



ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

PROPUESTA DE UN MÉTODO UTILIZANDO TÉCNICAS DE PROGRAMACIÓN SEGURAS PARA EL DESARROLLO DE APLICACIONES WEB EN ENTORNO PHP PARA MITIGAR RIESGOS POTENCIALES DE SEGURIDAD

JOFFRE STALIN MONAR MONAR

Trabajo de Titulación modalidad: Proyectos de Investigación y Desarrollo, presentado ante el Instituto de Posgrado y Educación Continua de la ESPOCH, como requisito parcial para la obtención del grado de:

MAGISTER EN SEGURIDAD TELEMÁTICA

Riobamba - Ecuador

Junio 2017



ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

CERTIFICACIÓN:

EL TRIBUNAL DE TRABAJO DE TITULACIÓN CERTIFICA QUE:

El **Trabajo de Titulación modalidad Proyectos de Investigación y Desarrollo**, titulado: “PROPUESTA DE UN MÉTODO UTILIZANDO TÉCNICAS DE PROGRAMACIÓN SEGURAS PARA EL DESARROLLO DE APLICACIONES WEB EN ENTORNO PHP PARA MITIGAR RIESGOS POTENCIALES DE SEGURIDAD”, de responsabilidad del Ing. Joffre Stalin Monar Monar, ha sido prolijamente revisada y se autoriza su presentación.

Tribunal:

Ing. Wilson Zúñiga Vinueza; M. Sc.

PRESIDENTE

Ing. Danilo Mauricio Pastor Ramírez; M. Sc.

DIRECTOR

Ing. Juan Carlos Díaz Ordoñez; M. Sc.

MIEMBRO

Ing. Lady Marieliza Espinoza Tinoco; M. Sc.

MIEMBRO

Riobamba, Junio 2017

DERECHOS INTELECTUALES

Yo, Joffre Stalin Monar Monar, declaro que soy responsable de las ideas, doctrinas y resultados expuestos en el **Trabajo de Titulación modalidad Proyectos de Investigación y Desarrollo**, y que el patrimonio intelectual generado por la misma pertenece exclusivamente a la Escuela Superior Politécnica de Chimborazo.

Joffre Stalin Monar Monar

No. Cédula: 120428768-2

DECLARACIÓN DE AUTENTICIDAD

Yo, Joffre Stalin Monar Monar, declaro que el presente **Trabajo de Titulación modalidad Proyectos de Investigación y Desarrollo**, es de mi autoría y que los resultados del mismo son auténticos y originales. Los textos constantes en el documento que provienen de otra fuente están debidamente citados y referenciados.

Como autor, asumo la responsabilidad legal y académica de los contenidos de este proyecto de investigación de maestría.

Riobamba, Junio de 2017

Joffre Stalin Monar Monar
No. Cédula: 120428768-2

DEDICATORIA

Agradezco en primer lugar a Dios quién supo guiarme y darme fortaleza para la culminación de este trabajo de titulación.

A mi amada esposa María del Carmen Vargas por su paciencia, comprensión y sabios consejos que me ayudaron a culminar el presente trabajo de titulación; además de ser el pilar fundamental en mi vida.

A mis queridos hijos Alisson, David y Renata Monar Vargas quienes con su inocencia siempre estuvieron a mi lado brindándome su apoyo incondicional.

A mis padres Antonio Monar y Ana Monar quienes estuvieron pendientes en todo momento además de ser mis garantes para iniciar esta carrera.

A mis hermanos Mabell, Darío y Daniel Monar Monar con quienes compartí muchos días de alegría y enseñanza.

A mi suegro Segundo Vargas por apoyarme y quién al momento está atravesando una situación difícil de salud y que esperamos todos pronto mejore.

Joffre Stalin

AGRADECIMIENTO

A la Escuela Superior Politécnica de Chimborazo por darme la oportunidad de obtener este nuevo título profesional.

Mi agradecimiento especial al Dr. Danilo Pastor Ramírez, Director de esta investigación quién aportó con sus conocimientos, apoyo y paciencia a la culminación de esta tesis.

A mis suegros Segundo Vargas y Teresa Velasteguí que me motivaron durante el desarrollo del presente trabajo de titulación.

A todos los catedráticos de esta maestría por su profesionalismo y dedicación en impartir sus conocimientos.

Joffre Stalin

TABLA DE CONTENIDO

	Páginas
RESUMEN.....	xiii
SUMMARY.....	xiv
CAPÍTULO I	
1. INTRODUCCIÓN.....	1
1.1 Planteamiento del problema	1
1.1.1 Situación Problemática.....	1
1.1.2 Formulación del problema.....	2
1.1.3 Preguntas directrices o específicas de la investigación	3
1.2 Justificación de la investigación	3
1.2.1 Justificación Teórica.....	3
1.2.2 Justificación Práctica.....	4
1.2.3 Justificación Metodológica.....	5
1.3 Objetivos de la investigación.....	5
1.3.1 Objetivo General	5
1.3.2 Objetivos Específicos	6
1.4 Hipótesis.....	6
CAPÍTULO II	
2. MARCO TEÓRICO.....	7
2.1 Antecedentes del problema.....	7
2.1.1 Análisis de metodologías propuestas para mejorar las seguridades en aplicaciones web ..	7
2.1.2 Pros y contras de las metodologías a estudiar.....	7
2.2 Bases teóricas	9
2.2.1 Aplicaciones Web	9
2.2.2 Seguridades en Aplicaciones Web.....	11
2.2.3 Seguridades en PHP	14
2.2.4 Principales vulnerabilidades detectadas en aplicaciones web	16
2.2.5 Analizadores de vulnerabilidades	19
2.2.6 Controles estándar de seguridad	21
2.2.7 Recomendaciones generales para programar de manera segura	22
2.2.8 Recomendaciones para evitar las diez (10) vulnerabilidades más críticas de OWASP ...	27

CAPÍTULO III

3.	METODOLOGÍA DE INVESTIGACIÓN.....	35
3.1	Introducción	35
3.2	Tipo y diseño de la Investigación	35
3.2.1	Tipo de Investigación	35
3.2.2	Diseño de la Investigación.....	35
3.3	Métodos de Investigación.....	36
3.4	Fuentes de información	36
3.4.1	Primarias	36
3.4.2	Secundarias	37
3.5	Técnicas de recolección de datos primarios y secundarios	37
3.6	Planteamiento de la Hipótesis.....	37
3.6.1	Hipótesis General	37
3.6.2	Identificación de variables.....	38
3.6.3	Operacionalización de variables.....	38
3.7	Población y muestra	39
3.7.1	Población.....	39
3.7.2	Selección de la muestra	39
3.8	Procedimientos generales	40
3.9	Instrumentos de recolección de datos primarios y secundarios.....	41
3.10	Instrumentos para procesar datos recopilados	43
3.11	Ambiente de Pruebas.....	44
3.11.1	Hardware y software utilizado.....	44
3.11.2	Prototipos de prueba.....	45
3.12	Selección de la metodología	45
3.12.1	Metodología seleccionada	46

CAPÍTULO IV

4.	RESULTADOS Y DISCUSIÓN.....	48
4.1	Presentación de resultados.....	48
4.2	Identificación de activos de información.....	48
4.2.1	Servicio Web.....	48
4.2.2	Servidor Principal.....	49
4.2.3	Servidor de Pruebas.....	50
4.2.4	Equipos de Protección	51
4.3	Procesamiento y análisis	51
4.4	Valoraciones de la variable independiente	51

4.4.1	Variable independiente: Método propuesto de programación segura para desarrollo de aplicaciones web en PHP	51
4.4.2	Indicador: Nivel de satisfacción del usuario	52
4.5	Valoraciones de la variable dependiente.....	67
4.5.1	Variable dependiente: Mejora en el nivel de la seguridad	67
4.5.2	Indicador: Número total de vulnerabilidades detectadas	67
4.6	Comprobación estadística de la hipótesis	72
CAPÍTULO V		
5.	PROPUESTA	76
5.1	Determinación de la propuesta	76
5.2	Propuesta de un método para el desarrollo de aplicaciones web utilizando técnicas de programación segura en entornos PHP.....	77
5.2.1	Parametrizar las consultas	77
5.2.2	Codificar los datos.....	80
5.2.3	Validar todas las entradas.....	81
5.2.4	Implementar controles de identidad y autenticación.....	84
5.2.5	Proteger los datos	91
5.2.6	Error y control de excepciones	92
5.2.7.	Configuración adecuada	93
CONCLUSIONES		100
RECOMENDACIONES		102
GLOSARIO		
BIBLIOGRAFÍA		
ANEXOS		

ÍNDICE DE TABLAS

Tabla 1-2:	Comparación de metodologías	8
Tabla 1-3:	Operacionalización conceptual de variables.....	38
Tabla 2-3:	Operacionalización Metodológica de variables.....	38
Tabla 3-3:	Vulnerabilidades a evaluar	40
Tabla 4-3:	Técnicas para la demostración de la hipótesis	41
Tabla 5-3:	Herramientas de detección de vulnerabilidades	42
Tabla 6-3:	Escala de Likert	42
Tabla 7-3:	Pesos asignados para la selección de la herramienta	43
Tabla 8-3:	Hardware utilizado para pruebas.....	44
Tabla 9-3:	Software utilizado para pruebas	45
Tabla 1-4:	Activos utilizados en la investigación	48
Tabla 2-4:	Servicio Web	49
Tabla 3-4:	Número de usuarios que utilizan el Servicio Web.....	49
Tabla 4-4:	Características del Servidor Principal	50
Tabla 5-4:	Características del Servidor de Pruebas	50
Tabla 6-4:	Equipos de Protección.....	51
Tabla 7-4:	Encuesta-Pregunta 1-antes de aplicar la propuesta.....	52
Tabla 8-4:	Encuesta- Pregunta 2-antes de aplicar la propuesta.....	53
Tabla 9-4:	Encuesta- Pregunta 3-antes de aplicar la propuesta.....	53
Tabla 10-4:	Encuesta- Pregunta 4-antes de aplicar la propuesta.....	54
Tabla 11-4:	Encuesta- Pregunta 5-antes de aplicar la propuesta.....	55
Tabla 12-4:	Encuesta- Pregunta 6-antes de aplicar la propuesta.....	56
Tabla 13-4:	Encuesta- Pregunta 7-antes de aplicar la propuesta.....	56
Tabla 14-4:	Encuesta- Pregunta 8-antes de aplicar la propuesta.....	57
Tabla 15-4:	Resumen de resultados de Encuesta-Antes de aplicar el método.....	58
Tabla 16-4:	Encuesta- Pregunta 1-después de aplicar la propuesta.....	59
Tabla 17-4:	Encuesta- Pregunta 2-después de aplicar la propuesta.....	60
Tabla 18-4:	Encuesta- Pregunta 3-después de aplicar la propuesta.....	60
Tabla 19-4:	Encuesta- Pregunta 4-después de aplicar la propuesta.....	61
Tabla 20-4:	Encuesta- Pregunta 5-después de aplicar la propuesta.....	62
Tabla 21-4:	Encuesta- Pregunta 6-después de aplicar la propuesta.....	62
Tabla 22-4:	Encuesta- Pregunta 7-después de aplicar la propuesta.....	63
Tabla 23-4:	Encuesta- Pregunta 8-después de aplicar la propuesta.....	64

Tabla 24-4:	Resumen de resultados de Encuesta-Después de aplicar el método.....	65
Tabla 25-4:	Resumen de resultados de Encuesta-Antes y Después de aplicar el método.....	66
Tabla 26-4:	Identificación de Vulnerabilidades con ACUNETIX antes del método.....	68
Tabla 27-4:	Total de Vulnerabilidades con ACUNETIX antes del método.....	68
Tabla 28-4:	Identificación de Vulnerabilidades con ACUNETIX después del método.....	70
Tabla 29-4:	Total de Vulnerabilidades con ACUNETIX después del método.....	70
Tabla 30-4:	Comparativa de Vulnerabilidades antes y después del método.....	71
Tabla 31-4:	Frecuencias de Valores Observados.....	72
Tabla 32-4:	Frecuencias de Valores Esperados... ..	73

ÍNDICE DE FIGURAS

Figura 1-2:	Arquitectura Web en 3 Capas	10
Figura 2-2:	Lenguajes de Programación	11
Figura 3-2:	Tipos de ataques o amenazas	13
Figura 4-2:	Descripción básica de SQL Injection	28
Figura 5-2:	Pérdida de autenticación	29
Figura 6-2:	Almacenamiento criptográfico inseguro	32

ÍNDICE DE GRÁFICOS

Gráfico 1-4:	Encuesta- Pregunta 1-antes de aplicar la propuesta.....	52
Gráfico 2-4:	Encuesta- Pregunta 2-antes de aplicar la propuesta.....	53
Gráfico 3-4:	Encuesta- Pregunta 3-antes de aplicar la propuesta.....	54
Gráfico 4-4:	Encuesta- Pregunta 4-antes de aplicar la propuesta.....	54
Gráfico 5-4:	Encuesta- Pregunta 5-antes de aplicar la propuesta.....	55
Gráfico 6-4:	Encuesta- Pregunta 6-antes de aplicar la propuesta.....	56
Gráfico 7-4:	Encuesta- Pregunta 7-antes de aplicar la propuesta.....	57
Gráfico 8-4:	Encuesta- Pregunta 8-antes de aplicar la propuesta.....	57
Gráfico 9-4:	Resumen de resultados de Encuesta-Antes de aplicar el método	58
Gráfico 10-4:	Encuesta- Pregunta 1-después de aplicar la propuesta.....	59
Gráfico 11-4:	Encuesta- Pregunta 2-después de aplicar la propuesta.....	60
Gráfico 12-4:	Encuesta- Pregunta 3-después de aplicar la propuesta.....	61
Gráfico 13-4:	Encuesta- Pregunta 4-después de aplicar la propuesta.....	61
Gráfico 14-4:	Encuesta- Pregunta 5-después de aplicar la propuesta.....	62
Gráfico 15-4:	Encuesta- Pregunta 6-después de aplicar la propuesta.....	63
Gráfico 16-4:	Encuesta- Pregunta 7-después de aplicar la propuesta.....	63
Gráfico 17-4:	Encuesta- Pregunta 8-después de aplicar la propuesta.....	64
Gráfico 18-4:	Resumen de resultados de Encuesta-Después de aplicar el método.....	65
Gráfico 19-4:	Resumen de resultados de Encuesta-Antes y Después de aplicar el método....	66
Gráfico 20-4:	Total de Vulnerabilidades antes del método.....	69
Gráfico 21-4:	Total de Vulnerabilidades después del método.....	70
Gráfico 22-4:	Comparativo de Vulnerabilidades antes y después del método.....	71
Gráfico 23-4:	Distribución Chi Cuadrado.....	75

RESUMEN

La investigación tuvo como objetivo proponer un método de desarrollo utilizando técnicas de programación segura para reducir el riesgo de recibir ataques informáticos en las aplicaciones web en lenguaje de programación Hypertext Pre-processor (PHP). El método propuesto fue desarrollado con técnicas de programación para PHP 5.6.x que contemplaron siete etapas: Parametrización de consultas, Codificación de datos, Validación de todas las entradas, Implementación de controles de identidad y autenticación, Protección de los datos, Error y control de excepciones y Configuración adecuada, las que fueron adaptadas de las diferentes guías y recomendaciones del Proyecto Open Web Application Security Project (OWASP). Se planteó la hipótesis “El método de desarrollo propuesto mejorará el nivel de seguridad de las aplicaciones web en entorno PHP ante posibles ataques informáticos”. Se aplicó Chi-Cuadrado con un nivel de significancia de 0.05. Los indicadores fueron: Mejora en el nivel de la seguridad y Número total de vulnerabilidades detectadas, de una muestra de ocho usuarios para la variable dependiente y siete vulnerabilidades para la variable independiente, utilizando encuestas y analizador de vulnerabilidades Acunetix Web Vulnerability Scanner respectivamente. De acuerdo a los resultados obtenidos el prototipo 2 Sistema de Seguimiento y Evaluación Seguro (S-SISEV) mejoró en un 97% y 87.95% respectivamente con relación al prototipo 1. Se concluye que el método propuesto evita algunos de los errores más comunes en programación y se recomienda desarrollar nuevas técnicas de programación en PHP para cubrir las vulnerabilidades que no fueron consideradas.

Palabras clave: <VULNERABILIDADES>,<OPEN WEB APPLICATION SECURITY PROJECT (OWASP)>,<MÉTODO PROPUESTO>,<PROGRAMACIÓN>,< LENGUAJE DE PROGRAMACIÓN (PHP)>,< SEGURIDAD>.

SUMMARY

The research aimed to propose a method of development using secure programming techniques to reduce the risk of receiving computer attacks in web applications in Hypertext Pre-processor (PHP) programming language. The proposed method was developed with programming techniques for PHP 5.6.x that included seven stages: Parameterization of queries, data encoding, validation of all entries, implementation of identity and authentication controls, data protection, error and control of exceptions and adequate configuration, which were adapted from the different guides and recommendations of the Open Web Application Security Project (OWASP). The hypothesis "The proposed development method will improve the level of security of web applications around PHP in the face of possible computer attacks." Chi-Square was applied with a significance level of 0.05. The indicators were: Improvement in the level of security and total number of detected vulnerabilities, from a sample of eight users for the dependent variable and seven vulnerabilities for the independent variable, using surveys and vulnerability analyzer Acunetix Web Vulnerability Scanner respectively. According to the results obtained, prototype 2 Monitoring System and Safe Evaluation (S-SISEV) improved by 97% and 87.95%, respectively, in relation to prototype 1. It is concluded that the proposed method avoids some of the most common errors in programming and it is recommended to develop new programming techniques in PHP to cover vulnerabilities that were not considered.

Key Words: <VULNERABILITIES>, <OPEN WEB APPLICATION SECURITY PROJECT (OWASP)>, <PROPOSED METHOD>, <PROGRAMMING>, <PROGRAMMING LANGUAGE (PHP)>, <SECURITY>.

CAPÍTULO I

1. INTRODUCCIÓN

1.1. Planteamiento del problema

1.1.1. *Situación Problemática*

En la actualidad las aplicaciones web se han vuelto indispensables para el manejo de la información en una organización, convirtiéndose en una herramienta que permite al usuario acceder y utilizar un sistema informático a través de internet mediante un navegador web, permitiendo el acceso a la información desde cualquier parte del mundo. (Salgado, 2014, pág. 1).

Se demuestra que si bien existe un importante nivel de madurez en materia de seguridad informática respecto de la infraestructura tecnológica organizacional no sucede lo mismo con los sistemas aplicativos que son soportados por dicha infraestructura, esto puede poner en riesgo uno de los activos más importantes que tiene una organización como es su información.

La mayoría de los problemas de seguridad en los sitios web se encuentran a nivel de aplicación y que son el resultado de escritura defectuosa de código, debemos entender que programar aplicaciones web seguras no es una tarea fácil, ya que requiere por parte del programador no únicamente mostrar atención en cumplir con el objetivo funcional básico de la aplicación sino una concepción general de los riesgos que puede correr la información contenida, solicitada y recibida por el sistema. En la actualidad aunque existen muchas publicaciones que permiten formar un criterio sobre el tema, no existen acuerdos básicos sobre lo que se debe o no se debe hacer, y lo que en algunas publicaciones se recomienda, en otras es atacado. (UNAM-CERT, 2016).

Sin embargo, en lo sustancial sí existen algunas recomendaciones que son generales y serán tomadas como base para la elaboración del método propuesto como la Guía de Pruebas OWASP (OWASP, Owasp Testing Guide 4.0, 2015).

Se han realizado varias investigaciones previas acerca de guías y/o metodologías propuestas para mejorar las seguridades en las aplicaciones web, entre las que destacamos:

- "Metodologías para el Desarrollo de Software Seguro". (Lopez, 2015).
- "Guía de buenas prácticas de desarrollo de aplicaciones web seguras aplicado al sistema control de nuevos aspirantes Empresa Grupo LAAR". (Yáñez, 2014).
- "Desarrollo de una Propuesta Metodológica para determinar la seguridad en una aplicación web". (Ascencio, M. y Moreno, P., 2011).

Estas investigaciones si bien son importantes para determinar el nivel de seguridad de una aplicación, no definen procedimientos o técnicas puntuales de programación que sirvan como base para el desarrollo de aplicaciones web, ni se enfocan a un lenguaje determinado.

Por lo que un método basado en técnicas de programación segura para el desarrollo de aplicaciones web en entorno PHP surge como propuesta para mitigar riesgos potenciales de seguridad.

1.1.2. Formulación del problema

¿Cómo contribuirá el método propuesto a mejorar el desarrollo de aplicaciones web seguras en entorno PHP?

1.1.3. Preguntas directrices o específicas de la investigación

- ¿Cuáles son los métodos existentes para el desarrollo de aplicaciones web seguras en entorno PHP?
- ¿Cuáles son las vulnerabilidades más comunes que está expuesta una aplicación web?
- ¿Cuáles son los principales riesgos que existen cuando hay un ataque a una aplicación web?
- ¿Cuáles son las herramientas y/o técnicas de programación más adecuadas a utilizar para mejorar el desarrollo de aplicaciones web seguras en PHP?

1.2. Justificación de la investigación

1.2.1. Justificación Teórica

Desde inicios de internet se han intentado valorar las formas que aprovechan los atacantes para penetrar en los sistemas, se pueden catalogar según desde donde se genera la vulnerabilidad y estas pueden ser:

- Falencias en la operación por parte del usuario.
- Falencias en la administración del sitio o el servidor.
- Falencias en la programación de la aplicación.
- Falencias en el servidor web.

De estas las "Falencias en la programación de la aplicación", es uno de los puntos más críticos que lamentablemente muchas personas a cargo del desarrollo de aplicaciones no han recibido una adecuada capacitación sobre cómo realizar programación segura. Este es uno de los puntos cruciales a la hora de aprovechar fallas de seguridad por parte de los atacantes. De hecho podríamos afirmar que de realizarse una programación segura la casi totalidad de los problemas de seguridad que aquejan a muchas aplicaciones dinámicas se podrían solucionar, con claras excepciones como es el hecho de que un usuario sucumba ante un ataque de Ingeniería Social por ejemplo (Perez, 2014, pág. 26).

Entonces una adecuada programación de un sitio web contribuirá a eliminar o al menos mitigar fallas de seguridad, pues el programador es el responsable de validar las entradas/salidas que reciben/envían sus programas hechos para la web.

Se establecerá un método de desarrollo que permita estandarizar la programación en PHP y evitar o minimizar los fallos de seguridad, basadas en técnicas ya existentes y probadas conjuntamente con nuevas técnicas que se propondrán en base a un análisis de las vulnerabilidades más frecuentes detectadas en aplicaciones web.

Es necesario entonces plantear un método basado en técnicas de programación segura que cubra el desarrollo de una aplicación web para mitigar los riesgos potenciales de seguridad ante ataques.

1.2.2. Justificación Práctica

Para validar el método propuesto se utilizarán dos (2) escenarios:

- El primero una aplicación web denominada SISEV (Sistema de Seguimiento y Evaluación) desarrollada en lenguaje de programación PHP 5.5 con una base de datos MySQL 5 para el Ministerio de Agricultura Ganadería Acuacultura y Pesca (MAGAP)- Zona 3, y que está alojado en un servidor web en Planta Central (Quito) a través de la URL: servicios.agricultura.gob.ec/smap/; esta aplicación fue desarrollada con código propio, y sin establecer un método adecuado de programación segura.
- El segundo escenario es una variante de la aplicación SISEV que se la denominará S-SISEV, la cual se implementará con el método propuesto de programación segura y estará alojado en un servidor de pruebas ubicado en Planta Central; utilizará un marco propio con librerías y funciones probadas para PHP 5.6.X y PHP 7 que son las últimas versiones con soporte.

Se pretende entonces verificar el nivel de seguridad que existe en estas dos aplicaciones mediante herramientas de detección de vulnerabilidades como: Nikto, ZAP, Acunetix, w3af, entre otros; y evaluar los resultados obtenidos.

Se debería observar que el escenario dos (método propuesto) presenta menos riesgos de vulnerabilidad, al menos de los ataques más conocidos, ya que se están aplicando técnicas de programación segura para entrada y salida de datos desde y hacia la base de datos; este escenario debería al menos mitigar siete (7) de las diez (10) vulnerabilidades más críticas de OWASP.

1.2.3. Justificación Metodológica

Se creará un método propio de desarrollo teniendo como base algunas metodologías existentes como: OSSTMM, Guía OWASP, ISO 27001, entre otras; que son utilizadas para análisis de vulnerabilidades y seguridades en aplicaciones web y software en general, así como técnicas de programación segura que se descubrirán o implementarán a lo largo del proyecto.

A continuación se describen brevemente dos (2) de las metodologías más importantes:

- **"OWASP Testing Project"** el cual describe toda una metodología de pruebas para verificar la seguridad de las aplicaciones web. En general se suele utilizar el término metodología OWASP para referirse al testeo de aplicaciones web, pero OWASP abarca muchos más proyectos; aquí utilizaremos como base la metodología definida en el proyecto de pruebas de OWASP.
- El manual **OSSTMM** (Manual de la Metodología Abierta de Testeo de Seguridad) describe el modo de realizar pruebas de seguridad, así como las métricas a emplear por profesionales al realizar las Auditorías de Seguridad.

1.3. Objetivos de la investigación

1.3.1. Objetivo General

Proponer un método de desarrollo utilizando técnicas de programación segura para reducir el riesgo de recibir ataques informáticos en las aplicaciones web en entorno PHP.

1.3.2. Objetivos Específicos

- Determinar las vulnerabilidades más conocidas en el desarrollo de aplicaciones web.
- Analizar las metodologías, guías o buenas prácticas existentes para desarrollar una aplicación web segura.
- Proponer un método que contemple las vulnerabilidades más críticas en el desarrollo de aplicaciones web en entorno PHP.
- Verificar la mejora de la seguridad aplicando el método desarrollado.

1.4. Hipótesis

El método de desarrollo propuesto mejorará el nivel de seguridad de las aplicaciones web en entorno PHP ante posibles ataques informáticos.

CAPÍTULO II

2. MARCO TEÓRICO

2.1. Antecedentes del problema

2.1.1. Análisis de metodologías propuestas para mejorar las seguridades en aplicaciones web

- "Metodologías para el Desarrollo de Software Seguro" (Lopez, 2015), propone una metodología a partir de diferentes metodologías de auditoría existentes (OSSTMM, OWASP, ISO 27001), manuales de buenas prácticas, metodologías de desarrollo y aportaciones de responsables del proyecto; **pero es muy general y no se centra sólo al desarrollo de aplicaciones web.**
- "Guía de buenas prácticas de desarrollo de aplicaciones web seguras aplicado al sistema control de nuevos aspirantes Empresa Grupo LAAR" (Yáñez, 2014), define una guía segura de aplicaciones web basadas en metodologías de evaluación de riesgos como CORAS o PTA; **pero no contempla otros tipos de amenazas como: redirecciones y reenvíos no validados y falsificación de peticiones en dominios cruzados, entre otros.**
- "Desarrollo de una Propuesta Metodológica para determinar la seguridad en una aplicación web" (Ascencio, M. y Moreno, P., 2011), presenta una propuesta metodológica para determinar la seguridad en una aplicación web bajo dos enfoques, el de caja negra y caja blanca, basado en un análisis de la seguridad en aplicaciones web, vulnerabilidades en la web, herramientas de pruebas de Intrusión, estándar ISO 27001 y Metodologías OWASP e ISSAF; **pero no incluyen políticas de programación segura.**

2.1.2. Pros y contras de las metodologías a estudiar

Se realizó una comparación entre las diferentes metodologías que se pretende estudiar por lo que se pone a consideración su análisis (Lopez, 2015):

Tabla 1-2: Comparación de metodologías

Metodología	Pros	Contras
ISO 27001	Ofrece una buena guía para la gestión de toda la documentación referente a la seguridad.	La complejidad de su lectura se asemeja a la de textos legales. No cubre la seguridad de un proyecto, solo a nivel de organización.
OSSTMM	Presenta diferentes módulos que se encargan de cubrir diferentes áreas de seguridad en una organización. Ofrece una métrica sobre el nivel de seguridad de la organización, así como la forma de utilizarla.	Algunos de los controles de seguridad propuestos pueden resultar excesivos para la mayoría de organizaciones. No se cubre la seguridad de un proyecto, solo a nivel de organización.
OWASP Testing Guide	Cubre la seguridad de un proyecto. Es la metodología que más se acerca a nuestras necesidades.	Algunas fases del ciclo de vida no se profundizan.

Realizado por: Monar Joffre, 2016

Fuente: (Lopez, 2015)

De este análisis se puede mencionar que se utilizará como base la metodología OWASP, específicamente asociada a las siguientes Guías:

- Controles Proactivos TOP 10 versión 2-2016.
- El Estándar de Verificación para la Seguridad de las Aplicaciones (ASVS) versión 3-2015.
- La Guía de Desarrollo OWASP versión 4.
- Hojas de Trucos para el manejo de sesiones y autenticación.
- La Guía de revisión de código en su versión 2.
- La Guía de Pruebas OWASP versión 3.

Crear una guía representa un esfuerzo enorme, que implica décadas de trabajo realizado por cientos de personas en todo el mundo. Hay muchas formas diferentes de probar fallos de seguridad, y esta guía aúna el consenso de los principales expertos sobre cómo realizar esta comprobación rápida, exacta y eficientemente. (OWASP, GUÍA DE PRUEBAS OWASP VERSIÓN 3, 2008).

