocados a proponer guías, recomendaciones y/o metodologías en donde se presentan alternativas de mejora de las seguridades de aplicaciones web. En el trabajo de [4] se presenta un enfoque totalmente automatizado para reforzar de forma segura las aplicaciones web, pero no se validan las estrategias propuestas. En otro estudio se describe una combinación de características estáticas y de tiempo de ejecución que garantiza la seguridad de aplicaciones web [1], no obstante, su propuesta se orienta un enfoque holístico y práctico. Además, en la propuesta de [5] se usa un entorno de desarrollo integrado recordando interactivamente a los programadores de las prácticas de programación segura. Finalmente, en los trabajos [6]-[8] proponen metodologías y buenas prácticas de programación segura, pero se basan en las recomendaciones de OWASP, ISO 27001, y OSSTMM, las cuales son muy generales y no se enfocan a un lenguaje de programación específico.

Además, se analizó estudios relacionados a revisiones sistemáticas de literatura. En el estudio de [9] se describe el estado del arte actual para asegurar aplicaciones web a partir de fallas importantes tales como defectos de inyección y lógica. Otro aporte [10] hace una evaluación de la situación actual de los sitios web chinos utilizando plataformas reales de revelación de vulnerabilidades. Otro trabajo [11], realizó enfoque sistemático para establecer una base de datos sustancial y completa de la literatura de última generación centrada en la detección de ataques. Finalmente, en la artículo [12] se lleva a cabo una revisión sistemática de la literatura sobre vulnerabilidades y ataques XSS hechas en trabajos afines.

El objetivo principal de este artículo es proponer un conjunto de técnicas de programación segura para reducir el riesgo de recibir ataques informáticos en las aplicaciones

web usando en el entorno de desarrollo PHP. Para verificar el nivel de seguridad de las técnicas propuestas se realizó un análisis de vulnerabilidades en dos escenarios; antes y después de la aplicación de las estrategias de programación seguras.

El artículo está organizado de la siguiente manera: en la segunda sección se presenta el método utilizado para definir las estrategias de programación y su validación respectiva; en la tercera sección de describe los resultados obtenidos de análisis de vulnerabilidades detectadas mediante la aplicación de las estrategias de programación seguras; y finalmente la cuarta sección se detalla algunas conclusiones del trabajo de investigación.

II. MÉTODO

Para lograr el propósito de esta investigación, se aplicó el presente procedimiento:

A. Determinación de vulnerabilidades de estudio

En esta investigación se tomó como base la utilización de los Controles Proactivos TOP-10 OWASP 2016 [13] que consideran las diez vulnerabilidades más críticas, de las cuales se seleccionaron siete: 1) Inyección, 2) Pérdida de autenticación y gestión de sesiones, 3) Secuencia de comandos en sitios cruzados (XSS), 4) Configuración de seguridad incorrecta, 5) Exposición de datos sensibles, 6) Falsificación de peticiones en sitios cruzados (CSRF) y 7) Redirecciones y reenvíos no validados.

Para la selección de estas vulnerabilidades se consideraron los errores más comunes de programación: validación de entradas, gestión de sesiones, cifrado de datos, Cross Site Scripting (XSS), instalación y configuración incorrecta del servidor web y base de datos y CSRF.

B. Técnicas Propuestas

Para definir esta propuesta se tomó como base las recomendaciones de: TOP-10 Controles Proactivos en su versión dos del 2016, la Guía de Pruebas OWASP en su versión cuatro del 2013, así como los Controles Estándar de Seguridad (ASVS) en su versión tres del 2015; todo esto implementado con las técnicas de programación de PHP.

Para cumplir con el objetivo de este estudio se proponen siete grupos de técnicas presentadas en la Tabla I.