Es por ello que se decide aplicar esta metodología, ya que existe un trabajo previo efectuado en relación a los controles a efectuarse y técnicas que deben tomarse en cuenta para mantener la seguridad en las aplicaciones web; el mismo entonces servirá como punto de partida para el desarrollo del método propuesto.

2.2. Bases teóricas

2.2.1. Aplicaciones Web

1. Definición

Una aplicación web se define como un tipo especial de aplicación cliente/servidor, donde tanto el cliente (el navegador web) como el servidor (el servidor web) y el protocolo de red (HTTP) interactúan de manera sincronizada para proporcionar las peticiones coherentes al servidor web (Mateu, 2010).

2. Estructura de las Aplicaciones Web

Las aplicaciones web son aplicaciones que generalmente se desarrollan distribuidas en 3 capas o niveles (Figura 1-2), se utiliza el navegador web como la primera capa, la lógica de negocio sería la capa intermedia, la cual puede desarrollarse en diferentes lenguajes de programación como PHP, Java, Python, Perl, ASP.NET, etc.; y la base de datos sería la última capa.

El navegador web, al cual toma el nombre de cliente envía peticiones a la capa media, la cual las procesa y accede a la base de datos para realizar las transacciones solicitadas. Las peticiones realizadas entre el cliente y el servidor se realizan a través del protocolo HTTP y se emplea el lenguaje HTML para mostrar el contenido de respuesta a la petición inicial entre cliente y servidor.

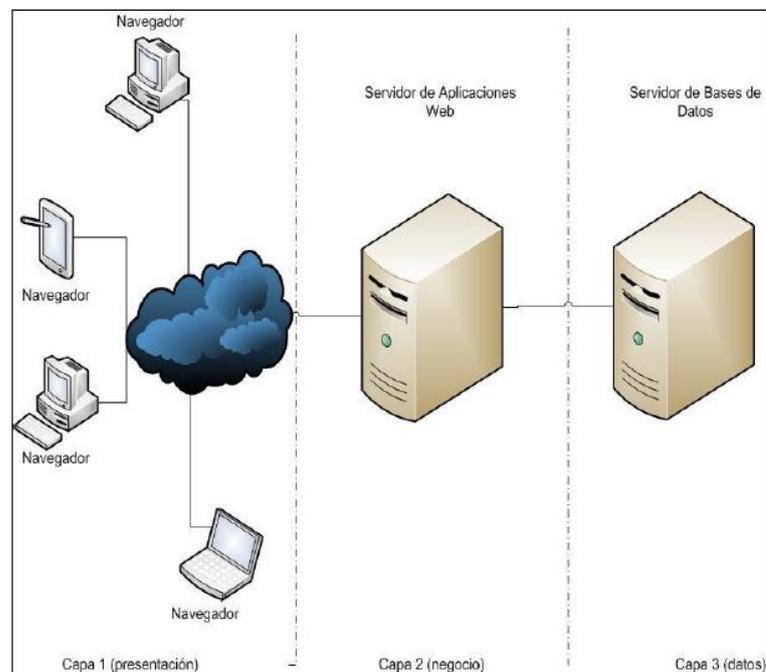


Figura1-2: Arquitectura Web en 3 Capas

Fuente: (Lopez, 2015)

3. Lenguajes de Programación

Actualmente existen diferentes lenguajes de programación para hacer desarrollos en la web (Figura 2-2), estos han ido surgiendo de acuerdo a las tendencias y necesidades de las plataformas.

En el inicio del Internet las aplicaciones web creadas fueron realizadas mediante lenguajes estáticos, posteriormente con el desarrollo y el avance de nuevas tecnologías surgen nuevas necesidades que dio lugar al desarrollo de Lenguajes Dinámicos de Programación que utilizan bases de datos y permiten interactuar con los usuarios.



Figura2-2: Lenguajes de Programación

Fuente: (Lopez, 2015)

De los diferentes lenguajes de programación existentes se hará hincapié en PHP que será el lenguaje utilizado para el método de desarrollo propuesto.

2.2.2. *Seguridades en Aplicaciones Web*

1. Introducción

Cualquier empresa u organización hoy en día tiene que realizar un esfuerzo significativo y una elevada inversión para asegurar que la información y recursos se encuentren protegidos siempre que se encuentren expuestos sus servicios a la red de redes. Internet es un factor primordial en las comunicaciones y también un evidente riesgo potencial de acceso y mal uso de los servicios e información disponibles. (UNAM-CERT, 2016).

Se clasifican como sistemas más críticos aquellos donde la seguridad debe de ser muy significativa, pero en general todas las aplicaciones web deben de estar protegidos y asegurados ante los principales ataques o al menos los más comunes.

En una aplicación web, la seguridad se divide en:

- Disponibilidad
- Autenticidad
- Integridad
- Confidencialidad
- Trazabilidad

Las aplicaciones web son consideradas como el punto más común para los ataques informáticos debido a su fácil acceso a través de internet, muchas de ellas contienen información sensible de instituciones que mueven todo su negocio mediante una aplicación web. Una institución u organización mientras más va automatizando sus procesos mediante aplicaciones web, se vuelve más importante la necesidad de implementar seguridad en sus procedimientos e información. (UNAM-CERT, 2016).

La mayoría de los problemas de seguridad en los sitios web se encuentran a nivel aplicación y son el resultado de la escritura de código con problemas. (Areito, 2008).

Programar aplicaciones web seguras no es una tarea fácil, pues se requiere por parte del programador, una concepción general de los riesgos que puede correr la información contenida, solicitada y recibida por el sistema. En la actualidad, aunque existen muchas publicaciones que permiten formar un criterio sobre el tema, no existen acuerdos básicos sobre lo que se debe o no se debe hacer. (Yáñez, 2014).

2. Necesidad de asegurarlas aplicaciones web

La seguridad del sitio web es hoy el aspecto más asegurado de la empresa y debe ser una prioridad en cualquier Organización. Cada vez más, los atacantes están concentrando sus esfuerzos en aplicaciones basadas en la Web: carritos de compra, formularios, páginas de inicio de sesión, de contenido dinámico, etc. (Al-Ibrahim, M. & Al-Deen, Y., 2014).

3. Tipos de ataques o amenazas

“Se entiende por amenaza una condición del entorno del sistema de información que, dada una oportunidad, podría dar lugar a que se produjese una violación de la seguridad” (Enríquez, Políticas de seguridad informática y la vulnerabilidad de los entornos web de la Empresa Turbotech durante el año 2010, 2011).

Según (Enríquez, Políticas de seguridad informática y la vulnerabilidad de los entornos web de la Empresa Turbotech durante el año 2010, 2011) las cuatro (4) categorías generales de amenazas o ataques son:

1. **Interrupción:** Un recurso del sistema es destruido o se vuelve no disponible. Este es un ataque contra la disponibilidad.
2. **Intercepción:** Una entidad no autorizada consigue acceso a un recurso. Este es un ataque contra la confidencialidad; la entidad no autorizada podría ser una persona, un programa o un ordenador.
3. **Modificación:** Una entidad no autorizada no sólo consigue acceder a un recurso, sino que es capaz de manipularlo. Este es un ataque contra la integridad.
4. **Fabricación:** Una entidad no autorizada inserta objetos falsificados en el sistema. Este es un ataque contra la autenticidad.

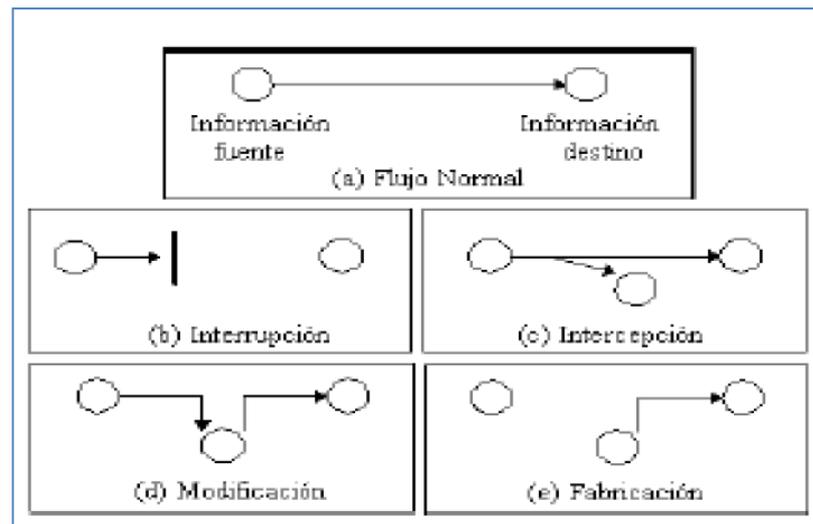


Figura3-2: Tipos de ataques o amenazas

Fuente: (Enríquez, Políticas de seguridad informática y la vulnerabilidad de los entornos web de la Empresa Turbotech durante el año 2010, 2011)

La identificación de las vulnerabilidades permite conocer los tipos de ataque que podrían ser efectuados, así como también sus consecuencias.

2.2.3. *Seguridades en PHP*

1. **Introducción**

"Uno de los lenguajes más populares para la creación de páginas web dinámicas es PHP (del inglés, PHP Hypertext Pre-processor)" (Waisen, J. & Pérez, F., 2012, pág. 9).

A pesar de que muchos programadores y desarrolladores puede que estén implementando PHP en sus sitios, el tema referente a la seguridad es a menudo dejado de lado cuando se construye nuestra web.

El código PHP es un lenguaje que funciona aún con cabos sueltos. Los atacantes buscan esos cabos, y en PHP, no son muy difíciles de encontrar. La seguridad PHP involucra la minimización de los errores de programación, y la colocación del código apropiado en su lugar para eliminar toda posible vulnerabilidad. (elwebmaster.com, 2009).

No existe ningún sistema libre de fallos y seguro en su totalidad por lo que la tarea de un programador de aplicaciones web PHP debe ser la de minimizar los riesgos intentando garantizar en la mayor medida que sea posible la integridad del propio sitio web y de la información que esta almacena. (Waisen, J. & Pérez, F., 2012, pág. 75).

2. **Principios de seguridad**

Independientemente del lenguaje y la arquitectura utilizados, el desarrollo de la aplicación y de los controles de seguridad necesarios deben llevarse a cabo teniendo en cuenta una serie de principios de seguridad que traten de reducir la probabilidad de la realización de amenazas y el impacto de éstas en el caso de que se hayan producido ya.

Aunque estos principios pueden servir como pautas generales, para que sean realmente útiles, deben ser evaluados, interpretados y aplicados para abordar cada problema específico. La consideración de cada uno de ellos puede derivar en la identificación de requisitos de seguridad, la toma de decisiones relativas a la arquitectura y la implementación e identificación de posibles debilidades en el sistema. (OWASP, Los diez riesgos más críticos en Aplicaciones Web, 2013).

A continuación se detalla cada uno de estos principios:

- **Minimizar la superficie de ataque:** Cada funcionalidad que se incorpora a una aplicación añade cierto riesgo al conjunto. Uno de los principios de la programación segura consiste en disminuir el riesgo reduciendo la superficie de ataque.
- **Seguridad por defecto:** Cuando una aplicación se despliega en su entorno de producción, utiliza una serie de opciones de configuración que se establecen por defecto. Estas opciones por defecto deben ser tales que la aplicación sea segura. Es responsabilidad del usuario modificar dichas opciones aún a costa de disminuir la seguridad.
- **Ejecución con los mínimos privilegios:** El principio de mínimos privilegios establece que las cuentas deben tener el menor nivel de privilegios posible para realizar las tareas de negocio. Este nivel de privilegios abarca tanto permisos de usuarios como permisos sobre recursos como CPU, memoria, red, sistema de ficheros, etc.
- **Defensa en profundidad:** El principio de defensa en profundidad sugiere que donde un único control de seguridad podría ser asumible, sería mejor utilizar varios controles que afronten el riesgo desde distintos puntos de vista. Los controles de seguridad, cuando se utilizan con el enfoque de defensa en profundidad, hacen que vulnerabilidades que pueden ser muy graves, sean tremendamente difíciles de explotar.
- **Fallar de forma segura:** En muchas ocasiones se producen errores cuando las aplicaciones realizan una transacción. El estado en que la aplicación queda cuando se produce dicho error, determina si la aplicación es segura o no
- **Detección de intrusos:** La detección de intrusiones requiere la existencia de tres (3) elementos:
 1. Capacidad para incluir en el log eventos relevantes de seguridad.
 2. Procedimientos que aseguren que los logs son monitorizados con regularidad.
 3. Procedimientos para responder adecuadamente a una intrusión una vez ha sido detectada.

- **Evitar la seguridad por ocultación:** La seguridad por ocultación es un mecanismo de seguridad débil y generalmente falla cuando es el único control existente. La seguridad de un sistema nunca debe recaer en la ocultación de secretos.
- **Mantener la simplicidad:** Cuando el código fuente es muy complejo es más complicado hacerlo seguro, es mejor mantenerlo simple.
Hay que tener en cuenta que cuando se desarrollan nuevas versiones o se solucionan defectos, los programadores que intentan hacer segura la aplicación pueden no ser los mismos que desarrollaron el código inicialmente. Cuantas más líneas de código hay, más probabilidades de introducir vulnerabilidades.
- **Solucionar correctamente los problemas de seguridad:** Una vez que se ha detectado un problema de seguridad, es fundamental desarrollar pruebas para reproducirlo y detectar la causa raíz. Una vez que se desarrolla una solución válida es clave garantizar que no se introducen problemas de regresión.

2.2.4. Principales vulnerabilidades detectadas en aplicaciones web

El proceso de explotar vulnerabilidades y realizar ataques en las aplicaciones web ha crecido y sigue creciendo. El OWASP Top 10 2013 se basa en ocho (8) conjuntos de datos de siete (7) firmas especializadas en seguridad de aplicaciones, incluyendo cuatro (4) empresas consultoras y tres (3) proveedores de herramientas SaaS. Estos datos abarcan más de 500.000 vulnerabilidades a través de cientos de organizaciones y miles de aplicaciones. Las vulnerabilidades del Top 10 son seleccionadas y priorizadas de acuerdo a estos datos de prevalencia, en combinación con estimaciones consensuadas de explotabilidad, detectabilidad e impacto. (Yáñez, 2014).

Según (OWASP, Los diez riesgos más críticos en Aplicaciones Web, 2013), Las principales vulnerabilidades en aplicaciones web son:

1. Inyección

Las fallas de inyección, tales como SQL, OS y LDAP, ocurren cuando datos no confiables son enviados a un intérprete como parte de un comando o consulta. Los datos hostiles del atacante pueden engañar al intérprete en ejecutar comandos no intencionados o acceder datos no autorizados.

2. Pérdida de autenticación y gestión de sesiones

Las funciones de la aplicación relacionadas a autenticación y gestión de sesiones son frecuentemente implementadas incorrectamente, permitiendo a los atacantes comprometer contraseñas, claves, token de sesiones, o explotar otras fallas de implementación para asumir la identidad de otros usuarios.

3. Secuencia de comandos en sitios cruzados (XSS)

Las fallas XSS ocurren cada vez que una aplicación toma datos no confiables y los envía al navegador web sin una validación y codificación apropiada. XSS permite a los atacantes ejecutar secuencia de comandos en el navegador de la víctima los cuales pueden secuestrar las sesiones de usuario, destruir sitios web, o dirigir al usuario hacia un sitio malicioso.

4. Referencia directa insegura a objetos

Una referencia directa a objetos ocurre cuando un desarrollador expone una referencia a un objeto de implementación interno, tal como un fichero, directorio, o base de datos. Sin un chequeo de control de acceso u otra protección, los atacantes pueden manipular estas referencias para acceder datos no autorizados.

5. Configuración de seguridad incorrecta

Una buena seguridad requiere tener definida e implementada una configuración segura para la aplicación, marcos de trabajo, servidor de aplicación, servidor web, base de datos, y plataforma.

Todas estas configuraciones deben ser definidas, implementadas, y mantenidas ya que por lo general no son seguras por defecto. Esto incluye mantener todo el software actualizado, incluidas las librerías de código utilizadas por la aplicación.

6. Exposición de datos sensibles

Muchas aplicaciones web no protegen adecuadamente datos sensibles tales como números de tarjeta de crédito o credenciales de autenticación.

Los atacantes pueden robar o modificar tales datos para llevar a cabo fraudes, robos de identidad u otros delitos. Los datos sensibles requieren de métodos de protección adicionales tales como el cifrado de datos, así como también de precauciones especiales en un intercambio de datos con el navegador.

7. Ausencia de control de acceso a funciones

La mayoría de aplicaciones web verifican los derechos de acceso a nivel de función antes de hacer visible en la misma interfaz de usuario. A pesar de esto, las aplicaciones necesitan verificar el control de acceso en el servidor cuando se accede a cada función. Si las solicitudes de acceso no se verifican, los atacantes podrán realizar peticiones sin la autorización apropiada.

8. Falsificación de peticiones en sitios cruzados (CSRF)

Un ataque CSRF obliga al navegador de una víctima autenticada a enviar una petición http falsificado, incluyendo la sesión del usuario y cualquier otra información de autenticación incluida automáticamente, a una aplicación web vulnerable. Esto permite al atacante forzar al navegador de la víctima para generar pedidos que la aplicación vulnerable piensa son peticiones legítimas provenientes de la víctima.

9. Utilización de componentes con vulnerabilidades conocidas

Algunos componentes tales como las librerías, los frameworks y otros módulos de software casi siempre funcionan con todos los privilegios. Si se ataca un componente vulnerable otro podría facilitar la intrusión en el servidor o una pérdida seria de datos. Las aplicaciones que utilicen componentes con vulnerabilidades conocidas debilitan las defensas de la aplicación y permiten ampliar el rango de posibles ataques e impactos.

10. Redirecciones y reenvíos no validados

Las aplicaciones web frecuentemente redirigen y reenvían a los usuarios hacia otras páginas o sitios web, y utilizan datos no confiables para determinar la página de destino. Sin una validación apropiada, los atacantes pueden redirigir a las víctimas hacia sitios de phishing o malware, o utilizar reenvíos para acceder páginas no autorizadas.

2.2.5. Analizadores de vulnerabilidades

1. Herramientas para detección de vulnerabilidades web

Dentro del proceso de detección de vulnerabilidades en aplicaciones web se utilizarán algunas herramientas que existen en el mercado y sobre todo open source para la revisión de código PHP, entre ellas:

- **Zed Attack Proxy (ZAP):** Herramienta de fácil uso para encontrar vulnerabilidades en aplicaciones web. Está diseñada para ser utilizada tanto por desarrolladores y probadores funcionales (que son nuevos en test de intrusión) como por personas con una amplia gama de experiencia en seguridad. Permite automatizar las pruebas y también facilita un número de herramientas para hacerlas manualmente. (Yáñez, 2014, pág. 35).
- **Acunetix Web Vulnerability Scanner:** Puede utilizarse para realizar escaneos de vulnerabilidades en aplicaciones web y para ejecutar pruebas de acceso frente a los problemas identificados. La herramienta provee sugerencias para mitigar las vulnerabilidades identificadas y puede utilizarse para incrementar la seguridad de servidores web o de las aplicaciones que se analizan; comprueba los sistemas en busca de múltiples vulnerabilidades incluyendo: SQL Injection, Cross Site Scripting y passwords débiles.
- **W3af:** Este software tiene como objetivo proporcionar una plataforma de pruebas de penetración de aplicaciones web mejor. Está desarrollado en Python. Mediante el uso de esta herramienta, usted será capaz de identificar más de 200 tipos de vulnerabilidades de las aplicaciones web, incluyendo la inyección SQL, Cross Site Scripting y muchos otros.

- **Nikto:** Es una herramienta utilizada por quienes realizan actividades de ethical hacking y pentest o test de penetración; puede ejecutar cualquier plataforma que permita Perl y puede evadir sistemas de detección de intrusos (IDS) (Culoccioni, 2015).

De ser necesario se incluirán otras herramientas para el análisis de vulnerabilidades.

2. Clasificación de los niveles de riesgo

(Astudillo, 2013) menciona que:

Los analizadores facilitan la labor del auditor porque permiten ejecutar desde una sola interfaz escaneos y enumeraciones sobre el objetivo, a la vez que identifican las vulnerabilidades presentes en dichos sistemas y las clasifican de acuerdo al nivel de riesgo presente.

La identificación se realiza de acuerdo a la versión del sistema operativo y de los servicios y aplicaciones detectados comparándolos contra una base de datos de vulnerabilidades que se actualiza frecuentemente conforme nuevos huecos de seguridad son descubiertos.

Los niveles de riesgo se clasifican usualmente en: bajo, medio y alto, conforme a la siguiente escala.

- **Riesgo Alto:** El equipo tiene una o más vulnerabilidades críticas que podrían ser explotadas fácilmente por un atacante y que podrían conllevar a tomar control total del sistema o comprometer la seguridad de la información de la organización. Los equipos con este nivel de riesgo requieren acciones correctivas inmediatas.
- **Riesgo Medio:** El equipo tiene una o más vulnerabilidades severas que requieren una mayor complejidad para poder ser explotadas y que podrían no brindar el mismo nivel de acceso al sistema afectado. Los equipos con riesgos severos requieren atención a corto plazo.

- **Riesgo Bajo:** El equipo tiene una o más vulnerabilidades moderadas que podrían brindar información a un atacante, la cual podría utilizarse para realizar ataques posteriores. Estos riesgos deben ser mitigados adecuadamente, pero no tienen un nivel de urgencia alto.

2.2.6. Controles estándar de seguridad

Dentro de las “Buenas prácticas de calidad en el desarrollo de aplicaciones” (propuesto por el Gobierno de España) menciona que durante el desarrollo de las aplicaciones deben implementarse distintos controles de seguridad. En general, pueden encajarse en los siguientes grupos en función del objeto para el cual se construyan:

- Controles de prevención de accesos no autorizados.
- Controles sobre las entradas de datos en la aplicación, para evitar que ciertos datos provoquen que la aplicación no se comporte de la manera deseada.
- Controles para asegurar el funcionamiento correcto de la aplicación cuando está siendo directamente atacada.
- Controles de gestión de la propia aplicación para monitorizar la actividad de la aplicación y configurar su comportamiento.

1. Piensa como el atacante

Los desarrolladores involucrados en la implementación de estos controles de seguridad deben ponerse en la piel de un usuario atacante, valorando en cada una de las funcionalidades que esté implementando los siguientes puntos:

- ¿el proceso que envuelve a la funcionalidad es seguro?
- ¿de qué manera podría abusar de esa funcionalidad?
- ¿es necesario que la funcionalidad esté activa por defecto? Si es así, ¿qué opciones minimizan el riesgo de uso de la misma?

2. Toda entrada es maliciosa

Además, el desarrollador debe tener en cuenta en todo momento durante el desarrollo que la aplicación debe considerar que toda entrada es potencialmente maliciosa y reaccionar aplicando las medidas necesarias que aseguren que un atacante no pueda utilizar estas entradas como medio para comprometer la aplicación.

3. No existe el software libre de errores

El software siempre tendrá errores y, por extensión, vulnerabilidades de seguridad. Por lo tanto, un objetivo práctico en el ciclo de vida de desarrollo de software seguro debería ser reducir al máximo (no necesariamente eliminar), el número de vulnerabilidades introducidas y la severidad de aquellas que no hayan sido eliminadas.

4. Una sola vulnerabilidad si es suficiente

La explotación de una única vulnerabilidad en una aplicación web es suficiente para interrumpir de manera significativa el negocio, provocar pérdida de datos, dañar la confianza del cliente, etc. Por ello, cuanto antes sean identificadas y corregidas las vulnerabilidades, menor será la oportunidad para un atacante de explotarlas maliciosamente.

2.2.7. Recomendaciones generales para programar de manera segura

Las prácticas recomendadas para la codificación segura cuando se desarrollen aplicaciones web se basan en (OWASP, CODE REVIEW GUIDE 2.0 RELEASE, 2013) y se describen a continuación:

1. Validación de entradas

- Validar datos provenientes de fuentes no.
- Especificar sets de caracteres apropiados para todas las fuentes de entrada.
- Validar los datos brindados por el cliente antes de procesarlos.

- Validar datos redireccionados.
- Si se desea permitir el ingreso de algún carácter considerado peligroso como: < > ” ’ % () & + \ / \’ \”; se debe implementar controles adicionales como la codificación de la salida, API de seguridad, etc.;

2. Codificación de salidas

- Sanear todas las salidas de datos no confiables hacia consultas SQL, XML y LDAP.

3. Administración en la autenticación y contraseñas

- Requerir autenticación para todos los recursos y páginas que manejen información sensible.
- Las respuestas a los fallos en la autenticación no deben indicar cual parte de la autenticación fue incorrecta.
- Utilizar conexiones encriptadas o datos encriptados para el envío de contraseñas que no sean temporales.
- Cumplir requerimientos de complejidad, longitud y cambio de contraseña.
- Deshabilitar las cuentas de usuario luego de un número establecido de intentos fallidos de acceso al sistema.
- Forzar el cambio de contraseñas temporales luego de su utilización.
- Notificar a los usuarios cada vez que se produzca un reseteo o cambio de contraseña.
- Llevar un registro de las últimas contraseñas registradas, además deben tener al menos un día de antigüedad antes de poder ser cambiadas.
- El último acceso (fallido o exitoso) debe ser reportado al usuario en su siguiente acceso válido.

4. Administración de sesiones

- La función de logout debe terminar completamente con la sesión.
- Establecer un tiempo de vida de la sesión lo más corto posible.
- Generar un nuevo identificador de sesión luego de cada reautenticación.

- No permitir logueos concurrentes con el mismo usuario.
- No exponer identificadores de sesión en URLs, mensajes de error, ni logs.

5. Control de Acceso

- Requerir controles de autorización en cada solicitud o pedido.
- Restringir acceso a ficheros u otros recursos.
- Restringir el acceso a servicios, URLs protegidas, funciones protegidas, referencias directas a objetos, información de la aplicación y su configuración.

6. Prácticas Criptográficas

- Establecer y utilizar una política de cómo manejar las claves criptográficas.

7. Manejo de errores y Logs

- No difundir información sensible en respuestas de error.
- Implementar mensajes de error genéricos y utilizar páginas de error adaptadas.
- Restringir el acceso a los logs.
- No guardar información sensible en logs, incluyendo detalles innecesarios del sistema.
- Registrar en un log las fallas de validación, intentos de autenticación, fallas en los controles de acceso, entre otros.

8. Protección de datos

- Implementar el mínimo privilegio a los usuarios
- Encriptar toda la información sensible almacenada.
- Proteger el código fuente del servidor.
- No almacenar contraseñas, cadenas de conexión u otra información sensible en texto claro o de forma que no sea criptográficamente segura del lado del cliente.
- Remover los comentarios en el código de producción que puedan revelar información sensible.

- Remover cualquier aplicación que no sea necesaria y la documentación que pueda revelar información útil a los atacantes.
- Deshabilitar las funcionalidades de autocompletar en aquellos formularios que contienen información sensible, incluyendo la autenticación.

9. Seguridad en las comunicaciones

- Implementar encriptación para todas las transmisiones de información sensible.

10. Configuración de los sistemas

- Asegurar que los servidores, los frameworks y los componentes del sistema estén corriendo con la última versión.
- Restringir el servidor web, los procesos y las cuentas de servicios con el mínimo privilegio posible.
- Remover todas las funcionalidades y archivos que no sean necesarios.
- Aislar los ambientes de desarrollo de los ambientes de producción y permitir el acceso solamente a los grupos de desarrollo.

11. Seguridad de Base de Datos

- Utilizar validación en las entradas y codificación en las salidas
- La aplicación debe utilizar el mínimo nivel de privilegios cuando accede a la base de datos.
- Utilizar credenciales seguras para acceder a la base de datos.
- Las cadenas de conexión a la base de datos no deben estar incluidas en el código de la aplicación.
- Cerrar la conexión a la base de datos tan pronto como sea posible.
- Remover o cambiar todas las contraseñas administrativas por defecto.
- Deshabilitar todas las funcionalidades innecesarias de la base de datos por ejemplo: procedimientos almacenados innecesarios, servicios no utilizados, paquetes de utilidades; instale solo el conjunto mínimo de funcionalidades y opciones requeridas.

- Deshabilitar las cuentas por omisión que no son necesarias para la funcionalidad del sistema.
- La aplicación debería conectarse a la base de datos con credenciales diferentes para cada nivel de confianza.

12. Manejo de Archivos

- No utilizar directamente información provista por el usuario en ninguna operación dinámica.
- Exigir autenticación antes de permitir la transferencia de un archivo al servidor.
- Transferir al servidor únicamente los tipos de archivo pdf, doc, entre otros; requeridos para la funcionalidad del sistema.
- Validar los tipos de archivo transferidos verificando la estructura de los encabezados.
- No guardar los archivos transferidos en el mismo contexto que la aplicación web.
- No incluir en parámetros nombres de directorios o rutas de archivos.
- Nunca enviar la ruta absoluta de un archivo al cliente.
- Asegurar que los archivos y recursos de la aplicación sean sólo de lectura.

13. Manejo de Memoria

- Utilizar controles a la entrada y salida de información no confiable.
- Verificar que el largo de los buffer sean los requeridos y especificados.
- Verificar los límites de los buffers si se llama a las funciones dentro de un ciclo y asegurarse de no correr riesgo de escribir fuera del espacio reservado.
- Truncar el largo de todas las cadenas de entrada a un tamaño razonable antes de pasarlos a una función de copia o concatenación.
- Evitar el uso de primitivas con vulnerabilidades conocidas
- Liberar adecuadamente la memoria previa a la salida de una función y de todos los puntos de finalización de la aplicación.

14. Prácticas Generales de Codificación

- Para las tareas habituales, utilizar código probado y verificado en lugar de crear códigos específicos.
- Utilizar funciones de control de integridad (hash) para verificar la integridad del código interpretado, librerías, ejecutables y archivos de configuración previo a su utilización.
- Explícitamente inicializar todas las variables y mecanismos de almacenamiento de información durante su declaración o antes de usarlos por primera vez.
- Las aplicaciones que requieran privilegios especiales, deberán elevar los privilegios sólo cuando sea necesario y devolverlos lo antes posible.
- Evitar errores de cálculo comprendiendo la forma en que el lenguaje de programación maneja las operaciones matemáticas y las representaciones numéricas.
- No utilizar datos provistos por el usuario para ninguna función dinámica.
- Evitar que los usuarios introduzcan o modifiquen código de la aplicación.
- Revisar todas las aplicaciones secundarias, código provisto por terceros y librerías

2.2.8. *Recomendaciones para evitar las 10 vulnerabilidades más críticas de OWASP*

Durante el desarrollo de una aplicación deben implementarse controles que garanticen la seguridad de la misma. Los errores de programación que se cometen durante la implementación pueden desembocar en vulnerabilidades de seguridad en las aplicaciones.

De acuerdo a (OWASP, Los diez riesgos más críticos en Aplicaciones Web, 2013), a continuación se explica cómo se propaga y cómo evitar cada tipo de vulnerabilidad en lenguaje PHP (objeto de este estudio):

1. Inyección SQL

Para que exista una vulnerabilidad de SQL Injection se requieren dos fallas por parte del programador:

1. Fallas en el filtrado de los datos.
2. Fallas en el escapado de los datos al enviarlos a la base de datos.

```
Sentencias del tipo:  
  
select * from usuarios where id=$id;  
  
Donde no se filtre correctamente la variable $id la entrada  
podría ser algo parecido a $id = '10 or 1=1;--'  
  
Dando como resultado:  
  
select * from usuarios where id=10 or 1=1;--;  
  
Podemos observar que el código inyectado nos devolvería toda  
la información al ser verdadera la condición inyectada.
```

Figura4-2: Descripción básica de SQL Injection

Fuente: (OWASP, Los diez riesgos más críticos en Aplicaciones Web, 2013).

Recomendación:

Ninguno de estos dos pasos cruciales debe ser omitido, y requieren especial atención para poder minimizar los errores. Afortunadamente los ataques de SQL Injection son fácilmente evitables, mientras filtremos y escapemos las salidas.

2. Pérdida de autenticación y gestión de sesiones

Cuando se presentan:

- Cuando se gestionan las contraseñas.
- Cuando expiran las sesiones o el proceso de cierre de sesión.
- Cuando hay recuperación de los valores del usuario de forma automática.
- Cuando se trae la famosa “Pregunta secreta.”
- Cuando se actualiza una cuenta.
- Cuando el usuario pierde su contraseña y solicita “Recordar contraseña”.

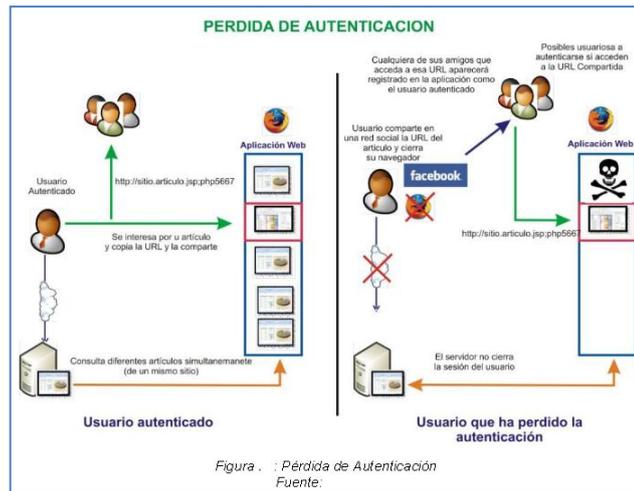


Figura5-2: Pérdida de autenticación

Fuente: (OWASP, Los diez riesgos más críticos en Aplicaciones Web, 2013).

Recomendación:

- Cierre de sesión correcto.
- Gestión de contraseñas adecuado.
- Tiempo de desconexión.
- Eliminar la opción recordar contraseña en los navegadores.
- Utilizar conexiones SSL.

3. Cross-Site Scripting (XSS)

Las vulnerabilidades de Cross-Site Scripting (XSS) se producen cuando:

- La aplicación web recibe datos que proceden de una fuente no confiable.
- Los datos son incluidos en el contenido dinámico que se devuelve al navegador del usuario sin haber comprobado que no contienen código malicioso.

El posible contenido malicioso que puede ser enviado al navegador del usuario generalmente toma la forma de un bloque de código Java Script, aunque también puede incluir HTML, Flash o cualquier otro tipo de código que pueda ser ejecutado por el navegador.

Recomendación:

Para evitar este ataque es conveniente filtrar todos los caracteres que tienen un significado especial en HTML como “, &, < y >.

4. Referencia directa insegura a objetos

Una referencia directa insegura a objetos ocurre cuando un desarrollador expone una referencia a un objeto de implementación interno, tal como un fichero, directorio, o base de datos. Sin un chequeo de control de acceso u otra protección, los atacantes pueden manipular estas referencias para acceder a datos no autorizados.

Ejemplo:

URL: <https://www.ejemplo/cuentas.php?cuenta=1234567890>

Con esto se cargarían los movimientos de dicha cuenta propiedad del usuario. Si se introduce otro número de cuenta cercano, por ejemplo 1234567891, en principio, si todo se hubiera hecho correctamente, debería mostrar un error al estar introduciendo una cuenta ajena, pero si somos vulnerables a la referencia directa a objetos, mostrará los datos de esta cuenta sin ser nuestra.

5. Configuración de seguridad incorrecta

- **Fuga de información del sistema**

Una fuga de información ocurre cuando los datos del sistema o la información de depuración salen del programa a través de un flujo de salida o función de registro.

Un mensaje de error en base de datos puede revelar que la aplicación es vulnerable a un ataque de inyección SQL; otros mensajes de error pueden revelar otras pistas indirectas sobre el sistema.

Ejemplos:

- **Intento crear una cuenta** => “Ese correo electrónico ya está asociado con una cuenta”.
- **Tratar de autenticarse con una cuenta** => “Esa cuenta no existe” / Error genérico.
- **Olvidé la contraseña** => “Se ha enviado un enlace a...” / “Esa cuenta no existe”.

Recomendación:

Escriba los mensajes de error pensando en la seguridad. En entornos de producción, deshabilite información detallada del error y en su lugar coloque mensajes breves.

- **Contraseñas escritas directamente en el código**

Pueden comprometer la seguridad de un sistema de una manera que puede no ser fácilmente remediable. Nunca es una buena idea escribir directamente una contraseña en el código fuente.

Recomendación:

Las contraseñas nunca deben ser escritas directamente en el código. Es preferible que sean mantenidas en un recurso externo.

6. Exposición de datos sensibles

Para asegurar que no se almacenan datos como texto en claro en la base de datos se pueden realizar procedimientos de hash a las cadenas almacenadas para que no sea entendible la información a simple vista. (OWASP, Los diez riesgos más críticos en Aplicaciones Web, 2013).

Pero no es suficiente sólo encriptar la información sino hacerlo con un algoritmo fuerte.

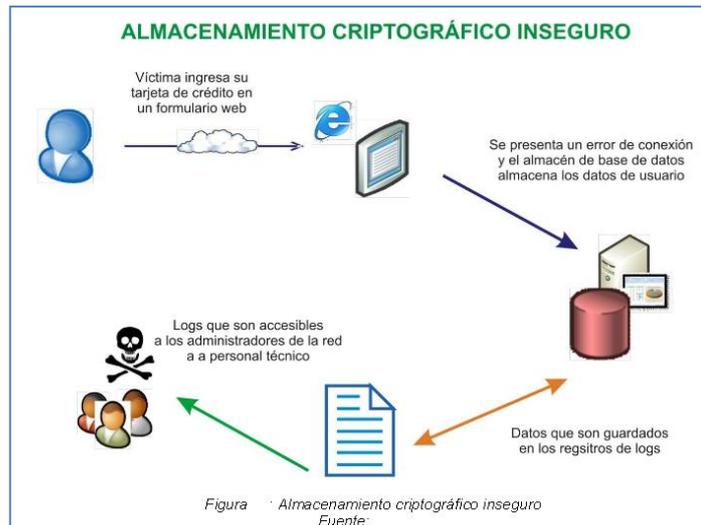


Figura6-2: Almacenamiento criptográfico inseguro

Fuente: (OWASP, Los diez riesgos más críticos en Aplicaciones Web, 2013).

Recomendación:

Se recomienda utilizar algoritmos de encriptación fuerte con claves de gran tamaño para proteger datos sensibles. Ejemplos de alternativas fuertes a DES son Rijndael (Advanced Encryption Standard o AES) y Triple DES (3DES).

7. Ausencia de control de acceso a funciones

Estos errores se dan cuando:

- El programa recibe datos que proceden de una fuente no confiable.
- Los datos son utilizados para especificar el valor de una clave primaria en una sentencia SQL.

Recomendación:

En vez de confiar en la interfaz de usuario para restringir los valores que pueden ser enviados por el usuario, la aplicación y la capa de base de datos deben implementar un control de acceso.

Bajo ninguna circunstancia se debe permitir que un usuario obtenga o modifique un registro de la base de datos sin tener los permisos adecuados.

8. Cross-Site Request Forgery (CSRF)

Una vulnerabilidad de tipo Cross-Site Request Forgery (CSRF) se dan cuando:

- Una aplicación web utiliza cookies de sesión.
- La aplicación procesa una petición HTTP sin verificar que ésta fue hecha con el consentimiento del usuario.

Recomendación:

Las aplicaciones que pasan el identificador de sesión en la URL en vez de usar una cookie, no son vulnerables a CSRF ya que no hay forma de que el atacante acceda a la información de sesión y la incluya en una petición.

9. Utilización de componentes con vulnerabilidades conocidas

Es una vulnerabilidad a través de la cual el atacante identifica o conoce mediante escaneos o análisis manuales, los componentes débiles de una aplicación web como los frameworks y ejecuta el ataque.

Recomendación:

- Identificar las versiones que están usando los componentes, incluyendo dependencias.
- Revisar la seguridad del componente en bases de datos públicas.
- Mantener actualizados los componentes.

10. Redirecciones y reenvíos no validados

El atacante crea enlaces para que la víctima haga clic que luego son llevados a una aplicación de confianza y allí los atacantes instalan el código malicioso.

Recomendación:

Este tipo de ataques son fáciles de evitar, basta con realizar las validaciones oportunas a la hora de hacer redirecciones. Toda redirección realizada a partir de un parámetro tendría que ser validada previamente comprobando que dicha redirección se va a realizar a un destino válido y confiable.

CAPÍTULO III

3. METODOLOGÍA DE INVESTIGACIÓN

3.1. Introducción

En esta sección se determinó los procedimientos y/o técnicas a utilizar para definir el método más adecuado para mitigar riesgos potenciales de seguridad al desarrollar aplicaciones web en entornos PHP así como su escenario de estudio, además se analizó los instrumentos que se utilizaron.

Debido al tipo de trabajo de investigación, la observación fue la herramienta más importante a la hora de detectar las vulnerabilidades más críticas que se presentaron sobre el escenario planteado, así también se clasificó la investigación.

3.2. Tipo y diseño de la Investigación

3.2.1. Tipo de Investigación

La investigación es considerada del tipo **descriptiva y aplicada**, ya que se utilizará el conocimiento adquirido producto de la investigación, para realizar un estudio comparativo de las vulnerabilidades detectadas en aplicaciones web con el antes y después de la utilización del nuevo método desarrollado basado en técnicas de programación segura en PHP con la finalidad de mitigar riesgos potenciales de seguridad.

3.2.2. Diseño de la Investigación

El diseño de la investigación es del tipo **cuasi experimental** ya que se seleccionarán muestras no probabilísticas para analizar los resultados y probar la hipótesis que estará relacionada al nivel de seguridad del aplicativo, donde para obtener los valores

cuantificables se implementarán analizadores de vulnerabilidades sobre los dos (2) prototipos en los dos (2) momentos (antes y después de aplicar el método).

3.3. Métodos de Investigación

Para esta investigación se utilizará los siguientes métodos:

- El **método científico** ya que se refiere a la serie de etapas que hay que recorrer para obtener un conocimiento válido desde el punto de vista científico, utilizando para esto instrumentos que resulten fiables; consta de las siguientes etapas:
 - Planteamiento del problema.
 - Formulación de la hipótesis.
 - Levantamiento de la información.
 - Análisis e interpretación de resultados.
 - Comprobación de la hipótesis.
 - Difusión de resultados.

- El **método deductivo** debido a que al estudiar de forma general las diferentes técnicas de programación para mitigar las vulnerabilidades más conocidas en aplicaciones web, se tratará de encontrar las técnicas más adecuadas y actuales al lenguaje PHP 5.6.X para el desarrollo del método.

3.4. Fuentes de información

Las principales fuentes utilizadas en la investigación fueron:

3.4.1 Primarias

- Pruebas.
- Observación de resultados.

3.4.2. *Secundarias*

- Tesis desarrolladas relacionadas con el tema de investigación.
- Trabajos de investigación nacional e internacional.
- Artículos científicos en base de datos de bibliotecas virtuales.
- Libros especializados en la biblioteca y electrónicos.
- Diccionarios especializados.
- Conferencias académicas, congresos, seminarios.
- Revistas indexadas y no indexadas publicadas de prestigio.
- Revistas electrónicas.
- Páginas de internet que brinden información confiable.

3.5. **Técnicas de recolección de datos primarios y secundarios**

Las técnicas que se utilizaron para la investigación fueron:

- **Búsqueda de información:** permite obtener la información necesaria acerca del objeto de estudio de la investigación para su desarrollo, utilizando las fuentes secundarias disponibles.
- **Pruebas:** realiza experimentos en escenarios de laboratorio.
- **Observación:** permite determinar los resultados de las pruebas realizadas en los escenarios de laboratorio.
- **Análisis:** determina los resultados de la investigación.

3.6. **Planteamiento de la Hipótesis**

3.6.1. *Hipótesis General*

El método de desarrollo propuesto mejorará el nivel de seguridad de las aplicaciones web en entorno PHP ante posibles ataques informáticos.

3.6.2. Identificación de variables

1. **Variable Independiente:** Método propuesto de programación segura para desarrollo de aplicaciones web en PHP.
2. **Variable Dependiente:** Mejora en el nivel de seguridad.

3.6.3. Operacionalización de variables

1. Operacionalización Conceptual de variables

Tabla 1-3: Operacionalización conceptual de variables

Variable	Tipo	Concepto
Método propuesto de programación segura para desarrollo de aplicaciones web en PHP	Independiente	Conjunto de técnicas de programación segura en PHP basado en normas existentes.
Mejora en el nivel de seguridad	Dependiente	Nivel de protección de la aplicación web contra las vulnerabilidades más conocidas

Fuente: Monar Joffre, 2016

Realizado por: Monar Joffre, 2016

2. Operacionalización Metodológica de variables

Tabla 2-3: Operacionalización Metodológica de variables

Hipótesis	Variables	Indicadores	Técnicas	Instrumentos
El método de desarrollo propuesto mejorará el nivel de seguridad de las aplicaciones web en entorno PHP ante posibles ataques informáticos	V. INDEPENDIENTE Método propuesto de programación segura para desarrollo de aplicaciones web en PHP	Nivel de satisfacción del usuario	Observación Encuesta	Software implementado utilizando el método propuesto
	V. DEPENDIENTE Mejora en el nivel de seguridad	Total de vulnerabilidad es detectadas	Observación Pruebas de software	Herramientas para detección de vulnerabilidades

Fuente: Monar Joffre, 2016

Realizado por: Monar Joffre, 2016

3.7. Población y muestra

3.7.1. Población

La población es el conjunto de todos los elementos a ser evaluados, en esta investigación se considera como base la utilización de los Controles Proactivos TOP-10 OWASP 2016 que pretenden controlar la seguridad de las aplicaciones web, sobre todo las diez (10) vulnerabilidades más críticas de OWASP que se detallan a continuación:

1. Inyección.
2. Pérdida de autenticación y gestión de sesiones.
3. Secuencia de comandos en sitios cruzados (XSS).
4. Referencia directa insegura a objetos.
5. Configuración de seguridad incorrecta.
6. Exposición de datos sensibles.
7. Ausencia de control de acceso a funciones.
8. Falsificación de peticiones en sitios cruzados (CSRF).
9. Utilización de componentes con vulnerabilidades conocidas.
10. Redirecciones y reenvíos no validados.

Partiendo del estudio de estos controles y la revisión de las vulnerabilidades se definirá las técnicas de programación más adecuadas en PHP para crear el nuevo método propuesto.

3.7.2. Selección de la muestra

Para determinar la muestra se tomó en consideración alguno de los errores más comunes en programación 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; los cuales derivan en la selección de siete (7) de las diez (10) vulnerabilidades más críticas de OWASP-2013 tomadas de forma no probabilística, que serán evaluadas para la creación del método y aplicación de las nuevas técnicas de programación segura en PHP en el nuevo prototipo desarrollado y que finalmente serán

probados en los dos (2) momentos (antes y después de aplicar el método) por los mejores analizadores de vulnerabilidades que podamos determinar.

Tabla 3-3: Vulnerabilidades a evaluar

N°	Vulnerabilidades
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)
7	Redirecciones y reenvíos no validados

Fuente: Monar Joffre, 2016

Realizado por: Monar Joffre, 2016

Del total de usuarios cincuenta (50) que utilizan la aplicación SISEV se escoge de forma no probabilística una muestra de dos (2) usuarios por provincia, en total ocho (8) usuarios, los cuales son los administradores del sistema, quienes son Profesionales en Sistemas que manejan Programación PHP y base de datos MySQL y conocen el funcionamiento del sistema y su código fuente antes y después de aplicar las técnicas de programación definidas en el método, ya que fueron previamente capacitados; los mismos que con sus conocimientos adquiridos validarán la seguridad del aplicativo en los dos (2) momentos mediante encuestas establecidas que consideran las siete (7) vulnerabilidades a mitigar.

3.8. Procedimientos generales

Para recolectar la información se usará en primer lugar los conocimientos adquiridos de los administradores de la aplicación SISEV de las cuatro (4) provincias donde funciona el sistema, con la finalidad de verificar si las técnicas de programación planteadas en el nuevo método desarrollado son las adecuadas, esto mediante encuestas; y en segundo lugar se revisarán los resultados obtenidos de la utilización de los analizadores de

vulnerabilidades en los dos (2) momentos del aplicativo (antes y después de aplicar el método).

Tabla 4-3: Técnicas para la demostración de la hipótesis

Variables	Indicadores	Técnicas
V. INDEPENDIENTE Método propuesto de programación segura para desarrollo de aplicaciones web en PHP	Nivel de satisfacción del usuario	ENCUESTA 1. ¿El sistema valida todas las entradas en los formularios establecidos? 2. ¿Considera que el sistema realiza una gestión de sesión adecuada? 3. ¿Existe un cifrado de datos adecuado para el manejo de contraseñas? 4. ¿El sistema maneja los errores y control de excepciones de manera segura? 5. ¿Considera que el sistema se encuentra en un servidor que mantiene una configuración de seguridad adecuada para el servidor web, la base de datos y el lenguaje PHP? 6. ¿Cree usted que la autenticación en el sistema está bien definida? 7. ¿Considera que se está aplicando en el sistema una autorización adecuada para áreas de acceso más restringidas? 8. ¿Considera usted que las técnicas de programación utilizadas en el sistema ayudarán a minimizar los ataques web?
V. DEPENDIENTE Mejora en el nivel de seguridad	Total de vulnerabilidades detectadas	PRUEBAS DE SOFTWARE Utilización de las herramientas ACUNETIX, OWASP-ZAP, entre otras para la detección de vulnerabilidades.

Fuente: Monar Joffre, 2016

Realizado por: Monar Joffre, 2016

3.9. Instrumentos de recolección de datos primarios y secundarios

Una vez determinada las vulnerabilidades a estudiar para el desarrollo de las técnicas de programación; se utilizará los siguientes programas open source definidos en la sección 2.2.5.2 “Herramientas para detección de vulnerabilidades web”, para la detección de vulnerabilidades en los dos (2) prototipos desarrollados en los dos (2) momentos de estudio (antes y después de aplicar el método).

Cabe mencionar que para seleccionar estos cuatro (4) analizadores se basó en estudios previos determinados en el Marco Teórico y en pruebas realizadas cuando se tenía únicamente el primer prototipo.

Tabla 5-3: Herramientas de detección de vulnerabilidades

Tipo de vulnerabilidad	Herramientas
Inyección	<p style="text-align: center;">Analizadores de Vulnerabilidades</p> <p style="text-align: center;">ACUNETIX OWASP ZAP W3AF NIKTO</p>
Pérdida de autenticación y gestión de sesiones	
Secuencia de comandos en sitios cruzados (XSS)	
Configuración de seguridad incorrecta	
Exposición de datos sensibles	
Falsificación de peticiones en sitios cruzados	
Redirecciones y reenvíos no validados	

Fuente: Monar Joffre, 2016

Realizado por: Monar Joffre, 2016

Tomando en cuenta que las herramientas descritas en la Tabla 5-3 son las más utilizadas para estos tipos de análisis, luego de las pruebas pertinentes en el prototipo 1 con cada una de estas herramientas se designó a ACUNETIX (ver tabla 7-3) como el principal analizadora utilizar en los dos (2) momentos (antes y después del método); para ello se asignó pesos o valores de acuerdo a la Escala de Likert.

Tabla 6-3: Escala de Likert

	Siempre	Casi siempre	Algunas veces	Muy pocas veces	Nunca
ITEMS POSITIVOS	5	4	3	2	1

Fuente: Escala de Likert

Realizado por: Monar Joffre, 2016

Tabla 7-3: Pesos asignados para la selección de la herramienta

Vulnerabilidad	Acunetix	Owasp-ZAP	W3af	Nikto
Inyección	5	4	3	4
Pérdida de autenticación y gestión de sesiones	5	5	4	3
Secuencia de comandos en sitios cruzados (XSS)	4	4	3	4
Configuración de seguridad incorrecta	3	3	3	3
Exposición de datos sensibles	4	3	3	3
Falsificación de peticiones en sitios cruzados (CSRF)	5	4	3	4
Redirecciones y reenvíos no validados	4	3	3	3
TOTAL	30	26	22	24

Fuente: MONAR Joffre, 2016

Realizado por: MONAR Joffre, 2016

El peso fue determinado no sólo por el nivel de vulnerabilidad que detecta sino además por su instalación y configuración, nivel de usabilidad, costo de licencias para el caso de versiones pagas, entre otros; de acuerdo a este resultado se determinó que Acunetix es la mejor herramienta para la detección de vulnerabilidades por lo que se utilizará en los dos (2) prototipos.

3.10. Instrumentos para procesar datos recopilados

Entre los instrumentos para procesar los datos tenemos:

- **Excel.**-Para el tratamiento de los datos, nos ayudará en la realización de gráficos, diagramas, etc.; prepara la información para facilitar su análisis posterior.
- **SPSS** (paquete estadístico).- Realiza un análisis estadístico de los datos (pruebas) y servirá además para graficar la Distribución.

3.11. Ambiente de Pruebas

3.11.1. Hardware y software utilizado

Se utilizará una variante del sistema SISEV que se encuentra en la URL:<http://servicios.agricultura.gob.ec/smap/>, ubicada en uno de los servidores del MAGAP (192.168.1.240); el cual se denominará S-SISEV y estará ubicada en un servidor de pruebas (192.168.4.69) que contará con las configuraciones necesarias para una adecuada seguridad y óptimo rendimiento.

Para el desarrollo de este aplicativo (S-SISEV) se tomó en cuenta algunas técnicas de programación que se describen en el método, con soluciones a siete (7) de las diez(10) vulnerabilidades más críticas de OWASP-2013.

Para la implementación de los escenarios de prueba se contó con el siguiente hardware y software:

Tabla 8-3: Hardware utilizado para pruebas

Cantidad	Equipo	Marca	Modelo	Especificaciones	Observaciones
1	PORTATIL	DELL	INSPIRON 3437	INTEL CORE I5 1.6 GHZ, 6 GB en RAM	Equipo personal para el desarrollo del prototipo y el proyecto de tesis
1	SERVIDOR PRINCIPAL	HP		INTEL XEON E5420 2.5 Ghz, 4 GB en RAM	Servidor utilizado para los servicios web: SISEV, CIVZ3, SISVIAT, GLPI, entre otros.
1	SERVIDOR DE PRUEBAS			INTEL Core 2 Duo 6400 .1 Ghz, 1.8 GB en RAM	Servidor sobre el cual está montado el prototipo para realizar los test de penetración

Fuente: Dirección de Soporte e Infraestructura-MAGAP-Planta Central

Realizado por: Monar Joffre, 2016

Tabla9-3: Software utilizado para pruebas

Nombre	Descripción	Observaciones
CENTOS 6.7	SISTEMA OPERATIVO LINUX	SOFTWARE UTILIZADO EN EL SERVIDOR DE PRUEBAS
MariaDB	SERVIDOR DE BASE DE DATOS	
APACHE 2.4.17	SERVIDOR DE APLICACIONES	
PHP 5.6.15	LENGUAJE DE PROGRAMACIÓN	

Fuente: Dirección de Soporte e Infraestructura-MAGAP-Planta Central

Realizado por: MONAR Joffre, 2016

3.11.2. *Prototipos de prueba*

Para definir las técnicas de programación que se aplicarán en el prototipo 2 se utilizará como guía siete (7) de los diez (10) Controles Proactivos TOP-10 OWASP 2016, los cuales se detallan a continuación:

- Parametrizar las consultas.
- Codificar los datos.
- Validar todas las entradas.
- Implementar controles de identidad y autenticación.
- Implementar controles de acceso apropiados.
- Proteger los datos.
- Error y control de excepciones.

3.12. Selección de la metodología

Aunque no existe un sistema 100% seguro, en este caso una aplicación web 100% segura; para generar técnicas de programación seguras no podemos descartar ninguna propuesta de seguridad existente, ya que, todas las medidas que se puedan tomar para prevenir riesgos de seguridad se deben tomar en consideración para poder mejorar los niveles de seguridad, y si bien es cierto, no se podrá combatir completamente los riesgos de seguridad, si se podrá hacer más difícil la tarea de quien intente vulnerar las seguridades de la aplicación.

En la elección de la metodología como se describe en la sección 2.1.3 (Pros y contras de metodologías a estudiar) se utilizará como base la metodología OWASP, que contiene una serie de guías, prácticas de codificación, controles de seguridad, entre otros; que nos servirán como base para crear el modelo que utilizará nuevas técnicas de programación segura en PHP.

3.12.1. Metodología seleccionada

Para definir el método o guía que utilizará técnicas de programación segura para el desarrollo de aplicaciones web en entornos PHP y mitigar al menos siete (7) de las diez (10) vulnerabilidades más conocidas (TOP 10–OWASP 2013), se utilizó como base el proyecto OWASP, el cual contiene prácticas de codificación, guías de desarrollo, controles de seguridad, entre otros; como: Controles Proactivos TOP 10 versión 2 de 2016, el Estándar de Verificación para la Seguridad de las Aplicaciones (ASVS) versión 3 de 2015, la Guía de Desarrollo OWASP versión 4, Hojas de Trucos para el manejo de sesiones y autenticación, la Guía de revisión de código en su versión 2 y Prácticas de Codificación Segura versión 2 de 2011, así como la Guía de Seguridad Española y recomendaciones de algunos expertos inmersos en la seguridad de las aplicaciones.

A continuación una breve descripción de algunas de ellas:

- **ASVS:** Es una guía que ajusta los requisitos de seguridad de las aplicaciones web que ya están construidas. (OWASP, Application Security Verification Standard 3.0, 2015).
- **Guía de Desarrollo OWASP y Hoja de Trucos para prevenir vulnerabilidades:** Se utilizan como punto de partida para añadir seguridad a las aplicaciones web que se van a construir.
- **Controles Proactivos:** Consiste en controlar las metas establecidas independientemente con un grado de aceptación que en su caso, puedan ser cambiadas si su caso lo requiere anticipadamente para evitar futuros problemas. Cuando se detectan problemas, se van haciendo ajustes, estos ajustes se van realizando de manera tal que no puedan crear problemas. Por consiguiente, el control proactivo se aplica antes de cualquier actividad sea realizada y va en

relación al trabajo y la productividad. (OWASP, OWASP Top 10 Proactive Controls 2016, 2016).