TABLA I
TÉCNICAS DE PROGRAMACIÓN SEGURA

Técnica	Implementación	Vulnerabilidad que previene
T1. Parametrizar consultas	- Consultar - Insertar - Actualizar - Eliminar - Restricciones en los parámetros	- Injection
T2. Codificar los datos	- Sanear el HTML - Configurar las cookies para que sean httponly y secure	- Injection - XSS
T3. Validar todas las entradas	- Expresiones regulares - Validación y saneamiento Validación de direcciones IP Redirecciones y reenvíos no validados Enlaces desde $ _SERVER Remover caracteres ASCII con Valor> 127	- Injection - XSS - Redireccion es y reenvíos no validados
T4. Implementar controles de identidad y autenticación	- Medidas durante la autenticación Prevenir ataques de fuerza bruta Captcha Recordar contraseña - Gestión de sesiones Inicio de sesión segura Estado de la sesión iniciada Cierre de sesión - Falsificación de peticiones en sitios cruzados (CSRF) Crear la clase y funciones Proteger un formulario POST	- Pérdida de autenticació n y gestión de sesiones - Falsificación de peticiones en sitios cruzados (CSRF)
T5. Proteger los datos	- Algoritmo cifrado - Cifrado de contraseñas Hash de Contraseñas	- Exposición de datos sensibles
T6. Error y control de excepciones	- Respuestas try/catch - Respuestas de autentificación	- Todas las OWASP top 10
T7. Configuración adecuada	- Instalar un servidor web, PHP y MySQL en el servidor Mantener el software actualizado - Seguridad de la base de datos Configuración de la base de datos MYSQL Conexión con la base de datos Desconexión con la base de datos Excepciones - Archivos de configuración - Reporte de errores - Estar al tanto de las actualizaciones de los productos utilizados - Comprobar la información que se obtiene de las cabeceras HTTP - Verificar los servicios visibles - Buscar vulnerabilidades - Revisar la configuración HTTPS	- Todas las OWASP top 10

Cada técnica cuenta con su descripción y código para implementar en PHP. Se muestra la siguiente sub-técnica:

- Técnica T3: Validar todas las entradas.
- Sub-técnica: Redirecciones y reenvíos no validados
- Descripción: No se debe confiar de los datos que provienen del exterior de su aplicación PHP. Siempre depurar y verificar los datos de entrada antes de usarlos en el código. Las funciones filter_var() y filter_input() ayudan a depurar los datos y verifican la

validez del formato del texto.

- Código:

```
<?php var_dump(filter_var('bob@example.com',
FILTER_VALIDATE_EMAIL));
var_dump(filter_var('http://example.com',
FILTER_VALIDATE_URL,
FILTER_FLAG_PATH_REQUIRED)); ?>
```

C. *Determinación del nivel de Seguridad*

La seguridad fue determinada mediante el nivel de protección de la aplicación web contra las vulnerabilidades más conocidas, es decir, el número de vulnerabilidades altas, medias y bajas detectadas utilizando *Acunetix Vulnerability Scanner* un software que permite el escaneo de sitios web, incluyendo aplicaciones web integradas, servidores web y chequea automáticamente vulnerabilidades que aparentemente no son detectadas.

D. *Escenarios de Prueba*

Para validar las técnicas propuestas se utilizó el sistema SISEV desarrollado en PHP 5.6.15, que automatiza el seguimiento y evaluación de las actividades de programas y proyectos de una institución pública del Ecuador; cuenta con 5 módulos y tiene un tamaño de 12.120 líneas de código. Este sistema fue ejecutado en dos escenarios: 1) Sin la aplicación de las técnicas de programación segura (Ambiente 1-N) y 2) Con la aplicación de técnicas de programación segura (Ambiente 2-T). En cada uno de estos escenarios se determinó la cantidad de vulnerabilidades de cada tipo. Así mismo se contabilizó la cantidad de vulnerabilidades eliminadas.

III. Análisis de Resultados

E. *Resultados de la determinación de vulnerabilidades*

Los resultados obtenidos en la etapa de determinación de vulnerabilidades se muestran en la Tabla II, diez vulnerabilidades fueron establecidas, de las cuales dos son de alto grado, dos de grado medio y seis de bajo grado de incidencia en la seguridad de una aplicación web.

La Tabla III describe los resultados de las vulnerabilidades detectadas luego de la aplicación de las estrategias de programación seguras, a fin de mitigar los riesgos de seguridad en aplicaciones web.

Por otra parte, en la Figura 1 se ilustra las frecuencias numéricas de las vulnerabilidades observadas antes y después de la utilización de las estrategias de programación seguras, en la que se evidencia la reducción del 100%, 87,69% y 85,71% de las vulnerabilidades altas, medias y bajas respectivamente. Obteniendo una mejora promedio del 91,13%. Estos resultados no son suficientes para concluir que la utilización de las técnicas propuestas mejora el nivel de seguridad de las aplicaciones web, por lo que es necesario realizar una prueba estadística inferencial que nos permita concluir esto con mayor certeza.