- **Guía de revisión de código:** Verifica la seguridad desde el código, es más efectivo (en dinero) que las pruebas de intrusión. (OWASP, CODE REVIEW GUIDE 2.0 RELEASE, 2013).
- **Prácticas de codificación segura:** Son prácticas asumidas por algunas Empresas o desarrolladores para la generación de código seguro de acuerdo a sus experiencias e investigación, se implementa para un lenguaje de programación determinado, en este caso para PHP (por ser objeto de estudio). (OWASP, OWASP Secure Coding Practices, 2011).

Cada una de estas guías busca mejorar la seguridad de las aplicaciones a través de controles o prácticas de codificación pero no todas se orientan específicamente al lenguaje PHP y algunos estudios que existen o están desactualizados (versiones anteriores a PHP 5) o utilizan funciones que su uso ya no es recomendado debido a mejores técnicas de ataque.

Es por ello que esta guía o método está determinada para las últimas versiones del lenguaje PHP que todavía tienen soporte total hasta el 2017 y 2018 como PHP 5.6.x y PHP 7.0.x respectivamente; se incluye las configuraciones para el servidor de aplicaciones (Apache) y el servidor de base de datos (MySQL) para una seguridad integral.

CAPÍTULO IV

4. RESULTADOS Y DISCUSIÓN

4.1. Presentación de resultados

En esta sección se realizará una discusión de los resultados, donde se interpretará los resultados obtenidos en la investigación y su relación con los objetivos y la hipótesis.

En base a los resultados obtenidos, se observará que muchos programadores sobre todo quienes utilizan código propio tienen una escritura defectuosa de código debido a que no han recibido una adecuada capacitación sobre cómo realizar programación segura.

4.2. Identificación de activos de información

La identificación de activos es importante ya que permite materializar con precisión el alcance de la investigación, permite valorar los activos con veracidad e identificar las amenazas a las que se encuentran expuestos dichos activos.

Tabla 1-4: Activos utilizados en la investigación

Número de Activo	Activo
Activo 1	Servicio Web
Activo 2	Servidor principal
Activo 3	Servidor de pruebas
Activo 4	Equipos de protección (firewall)

Fuente: Dirección de Soporte e Infraestructura (MAGAP)

Realizado por: Monar Joffre, 2016

4.2.1. Servicio Web

En la Tabla 2-4 se enumera el servicio Web (SISEV) que brinda la Plataforma de MAGAP-Planta Central a sus funcionarios a nivel de zona 3.

Tabla 2-4: Servicio Web

Servicio	URL
Sistema de Seguimiento y Evaluación (SISEV)	http://servicios.agricultura.gob.ec/smap

Fuente: Dirección de Soporte e Infraestructura (MAGAP)

Realizado por: Monar Joffre, 2016

Este servicio web es utilizado por cincuenta (50) funcionarios del MAGAP Zona 3, los cuales son Técnicos de Territorio que requieren llevar de manera ordenada los Planes de Intervención y Planificaciones Semanales en el sistema y que se distribuyen por provincias de la siguiente manera:

Tabla 3-4: Número de usuarios que utilizan el Servicio Web

	Provincias	# Usuarios	Total
ZONA 3	CHIMBORAZO	17	50
	TUNGURAHUA	13	
	COTOPAXI	12	
	PASTAZA	8	

Fuente: Base de datos SISEV

Realizado por: Monar Joffre, 2016

Existen dos (2) Informáticos por cada provincia que hacen de administradores locales y que se incluyen en el total de funcionarios que utilizan el sistema.

4.2.2. Servidor Principal

En Tabla 4-4 se detalla las características del servidor principal (IP: 192.168.1.240) donde se encuentra ubicada la base de datos (magap_civz3) y el código fuente de la aplicación SISEV.

Tabla 4-4: Características del Servidor Principal

Equipo Físico Hp De Rack	
Procesador	INTEL XEON E5420 2.5 Ghz
Memoria	4Gb
Disco Duro	273 Gb
S.O	Windows Server 2008 R2
B.D	MYSQL 5.5.16
Apache	2.2.21
PHP	5.3.8

Fuente: Dirección de Soporte e Infraestructura (MAGAP)

Realizado por: Monar Joffre, 2016

Cabe indicar que este servidor es utilizado para los servicios web: SISEV, CIVZ3, SISVIAT, GLPI, entre otros.

4.2.3. Servidor de Pruebas

En la Tabla 5-4 se detalla las características del servidor de pruebas (IP: 192.168.4.69) donde se ubica la aplicación web (S-SISEV) que utiliza las técnicas de programación segura descritas en este estudio:

Tabla 5-4: Características del Servidor de Pruebas

Equipo Informático	
Procesador	INTEL Core 2 Duo 6400 .1 Ghz
Memoria	1.8Gb
Disco Duro	150 Gb
S.O	Centos 7
B.D	MariaDB
Apache	2.4.17
PHP	5.6.30

Fuente: Dirección de Soporte e Infraestructura (MAGAP)

Realizado por: Monar Joffre, 2016

4.2.4. Equipos de Protección

Aquí citamos las características de los equipos de protección.

Tabla 6-4: Equipos de Protección

Protección	Características del equipo	Observaciones
Firewall de perímetro	Checkpoint 12200 con módulos de FW, IPS, DLP, Threat Prevention, Anti Virus y Anti spam	Primera línea de protección
Firewall de Data Center	ASA 5400	Segunda línea de protección: permite acceso externo e interno solo a los puertos e IPs pedidas por los administradores de cada sistema

Fuente: Dirección de Soporte e Infraestructura (MAGAP)

Realizado por: Monar Joffre, 2016

4.3. Procesamiento y análisis

Para esta investigación se aplicó la encuesta (Anexo A) a los administradores locales y zonales de las cuatro (4) provincias que utilizan el sistema para el caso de la demostración de la variable independiente y la ejecución de dos (2) analizadores de vulnerabilidades sobre los dos (2) prototipos para el caso de la demostración de la variable dependiente; luego para la verificación de la hipótesis se analizarán los resultados obtenidos.

4.4. Valoraciones de la variable independiente

4.4.1. Variable independiente: Método propuesto de programación segura para desarrollo de aplicaciones web en PHP

Para su valoración se consideró una encuesta a ocho (8) administradores locales del sistema SISEV.

4.4.2. *Indicador: Nivel de satisfacción del usuario*

1. Encuesta realizada antes de aplicar el método a los administradores locales zonales que utilizan el sistema SISEV

Pregunta 1. ¿El sistema valida todas las entradas en los formularios establecidos?

Tabla 7-4: Encuesta-Pregunta 1-antes de aplicar la propuesta

Alternativas	Frecuencia	Porcentaje
SI	2	25%
NO	6	75%
TOTAL	8	100%

Fuente: Resultados Encuesta

Realizado por: Monar Joffre, 2017

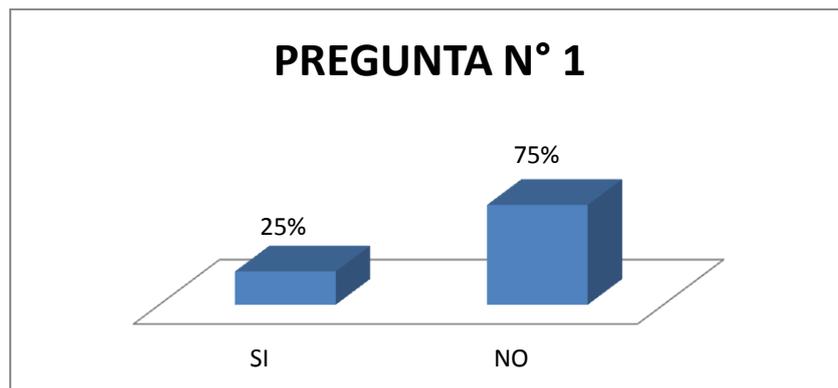


Gráfico 1-4: Encuesta-Pregunta 1-antes de aplicar la propuesta

Realizado por: Monar Joffre, 2017

Análisis e Interpretación de resultados:

El 75% afirma que no se encuentra validado correctamente las entradas ni tampoco restringe el tamaño de los campos, además permite vulnerabilidades sobre todo de tipo SQL Injection y XSS.

Pregunta 2. ¿Considera que el sistema realiza una gestión de sesión adecuada?

Tabla 8-4: Encuesta-Pregunta 2-antes de aplicar la propuesta

Alternativas	Frecuencia	Porcentaje
SI	3	37.5%
NO	5	62.5%
TOTAL	8	100%

Fuente: Resultados Encuesta

Realizado por: Monar Joffre, 2017

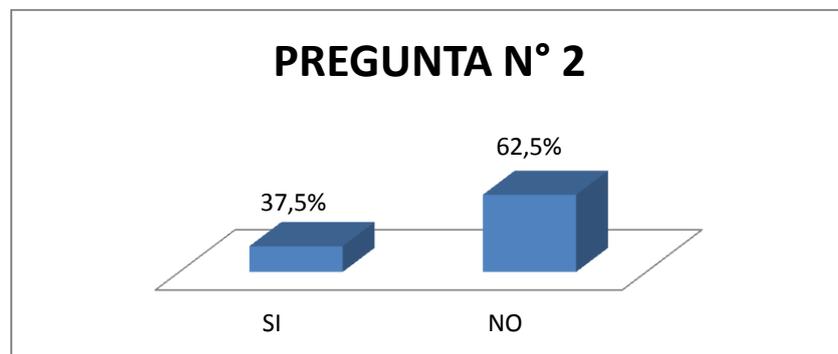


Gráfico 2-4: Encuesta-Pregunta 2-antes de aplicar la propuesta

Realizado por: Monar Joffre, 2017

Análisis e Interpretación de resultados:

El 62.5% cree que no se está realizando una gestión de sesión adecuada, ya que no se renueva la sesión de manera aleatoria, además no se encuentra encriptado lo que permitiría un secuestro o robo de sesión; adicional el tiempo de sesión es muy largo y el cierre no se lo realiza correctamente.

Pregunta 3. ¿Existe un cifrado de datos adecuado para el manejo de contraseñas?

Tabla 9-4: Encuesta-Pregunta 3-antes de aplicar la propuesta

Alternativas	Frecuencia	Porcentaje
SI	0	0%
NO	8	100%
TOTAL	8	100%

Fuente: Resultados Encuesta

Realizado por: Monar Joffre, 2017

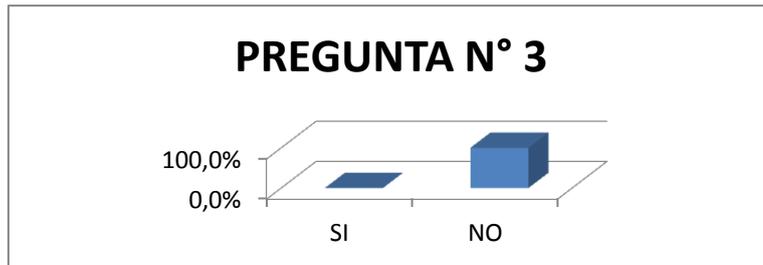


Gráfico 3-4: Encuesta-Pregunta 3-antes de aplicar la propuesta

Realizado por: Monar Joffre, 2017

Análisis e Interpretación de resultados:

El 100% considera que el algoritmo de encriptación md5 que utiliza para las contraseñas no es el adecuado, ya que dejó de utilizarse por ser muy inseguro; existen otros algoritmos fuertes como SHA 256 o SHA 512 que podría implementarse.

Pregunta 4. ¿El sistema maneja los errores y control de excepciones de manera segura?

Tabla 10-4: Encuesta-Pregunta 4-antes de aplicar la propuesta

Alternativas	Frecuencia	Porcentaje
SI	2	25%
NO	6	75%
TOTAL	8	100%

Fuente: Resultados Encuesta

Realizado por: Monar Joffre, 2017

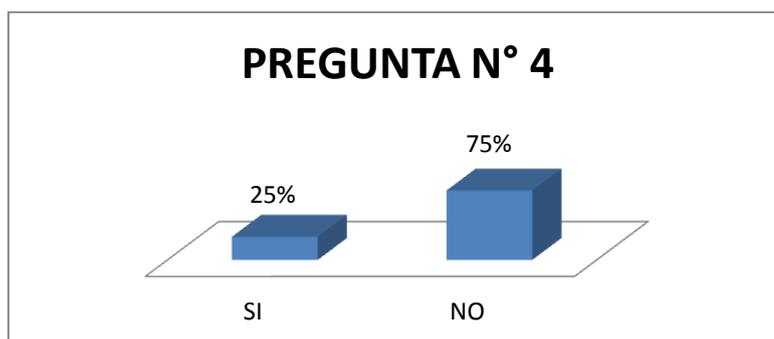


Gráfico 4-4: Encuesta-Pregunta 4-antes de aplicar la propuesta

Realizado por: Monar Joffre, 2017

Análisis e Interpretación de resultados:

El 75% cree que los mensajes de error que muestra el sistema son muy específicos e informan al atacante donde se encuentra el error, uno de los mensajes lo encontramos en la pantalla de login donde indica que existe un error al ingresar la contraseña.

Pregunta 5. ¿Considera que el sistema se encuentra en un servidor que mantiene una configuración de seguridad adecuada para el servidor web, la base de datos y el lenguaje PHP?

Tabla 11-4: Encuesta-Pregunta 5-antes de aplicar la propuesta

Alternativas	Frecuencia	Porcentaje
SI	4	50%
NO	4	50%
TOTAL	8	100%

Fuente: Resultados Encuesta

Realizado por: Monar Joffre, 2017

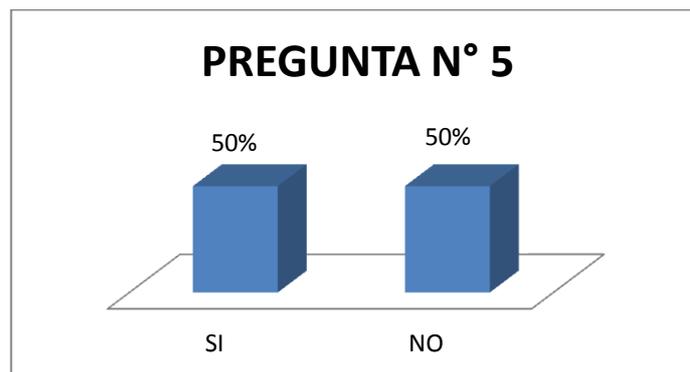


Gráfico 5-4: Encuesta-Pregunta 5-antes de aplicar la propuesta

Realizado por: Monar Joffre, 2017

Análisis e Interpretación de resultados:

El 50% considera que el sistema se encuentra en un servidor que implementa adecuadamente APACHE, MySQL y PHP ya que mantiene configuraciones de seguridad adecuadas e integrado a un firewall de protección; pero el otro 50% no lo considera así ya que XAMPP no es una configuración adecuada porque se debería configurar por separado.

Pregunta 6. ¿Cree usted que la autenticación en el sistema está bien definida?

Tabla 12-4: Encuesta-Pregunta 6-antes de aplicar la propuesta

Alternativas	Frecuencia	Porcentaje
SI	0	0%
NO	8	100%
TOTAL	8	100%

Fuente: Resultados Encuesta

Realizado por: Monar Joffre, 2017

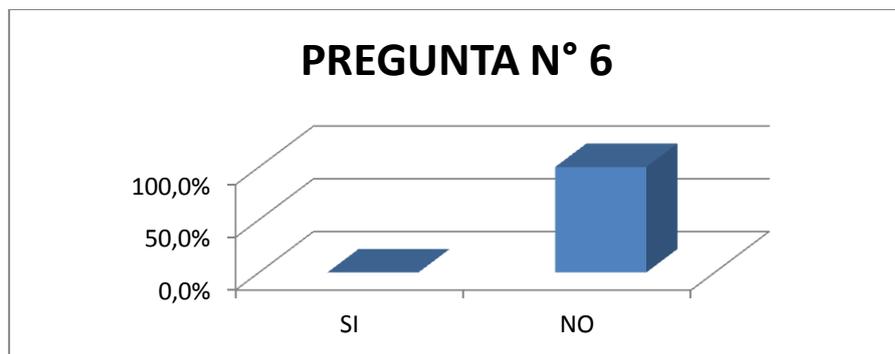


Gráfico 6-4: Encuesta-Pregunta 6-antes de aplicar la propuesta

Realizado por: Monar Joffre, 2017

Análisis e Interpretación de resultados:

El 100% cree que la autenticación no es la adecuada, ya que no se implementa un captcha que dificulta los ataques de fuerza bruta, además no bloquea el usuario después de un número de intentos determinado.

Pregunta 7. ¿Considera que se está aplicando en el sistema una autorización adecuada para áreas de acceso más restringidas?

Tabla 13-4: Encuesta-Pregunta 7-antes de aplicar la propuesta

Alternativas	Frecuencia	Porcentaje
SI	2	25%
NO	6	75%
TOTAL	8	100%

Fuente: Resultados Encuesta

Realizado por: Monar Joffre, 2017

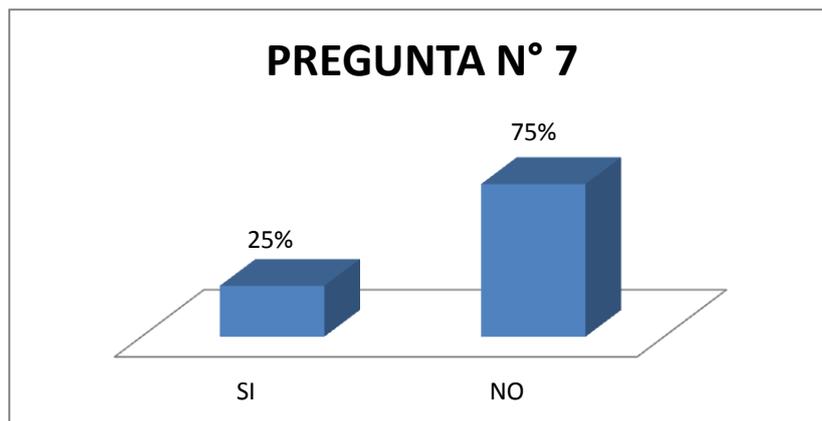


Gráfico 7-4: Encuesta-Pregunta 7-antes de aplicar la propuesta

Realizado por: Monar Joffre, 2017

Análisis e Interpretación de resultados:

El 75% considera que el acceso a áreas restringidas no es el adecuado ya que para páginas como cambio de contraseña no solicita re autenticación del usuario.

Pregunta 8. ¿Considera usted que las técnicas de programación utilizadas en el sistema ayudarán a minimizar los ataques web?

Tabla 14-4: Encuesta-Pregunta 8-antes de aplicar la propuesta

Alternativas	Frecuencia	Porcentaje
SI	0	0%
NO	8	100%
TOTAL	8	100%

Fuente: Resultados Encuesta

Realizado por: Monar Joffre, 2017

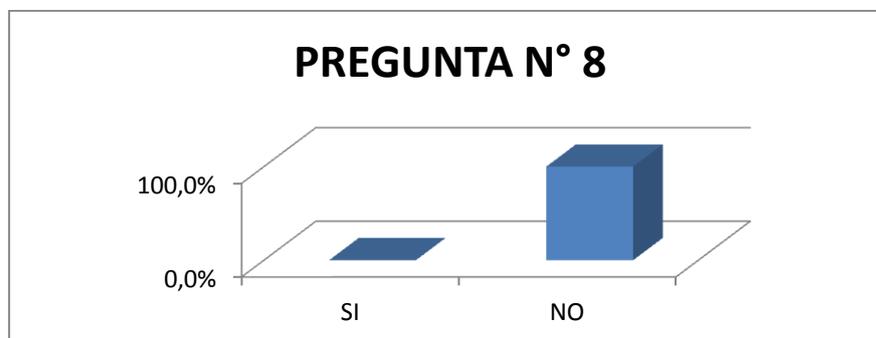


Gráfico 8-4: Encuesta-Pregunta 8-antes de aplicar la propuesta

Realizado por: Monar Joffre, 2017

Análisis e Interpretación de resultados:

El 100% considera que las técnicas de programación utilizadas son vulnerables a los principales ataques web como SQL Injection, XSS, CSRF, entre otros; ya que la programación sobre todo no valida de manera adecuada los campos de formulario permitiendo acceder directamente a las consultas.

- **Resumen de resultados antes de aplicar el método**

Tabla 15-4: Resumen de resultados de Encuesta-Antes de aplicar el método

N°	Preguntas	Frecuencias		Total
		SI	NO	
1	¿El sistema valida todas las entradas en los formularios establecidos?	2	6	8
2	¿Considera que el sistema realiza una gestión de sesión adecuada?	3	5	8
3	¿Existe un cifrado de datos adecuado para el manejo de contraseñas?	0	8	8
4	¿El sistema maneja los errores y control de excepciones de manera segura?	2	6	8
5	¿Considera que el sistema se encuentra en un servidor que mantiene una configuración de seguridad adecuada para el servidor web, la base de datos y el lenguaje PHP?	4	4	8
6	¿Cree usted que la autenticación en el sistema está bien definida?	0	8	8
7	¿Considera que se está aplicando en el sistema una autorización adecuada para áreas de acceso más restringidas?	2	6	8
8	¿Considera usted que las técnicas de programación utilizadas en el sistema ayudarán a minimizar los ataques web?	0	8	8
TOTAL		13	51	64

Fuente: Resultados Encuesta

Realizado por: Monar Joffre, 2017

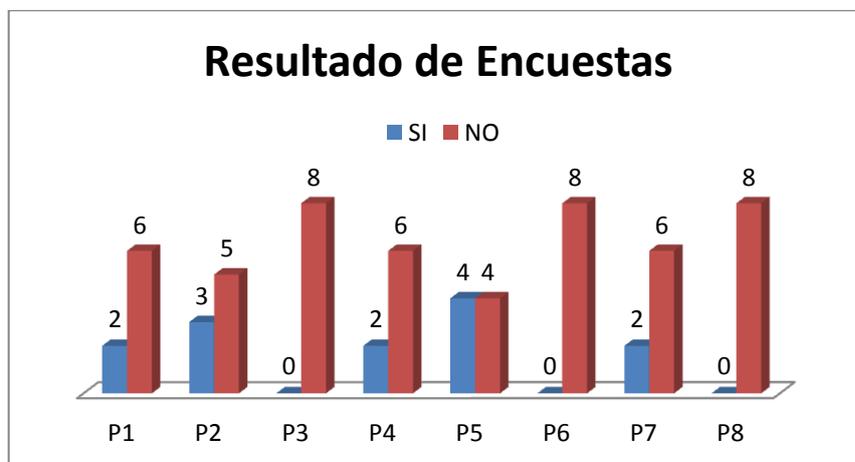


Gráfico 9-4: Resumen de resultados de Encuesta-Antes de aplicar el método

Realizado por: Monar Joffre, 2017

2. Encuesta realizada después de aplicar el método a los administradores locales y zonales que utilizan el sistema S-SISEV

Pregunta 1. ¿El sistema valida todas las entradas en los formularios establecidos?

Tabla 16-4: Encuesta-Pregunta 1-después de aplicar la propuesta

Alternativas	Frecuencia	Porcentaje
SI	8	100%
NO	0	0%
TOTAL	8	100%

Fuente: Resultados Encuesta

Realizado por: Monar Joffre, 2017

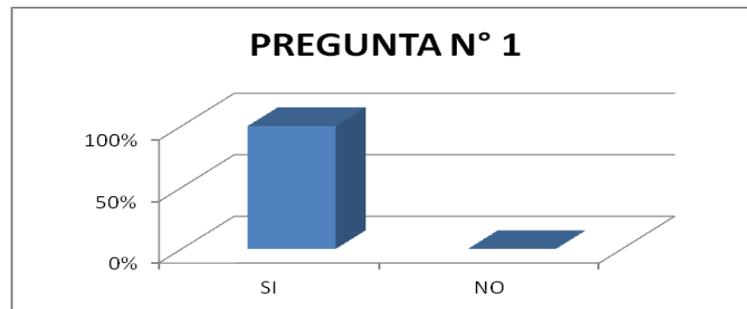


Gráfico 10-4: Encuesta-Pregunta 1-después de aplicar la propuesta

Realizado por: Monar Joffre, 2017

Análisis e Interpretación de resultados:

El 100% afirma que probaron cada una de las entradas en los formularios para determinar si existía vulnerabilidad del tipo SQL Inyección, XSS, entre otros; llegando a la conclusión que la utilización de consultas parametrizadas conjuntamente con controles adicionales para validar los datos ayuda a mitigar las vulnerabilidades más conocidas.

Pregunta 2. ¿Considera que el sistema realiza una gestión de sesión adecuada?

Tabla 17-4: Encuesta-Pregunta 2-después de aplicar la propuesta

Alternativas	Frecuencia	Porcentaje
SI	8	100%
NO	0	0%
TOTAL	8	100%

Fuente: Resultados Encuesta

Realizado por: Monar Joffre, 2017

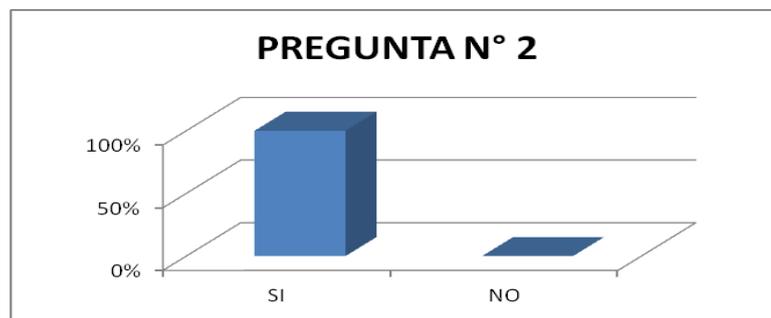


Gráfico11-4: Encuesta-Pregunta 2-después de aplicar la propuesta

Realizado por: Monar Joffre, 2017

Análisis e Interpretación de resultados:

El 100% cree que se está realizando una gestión adecuada de la sesión, ya que al regenerarla y cifrarla permite evitar un secuestro de sesión.

Pregunta 3. ¿Existe un cifrado de datos adecuado para el manejo de contraseñas?

Tabla 18-4: Encuesta-Pregunta 3-después de aplicar la propuesta

Alternativas	Frecuencia	Porcentaje
SI	8	100%
NO	0	0%
TOTAL	8	100%

Fuente: Resultados Encuesta

Realizado por: Monar Joffre, 2017

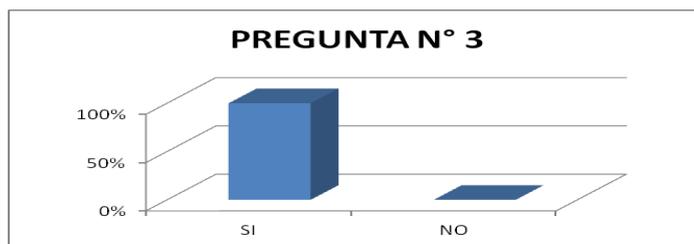


Gráfico 12-4: Encuesta-Pregunta 3-después de aplicar la propuesta

Realizado por: Monar Joffre, 2017

Análisis e Interpretación de resultados:

El 100% considera que el hash (sha512) utilizado para el cifrado de contraseñas es uno de los algoritmos más seguros actualmente.

Pregunta 4. ¿El sistema maneja los errores y control de excepciones de manera segura?

Tabla 19-4: Encuesta- Pregunta 4-después de aplicar la propuesta

Alternativas	Frecuencia	Porcentaje
SI	7	87,5%
NO	1	12,5%
TOTAL	8	100%

Fuente: Resultados Encuesta

Realizado por: Monar Joffre, 2017

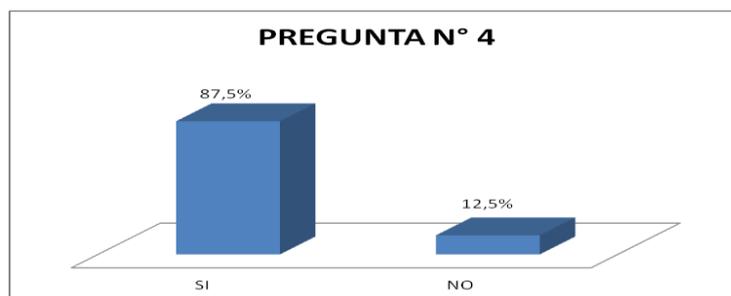


Gráfico 13-4: Encuesta-Pregunta 4-después de aplicar la propuesta

Realizado por: Monar Joffre, 2017

Análisis e Interpretación de resultados:

El 87.5% cree que el manejo de errores y control de excepciones que realiza el sistema es el adecuado, ya que muestra información general del error lo que dificulta al atacante conocer dónde está el error; por ejemplo en un formulario de login nos mostrará usuario o contraseña incorrectos.

Pregunta 5. ¿Considera que el sistema se encuentra en un servidor que mantiene una configuración de seguridad adecuada para el servidor web, la base de datos y el lenguaje PHP?

Tabla 20-4: Encuesta-Pregunta 5-después de aplicar la propuesta

Alternativas	Frecuencia	Porcentaje
SI	8	100%
NO	0	0%
TOTAL	8	100%

Fuente: Resultados Encuesta

Realizado por: Monar Joffre, 2017

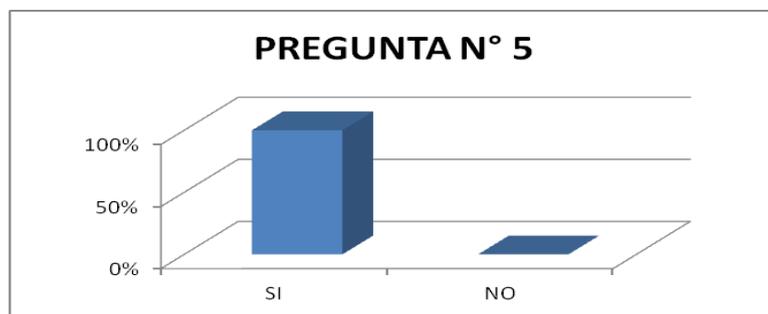


Gráfico 14-4: Encuesta-Pregunta 5-después de aplicar la propuesta

Realizado por: Monar Joffre, 2017

Análisis e Interpretación de resultados:

El 100% considera que el método cubre una configuración e implementación adecuada de APACHE, MySQL y PHP; además de que integra algunas configuraciones de seguridad para el Sistema Operativo Centos 7.

Pregunta 6. ¿Cree usted que la autenticación en el sistema está bien definida?

Tabla 21-4: Encuesta-Pregunta 6-después de aplicar la propuesta

Alternativas	Frecuencia	Porcentaje
SI	7	87,5%
NO	1	12,5%
TOTAL	8	100%

Fuente: Resultados Encuesta

Realizado por: Monar Joffre, 2017

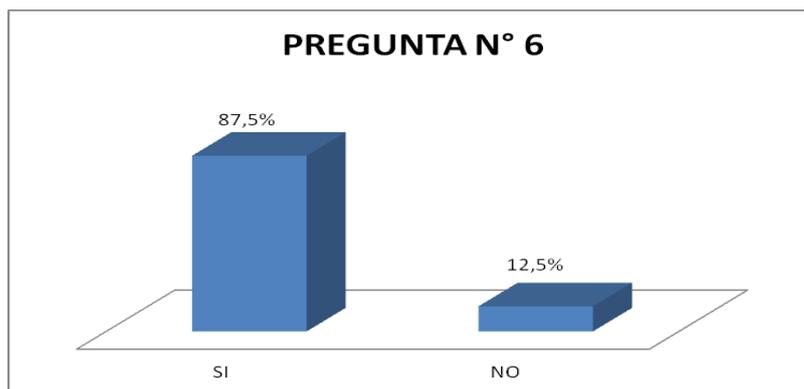


Gráfico 15-4: Encuesta-Pregunta 6-después de aplicar la propuesta

Realizado por: Monar Joffre, 2017

Análisis e Interpretación de resultados:

El 87.5% cree que la autenticación es la adecuada ya que el sistema ofrece un bloqueo de usuario por intentos fallidos así como implementa un captcha que dificulta los ataques de fuerza bruta.

Pregunta 7. ¿Considera que se está aplicando en el sistema una autorización adecuada para áreas de acceso más restringidas?

Tabla 22-4: Encuesta- Pregunta 7-después de aplicar la propuesta

Alternativas	Frecuencia	Porcentaje
SI	8	100%
NO	0	0%
TOTAL	8	100%

Fuente: Resultados Encuesta

Realizado por: Monar Joffre, 2017

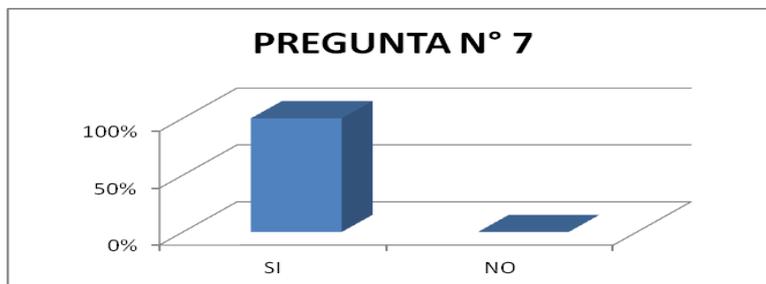


Gráfico 16-4: Encuesta-Pregunta 7-después de aplicar la propuesta

Realizado por: Monar Joffre, 2017

Análisis e Interpretación de resultados:

El 100% considera que el acceso a áreas restringidas o que manejan información sensible están garantizados por la re autenticación del usuario.

Pregunta 8. ¿Considera usted que las técnicas de programación utilizadas en el sistema ayudarán a minimizar los ataques web?

Tabla 23-4: Encuesta-Pregunta 8-después de aplicar la propuesta

Alternativas	Frecuencia	Porcentaje
SI	8	100%
NO	0	0%
TOTAL	8	100%

Fuente: Resultados Encuesta

Realizado por: Monar Joffre, 2017

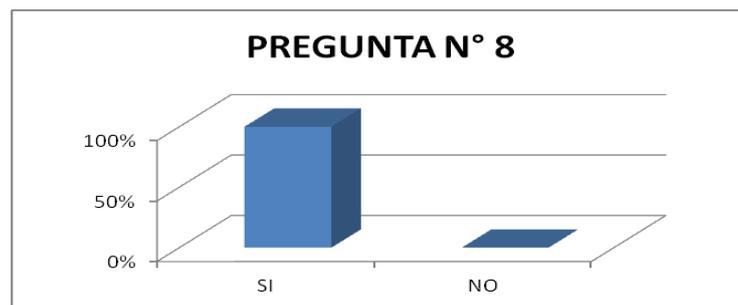


Gráfico 17-4: Encuesta-Pregunta 8-después de aplicar la propuesta

Realizado por: Monar Joffre, 2017

Análisis e Interpretación de resultados:

El 100% considera que las técnicas de programación aplicadas en el método propuesto ayudarán a mitigar sobre todo las vulnerabilidades más conocidas.

- **Resumen de resultados después de aplicar el método**

Tabla 24-4: Resumen de resultados de Encuesta-Después de aplicar el método

N°	Preguntas	Frecuencias		Total
		SI	NO	
1	¿El sistema valida todas las entradas en los formularios establecidos?	8	0	8
2	¿Considera que el sistema realiza una gestión de sesión adecuada?	8	0	8
3	¿Existe un cifrado de datos adecuado para el manejo de contraseñas?	8	0	8
4	¿El sistema maneja los errores y control de excepciones de manera segura?	7	1	8
5	¿Considera que el método propuesto ofrece una configuración de seguridad adecuada para el servidor web, la base de datos y el lenguaje PHP?	8	0	8
6	¿Cree usted que la autenticación en el sistema está bien definida?	7	1	8
7	¿Considera que se está aplicando en el sistema una autorización adecuada para áreas de acceso más restringidas?	8	0	8
8	¿Considera usted que las técnicas de programación utilizadas en el sistema ayudarán a minimizar los ataques web?	8	0	8
	TOTAL	62	2	64

Fuente: Resultados Encuesta

Realizado por: Monar Joffre, 2017



Gráfico 18-4: Resumen de resultados de Encuesta-Después de aplicar el método

Realizado por: Monar Joffre, 2017

3. Resumen de resultados antes y después de aplicar el método

Tabla 25-4: Resumen de resultados de Encuesta-Antes y Después de aplicar el método

N°	Preguntas	Antes		Después	
		SI	NO	SI	NO
1	¿El sistema valida todas las entradas en los formularios establecidos?	2	6	8	0
2	¿Considera que el sistema realiza una gestión de sesión adecuada?	3	5	8	0
3	¿Existe un cifrado de datos adecuado para el manejo de contraseñas?	0	8	8	0
4	¿El sistema maneja los errores y control de excepciones de manera segura?	2	6	7	1
5	¿Considera que el método propuesto ofrece una configuración de seguridad adecuada para el servidor web, la base de datos y el lenguaje PHP?	4	4	8	0
6	¿Cree usted que la autenticación en el sistema está bien definida?	0	8	7	1
7	¿Considera que se está aplicando en el sistema una autorización adecuada para áreas de acceso más restringidas?	2	6	8	0
8	¿Considera usted que las técnicas de programación utilizadas en el sistema ayudarán a minimizar los ataques web?	0	8	8	0
TOTAL		13	51	62	2

Fuente: Resultados Encuesta

Realizado por: Monar Joffre, 2017

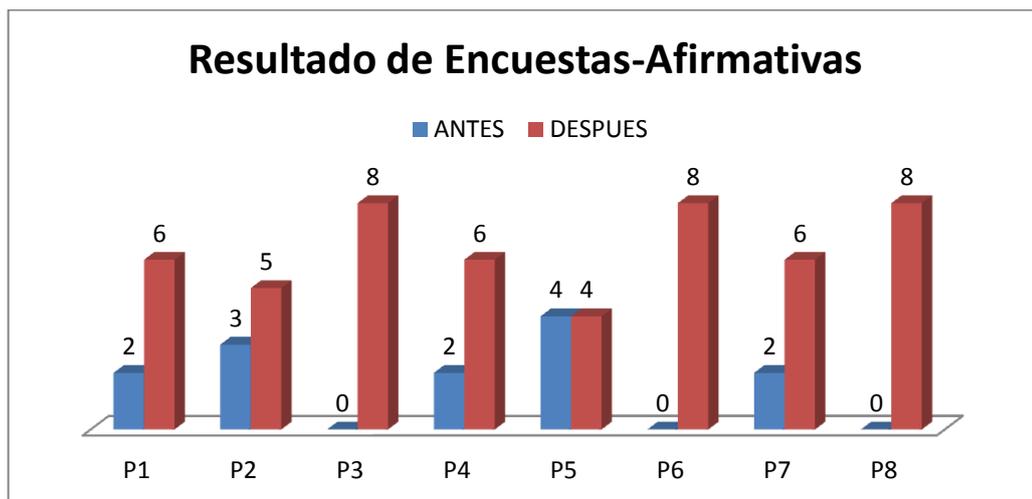


Gráfico 19-4: Resumen de resultados de Encuesta-Antes y Después de aplicar el método

Realizado por: Monar Joffre, 2017

Análisis e Interpretación de resultados:

Se determina que existe una visible mejora en los resultados de las encuestas del antes y después de aplicar las técnicas de programación segura para entornos PHP; por lo que el método propuesto reducirá considerablemente las siete (7) vulnerabilidades tomadas como muestra.

4.5. Valoraciones de la variable dependiente

4.5.1. Variable dependiente: Mejora en el nivel de la seguridad

Para su valoración se consideró la utilización del analizador de vulnerabilidades ACUNETIX sobre los dos (2) prototipos considerados (antes y después del método aplicado).

4.5.2. Indicador: Número total de vulnerabilidades detectadas

1. Identificación de vulnerabilidades antes de aplicar la propuesta

A continuación se detallan las pruebas realizadas y los resultados obtenidos de la URL donde se encuentra ubicada el servicio web: SISEV.

- **Utilizando Acunetix Web Vulnerability Scanner 10.5**

El modo de trabajo de este analizador se describe en la gráfica (Anexo B).

Servicio Web: Sistema de Seguimiento y Evaluación (SISEV)

URL: <http://servicios.agricultura.gob.ec/smap/acceso.php>

Tabla 26-4: Identificación de Vulnerabilidades con ACUNETIX antes del método

Vulnerabilidad	Grado	Numero	Descripción
Blind SQL Injection	ALTO	3	Técnica de ataque que utiliza Inyección SQL
PHP Hash Collision denial of service vulnerability	ALTO	1	Vulnerabilidad de tipo denegación de servicio
Directory listing	MEDIO	63	Es una función de servidor web que muestra una lista de todos los archivos cuando no hay un archivo de índice
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

Fuente: Acunetix Web Vulnerability Scanner

Realizado por: Monar Joffre, 2017

Tabla 27-4: Total de Vulnerabilidades con ACUNETIX antes del método

Importancia	Número
ALTAS	4
MEDIAS	65
BAJAS	14
TOTAL	83

Fuente: Resultados de análisis ACUNETIX

Realizado por: Monar Joffre, 2017

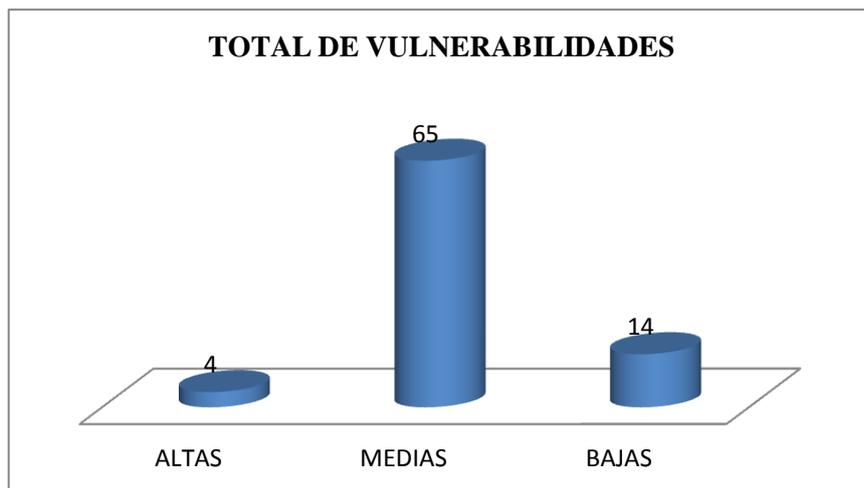


Gráfico 20-4: Total de Vulnerabilidades antes del método

Realizado por: Monar Joffre, 2017

Análisis e Interpretación de resultados:

Se detectaron en total ochenta y tres (83) vulnerabilidades, de las cuales al menos cuatro (4) altas y sesenta y cinco (65) medias, que son al menos las vulnerabilidades que deberíamos corregir en el nuevo sistema que se relacionan a ataques de tipo SQL Injection, XSS y CSRF.

.

2. Identificación de vulnerabilidades después de aplicar la propuesta

A continuación se detallan las pruebas realizadas y los resultados obtenidos de la URL donde se encuentra ubicada el servicio web: S-SISEV.

- **Utilizando Acunetix Web Vulnerability Scanner 10.5**

Servicio Web: Sistema de Seguimiento y Evaluación Seguro (S-SISEV)

URL: <http://192.168.4.69/s-sisev/acceso.php>

Tabla 28-4: Identificación de Vulnerabilidades con ACUNETIX después del método

Vulnerabilidad	Grado	Numero	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

Fuente: Acunetix Web Vulnerability Scanner

Realizado por: Monar Joffre, 2017

Tabla 29-4: Total de Vulnerabilidades con ACUNETIX después del método

Importancia	Número
ALTAS	0
MEDIAS	8
BAJAS	2
TOTAL	10

Fuente: Acunetix Web Vulnerability Scanner

Realizado por: Monar Joffre, 2017

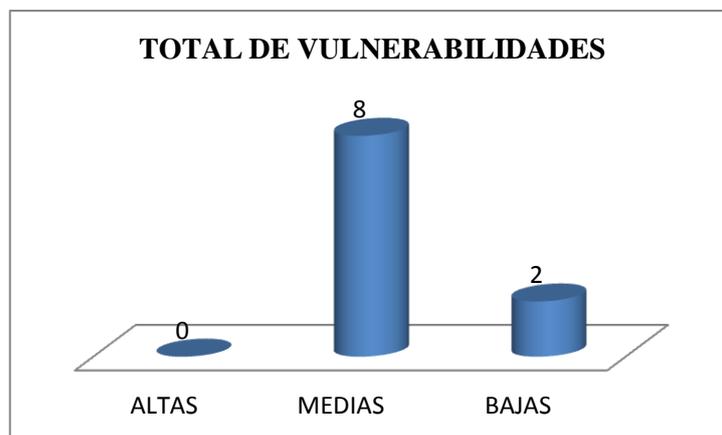


Gráfico 21-4: Total de Vulnerabilidades después del método

Realizado por: Monar Joffre, 2017

Análisis e Interpretación de resultados:

Se detectaron en total diez (10) vulnerabilidades, de las cuales ocho (8) son medias y dos (2) son bajas.

3. Comparativa de vulnerabilidades antes y después de aplicar la propuesta

Tabla 30-4: Comparativa de vulnerabilidades antes y después del método

Vulnerabilidades	Frecuencias		Porcentaje de reducción
	ANTES	DESPUÉS	
ALTAS	4	0	100%
MEDIAS	65	8	87.69%
BAJAS	14	2	85.71%
TOTAL	83	10	87.95%

Fuente: Acunetix Web Vulnerability Scanner
Realizado por: Monar Joffre, 2017

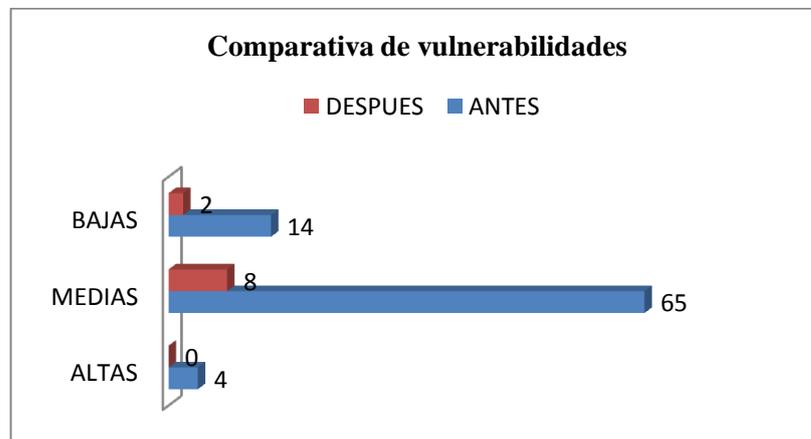


Gráfico 22-4: Comparativo de Vulnerabilidades antes y después del método

Realizado por: Monar Joffre, 2017

Análisis e Interpretación de resultados:

Para medir la reducción de las vulnerabilidades con la aplicación del nuevo método se determina que del 100% de las vulnerabilidades existentes (altas, medias y bajas) se redujo un 87.95%; el mayor porcentaje de reducción se produjo en las vulnerabilidades altas y medias con un 100% y 87.69% respectivamente.

4.6 Comprobación estadística de la hipótesis

Para la comprobación de la Hipótesis General “El método desarrollado propuesto mejorará el nivel de seguridad de las aplicaciones web en entorno PHP ante posibles ataques informáticos” se utilizó estadística inferencial aplicando **Chi-Cuadrado (X^2)**. Luego de realizar un análisis de los resultados obtenidos de las vulnerabilidades encontradas en las aplicaciones web SISEV y S-SISEV se determinó la siguiente hipótesis nula H_0 y la Alternativa H_1 :

- La **hipótesis Nula (H_0)** “El método desarrollado propuesto no mejorará el nivel de seguridad de las aplicaciones web en entorno PHP ante posibles ataques informáticos” con un nivel de significancia del 5% en la prueba de Chi cuadrado X^2 .
- La **hipótesis Alternativa (H_1)** “El método desarrollado propuesto si mejorará el nivel de seguridad de las aplicaciones web en entorno PHP ante posibles ataques informáticos” con un nivel de significancia del 5% en la prueba de Chi cuadrado X^2 .

Para comprobar la hipótesis se utilizó el analizador de vulnerabilidades ACUNETIX que permitió visualizar las vulnerabilidades de los dos (2) prototipos planteados.

Se obtuvo los siguientes resultados preliminares que consideramos como frecuencias de valores observados.

Tabla 31-4: Frecuencias de Valores Observados

Valores Observados			
	ANTES	DESPUÉS	TOTAL
Vulnerabilidades totales detectadas	83	10	93
Vulnerabilidades totales eliminadas	0	73	73
TOTAL	83	83	166

Fuente: Acunetix

Realizado por: Monar Joffre, 2017

La tabla de frecuencias esperadas se obtiene de la siguiente manera:

$$E = \frac{\text{total columna (para cada celda)} * \text{total fila (para cada celda)}}{\text{suma total}}$$

$$E(\text{Si Antes}) = \frac{83 * 93}{166} = 46.5$$

$$E(\text{No Antes}) = \frac{83 * 73}{166} = 36.5$$

$$E(\text{Si Despues}) = \frac{83 * 93}{166} = 46.5$$

$$E(\text{No Despues}) = \frac{83 * 73}{166} = 36.5$$

Tabla 32-4: Frecuencias de Valores Esperados

Valores Esperados			
	ANTES	DESPUÉS	TOTAL
Vulnerabilidades totales detectadas	46.5	46.5	93
Vulnerabilidades totales eliminadas	36.5	36.5	73
TOTAL	83	83	166

Fuente: Acunetix

Realizado por: Monar Joffre, 2017

Una vez obtenido los Valores Esperados el siguiente paso es determinar el valor de X^2 calculado para lo cual se aplica la siguiente ecuación:

$$X^2_{calc.} = \sum_{i=1}^r \sum_{j=1}^k \frac{(O_{ij} - E_{ij})^2}{E_{ij}}$$

Dónde:

O_{ij} : son las frecuencias observadas, es el número de casos observados clasificados en la fila i columna j

E_{ij} : son las frecuencias esperadas, es el número de casos esperados correspondiente a cada fila y columna

$$X^2_{calc} = \frac{(O_{11} - E_{11})^2}{E_{11}} + \frac{(O_{22} - E_{22})^2}{E_{22}} + \dots + \frac{(O_{rk} - E_{rk})^2}{E_{rk}}$$

$$X^2_{calc} = \frac{(83 - 46.5)^2}{46.5} + \frac{(0 - 36.5)^2}{36.5} + \frac{(10 - 46.5)^2}{46.5} + \frac{(73 - 36.5)^2}{36.5}$$

$$X^2_{calc} = 28.65 + 36.5 + 28.65 + 36.5$$

$$X^2_{calculado} = 130.3$$

Ahora el siguiente paso es determinar los grados de libertad (v) para lo cual se utiliza la siguiente fórmula:

$$v = (r - 1) * (k - 1)$$

Donde $r=N^\circ$ de filas y $k= N^\circ$ de columnas

Para este caso tenemos una matriz de $2r \times 2k$

$$v = (2 - 1) * (2 - 1)$$

$v = 1$ Grado de libertad

El nivel de significancia es el error que se puede cometer al rechazar la H_0 siendo verdadera, por lo general se trabaja con un 5%, es decir 0,05, que indica que hay una probabilidad (p) del 0,95 de que la H_0 sea verdadera.

Por lo tanto buscando en la tabla para valores de Chi-Cuadrado (Anexo C) el valor para X^2 de la tabla con 1 grado de libertad y un nivel de significancia de 0,05; generará un valor de:

$$X^2_{tabla} = 3,841$$

Criterio de decisión:

Se acepta H_0 cuando: $X^2_{calculado} \leq X^2_{tabla}$

Caso contrario se rechaza H_0 y se acepta H_1

Resultados de la investigación:

$$X^2_{calculado} = 130.3$$

$$X^2_{tabla} = 3,841$$

Al aplicar el criterio de decisión obtenemos:

$$X^2_{calculado} = 130.3 > X^2_{tabla} = 3,841$$

Al no cumplirse la condición se rechaza la hipótesis nula (H_0) y se acepta la hipótesis alternativa (H_1)

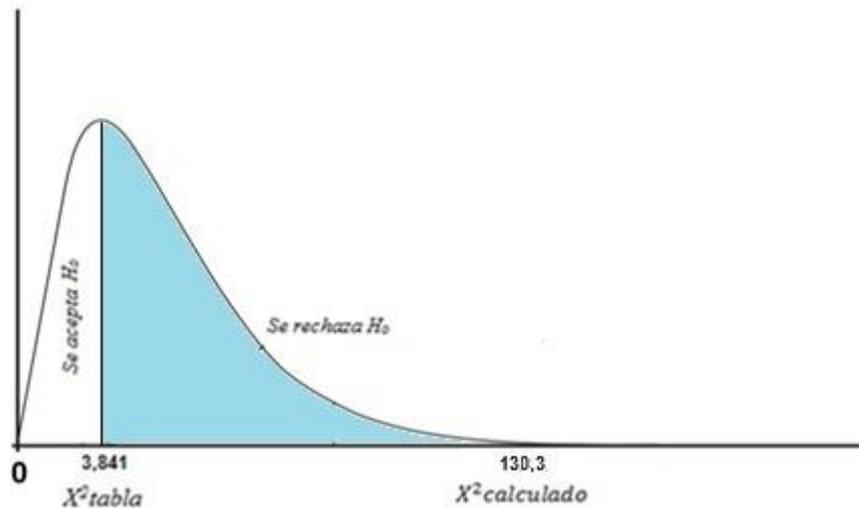


Gráfico 23-4: Distribución Chi Cuadrado

Realizado por: Monar Joffre, 2017

Interpretación:

De acuerdo a los datos obtenidos en el cálculo de X^2 tabla y X^2 calculado como se aprecia en la Gráfica 23-4 podemos llegar a la conclusión de que se rechaza la hipótesis nula (H_0) y se acepta la hipótesis alternativa (H_1), con un nivel de significancia del 5% para obtener un nivel de confianza del 95%.

Quedando demostrado que:

H_1 : “El método desarrollado propuesto si mejorará el nivel de seguridad de las aplicaciones web en entorno PHP ante posibles ataques informáticos”.

Por lo que la comprobación de la hipótesis por el método del X^2 permite identificar que si el Ministerio de Agricultura Ganadería Acuicultura y Pesca (MAGAP) implementa esta propuesta en la zona 3 mejorará el nivel de seguridad con el prototipo 2 (S-SISEV) ante posibles ataques informáticos.

CAPÍTULO V

5. PROPUESTA

5.1. Determinación de la propuesta

Una de las etapas del ciclo de vida de desarrollo de software es la de codificación y el estudio se centra en ella, dado que es la etapa en la que se presentan mayores vulnerabilidades, es la que requiere mayor profundidad y mayor cuidado en el desarrollo de un proyecto web.

Para esta etapa, se tomó como base el “proyecto abierto de seguridad en aplicaciones Web” (OWASP, Open Web Application Security Project), liderado por la fundación OWASP que es una entidad sin fines de lucro, es tomada como referente dado que es un proyecto abierto y por su amplia aceptación por la comunidad de desarrolladores web, la fundación está integrada por personas con un vasto conocimiento en los sistemas y más en lo que se refiere al tema de seguridad informática, dedicado a la búsqueda y lucha contra las causas de software inseguro; es una guía de consulta obligatoria para los desarrolladores de software.

Es por ello que se pretende establecer un método o guía que integra nuevos estudios para el desarrollo de código seguro en entorno PHP 5.6 y PHP 7 que son las últimas versiones disponibles con soporte total de usuarios.

5.2. Propuesta de un método para el desarrollo de aplicaciones web utilizando técnicas de programación segura en entornos PHP

Para definir esta propuesta se tomó como base las recomendaciones de: TOP-10 Controles Proactivos en su versión dos (2) del 2016, la Guía de Pruebas OWASP en su versión cuatro (4) del 2013, así como los Controles Estándar de Seguridad (ASVS) en su versión tres (3) del 2015; todo esto implementado con las técnicas de programación para las versiones 5.6 y 7 de PHP que actualmente tienen soporte.

5.2.1. Parametrizar las consultas

Inyección SQL es uno de los riesgos más peligrosos en una aplicación web, es fácil de explotar con muchas herramientas de ataque open source automatizadas.

La principal preocupación con la inyección de SQL es el hecho, de que la consulta SQL y sus parámetros están contenidos en una cadena de consulta.

Con el fin de mitigar la inyección de SQL, de entrada que no sea de confianza se debe impedir que se interprete como parte de un comando SQL. La mejor manera de hacerlo es con la técnica de programación conocida como "parametrización de consultas". En este caso, las sentencias SQL son enviadas y analizadas por el servidor de base de datos de forma independiente de los parámetros.

Ejemplo:

```
# Marcadores conocidos
$stmt = $dbh->prepare("update users set email=:new_email where id=:user_id");
$stmt->bindParam(':new_email', $email);
$stmt->bindParam(':user_id', $id);
```

Implementación:

1. Consulta (SELECT)

Ejemplo: Consulta con parámetros

```
$nombre = $_REQUEST["nombre"];
$apellidos = $_REQUEST["apellidos"];
$consulta = "SELECT COUNT(*) FROM $dbTabla
    WHERE nombre=:nombre
    AND apellidos=:apellidos";
$result = $db->prepare($consulta);
$result->execute(array(":nombre" =>$nombre, ":apellidos" =>$apellidos));
if (!$result) {
    print " <p>Error en la consulta.</p>\n";
    ...
}
```

Ejemplo: Consulta sin parámetros

```
$stmt = $pdo->query('SELECT name, addr, city from colegas');
# Indicamos en qué formato queremos obtener los datos de la tabla en formato de array
asociativo.
    $stmt->setFetchMode(PDO::FETCH_ASSOC);
# Si no indicamos nada por defecto se usará FETCH_BOTH lo que nos permitirá acceder como
un array asociativo o array numérico.

# Leemos los datos del recordset con el método ->fetch()
    while ($row = $stmt->fetch()) {
        echo $row['name'] . "<br/>";
        echo $row['addr'] . "<br/>";
        echo $row['city'] . "<br/>";
    }
# Para liberar los recursos utilizados en la consulta SELECT
    $stmt = null;
```

2. Insertar (INSERT)

```
$consulta = "INSERT INTO $dbTabla
(nombre, apellidos)
VALUES (:nombre, :apellidos)";
$result = $db->prepare($consulta);
if ($result->execute(array(":nombre" => $nombre, ":apellidos" => $apellidos))) {
    print " <p>Registro creado correctamente.</p>\n";
} else {
    print " <p>Error al crear el registro.</p>\n";
}
```

3. Actualizar (UPDATE)

// Consulta de modificación de registro

```
$consulta = "UPDATE $dbTabla
SET nombre=:nombre, apellidos=:apellidos
WHERE id=:id";
$result = $db->prepare($consulta);
if ($result->execute(array(":nombre" =>$nombre, ":apellidos" =>$apellidos, ":id" =>$id))) {
print" <p>Registro modificado correctamente.</p>\n";
} else {
print" <p>Error al modificar el registro.</p>\n";
}
```

4. Eliminar (DELETE)

// Consulta de borrado de registro

```
$consulta = "DELETE FROM $dbTabla
WHERE id=:indice";
$result = $db->prepare($consulta);
if ($result->execute(array(":indice" =>$indice))) {
print" <p>Registro borrado correctamente.</p>\n";
} else {
print" <p>Error al borrar el registro.</p>\n";
}
```

5. Restricciones en los parámetros

Si no podemos usar parámetros, no queda más remedio que incluir los datos en la consulta. Como en este caso PHP no hace ninguna desinfección de los datos, se tiene que hacer previamente.