TABLA II
VULNERABILIDADES DETECTADAS ANTES DE LA APLICACIÓN DE LAS ESTRATEGIAS DE PROGRAMACIÓN SEGURAS

Vulnerabilidad	Grado	Cant.	Descripción
Blind SQL Injection	Alto	3	Técnica de ataque que utiliza Inyección SQL
PHP Hash Collision	Alto	1	Vulnerabilidad de tipo denial of service vulnerability			denegación de servicio
Directory listing	Medio	63	Es una función que lista todos los archivos cuando no hay un archivo de índex
HTML form without CSRF protection	Medio	2	Formulario HTML sin protección CSRF
Clickjacking	Bajo	1	Secuestro de sesión
Cookie without HttpOnly flag set	Bajo	1	Puede causar un desbordamiento de memoria.
Login page password-guesting attack	Bajo	1	Página de inicio de sesión puede ser atacada
OPTIONS method is enabled	Bajo	1	El método OPTIONS está activado
Possible relative path overwrite	Bajo	9	Posible sobre escritura de ruta
TRACE method is enabled	Bajo	1	El método TRACE está habilitado

TABLA III
VULNERABILIDADES DETECTADAS DESPUÉS DE LA APLICACIÓN DE LAS ESTRATEGIAS DE PROGRAMACIÓN SEGURAS

Vulnerabilidad	Grado	Cant.	Descripción
Directory listing	Medio	8	Es una función de servidor web que muestra una lista de todos los archivos cuando no hay un archivo de índice
Possible relative path overwrite	Bajo	1	Posible sobre escritura de ruta
TRACE method is enabled	Bajo	1	El método TRACE está habilitado

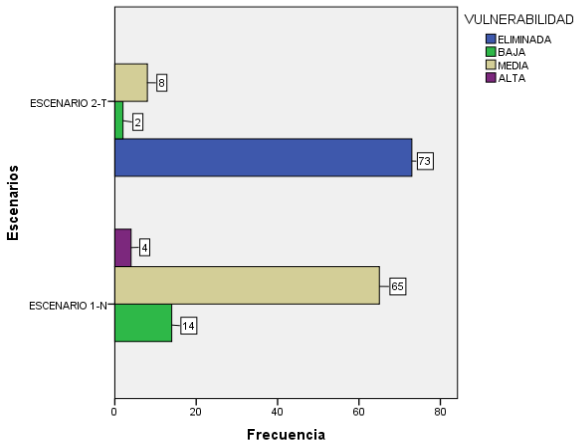

Fig. 1. Frecuencia de vulnerabilidades en dos escenarios

F. *Prueba de Hipótesis*

Lo que corresponde en este trabajo de investigación es analizar los resultados obtenidos a fin de poder demostrar la incidencia positiva de la utilización de técnicas de programación segura con el fin de mejorar la seguridad de las aplicaciones web, las mismas son:

- Ha: La utilización de estrategias de programación seguras se relaciona significativamente con la cantidad de vulnerabilidades encontradas en la aplicación web.
- H0: La utilización de estrategias de programación seguras no se relaciona significativamente con la cantidad de vulnerabilidades encontradas en la aplicación web.

El análisis de las variables definidas para demostrar la

hipótesis se detalla en la Tabla IV.

TABLA IV
TABLA DE CONTINGENCIA ESCENARIOS - VULNERABILIDAD

ESCENARIOS		VULNERABILIDAD				Total
		Eliminada	Baja	Media	Alta	
Escenario 1-N	Recuento	0	14	65	4	83
	Recuento esperado	36,5	8,0	36,5	2,0	83,0
	% del total	0,0%	16,9%	78,3%	4,8%	100,0%
Escenario 2-T	Recuento	73	2	8	0	83
	Recuento esperado	36,5	8,0	36,5	2,0	83,0
	% del total	88,0%	2,4%	9,6%	0,0%	100,0%
Total	Recuento	73	16	73	4	166
	Recuento esperado	73,0	16,0	73,0	4,0	166,0

Como se observa en la Tabla V, el valor de Chi-cuadrado es de 130.507 con 3 grados de libertad y un nivel de significancia asintótica P = 0.000, P > 0.05 por lo que se rechaza la H0 y se acepta la Ha, es decir que a utilización de estrategias de programación seguras se relaciona significativamente con la cantidad de vulnerabilidades encontradas en la aplicación web y por lo tanto mejora el nivel de seguridad de las aplicaciones web en entorno PHP.