Podemos crear una función de recogida de datos específica que impida cualquier tipo de ataque de inyección por parte del usuario, como muestra el siguiente ejemplo:

```
// función de recogida de un dato que sólo puede tomar valores determinados
$campos = array("nombre", "apellidos");
```

```
function recogeCampo($var, $var2)
{
global $campos;
foreach($campos as $campo) {
if (isset($_REQUEST[$var]) && $_REQUEST[$var] == $campo) {
return $campo;
}
}
return $var2;
}
```

```
// ejemplo de uso de la función anterior
$campo = recogeCampo("campo", "apellidos");
```

```

$nombre = $_REQUEST["nombre"];

$con consulta = "SELECT * FROM $dbTabla
WHERE nombre=:nombre
ORDER BY $campo ASC";
$result = $db->prepare($con consulta);
$result->execute(array(":nombre" =>$nombre));
if (!$result) {
print " <p>Error en la consulta.</p>\n";
...

```

Vulnerabilidades que previene: A1-Inyección

5.2.2. Codificar los datos

La codificación es un mecanismo poderoso para ayudar a proteger contra muchos tipos de ataques, especialmente los ataques de inyección. En esencia, la codificación consiste en la traducción de caracteres especiales en alguna forma equivalente que ya no es peligroso en el intérprete de destino. Otro ejemplo de codificación es la codificación de salida que es necesario para prevenir XSS (codificación de entidades HTML, codificación de Java Script, etc.).

Implementación:

1. Saneando el HTML

Lo primero que hacemos es asegurarnos de que hay un comentario, eliminando además todos los espacios en blanco que haya al inicio y al final del comentario; luego saneamos el comentario eliminando las etiquetas HTML que pudiera tener. Finalmente guardamos los comentarios y ya estarán preparados para ser mostrados al usuario.

```

$comentario=strip_tags($_POST['comentario']);

// Sanear comentario
$comentario=strip_tags($comentario);
/* Ahora podemos guardarlos*/
file_put_contents("comentarios.txt",$comentario,FILE_APPEND);
/* Escapamos los comentarios antes de mostrarlos */
$comentarios=file_get_contents("comentarios.txt");
echohtmlspecialchars($comentarios);

```

2. Configurar las cookies para que sean httponly y secure

Al configurar la cookie como secure sólo se intercambian entre el navegador y tu aplicación por https. Al marcarlas como httponly evitas que scripts accedan a la misma, limitando las posibilidades de los ataques vía cross-site scripting.

```
// Asegúrate de que la cookie de sesión no pueda accederse por javascript.  
$httponly = true;
```

Vulnerabilidades que previene: A1-Inyección y A3-XSS (en parte)

5.2.3. Validar todas las entradas

Cualquier dato que se introduce directamente, o influenciados por, los usuarios deben ser tratados como no fiables.

Una aplicación debe verificar que estos datos son tanto sintácticamente y semánticamente válidos (en ese orden) antes de usarlo en cualquier forma (incluyendo mostrar al usuario). Además, las aplicaciones más seguras tratan todas las variables como no fiables y proporcionan controles de seguridad independientemente de la fuente de los datos.

Fondo: La gran mayoría de vulnerabilidades de aplicaciones web derivan de no haber validado correctamente la entrada, o no validar completamente la entrada. Esta "entrada" no es necesariamente ingresada por usuarios usando una interfaz de usuario.

En el contexto de las aplicaciones web (y servicios web), esto podría incluir, pero no se limitan a:

- Cabeceras HTTP
- Las cookies
- Parámetros GET y POST (incluidos campos ocultos)
- Carga de archivos (incluyendo información como el nombre del archivo)

Implementación:

1. Expresiones regulares

Las expresiones regulares ofrecen una manera de comprobar si los datos coinciden con un patrón específico.

Patrón para el campo de contraseña:

```
^(?=.*[a-z])(?=.*[A-Z]) (?=.*\d) (?=.*[!@#$%]).{10,4000}$
```

Patrón para dirección email:

```
^[a-zA-Z0-9.!#$%&'*/+=?^_`{|}~-]+@[a-zA-Z0-9-]+(?:\.[a-zA-Z0-9-]+)*$
```

Ejemplo:

```
$_MensajeError="ERROR: caracteres no admitidos";
if (ereg("[^A-Za-z0-9]+",$_POST['consulta'])) {
echo $_MensajeError;
}
else{
echo '<div class="result">Tu nombre es: '.$consulta.'</div>';
}
}
```

2. Validación y Saneamiento

La extensión de filtro PHP contiene un conjunto de funciones que pueden ser utilizadas para validar la entrada del usuario, pero también para sanitizar (sanear) mediante la eliminación de caracteres no válidos.

También proporcionan una estrategia estándar para el filtrado de datos.

• Validación de direcciones IP

```
<?php
$ip_a = '127.0.0.1';
if (filter_var($ip_a, FILTER_VALIDATE_IP)) {
    echo "Esta dirección IP (ip_a) es válida.";
}
?>
```

- **Redirecciones y reenvíos no validados**

Nunca, jamás, se confíe de los datos que provienen del exterior de su aplicación PHP. Siempre sanee y verifique los datos de entrada antes de usarlos en su código. Las funciones `filter_var()` y `filter_input()` proporcionan saneamiento de los datos y verifican la validez del formato del texto.

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

- **Enlaces desde \$ _SERVER ['PHP_SELF']**

Nos muestra sólo enlaces que vienen de \$ _SERVER ['PHP_SELF'].

```
function esc_url($url) {
    if (" == $url) {
        return $url;
    }

    $url = preg_replace('[^a-z0-9-~+_.?#=!&,:% @$\|\*\(\)\x80-\xff]i', "", $url);
    $strip = array("%0d", "%0a", "%0D", "%0A");
    $url = (string) $url;

    $count = 1;
    while ($count) {
        $url = str_replace($strip, "", $url, $count);
    }

    $url = str_replace('/:/', '://', $url);
    $url = htmlentities($url);
    $url = str_replace('&', '&#038;', $url);
    $url = str_replace('"', '&#039;', $url);

    if ($url[0] !== '/') {
        // We're only interested in relative links from $_SERVER['PHP_SELF']
        return "";
    } else {
        return $url;
    }
}
```

- **Remover caracteres ASCII Con Valor > 127**

Se eliminará todas las etiquetas HTML y todos los caracteres con valor ASCII > 127, de la cadena:

```
<?php
$str = "<h1>Hello World!</h1>";

$newstr = filter_var($str, FILTER_SANITIZE_STRING, FILTER_FLAG_STRIP_HIGH);
echo $newstr;
?>
```

Vulnerabilidades que previene: A1-Inyección (en parte), A3-XSS (en parte) y A10-Redirecciones y reenvíos no validados

5.2.4. *Implementar controles de identidad y autenticación*

La autenticación es el proceso de verificar que un individuo o una entidad es quien dice ser. La autenticación se realiza con frecuencia mediante la presentación de un nombre de usuario o ID y uno o más elementos de información privada que sólo un determinado usuario debe saber.

Implementación:

1. Medidas durante la Autenticación

- **Prevenir ataques de fuerza bruta**

Este tipo de ataque es un método de ensayo y error utilizado para obtener información de una contraseña, clave o número de identificación personal, entre otros. Funciona mediante la generación de un gran número de intentos consecutivos para el valor de los datos deseados.

```
function checkbrute($user_id, $mysqli) {
// Obtiene el timestamp del tiempo actual.
    $now = time();
    // Todos los intentos de inicio de sesión se cuentan desde las 2 horas anteriores.
    $valid_attempts = $now - (2 * 60 * 60);
```

```

if ($stmt = $mysqli->prepare("SELECT time
    FROM login_attempts
    WHERE user_id = ?
    AND time > '$valid_attempts'")) {
    $stmt->bind_param('i', $user_id);

    // Ejecuta la consulta preparada.
    $stmt->execute();
    $stmt->store_result();

    // Si ha habido más de 5 intentos de inicio de sesión fallidos.
    if ($stmt->num_rows > 5) {
        return true;
    } else {
        return false;
    }
}
}

```

- **Captcha**

El código captcha se coloca antes del botón submit del formulario.

```

<?php
    // show captcha HTML using Securimage::getCaptchaHtml()
    require_once 'securimage.php';
    $options = array();
    $options['input_name'] = 'ct_captcha'; // change name of input element for form post
    $options['disable_flash_fallback'] = false; // allow flash fallback
    if (!empty($_SESSION['ctform']['captcha_error'])) {
        // error html to show in captcha output
        $options['error_html'] = $_SESSION['ctform']['captcha_error'];
    }
    echo "<div id='captcha_container_1'>\n";
    echo Securimage::getCaptchaHtml($options);
    echo "\n</div>\n";
?>

```

- **Recordar contraseña**

Al ingresar la contraseña es importante que se configure autocomplete en off.

Ejemplo: <INPUT TYPE="password" AUTOCOMPLETE="off">

2. Gestión de Sesiones

En cualquier autenticación correcta y re autenticación de software debería generar una nueva sesión y ID de sesión.

Con el fin de minimizar el período de tiempo que un atacante puede lanzar ataques durante las sesiones activas y secuestrarlas, es obligatorio establecer tiempos de caducidad para cada sesión, después de un período de inactividad. La longitud de tiempo de espera debería ser inversamente proporcional con el valor de los datos protegidos.

- **Inicio de sesión seguro**

```
<?php
include_once 'psl-config.php';
function sec_session_start() {
    $session_name = 'sec_session_id'; // Configura un nombre de sesión personalizado.
    $secure = SECURE;
    // Esto detiene que JavaScript sea capaz de acceder a la identificación de la sesión.
    $httponly = true;
    // Obliga a las sesiones a solo utilizar cookies.
    if (ini_set('session.use_only_cookies', 1) === FALSE) {
        header("Location: ../error.php?err=Could not initiate a safe session (ini_set)");
        exit();
    }
    // Obtiene los params de los cookies actuales.
    $cookieParams = session_get_cookie_params();
    session_set_cookie_params($cookieParams["lifetime"],
        $cookieParams["path"],
        $cookieParams["domain"],
        $secure,
        $httponly);
    // Configura el nombre de sesión al configurado arriba.
    session_name($session_name);
    session_start(); // Inicia la sesión PHP.
    session_regenerate_id(); // Regenera la sesión, borra la previa.
}
```

- **Estado de la sesión iniciada**

Agrega esta función a tu archivo “functions.php” en la carpeta “includes de tu aplicación:

```
function login_check($mysqli) {
    // Revisa si todas las variables de sesión están configuradas.
    if (isset($_SESSION['user_id'],
        $_SESSION['username'],
```

```

        $_SESSION['login_string'])) {

        $user_id = $_SESSION['user_id'];
        $login_string = $_SESSION['login_string'];
        $username = $_SESSION['username'];
        // Obtiene la cadena de agente de usuario del usuario.
        $user_browser = $_SERVER['HTTP_USER_AGENT'];

        if ($stmt = $mysqli->prepare("SELECT password
                                    FROM members
                                    WHERE id = ? LIMIT 1")) {
// Une "$user_id" al parámetro.
$stmt->bind_param('i', $user_id);
$stmt->execute(); // Ejecuta la consulta preparada.
$stmt->store_result();
        if ($stmt->num_rows == 1) {
// Si el usuario existe, obtiene las variables del resultado.
$stmt->bind_result($password);
        $stmt->fetch();
        $login_check = hash('sha512', $password . $user_browser);
        if ($login_check == $login_string) {
            // ¡¡Conectado!!
            return true;
        } else {
            // No conectado.
            return false;
        }
        } else {
            // No conectado.
            return false;
        }
        } else {
// No conectado.
        return false;
        }
        } else {
            // No conectado.
            return false;
        }
        }
    }
}

```

- **Cierre de sesión**

El código necesario a implementar dentro de un botón Cerrar Sesión es:

```

<?php
include_once 'includes/functions.php';
sec_session_start();
// Desconfigura todos los valores de sesión.
$_SESSION = array();
// Obtiene los parámetros de sesión.
$params = session_get_cookie_params();
// Borra el cookie actual.
setcookie(session_name(),
            "", time() - 42000,
            $params["path"],
            $params["domain"],
            $params["secure"],

```

```
$params["httponly"]);  
// Destruye sesión.  
session_destroy();  
header('Location: ../ index.php');
```

3. Falsificación de peticiones en sitios cruzados (CSRF)

Para evitar ataques CSRF en tus pedidos GET y POST recomiendo realizar lo siguiente:

- **Crea la clase y funciones**

```
<?php  
class csrf {  
public function get_token_id() {  
    if(isset($_SESSION['token_id'])) {  
        return $_SESSION['token_id'];  
    } else {  
        $token_id = $this->random(10);  
        $_SESSION['token_id'] = $token_id;  
        return $token_id;  
    }  
}  
public function get_token() {  
    if(isset($_SESSION['token_value'])) {  
        return $_SESSION['token_value'];  
    } else {  
        $token = hash('sha256', $this->random(500));  
        $_SESSION['token_value'] = $token;  
        return $token;  
    }  
}  
public function check_valid($method) {  
    if($method == 'post' || $method == 'get') {  
        $post = $_POST;  
        $get = $_GET;  
        if(isset(${$method}[$this->get_token_id()]) && (${$method}[$this->get_token_id()]  
== $this->get_token())) {  
            return true;  
        } else {  
            return false;  
        }  
    }  
}
```

```

    } else {
        return false;
    }
}
public function form_names($names, $regenerate) {
    $values = array();
    foreach ($names as $n) {
        if($regenerate == true) {
            unset($_SESSION[$n]);
        }
        $s = isset($_SESSION[$n]) ? $_SESSION[$n] : $this->random(10);
        $_SESSION[$n] = $s;
        $values[$n] = $s;
    }
    return $values;
}
private function random($len) {
    if (@is_readable('/dev/urandom')) {
        $f=fopen('/dev/urandom', 'r');
        $urandom=fread($f, $len);
        fclose($f);
    }
    $return="";
    for ($i=0;$i<$len;++$i) {
        if (!isset($urandom)) {
            if ($i%2==0) mt_srand(time()%2147 * 1000000 + (double)microtime() *
1000000);
            $rand=48+mt_rand()%64;
        } else $rand=48+ord($urandom[$i])%64;
        if ($rand>57)
            $rand+=7;
        if ($rand>90)
            $rand+=6;
        if ($rand==123) $rand=52;
        if ($rand==124) $rand=53;
        $return.=chr($rand);
    }
    return $return;
}
}
}

```

- **Protege un formulario POST**

El código a continuación muestra cómo implementar la clase CSRF en un formulario.

```
<?php
session_start();
include 'csrf.class.php';
$csrf = new csrf();
// Genera un identificador y lo valida
$token_id = $csrf->get_token_id();
$token_value = $csrf->get_token($token_id);
// Genera nombres aleatorios para el formulario
$form_names = $csrf->form_names(array('user', 'password'), false);
if(isset($_POST[$form_names['user']], $_POST[$form_names['password']])) {
// Revisa si el identificador y su valor son válidos.
if($csrf->check_valid('post')) {
    // Get the Form Variables.
    $user = $_POST[$form_names['user']];
    $password = $_POST[$form_names['password']];
// La función Form va aquí
}
// Regenera un valor aleatorio nuevo para el formulario.
$form_names = $csrf->form_names(array('user', 'password'), true);
}
?>

<form action="index.php" method="post">
<input type="hidden" name="<?=$token_id;?>" value="<?=$token_value;?>" />
<input type="text" name="<?=$form_names['user'];?>" /><br/>
<input type="text" name="<?=$form_names['password'];?>" />
<input type="submit" value="Login"/>
</form>
```

Vulnerabilidades que previene: A2-Pérdida de autenticación y gestión de sesiones, A8-Falsificación de peticiones en sitios cruzados (CSRF)

5.2.5. *Proteger los datos*

1. **Algoritmo de cifrado**

Este archivo es una implementación en Java Script del algoritmo hash sha512. Haremos uso de la función hash para que las contraseñas no se envíen en texto simple.

2. **Cifrado de contraseñas**

- **Hash de contraseñas**

Con el fin de proporcionar fuertes controles de autenticación, una aplicación con seguridad debe almacenar credenciales de usuario. Por otra parte, los controles criptográficos deben estar en su lugar de manera que si se ve comprometida una credencial (por ejemplo, una contraseña), el atacante no tiene acceso inmediato a esta información.

Ejemplo: Proceso para registrar una contraseña con hash

1. Verifico que haya ingresado todos los campos del login
if (isset(\$_POST['username'], \$_POST['password']))
2. Valido y sanitizo cada uno de los campos
3. Asigno a una variable el password ya sanitizado
\$password = filter_input(INPUT_POST, 'p', FILTER_SANITIZE_STRING);
4. Se crea la variable salt con un hash de openssl
\$random_salt = hash('sha512', uniqid(openssl_random_pseudo_bytes(16), TRUE));
5. Se crea el hash del password con el salt obtenido
\$password = hash('sha512', \$password . \$random_salt);
6. Ingreso en la tabla usuarios los datos como: usuario, password y salt

Ejemplo: Proceso para verificar contraseña que ingresa en el login vs la que está encriptada en la base de datos

1. Llamo a la función login
function login(\$email, \$password, \$mysqli)
2. Dentro de la function se hace un hash con el password recibido del login mas la salt obtenida de la bd
\$password = hash('sha512', \$password . \$salt);
3. Chequea si el password en la bd coincide con el password del usuario enviado en el login
if (\$db_password == \$password)
4. Si coincide re direcciona a la página principal caso contrario vuelve a la página de login

Vulnerabilidades que previene: A6-Exposición de datos sensibles

5.2.6. *Error y control de excepciones*

Es una parte importante de la codificación defensiva, crítica para hacer un sistema confiable, así como seguro.

Errores en el manejo de errores pueden conducir a diferentes tipos de vulnerabilidades de seguridad:

- Dando información a los atacantes: Ayudándoles a entender más acerca de su plataforma y diseño. Devolver los diferentes tipos de errores en diferentes situaciones (por ejemplo, "usuario no válido" o "contraseña no válida" en los errores de autenticación) también puede ayudar a los atacantes a encontrar su camino.
- No comprobando errores: Lo que conlleva a que los errores no se detecten, o a resultados impredecibles.

Las cuidadosas revisiones de código y pruebas negativas, fuzzing y fallas de inyección pueden ayudar a encontrar problemas en el manejo de errores.

Implementación:

- Administrar las excepciones de forma centralizada para evitar duplicar bloques try / catch en el código y para asegurarse de que todos los comportamientos inesperados se manejan correctamente dentro de la aplicación.
- Asegúrese de que los mensajes de error que se muestran a los usuarios no generen fugas de datos críticos, pero sean lo suficientemente detalladas como para explicar el problema al usuario.
- Asegúrese de que las excepciones se registren de una manera que proporcione suficiente información para entender el problema.

1. Respuestas try/catch

Se recomienda activar esta opción para gestionar los errores con PDOException.

```
$pdo->setAttribute( PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);  
try {  
# Otro Ejemplo de error ! DELECT en lugar de SELECT!  
$pdo->exec('SELECT name FROM people');
```

```

}
catch(PDOException $e){
echo"Se ha producido un error en la ejecucion de la consulta: ".$e->getMessage();

# En este caso hemos mostrado el mensaje de error y además almacenamos en un fichero los
errores generados.
file_put_contents('PDOErrors.txt',$e->getMessage(), FILE_APPEND);
}

```

2. Respuestas de autenticación

Una aplicación debería responder mensajes de error genéricos independientemente de si era incorrecto el identificador de usuario o la contraseña.

La respuesta correcta no debería indicar si el identificador de usuario o la contraseña es el parámetro incorrecto y por lo tanto, inferir un identificador de usuario válido.

Ejemplo:"Falló el inicio de sesión: Usuario o contraseña incorrectos"

Vulnerabilidades que previene: Todas las Top-10

5.2.7. Configuración adecuada

Asegúrese de que una aplicación cumpla los siguientes requisitos:

- Bibliotecas y plataforma (s) actualizadas.
- Un seguro de configuración por defecto.
- Endurecimiento suficiente de los cambios iniciados por el usuario para una configuración por defecto, no exponer innecesariamente o crear debilidades de seguridad o defectos de los sistemas subyacentes.

Implementación:

1. Instala un servidor web, PHP y MySQL en tu servidor

La mayoría de los servicios de alojamiento tiene PHP y MySQL ya instalados, pero tendrás que revisar que tengan las versiones más recientes de PHP y MySQL para que esta guía pueda serte de ayuda recomendando tener instalado al menos APACHE 2.4, PHP5.6 y MySQL5. La forma cómo instalar y configurar lo encuentras en los Anexos D, E y F.

- **Mantener tu software actualizado es parte del proceso de seguridad.**

Si tienes tu propio servidor o computadora, deberás instalar el software requerido normalmente según tu sistema. En general, si no vas a usar la configuración por motivos de producción y vas a desarrollar en Windows o Linux, instalar un paquete de aplicaciones “stack” como XAMPP será lo más recomendable; consigue la versión apropiada para tu sistema operativo y de preferencia la última estable.

Recomendación:

Ten presente que bajo ninguna circunstancia deberás utilizar XAMPP para crearte un ambiente de servidor de producción ya sea para Windows o Linux.

En su lugar debes instalar por separado (servidor web, base de datos y lenguaje de programación) y de preferencia en un ambiente Linux en su última versión estable.

2. Seguridad en la base de datos

- **Configuración de la Base de datos MYSQL**

No se debe configurar la BD con las opciones por defecto, asegúrate de cumplir al menos con lo siguiente:

- Asegúrate de configurar MySQL con una contraseña de raíz segura.
- Crea un usuario solo con los privilegios SELECCIONAR, ACTUALIZAR e INSERTAR.

```
CREATE USER 'sec_user'@'localhost' IDENTIFIED BY 'eKcGZr59zAa2BEWU';  
GRANT SELECT, INSERT, UPDATE ON `basse_datos`.* TO 'sec_user'@'localhost';
```

- **Conexión con la base de datos**

Se establecerá la función de conexión a la BD así como la estructura para la utilización con otros motores.

```
// biblioteca_mysql.php  
// Función de conexión con la base de datos MySQL
```

```

function conectaDb()
{
try {
$pdo=new PDO("mysql:host=$host;dbname=$dbname;charset=utf8",$user,$pass);
$pdo->setAttribute( PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
$db->exec("set names utf8mb4");
return($db);
}
catch(PDOException $e){
echo"Se ha producido un error al intentar conectar al servidor MySQL: ".$e->getMessage();
exit();
}
}
// La conexión se debe realizar en cada página que acceda a la base de datos
require_once "biblioteca.php";
$db = conectaDB();

// biblioteca.php, donde se define los diferentes motores de BD
define("MYSQL", "MySQL"); // Base de datos MySQL
define("SQLITE", "SQLite"); // Base de datos SQLITE
$dbMotor = SQLITE; // Base de datos empleada (MYSQL o SQLITE)
if ($dbMotor == MYSQL) {
require_once "biblioteca_mysql.php";
} elseif ($dbMotor == SQLITE) {

require_once "biblioteca_sqlite.php";
}

```

- **Desconexión con la base de datos**

Para cerrar la conexión con la BD se establece:

```
$db = null;
```

- **Excepciones**

PDO puede utilizar las excepciones para gestionar los errores, lo que significa que cualquier cosa que hagamos con PDO podríamos encapsularla en un bloque **try/catch** para gestionar si produce algún error.

```

// Se recomienda activar esta opción para gestionar los errores con PDOException
$pdo->setAttribute( PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

```

3. Archivos de configuración

Cuando esté trabajando con archivos de configuración para su aplicación, las mejores prácticas dictan que utilice uno de los métodos siguientes:

- Es recomendable que almacene sus archivos de configuración donde no se pueda acceder a ellos directamente por medio del sistema de archivos.
- Si no tiene otra alternativa más que almacenar sus archivos de configuración en la raíz de documentos de su aplicación, debe adjuntar la extensión.php al nombre de sus archivos.
- La información en los archivos de configuración debe ser protegida ya sea por medio de codificación o con los permisos de grupo.

4. Reporte de errores

Puede exponer información acerca de la estructura de su aplicación al exterior. Para proteger efectivamente su aplicación se necesita configurar su servidor de desarrollo de diferente manera que su servidor de producción.

Desarrollo

Configure las siguientes opciones en su archivo php.ini:

```
display_errors: On  
error_reporting: E_ALL  
log_errors: On
```

Producción

Configure su archivo php.ini de la siguiente manera:

```
display_errors: Off  
error_reporting: E_ALL  
log_errors: On
```

Con estas opciones en su entorno de producción, los errores seguirán siendo registrados en los registros de errores de su servidor web, pero no serán mostrados al usuario.

5. Estar al tanto de las actualizaciones de los productos utilizados

Una aplicación está sustentada en una serie de componentes software. Todos los días se descubren vulnerabilidades que afectan a esos componentes y se publican actualizaciones para parchearlas. Si no aplicamos esas actualizaciones tenemos

componentes con vulnerabilidades conocidas y publicadas, por ejemplo: si un atacante averigua que en una aplicación utiliza PHP con una versión $\leq 5.5.X$ (actualmente sin soporte) conoce las vulnerabilidades no parcheadas desde el fin de soporte de estas versiones y por lo tanto puede explotarlas.

Recomendación:

Utilizar cualquiera de las siguientes versiones de PHP en el desarrollo de aplicaciones web:

- PHP 5.6.X con soporte para errores generales y de seguridad hasta el 2017.
- PHP 7.0.X con soporte total hasta el 2018.

Aunque por estabilidad, compatibilidad y soporte recomiendo PHP 5.6.X, por lo que todos los proyectos deben ser actualizados a esta versión, tomando en cuenta que puede variar parte de la programación debido a cambios propios de la versión.

6. Comprobar la información que se obtiene de las cabeceras HTTP

Las cabeceras HTTP suelen aportar más información de lo deseable acerca del sistema operativo, versiones de software, productos, etc. Para ver las cabeceras se puede utilizar Nmap y Netcat:

```
nmap -O -Pn -n NOMBRE_DEL_HOST  
nc -w 10 NOMBRE_DEL_HOST 80
```

Se puede determinar tanto el servidor (cabecera “Server”) como la versión de PHP utilizada (cabecera “X-Powered-By”), de la que se podría deducir el repositorio del que se instaló y el sistema operativo del servidor. La segunda cabecera es la que más preocupa, ya que sabiendo la versión de PHP y el sistema operativo utilizado en el servidor es más sencillo explotar vulnerabilidades.

Recomendación:

Desactivar esta cabecera es tan sencillo como cambiar una línea del archivo de configuración php.ini; en la que basta con buscar la línea “expose_php=On” y cambiarlo a “**expose_php=Off**”, tras ello hay que reiniciar el servidor web.

7. Verificar los servicios visibles

Se debe comprobar que sólo se está exponiendo los servicios deseados (típicamente, el servidor web por los puertos 80 y 443). Para comprobarlo se puede usar Nmap:

```
sudo nmap -sV -n -Pn -p 80,443 NOMBRE_DEL_HOST
```

Recomendación:

Bloquear el acceso a todos los puertos que no sean necesarios para la implementación del proyecto web, esto se puede lograr mediante el establecimiento de reglas dentro del firewall.

8. Buscar vulnerabilidades

Utilizando herramientas más avanzadas como Nikto y W3AF se puede automatizar la búsqueda de vulnerabilidades e información interesante:

```
nikto -h NOMBRE_DEL_HOST
```

Hay que tener en cuenta que herramientas como Nikto o W3af realizan muchas peticiones HTTP para tratar de obtener información.

Recomendación:

Es recomendable tener instalado algún Web Application Firewall o al menos un plugin que bloquee las direcciones IP que realicen demasiadas peticiones en poco tiempo.

9. Revisar la configuración de HTTPS

Aunque todos deberíamos usar HTTPS, en muchas ocasiones aceptamos la configuración por defecto del software que utilizemos sin dedicarle mayor tiempo.

Recomendación:

Conviene prestarle atención a aspectos como:

- En la utilización del certificado usar SHA-2, además el certificado debe ser de 2048b.
- Usar las cabeceras STS.
- Evitar las redirecciones HTTP -> HTTPS, ya que son vulnerables a SSL Strip.
- Asegurarse de que el certificado utilizado es designado como confiable por los navegadores.

CONCLUSIONES

- Para la demostración de la variable independiente (Método propuesto de programación segura para desarrollo de aplicaciones web en PHP) se realizó una encuesta a ocho (8) administradores del MAGAP-Zona 3 sobre la seguridad que brinda los sistemas SISEV (prototipo 1) y S-SIVEV (prototipo 2), dando como resultado que el 97% considera que el prototipo 2 (al que se aplicó el método) proporciona mayor seguridad ante las vulnerabilidades más comunes, que contrasta con el 75% que consideró que el prototipo 1 brindaba poca seguridad.
- Para la demostración de la variable dependiente (Mejora en el nivel de seguridad) se escaneó los dos (2) prototipos mediante el analizador de vulnerabilidades ACUNETIX, el cual detectó para el prototipo 1 un total de 83 vulnerabilidades vs el prototipo 2 que detectó un total de 10, es decir se redujo las vulnerabilidades en el segundo prototipo en un 87.95%.
- Se desarrolló un método con técnicas de programación para PHP 5.6.x que contempla siete (7) etapas: Parametrización de consultas, Codificación de datos, Validación de todas las entradas, Implementación de controles de identidad y autenticación, Protección de los datos, Error y control de excepciones y Configuración adecuada, las cuales se tomaron como base y fueron adaptadas de las diferentes guías, manuales, controles, normas, recomendaciones en sus últimas versiones del Proyecto OWASP.
- El método propuesto evita algunos de los errores más comunes en programación 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; los cuales derivan en la selección de siete (7) de las diez (10) vulnerabilidades más críticas del TOP-TEN-OWASP que fueron tomadas de forma no probabilística.

- Con la utilización de Chi-Cuadrado y un nivel de significancia del 0.05, de acuerdo a la tabla de distribución se obtuvo que $X^2_{tabla} = 3,841$ y $X^2_{calculado} = 130.3$, dando como resultado que se rechaza la hipótesis nula H_0 y se acepta la hipótesis alternativa H_1 , es decir que “El método desarrollado propuesto si mejorará el nivel de seguridad de las aplicaciones web en entorno PHP ante posibles ataques informáticos”.

RECOMENDACIONES

- Considerar las recomendaciones y técnicas de programación desarrolladas en el método para la construcción de aplicaciones web en entornos PHP para quienes desarrollan con código propio y revisar nuevas guías y controles que surjan del Proyecto OWASP que nos permitan mejorar la seguridad de nuestro aplicativo.
- Para quienes programan con framework, utilizar las librerías o funciones del método que se acoplen a su proyecto para brindarle una mejor seguridad al desarrollo de sus aplicaciones.
- Desarrollar nuevas técnicas de programación en PHP para cubrir vulnerabilidades que no fueron consideradas como: Referencia directa insegura a objetos (A4), Ausencia de control de acceso a funciones (A7), Utilización de componentes con vulnerabilidades conocidas (A9); con el objetivo de ampliar el método y mejorar aún más la seguridad de las aplicaciones ya que cubre las diez (10) vulnerabilidades más críticas de OWASP.
- Se propone para trabajos futuros:
 - Desarrollar nuevas técnicas de programación que nazcan del análisis del método para PHP 7.0 que es la última versión con soporte para el desarrollo de aplicaciones web.
 - Investigar nuevas técnicas de programación en otros lenguajes como: .NET o Java, que son muy utilizados para la construcción de aplicaciones web, que mitiguen al menos las diez (10) vulnerabilidades más críticas de OWASP.

GLOSARIO

Apache: Es un software libre servidor HTTP de código abierto para plataforma Unix (BSD, GNU/Linux, etc.), Windows, Macintosh y otras, que implementa el protocolo HTTP/1.1, y la noción del sitio virtual.

Autenticación: Es el proceso de verificación de la identidad o localización de un usuario, servicio o aplicación. La autenticación se realiza utilizando al menos uno de tres mecanismos: "algo que tienes", "algo que sé" o "algo que es". La aplicación puede autenticar y proporcionar diferentes servicios basados en la ubicación, método de acceso, el tiempo de permanencia y los hábitos de uso de la aplicación web.

Autorización: La determinación de los recursos que un usuario, un servicio o aplicación tienen como permisos de acceso. Recursos accesibles que pueden ser URL, archivos, directorios, bases de datos, servlets, caminos de ejecución.

Brute Force: Un proceso automatizado de prueba y error utilizado para adivinar el "secreto" de la protección de un sistema. Ejemplos de estos secretos son nombres de usuario, contraseñas o claves criptográficas.

Caracteres considerados peligrosos: Cualquier carácter o representación codificada de un carácter que puede afectar la operación intencionada de la aplicación o sistema asociado por ser interpretado de una manera especial, fuera del uso intencionado del carácter.

Codificación de Salida: Conjunto de controles que apuntan al uso de una codificación para asegurar que los datos producidos por la aplicación son seguros.

Confidencialidad: Se garantiza que la información sea accesible sólo a aquellas personas autorizadas a tener acceso a la misma.

Configuración del sistema: Conjunto de controles que ayuda a asegurar que los componentes de infraestructura que brindan soporte al software fueron desplegados de manera segura.

Consultas Parametrizadas: Mantiene la consulta y los datos separados a través del uso de marcadores. La estructura de la consulta es definida utilizando marcadores, la consulta SQL es enviada a la base de datos y preparada, para luego ser combinada con los valores de los parámetros. Esto previene a las consultas de ser alteradas debido a que los valores de los parámetros son combinados con la consulta compilada y con el string de SQL.

Control de Acceso: Un conjunto de controles que permiten o niegan el acceso a un recurso de un usuario o entidad dado.

Cookie: Pequeña cantidad de datos enviados por el servidor web, a una web de un cliente, que puede ser almacenada y recuperada en un momento posterior. Típicamente se utilizan cookies para realizar un seguimiento de un estado de los usuarios a medida que usan, navegan o interactúan con un sitio web.

Cross-Site Scripting:(acrónimo -XSS): Una técnica de ataque que obliga a un sitio web para hacerse eco de los datos suministrados por el cliente, que se ejecutan en un navegador web por parte del usuario. Cuando un usuario interactúa entre varios sitios y aplicaciones que están haciendo eco de datos entre sí, el atacante tendrá acceso a todo el contenido navegador web (cookies, historial, versión de la aplicación, etc).

Escáner de vulnerabilidades: Aplicación que permite comprobar si un sistema es vulnerable a un conjunto de deficiencias de seguridad.

Exploit: Aplicación, generalmente escrita en C o ensamblador, que fuerza las condiciones necesarias para aprovecharse de un error de programación que permite vulnerar su seguridad.

Desbordamiento de Buffer: Una técnica de explotación que altera el flujo de una aplicación sobrescribiendo partes de la memoria.

Disponibilidad: Se garantiza que los usuarios autorizados tengan acceso a la información y a los recursos relacionados con la misma, toda vez que lo requieran

Información: Es uno de los activos más importantes de las instituciones, en las formas que esta se manifieste: textuales, numéricas, gráficas, cartográficas, narrativas o audiovisuales, y en cualquier medio, magnético, papel, electrónico, computadoras, audiovisual y otros.

Falsificación de petición en sitios cruzados (CSRF): Una aplicación externa o sitio web fuerza a un cliente a realizar un pedido a otra aplicación en la que el cliente posee una sesión activa. Las Aplicaciones son vulnerables cuando utilizan parámetros o URLs predecibles o conocidas y cuando el navegador transmite automáticamente toda la información de sesión con cada pedido a la aplicación vulnerable.

Gestión de Archivos: Conjunto de controles que cubren la interacción entre el código y otro sistema de archivos.

Gestión de memoria: Conjunto de controles de direccionamiento de memoria y uso de buffers.

Gestión de sesión: Conjunto de controles que ayudan a asegurar que la aplicación web maneja las sesiones HTTP de forma segura.

Impacto: Medida del efecto negativo en el negocio que resulta de la ocurrencia de un evento indeseado; pudiendo ser el resultado la explotación de una vulnerabilidad.

Insuficiente Autenticación: Cuando un sitio web permite a un atacante acceder a contenido sensible o a sus funcionalidades sin verificar su identidad.

Insuficiente Autorización: Cuando un sitio web permite a un atacante acceder a contenido sensible o a sus funcionalidades que deberían requerir mayores restricciones de acceso y control.

Integridad: los datos reflejen la realidad y que correspondan con lo que debe ser y no ha sido modificadas indebidamente.

Java Script: Lenguaje de scripting del lado cliente utilizado por el navegador web, para crear contenido dinámico página web.

LDAP Inyección: Una técnica para la explotación de un sitio web mediante la alteración de backend sentencias LDAP a través de la manipulación de entrada de la aplicación. Al igual que en la metodología de inyección SQL.

Manejo de Errores: Conjunto de prácticas que aseguran que las operaciones de manejo de errores se manejen correctamente.

Manipulación Cookie: La alteración o modificación de los valores de las cookies del navegador web del cliente, para explotar aspectos de seguridad dentro de una aplicación web. Los atacantes normalmente manipulan los valores de los cookies para autenticarse y realizar fraudes a la aplicación web.

Manipulación de Nombre de Archivo: Una técnica de ataque usada para explotar sitios web mediante la manipulación de los nombres de archivo de URL para provocar errores de aplicación, descubrir el contenido oculto, o visualizar el código fuente de una aplicación.

Mitigar: Pasos tomados para reducir la severidad de una vulnerabilidad. Estos pueden incluir remover una vulnerabilidad, hacer una vulnerabilidad más difícil de explotar, o reducir el impacto negativo de una explotación exitosa.

MySQL: Es un sistema de gestión de base de datos relacional y multiusuario ubica las tablas en ficheros diferenciados, es recomendable para desarrollos que necesiten manejar numerosos registros y sesiones simultáneas.

NMAP: Programa de código abierto que abierto que sirve para efectuar rastreo de puertos TCP y UDP. Se usa para evaluar la seguridad de sistemas informáticos así como para descubrir servicios o servidores en una red informática.

OWASP: (The Open Web Application Security Project).

PHP: Es Un lenguaje de programación interpretado, diseñado originalmente para la creación de páginas web dinámicas. Es usado principalmente en interpretación del lado del servidor pero actualmente puede ser utilizado desde una interfaz de línea de comandos o en la creación de otros tipos de programas incluyendo aplicaciones con interfaz grafica.

Prácticas Criptográficas: Conjunto de controles que aseguran que las operaciones de criptografía dentro de la aplicación son manejadas de manera segura.

Prácticas de Codificación Generales: Conjunto de controles que cubren las prácticas de codificación que no son parte otras categorías.

Protección de datos: Conjunto de controles que ayudan a asegurar que el software maneja de forma segura el almacenamiento de la información.

Requerimiento de Seguridad: Conjunto de requerimientos funcionales y de diseño que ayudan a asegurar que el software se construye y despliega de forma segura.

Sanitizar: El proceso de hacer seguros datos potencialmente peligrosos a través de la utilización de remoción, reemplazo, codificación de los caracteres que lo componen.

Seguridad de Base de Datos: Conjunto de controles que aseguran la interacción del software con la base de datos de una forma segura y que la base de datos se encuentra configurada de forma segura.

Servidor de aplicaciones: Un servidor de software, normalmente a través de HTTP, que tiene la capacidad de ejecutar aplicaciones web dinámicas. También se conoce como middleware

Sistema: Término genérico que cubre sistemas operativos, servidores web, frameworks de aplicaciones e infraestructura relacionada.

Validación de entrada: Conjunto de controles que verifican que las propiedades de los datos ingresados coinciden con las esperadas por la aplicación, incluyendo tipos, largos, rangos, conjuntos de caracteres aceptados excluyendo caracteres peligrosos conocidos.

Validación de proceso insuficiente: cuando un sitio web permite a un atacante para eludir o evadir el control de flujo previsto de una aplicación.

Vulnerabilidad: Debilidad en un sistema que lo hace susceptible a ataque o daño.

BIBLIOGRAFÍA

- [1]. **Al-Ibrahim, M. & Al-Deen, Y.** (2014). *The Reality of Applying Security in Web Applications in Education*. DOI: 10.1109/SAI.2014.6918307.
- [2]. **Areito, J.** (2008). *Seguridad de la Información. Redes, informática y sistemas de información* . España: Paraninfo.
- [3]. **Ascencio, M. y Moreno, P.** (2011). *Desarrollo de una Propuesta Metodológica para determinar la seguridad en una aplicación web*. Colombia: Universidad Tecnológica de Pereira.
- [4]. **Astudillo, K.** (2013). *Hacking Ético 101 – Cómo hackear profesionalmente en 21 días o menos*. Ecuador: Createspace Independent Pub.
- [5]. **Blogfactores.** (2016). *Guía de configuración de seguridad en Centos 7*. Obtenido de <https://blogfactores.wordpress.com/2016/03/30/seguridad-en-centos-configuracion-elemental/>
- [6]. **Culoccioni, S.** (2015). *Escanear vulnerabilidades en servidores web con Nikto*. Obtenido de <https://www.solvetic.com/tutoriales/article/2273-escanear-vulnerabilidades-en-servidores-web-con-nikto/>
- [7]. **Elwebmaster.com** (2009). *¿Tu PHP es seguro? Tips y herramientas para asegurar tu sitio*. Obtenido de <http://www.elwebmaster.com/articulos/%C2%BFtu-php-es-seguro-tips-y-herramientas-para-asegurar-tu-sitio>
- [8]. **Enríquez, J.** (2011). *Políticas de seguridad informática y la vulnerabilidad de los entornos web de la Empresa Turbotech durante el año 2010*.
- [9]. **Lopez, F.** (2015). *Metodologías para el Desarrollo de Software Seguro*. España: Universidad de Cataluña.
- [10]. **Mateu, C.** (2010). *Desarrollo de Aplicaciones Web*. Barcelona: Eureka Media.

- [11]. **OWASP.** (2015). *Application Security Verification Standard 3.0*. Obtenido de <https://www.owasp.org/images/6/67/OWASPApplicationSecurityVerificationStandard3.0.pdf>
- [12]. **OWASP.** (2013). *CODE REVIEW GUIDE 2.0 RELEASE*. Obtenido de https://www.owasp.org/images/7/78/OWASP_AlphaRelease_CodeReviewGuide2.0.pdf
- [13]. **OWASP.** (2008). *GUÍA DE PRUEBAS OWASP V3*. Obtenido de https://www.owasp.org/images/8/80/Gu%C3%ADa_de_pruebas_de_OWASP_ver_3.0.pdf
- [14]. **OWASP.** (2013). *Los diez riesgos más críticos en Aplicaciones Web*. Obtenido de https://www.OWASP.org/images/5/5f/OWASP_Top_10_-_2013_Final_-_Espa%C3%B1ol.pdf
- [15]. **OWASP.** (2011). *OWASP Secure Coding Practices*. Obtenido de https://www.owasp.org/images/a/aa/OWASP_SCP_Quick_Reference_Guide_SPA.pdf
- [16]. **OWASP.** (2015). *Owasp Testing Guide 4.0*. Obtenido de <https://www.owasp.org/images/1/19/OTGv4.pdf>
- [17]. **OWASP.** (2016). *OWASP Top 10 Proactive Controls 2016*. Obtenido de https://www.owasp.org/images/5/57/OWASP_Proactive_Controls_2.pdf
- [18]. **Perez, E.** (2014). *Estudio de las características de seguridad de servidores web en entornos de Software Libre aplicables a la protección de sitios dinámicos*. Ambato: Universidad Técnica de Ambato.
- [19]. **Salgado, A.** (2014). *Análisis de las aplicaciones web de la Superintendencia de Bancos y Seguros*. Sangolqui: Universidad de las Fuerzas Armadas ESPE.
- [20]. **Techoism.** (2016). *how-to-upgrade-php-version-5-4-to-5-6-on-centosrhel*. Obtenido de <http://www.techoism.com/how-to-upgrade-php-version-5-4-to-5-6-on-centosrhel/>
- [21]. **UNAM-CERT.** (2016). *Aspectos Básicos de la Seguridad en Aplicaciones Web*. Obtenido de <http://www.seguridad.unam.mx/documento/?id=17>

- [22]. **Vilar, J.** (2013). *Diseño de un Framework para el diseño y desarrollo rápido de aplicaciones web en PHP, Java script y MySQL para entornos emprendedores.* España: Universidad Politécnica de Valencia.
- [23]. **Waisen, J. & Pérez, F.** (2012). *WEB VULNERABLE DVWA.* Almeria: Universidad de Almería.
- [24]. **Websetnet.** (2015). *how-to-migrate-mysql-to-mariadb-on-linux.* Obtenido de <https://websetnet.com/es/how-to-migrate-mysql-to-mariadb-on-linux/>
- [25]. **Yáñez, E.** (2014). *Guía de buenas prácticas de desarrollo de aplicaciones web seguras aplicado al sistema control de nuevos aspirantes Empresa Grupo LAAR.* Riobamba: ESPOCH.

ANEXOS

ANEXO A. ENCUESTAS APLICADAS A LOS ADMINISTRADORES DEL SISTEMA

PROTOTIPO 1: SISEV



MINISTERIO DE AGRICULTURA GANADERÍA ACUACULTURA Y PESCA (MAGAP)

ENCUESTA N° 1

Dirigida a: Administradores locales del sistema SISEV en la zonas 3

Objetivo: La encuesta tiene como finalidad recabar información acerca de la seguridad del sistema SISEV que funciona en cada una de sus Direcciones Provinciales.

Su opinión es relevante y contribuirá al desarrollo del trabajo de titulación titulado "PROPUESTA DE UN MÉTODO UTILIZANDO TÉCNICAS DE PROGRAMACIÓN SEGURAS PARA EL DESARROLLO DE APLICACIONES WEB EN ENTORNO PHP PARA MITIGAR RIESGOS POTENCIALES DE SEGURIDAD", por lo que es muy importante que responda con honestidad.

Nombre del encuestado:

Henry Velasco

DPA a la que pertenece:

Japa. Pastaza.

Fecha: 18/01/2017

PREGUNTAS

1. ¿El sistema valida todas las entradas en los formularios establecidos?
SI----- NO
2. ¿Considera que el sistema realiza una gestión de sesión adecuada?
SI----- NO
3. ¿Existe un cifrado de datos adecuado para el manejo de contraseñas?
SI----- NO
4. ¿El sistema maneja los errores y control de excepciones de manera segura?
SI NO-----
5. ¿Considera que el sistema se encuentra en un servidor que mantiene una configuración de seguridad adecuada para el servidor web, la base de datos y el lenguaje PHP?
SI NO-----
6. ¿Cree usted que la autenticación en el sistema está bien definida?
SI----- NO
7. ¿Considera que se está aplicando en el sistema una autorización adecuada para áreas de acceso más restringidas?
SI----- NO
8. ¿Considera usted que las técnicas de programación utilizadas en el sistema ayudarán a minimizar los ataques web?
SI----- NO

Henry Velasco
Firma



PROTOTIPO 2: S-SISEV



MINISTERIO DE AGRICULTURA GANADERÍA ACUACULTURA Y PESCA (MAGAP)

ENCUESTA N° 2

Dirigida a: Administradores locales del sistema SISEV en las zonas 3

Objetivo: La encuesta tiene como finalidad recabar información acerca de la seguridad del nuevo sistema S-SISEV que se encuentra en etapa de pruebas para su implantación.

Su opinión es relevante y contribuirá al desarrollo del trabajo de titulación titulado "PROPUESTA DE UN MÉTODO UTILIZANDO TÉCNICAS DE PROGRAMACIÓN SEGURAS PARA EL DESARROLLO DE APLICACIONES WEB EN ENTORNO PHP PARA MITIGAR RIESGOS POTENCIALES DE SEGURIDAD", por lo que es muy importante que responda con honestidad.

Nombre del encuestado: Henry Celaxco

DPA a la que pertenece: Dpto. Pastaza.

Fecha: 25/01/2017.

PREGUNTAS

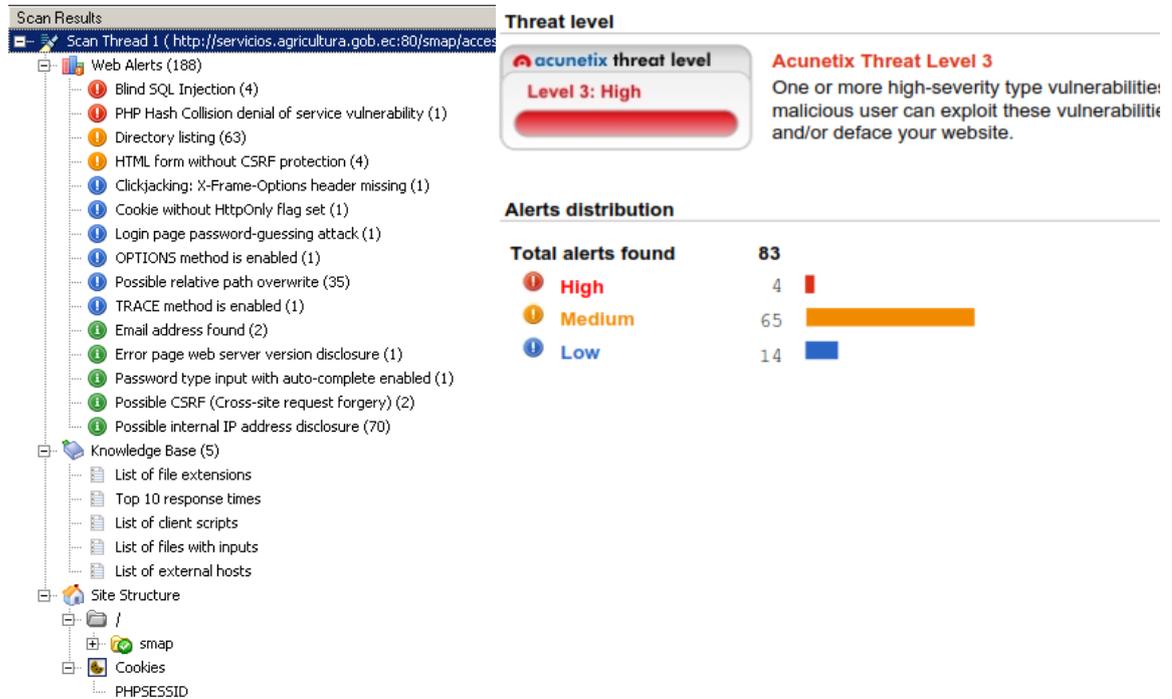
1. ¿El sistema valida todas las entradas en los formularios establecidos?
SI NO
2. ¿Considera que el sistema realiza una gestión de sesión adecuada?
SI NO
3. ¿Existe un cifrado de datos adecuado para el manejo de contraseñas?
SI NO
4. ¿El sistema maneja los errores y control de excepciones de manera segura?
SI NO
5. ¿Considera que el sistema se encuentra en un servidor que mantiene una configuración de seguridad adecuada para el servidor web, la base de datos y el lenguaje PHP?
SI NO
6. ¿Cree usted que la autenticación en el sistema está bien definida?
SI NO
7. ¿Considera que se está aplicando en el sistema una autorización adecuada para áreas de acceso más restringidas?
SI NO
8. ¿Considera usted que las técnicas de programación utilizadas en el sistema ayudarán a minimizar los ataques web?
SI NO


Firma



ANEXO B. RESULTADOS DE ACUNETIX WEB VULNERABILITY SCANNER EN LOS DOS PROTOTIPOS

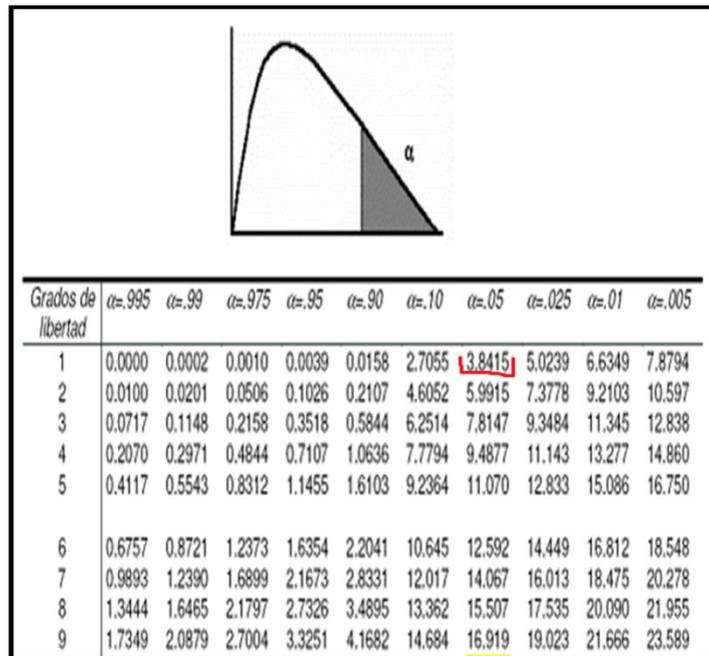
PROTOTIPO 1: SISEV



PROTOTIPO 2: S-SISEV



ANEXO C. TABLA DE CHI CUADRADO UTILIZADO PARA LA DEMOSTRACIÓN DE LA HIPÓTESIS



ANEXO D. CARACTERÍSTICAS DEL SERVIDOR WEB Y DE BASE DE DATOS ALOJADO EN EL SERVIDOR DE PRUEBAS

Servidor de base de datos

- Servidor: Localhost via UNIX socket
- Tipo de servidor: MariaDB
- Versión del servidor: 5.5.47-MariaDB - MariaDB Server
- Versión del protocolo: 10
- Usuario: root@localhost
- Conjunto de caracteres del servidor: UTF-8 Unicode (utf8)

Servidor web

- Apache/2.4.6 (CentOS) PHP/5.6.22
- Versión del cliente de base de datos: libmysql - 5.5.47-MariaDB
- Extensión PHP: mysqli
- Versión de PHP: 5.6.22

phpMyAdmin

- Acerca de esta versión: 4.4.15.6, versión estable más reciente: 4.6.2

ANEXO E. GUÍA DE CONFIGURACIÓN DE SEGURIDAD EN CENTOS 7

El sitio (blogfactores, 2016) recomienda configurar Centos 7 de la siguiente manera:

1. Instalación del sistema

Se partirá de la instalación de la imagen **CentOS-7-x86_64-Minimal-1511.iso** que tiene, como su nombre indica, sólo lo mínimo. Una vez logados en el tty correspondiente actualizamos el sistema.

```
sudo yum update
```

2. Organización del disco en particiones

Se muestra esta organización como ejemplo, siendo viables alternativas en función de los usos del sistema.

```
/          120 GiB xfs
/boot      500 MiB xfs
/var       50 GiB xfs
/tmp       25 GiB xfs
/var/tmp   35 GiB ext4
/var/log   35 GiB xfs
/home      175 GiB xfs
/var/log/audit 10 GiB xfs
/var/www   20 GiB xfs
```

3. Forzar acceso root a single user mode

Editamos el fichero correspondiente.

```
sudo vim /etc/sysconfig/init
```

Añadiendo la línea.

```
SINGLE=/sbin/sulogin
```

4. Cambiar de md5 a sha512

Simplemente

```
authconfig --passalgo=sha512 --update
```

5. Notificación de accesos

Añadimos en el archivo `/etc/pam.d/system-auth` la siguiente línea.

```
session required pam_lastlog.so showfailed
```

6. Denegar acceso a intentos fallidos

Añadimos las líneas siguientes a los archivos `/etc/pam.d/system-auth` y `/etc/pam.d/password-auth` justo bajo `auth ... pam_unix.so`

```
auth [default=die] pam_faillock.so authfail deny=3 unlock_time=604800 fail_interval=900
auth required pam_faillock.so authsucc deny=3 unlock_time=604800 fail_interval=900
```

7. Permitir el bloqueo de la pantalla

Screen permite bloquear la pantalla, lo instalamos.

```
sudo yum install screen
```

Para poder bloquear la pantalla con una contraseña debemos ejecutar `screen` y cuando se deja el puesto pulsamos `ctrl+(a x)`.