TABLA V
RESULTADOS DE LA PRUEBA DE CHI-CUADRADO

	Valor	**gl**	**Sig. asintótica (2 caras)**
Chi-cuadrado de Pearson	130,507	3	0,000

IV. CONCLUSIONES

El principal aporte de este trabajo de investigación es proponer treinta y cinco técnicas de programación segura organizada en siete grupos, las cuales se enfocan a evitar algunos de los errores muy comunes en programación, tales como: validación de entradas, gestión de sesiones, cifrado de datos, Cross Site Scripting (XSS), instalación y configuración incorrecta del servidor web y base de datos, y CSRF. Por lo que se logra reducir el riesgo de recibir ataques informáticos en las aplicaciones web.

En este trabajo se comprueba que la utilización de técnicas de programación segura se relaciona significativamente con la cantidad de vulnerabilidades encontradas en la aplicación web y por lo tanto mejora el nivel de seguridad de las aplicaciones web en entorno PHP.

El presente estudio puede ser complementado añadiendo otras técnicas de programación segura para cubrir vulnerabilidades que no fueron consideradas en el presente documento tales como: referencia directa insegura a objetos, ausencia de control de acceso a funciones y uso de componentes con vulnerabilidades conocidas.

REFERENCIAS

[1] Y.-W. Huang, F. Yu, C. Hang, C.-H. Tsai, D.-T. Lee, y S.-Y. Kuo, "Securing Web Application Code by Static Analysis and Runtime Protection," in *Proc. of the 13th International Conference on World Wide Web*, New York, NY, USA, 2004, pp. 40–52.

[2] K. Chandrasekar et al., "Internet Security Threat Report," *Symantec*, Reporte, abr. 2017.

[3] A. Mier y Terán y M. V. Martínez, "Aspectos Básicos de la Seguridad en Aplicaciones Web | Documentos - CSI -," 25-may-2016. [En línea]. Disponible en: https://www.seguridad.unam.mx/historico/documento/index.html-id=17. [Accedido: 13-abr-2018].

[4] A. Nguyen-Tuong, S. Guarnieri, D. Greene, J. Shirley, y D. Evans, "Automatically Hardening Web Applications Using Precise Tainting," in *Security and Privacy in the Age of Ubiquitous Computing*, 2005, pp. 295-307.

[5] J. Xie, B. Chu, H. R. Lipford, y J. T. Melton, "ASIDE: IDE Support for Web Application Security," in *Proc. of the 27th Annual Computer Security Applications Conference*, New York, NY, USA, 2011, pp. 267–276.

[6] M. A. Mendoza, M. Patiño, y P. Julián, "Desarrollo de una propuesta metodológica para determinar la seguridad en una aplicación web," 2011.

[7] Y. Romero y E. Nataly, "Guía de buenas prácticas de desarrollo de aplicaciones web seguras aplicado al sistema control de nuevos aspirantes Empresa Grupo LAAR," Dic. 2014.

[8] F. López Provencio, "Desarrollo dirigido por la seguridad," 2015.

[9] G. Deepa y P. S. Thilagam, "Securing web applications from injection and logic vulnerabilities: Approaches and challenges," *Inf. Softw. Technol.*, vol. 74, pp. 160-180, jun. 2016.

[10] C. Huang, J. Liu, Y. Fang, y Z. Zuo, "A study on Web security incidents in China by analyzing vulnerability disclosure platforms," *Comput. Secur.*, vol. 58, pp. 47-62, may 2016.

[11] K. Singh, P. Singh, y K. Kumar, "Application layer HTTP-GET flood DDoS attacks: Research landscape and challenges," *Comput. Secur.*, vol. 65, pp. 344-372, mar. 2017.

[12] I. Hydara, A. B. M. Sultan, H. Zulzalil, y N. Admodisastro, "Current state of research on cross-site scripting (XSS) – A systematic literature review," *Inf. Softw. Technol.*, vol. 58, pp. 170-186, feb. 2015.

[13] OWASP Foundation, "OWASP Top 10 Proactive Controls 2016," 2016.