8. Desactivar Zeroconf

Modificamos el fichero `/etc/sysconfig/network` para que muestre.

```
NOZEROCONF=yes
```

9. Bloqueo de IPv6

Creamos `/etc/modprobe.d/disabled.conf` añadiendo

```
options ipv6 disable=1
```

Editamos `/etc/sysconfig/network` añadiendo

```
NETWORKING_IPV6=no  
IPV6INIT=no
```

Y deshabilitamos el soporte RPC en IPv6 comentando las siguientes líneas en `/etc/netconfig` si es que existe.

```
udp6    tpi_clts    v    inet6  udp    -    -  
tcp6    tpi_cots_ord v    inet6  tcp    -    -
```

10. Seguridad en Sysctl

Editamos el archivo siguiente.

```
sudo vim /etc/sysctl.conf
```

Para añadir...

```
net.ipv4.ip_forward = 0  
net.ipv4.conf.all.send_redirects = 0  
net.ipv4.conf.default.send_redirects = 0  
net.ipv4.tcp_max_syn_backlog = 1280  
net.ipv4.icmp_echo_ignore_broadcasts = 1  
net.ipv4.conf.all.accept_source_route = 0
```

```
net.ipv4.conf.all.accept_redirects = 0
net.ipv4.conf.all.secure_redirects = 0
net.ipv4.conf.all.log_martians = 1
net.ipv4.conf.default.accept_source_route = 0
net.ipv4.conf.default.accept_redirects = 0
net.ipv4.conf.default.secure_redirects = 0
net.ipv4.icmp_echo_ignore_broadcasts = 1
net.ipv4.icmp_ignore_bogus_error_responses = 1
net.ipv4.tcp_syncookies = 1
net.ipv4.conf.all.rp_filter = 1
net.ipv4.conf.default.rp_filter = 1
net.ipv4.tcp_timestamps = 0
#Bloqueo de ipv6
net.ipv6.conf.all.disable_ipv6 = 1
net.ipv6.conf.default.disable_ipv6 = 1
#Bloqueo de core dumps de programas con SUID
fs.suid_dumpable = 0
```

11. Lista de servicios habilitados (enabled)

La lista de los servicios habilitados se obtiene con el siguiente comando.

```
sudo systemctl list-unit-files --type=service
```

Todos los servicios que no se utilicen deben ser deshabilitados mediante un comando como el siguiente y después reiniciando el sistema.

```
sudo systemctl disable servicio
```

12. Deshabilitar protocolos no utilizados

Para prevenir ataques a las posibles vulnerabilidades en la implementación de algunos protocolos de la pila de red de linux añadimos archivos .conf al directorio /etc/modprobe.d para que se ejecute /bin/false en lugar de cargar el módulo del protocolo indicado.

```
echo "install dccp /bin/false" > /etc/modprobe.d/dccp.conf
echo "install sctp /bin/false" > /etc/modprobe.d/sctp.conf
```

```
echo "install rds /bin/false" > /etc/modprobe.d/rds.conf
echo "install tipc /bin/false" > /etc/modprobe.d/tipc.conf
```

13. Instalación de rsyslog

Si no está instalado lo hacemos.

```
sudo yum install rsyslog -y
```

Y si no está activo lo activamos.

```
systemctl enable rsyslog.service
systemctl start rsyslog.service
```

14. Incrementamos protección frente a Buffer Overflow

- *ASLR*

Escribimos en /etc/sysctl.conf la siguiente línea y reiniciamos.

```
kernel.randomize_va_space = 2
```

- *ExecShield*

Ejecutando el siguiente comando comprobaremos si existe protección en el hardware.

```
dmesg | grep --color '[NX|DX]*protection'
```

Si aparece algo como esto la siguiente línea tu sistema protege contra ciertas escrituras en el stack.

```
[ 0.000000] NX (Execute Disable) protection: active
```

Si no apareciese es necesario usar las capacidades del kernel indicándoselo explícitamente añadiendo a /etc/sysctl.conf la siguiente línea y reiniciando.

```
kernel.exec-shield = 1
```

SELinux

Comprobar que SELinux está marcado como Targeted / Enforcing viendo el estado de las variables siguiente en el archivo /etc/selinux/config.

```
SELINUXTYPE=targeted
```

```
SELINUXTYPE=enforcing
```

Comprobamos qué daemons no están confinados por SELinux con la ejecución de ese comando.

```
sudo ps -eZ | egrep "initrc" | egrep -vw "tr|ps|egrep|bash|awk" | tr ':' ' ' | awk '{ print $NF }'
```

Si no devuelve ningún resultado es que todos están confinados.

ANEXO F. CONFIGURACIONES ADICIONALES UNA VEZ INSTALADO Y CONFIGURADO CENTOS 7

El sitio (techoism, 2016) recomienda lo siguiente:

1. Cambiar la versión de PHP (de ser necesario)

- Para saber que versión de PHP está instalada digitamos
`# php -v`
- Para actualizar la versión PHP de 5.4 a 5.6, descargamos el repositorio “webtatic-release.rpm” de <https://mirror.webtatic.com/yum/el7/webtatic-release.rpm> lo instalamos con:
`yum install webtatic-release.rpm`
- Removemos el php con la version anterior con:
`# yum remove php-common`
Al remover da una lista de todos los archivos que se afectan
- Instalar la nueva versión de PHP en este caso la 5.6
`yum install php56w php56w-cli php56w-mysql`
- Reiniciamos el servicio httpd y verificamos la versión PHP
`systemctl restart httpd.service`
`# php -v`
- Y verificamos que se cambió la versión

2. Instalando phpmyadmin

- Descargar e instalar el repositorio para phpMyAdmin en la dirección http://pkgs.repoforge.org/rpmforge-release/rpmforge-release-0.5.3-1.el7.rf.x86_64.rpm
`yum install rpmforge-release-0.5.3-1.el7.rf.x86_64.rpm`
- Instalar phpMyAdmin
`yum install phpMyAdmin`
- Modificar el fichero `/etc/httpd/conf.d/phpMyAdmin.conf` para establecer desde qué IP (equipos) se permitirá acceso:

```
[root@SRVWEBMYSQL ~]# vi /etc/httpd/conf.d/phpMyAdmin.conf
```

- Por ejemplo para habilitar la IP: 192.168.1.120 quedaría de la siguiente manera:

```
# phpMyAdmin - Web based MySQL browser written in php
#
# Allows only localhost by default
#
# But allowing phpMyAdmin to anyone other than localhost should be considered
# dangerous unless properly secured by SSL
Alias /phpMyAdmin /usr/share/phpMyAdmin
Alias /phpmyadmin /usr/share/phpMyAdmin

<Directory /usr/share/phpMyAdmin/>
  <IfModule mod_authz_core.c>
    # Apache 2.4
    <RequireAny>
      Require ip 127.0.0.1
      Require ip 192.168.1.120
      Require ip ::1
    </RequireAny>
  </IfModule>
  <IfModule !mod_authz_core.c>
    # Apache 2.2
    Order Deny,Allow
    Deny from All
    Allow from 127.0.0.1
    Allow from 192.168.1.120
    Allow from ::1
  </IfModule>
</Directory>

<Directory /usr/share/phpMyAdmin/setup/>
  <IfModule mod_authz_core.c>
    # Apache 2.4
    <RequireAny>
      Require ip 127.0.0.1
      Require ip 192.168.1.120
      Require ip ::1
    </RequireAny>
  </IfModule>
  <IfModule !mod_authz_core.c>
    # Apache 2.2
    Order Deny,Allow
    Deny from All
    Allow from 127.0.0.1
    Allow from 192.168.1.120
    Allow from ::1
  </IfModule>
</Directory>
```

- Reiniciamos el servicio httpd
`systemctl restart httpd.service`
- Desde el equipo al que hemos permitido el acceso, en el navegador web, escribiremos la [URL:http://192.168.216.41/phpmyadmin](http://192.168.216.41/phpmyadmin)



CIV 23

phpMyAdmin

192.168.216.41/phpmyadmin/index.php?db=&token=decb1b14e163d25b5afa3952a836e5c&old_usr=root

phpMyAdmin

Bienvenido a phpMyAdmin

Idioma - Language

Español - Spanish

Iniciar sesión

Usuario: root

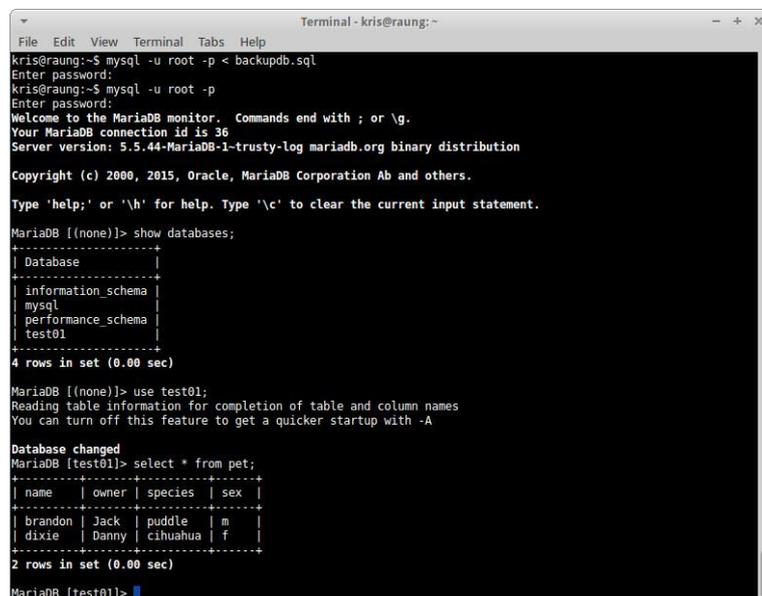
Contraseña:

Continuar

3. Importar la BD de MariaDB al servidor

El sitio (websetnet, 2015) recomienda lo siguiente:

- Obtenido el backup de nuestra base de datos magap_civz3.sql, procedemos a copiarlo en el servidor para luego ejecutar el siguiente comando:
mysql -u root -p magap_civz3 < magap_civz3.sql
- Nos solicita a continuación la contraseña y lo digitamos, esperamos un momento y finalmente está importada nuestra bd.
- Para verificar puedes acceder a la base mediante la consola o mediante phpMyAdmin.



```
Terminal - kris@raung:~
File Edit View Terminal Tabs Help
kris@raung:~$ mysql -u root -p < backupdb.sql
Enter password:
kris@raung:~$ mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 36
Server version: 5.5.44-MariaDB-1-trusty-log mariadb.org binary distribution
Copyright (c) 2000, 2015, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| test01 |
+-----+
4 rows in set (0.00 sec)

MariaDB [(none)]> use test01;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [test01]> select * from pet;
+-----+
| name | owner | species | sex |
+-----+
| brandon | Jack | puddle | m |
| dixie | Danny | chihuahua | f |
+-----+
2 rows in set (0.00 sec)

MariaDB [test01]>
```

4. Copiar el código fuente de la aplicación al servidor

- En el directorio /var/www/html copiamos la carpeta SISEV con todos sus archivos.
- Para verificar que funciona ingresamos en la URL lo siguiente:
<http://192.168.216.41/sisev>



Ministerio
de Agricultura, Ganadería,
Acuacultura y Pesca

S-SISEV

Sistema de Seguimiento y Evaluación

Esta zona tiene el acceso restringido.
" " Para entrar debe identificarse

Registre sus datos

Usuario:

Contraseña:



Type the text:

Iniciar

NOTA: si no dispone de identificación o tiene problemas " "para entrar póngase en contacto con el " "administrador del sitio

ANEXO G. LISTA DE FILTROS DE SANEAMIENTO Y DE VALIDACIÓN

LISTA DE FILTROS DE SANEAMIENTO			
ID	Nombre	Banderas	Descripción
FILTER_SANITIZE_EMAIL	"email"		Elimina todos los caracteres menos letras, dígitos y <i>!#\$%&'*+,-= ?^_`{ }~@.[]</i> .
FILTER_SANITIZE_NUMBER_FLOAT	"number_float"	FILTER_FLAG_ALLOW_FRAC TION, FILTER_FLAG_ALLOW_THOU SAND, FILTER_FLAG_ALLOW_SCIEN TIFIC	Elimina todos los caracteres a excepción de los dígitos, +- y, opcionalmente, <i>,eE</i> .
FILTER_SANITIZE_NUMBER_INT	"number_int"		Elimina todos los caracteres excepto dígitos y los signos de suma y resta.
FILTER_SANITIZE_SPECIAL_CHARS	"special_chars"	FILTER_FLAG_STRIP_LOW, FILTER_FLAG_STRIP_HIGH, FILTER_FLAG_ENCODE_HIGH	Escapa caracteres HTML <i>''<>&</i> y caracteres con valores ASCII menores que 32, opcionalmente elimina o codifica caracteres especiales.
FILTER_SANITIZE_STRING	"string"	FILTER_FLAG_NO_ENCODE_QUOTES, FILTER_FLAG_STRIP_LOW, FILTER_FLAG_STRIP_HIGH, FILTER_FLAG_ENCODE_LOW , FILTER_FLAG_ENCODE_HIGH, FILTER_FLAG_ENCODE_AMP	Elimina etiquetas, opcionalmente elimina o codifica caracteres especiales.
FILTER_SANITIZE_URL	"url"		Elimina todos los caracteres excepto letras, dígitos y <i>\$_-+!*(){} \^~[]`<>#%";/?:@&=</i> .

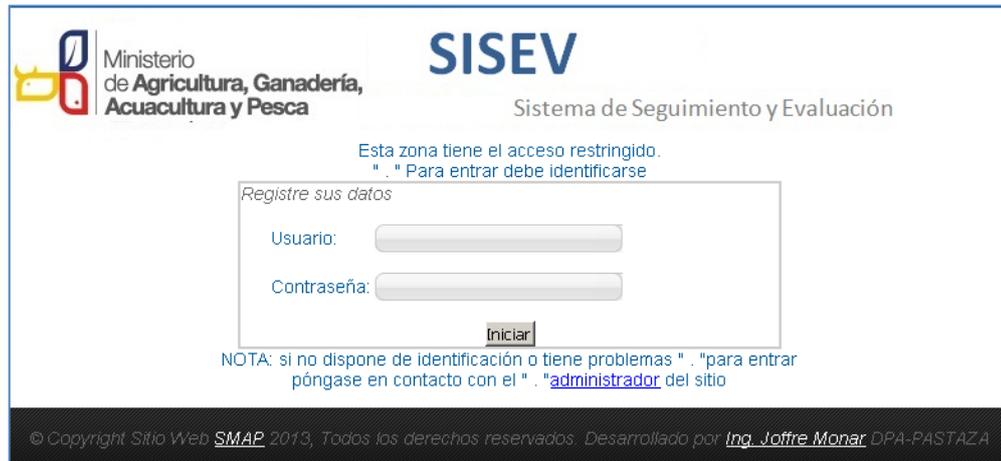
LISTADO DE FILTROS PARA VALIDACIÓN

ID	Nombre	Opciones	Banderas	Descripción
FILTER_VALIDATE_BOOLEAN	"boolean"	default	FILTER_NULL_ON_FAILURE	Devuelve TRUE para "1", "true", "on" y "yes". Devuelve FALSE en caso contrario. Si FILTER_NULL_ON_FAILURE está declarado, se devolverá FALSE sólo para "0", "false", "off", "no", y "", y NULL para cualquier valor no booleano.
FILTER_VALIDATE_EMAIL	"validate_email"	default		Valida una dirección de correo electrónico. En general, se validan direcciones de correo electrónico con la sintaxis de RFC 822, con la excepción de no admitir el pegamiento de comentarios y espacios en blanco.
FILTER_VALIDATE_FLOAT	"float"	default, decimal	FILTER_FLAG_ALLOW_THOUSAND	Valida si el valor es un float.
FILTER_VALIDATE_INT	"int"	default, min_range, max_range	FILTER_FLAG_ALLOW_OCTAL, FILTER_FLAG_ALLOW_HEX	Valida un valor como integer, opcionalmente desde el rango especificado, y lo convierte a int en case de éxito.
FILTER_VALIDATE_IP	"validate_ip"	default	FILTER_FLAG_IPV4, FILTER_FLAG_IPV6, FILTER_FLAG_NO_PRIV_RANGE, FILTER_FLAG_NO_RES_RANGE	Valida si es valor es una dirección IP, opcionalmente se puede indicar que sea sólo IPv4 o IPv6 o que no sea de rangos privados o reservados.
FILTER_VALIDATE_URL	"validate_url"	default	FILTER_FLAG_PATH_REQUIRED, FILTER_FLAG_QUERY_REQUIRED	Valida si su valor es una URL. Se ha de tener cuidado ya que un URL validado podría no especificar el protocolo HTTP <i>http://</i> , por lo que podrían ser necesarias validaciones posteriores para determinar que el URL utiliza un protocolo esperado, p.ej., <i>ssh://</i> o <i>mailto:</i> . Nótese que esta función sólo buscará para ser validadas URLs ASCII; los nombres de dominio internacionales (que contienen no-ASCII caracteres) fallarán en la validación.

ANEXO H. MANUAL DE USUARIO SISEV

Acceso al Sistema.

Para acceder al sistema nos vamos a la siguiente dirección web: servicios.agricultura.gob.ec/smap. Al ingresar a esta dirección web nos solicitará el sistema registrar el usuario y contraseña previamente establecido por su Administrador en Provincia.



Recuerde que usted luego podrá cambiar la contraseña asignada inicialmente.

1. Registros

1.1. Plan de Intervención

Una vez ingresado al sistema se visualizará la siguiente pantalla.



1.1.1. Programado Anual

Para acceder y registrar la programación anual del Plan de Intervención Parroquial, deberá seleccionar la opción **Registros/Plan de Intervención/Programado Anual**.

The screenshot shows a web form titled "Programación Anual" with a subtitle "Plan de Intervención (Parroquial)". Under "Datos del Formulario", there are dropdown menus for "Año" (2017), "Provincia: *" (PASTAZA), "Cantón: *" (PASTAZA), "Parroquia: *" (PUYO), and "Componente: *" (1-Tierra). Below this, there are two sections: "Apoyo Legalización de predios" and "Apoyo Redistribución de predios", each with input fields for the months from Enero to Diciembre. At the bottom, there are "Registrar" and "Borrar Datos" buttons, and a red note: "Nota: los datos marcados con (*) deben ser rellenados obligatoriamente".

Aquí debe registrar componente por componente (en total 13) las metas establecidas para todo el año de cada uno de los indicadores de su parroquia, si el técnico se encuentra en más de una parroquia deberá registrar la información de las parroquias en las que está realizando su trabajo, y a continuación dar clic en la opción Registrar.

Existen algunos técnicos que hacen un trabajo provincial, es decir laboran en la mayoría de cantones y parroquias de su provincia, en ese caso deberán registrar la opción **Plan Provincial Anual** que está ubicado en un submenú al lado derecho de la pantalla.

The screenshot shows a web form titled "Programación Anual" with a subtitle "Plan de Intervención (Provincial)". Under "Datos del Formulario", there are dropdown menus for "Año" (2017) and "Componente: *" (2-Riego). Below this, there are three sections: "Num. Sistemas Riego Parcelario Implementados", "Num. Microreservorios Implementados", and "Realizar Seguimiento Riego Parcelario", each with input fields for the months from Enero to Diciembre. At the bottom, there are "Registrar" and "Borrar Datos" buttons, and a red note: "Nota: los datos marcados con (*) deben ser rellenados obligatoriamente".

De igual manera que en el caso anterior para ingresar este formulario deberá registrar componente por componente (entotal13) las metas establecidas para todo el año pero en esta ocasión de toda la provincia, y a continuación dar clic en la opción Registrar.

1.1.2. Plan de Intervención Parroquial

Para acceder y registrar las metas mensuales del Plan de Intervención Parroquial, deberá seleccionar la opción Registros/Plan de Intervención/Parroquial.

Registro de Actividades
Ejecución Mensual del Plan de Intervención (Parroquial)

Datos del Formulario

Año: 2017
Mes: * MAYO
Provincia: * PASTAZA
Cantón: * PASTAZA
Parroquia: * PUYO
Componente: * 9-Organizacion

Número de organizaciones intervenidas:
Número de organizaciones fortalecidas:
Número de organizaciones acreditadas:
Número de productores de la organización que son apoyados:
Número de reuniones del comité de desarrollo rural:
Número de diagnósticos actualizados:
Número de reuniones cantonales participativas:

Registrar Borrar Datos

Nota: los datos marcados con (*) deben ser rellenos obligatoriamente

Aquí debe registrar componente por componente (únicamente con los que trabaja en la parroquia) la ejecución de metas de cada uno de los indicadores de ese componente, si el técnico se encuentra en más de una parroquia deberá registrar la información de las parroquias en las que está realizando su trabajo, y a continuación dar clic en la opción Registrar.

En el caso de los componentes 3(semilla), 5(Agrícola), 6(Pecuario), 7(Acuícola) y 8(Forestal) deberá registrar la información rubro por rubro cada uno de los indicadores.

2. Planificaciones

Para acceder y registrar su planificación, deberá seleccionar la opción Planificaciones y dependiendo si es Administrativo o Técnico seleccionará la opción Personal Administrativo/SemanalPlanificadooPersonalTécnico/SemanalPlanificado.

2.1. Personal Administrativo/Semanal Planificado

Una vez en el formulario deberán registrar todos los campos solicitados, recuerde que es la misma información que se manejaba en los reportes de Excel.

Para guardar la información en la base de datos damos clic en **Registrar**.



The screenshot shows a web form titled "Planificación Administrativa" with the subtitle "Registra día a día tu Plan Semanal de Actividades". The form includes the following fields and elements:

- Fecha:** A date input field with the label "Dia: *" and the value "3/5/2017".
- Lugar:** A dropdown menu with the label "Lugar/Sector: *" and the selected value "oficina".
- Actividades y Resultados:** A text area with the label "Actividad: *" containing the text "mantenimiento 5 equipos".
- Resultados:** A text area with the label "Resultados: *" containing the text "5 mantenimientos programados".
- Buttons:** "Registrar" and "Borrar Datos".
- Footer:** A red note: "Nota: los datos marcados con (*) deben ser rellenos obligatoriamente".

2.2. Personal Técnico / Semanal Planificado

Una vez en el formulario deberán registrar todos los campos solicitados, recuerde que esta información está relacionada con los 13 componentes que manejan a nivel del PIP. Para guardar la información en la base de datos damos clic en **Registrar**.

Planificación Técnica

Registra día a día tu Plan Semanal de Actividades de la Parroquia

Fecha
Fecha: (dd/mm/aaaa) * 8/5/2017

Hora (Inicio y Fin)
Hora: (hh:mm-hh:mm) * 08:00-11:00

Lugar (Cantón/Parroquia/Sector)
Lugar: Arajuno

Actividades, Metas y Requerimientos

Componente: * 1-Tierra

Actividad: * Apoyar en la legalizacion de pr

Meta Programada: * 1

Requerimiento: *

Observaciones: *

Nota: los datos marcados con (*) deben ser rellenados obligatoriamente

3. Reportes

3.1. Planificaciones

3.1.1. Personal Administrativo Planificado

Una vez registrada la información podrá consultar o imprimir la planificación semanal en la opción Reportes/Planificaciones/Personal Administrativo.

Ministerio
de Agricultura, Ganadería,
Acuacultura y Pesca

SISEV

Sistema de Seguimiento y Evaluación - PASTAZA

Bienvenido: Joffre Stalin Monar Monar

Inicio
Registros
Planificaciones
Reportes
Cambio Contraseña
Sugerencias
Cerrar Sesión

Reportes

Planificación Semanal de Actividades

Por favor seleccione el funcionario y la fecha a consultar

Funcionario * Monar Monar Joffre Stalin

Desde: (dd/mm/aaaa) * 8/5/2017

Hasta: (dd/mm/aaaa) * 12/5/2017

.: Planificaciones

Por Técnico

Por Unidad

Consolidado x Técnico

Seleccionamos el funcionario y el rango de fechas a consultar y se visualizará el siguiente reporte.

Ministerio de Agricultura, Ganadería, Acuacultura y Pesca		DIRECCION PROVINCIAL AGROPECUARIA DE PASTAZA			
PLANIFICACION DE ACTIVIDADES					
Semana: del 08-Mayo-2017 al 12-Mayo-2017					
Unidad: UNIDAD ZONAL DE INFORMACION					
Fecha y Hora de Registro: 2017-06-27, 14:20					
Día	Fecha	Actividad	Lugar	Tecnico Responsable	Resultados Esperados
Lunes	2017-05-08	mantenimiento 5 equipos	oficina	Joffre Stalin Monar Monar	5 mantenimientos programados
Martes	2017-05-09	capacitación quipux	oficina	Joffre Stalin Monar Monar	10 usuarios capacitados
Miércoles	2017-05-10	habilitación de puntos de red	oficina	Joffre Stalin Monar Monar	4 puntos de red habilitados
Viernes	2017-05-12	configuración de dominio y correo	oficina	Joffre Stalin Monar Monar	2 equipos configurados
Registrado Por:					
f).....					
Joffre Stalin Monar Monar					
TECNICO INFORMATICO					

3.1.2. Personal Técnico Planificado

Una vez registrada la información podrá consultar o imprimir la planificación semanal en la opción Reportes/Planificaciones/Personal Técnico Planificado.

Seleccionamos el funcionario y el rango de fechas a consultar y se visualizará el siguiente reporte.

Ministerio de Agricultura, Ganadería, Acuacultura y Pesca		Cotopaxi - Tungurahua - Chimborazo - Pastaza							
Coordinación Zonal 3		Av. 9 de Octubre S/N junto a la Quinta Macaji, Riobamba Tel: (03) 2610 057 / 2610 043 / 2610 038							
DIRECCION PROVINCIAL AGROPECUARIA DE PASTAZA									
PLANIFICACION DE ACTIVIDADES									
Semana: del 08-Mayo-2017 al 12-Mayo-2017									
Unidad: DIRECCION PROVINCIAL/ Responsable:									
Fecha y Hora de Registro: 2017-06-27, 14:20									
Fecha	Día	Hora	Actividad	Componente	Meta Programada	Requerimiento	Funcionario	Lugar	Observaciones
2017-05-08	Lunes	08:00-11:00	Apoyar en la legalización de predios de la parroquia	Tierra	1	movilización	Rene Ivan Aguirre Naranjo	Arajuno	reunión para determinar listado de beneficiarios
Registrado Por:									
f).....									
Rene Ivan Aguirre Naranjo									
TECNICO DE CAMPO									

3.2.3. Reporte de Metas

Para visualizar el reporte de metas mensual del PIP deberá ir a la opción Reportes/Plan de Intervención/Reporte de Metas.

Reporte de Metas

Matriz de Reporte Mensual x Técnico

Por favor seleccione el funcionario y la fecha a consultar

Año:

Mes:

Funcionario: *

Provincia: *

Cantón: *

Parroquia: *

.: Matriz de Reporte de Metas

Mensual x Técnico

Mensual Acumulado x Técnico

Mensual x Cantón

Mensual x Parroquia

Aquí seleccionará el mes, funcionario y la parroquia, a continuación dar clic en **Consultar**.

Ministerio de Agricultura, Ganadería, Acuicultura y Pesca

Cotopaxi - Tungurahua - Chimborazo - Pastaza

Av. 9 de Octubre S/N junto a la Quinta Macaj, Píotamba

Tel: (03) 2610 057 / 2610 043 / 2610 038

Coordinación Zonal 3

MATRIZ DE REPORTE DE METAS - MENSUAL

Fecha: Septiembre 2014
Unidad: DIRECCION PROVINCIAL/ Responsable:

Provincia	Cantón	Parroquia	COMPONENTE 1. TIERRA				COMPONENTE 2. RIEGO				Rubro Semilla	Num Semillistas Leg		
			Num. Exped	Implem	Num. Predios Redi	Num. HA Int	Num. Beneficiarios	Num. Sist. Riego Parc	Num. Microresenoind	Num. Seg. Riego Parc			Num. HA Int	Num. Beneficiarios
PASTAZA	PASTAZA	POMONA	1	0	0	0	0	0	0	0	0	0	0	0
PASTAZA	PASTAZA	POMONA	0	0	0	0	0	0	0	0	0	0	ND	0
PASTAZA	PASTAZA	POMONA	0	0	0	0	0	0	0	0	0	0	ND	0
PASTAZA	PASTAZA	POMONA	0	0	0	0	0	0	0	0	0	0	ND	0
PASTAZA	PASTAZA	POMONA	0	0	0	0	0	0	0	0	0	0	ND	0
PASTAZA	PASTAZA	POMONA	0	0	0	0	0	0	0	0	0	0	ND	0
PASTAZA	PASTAZA	POMONA	0	0	0	0	0	0	0	0	0	0	ND	0
PASTAZA	PASTAZA	POMONA	0	0	0	0	0	0	0	0	0	0	ND	0
PASTAZA	PASTAZA	POMONA	0	0	0	0	0	0	0	0	0	0	ND	0
PASTAZA	PASTAZA	POMONA	0	0	0	0	0	0	0	0	0	0	ND	0
			1	0	0	0	0	0	0	0	0	0	ND	0

Registrado Por: _____
Rene Ivan Aguirre Naranjo
TECNICO DE CAMPO

3.2.4. Informe Mensual

Para visualizar el informe mensual del PIP deberá ir a la opción Reportes/Plan de Intervención/Informe Mensual.

Informe Mensual

Plan de Intervención x Técnico

Por favor seleccione el funcionario y la fecha a consultar

Año:

Mes:

Funcionario: *

Provincia: *

Cantón: *

Parroquia: *

.: Informe Mensual

Plan Intervención x Técnico

Plan Intervención x Cantón

Plan Intervención x Parroquia

Aquí seleccionará el mes, funcionario y la parroquia, a continuación dar clic en **Consultar**.

DIRECCION PROVINCIAL AGROPECUARIA DE PASTAZA						
INFORME MENSUAL DE EJECUCIÓN DEL PLAN DE INTERVENCIÓN PARROQUIAL						
Fecha: Febrero/ 2014						
Funcionario: Rene Ivan Aguirre Naranjo / DIRECCION PROVINCIAL						
Cantón: PASTAZA						
Parroquia: POMONA						
NUM	COMPONENTE	INDICADOR	PLANIFICADO		CUMPLIDO	
			META ANUAL	META MENSUAL	META CUMPLIDA	% CUMPLIMIENTO
1	Tierra	Apoyar en la legalizacion de predios de la parroquia	1	0	0	100
1	Tierra	Apoyar en la redistribucion de predios	0	0	0	0
2	Riego	Numero de sistemas de riego parcelario	0	0	0	0
2	Riego	Numero de microreservorios implementados	0	0	0	0
2	Riego	Numero de seguimientos a riego parcelario	0	0	0	0
3	Semilla	Numero de grupos de semilleros legalizados	0	0	0	0
3	Semilla	Numero de nucleos semilleros fortalecidos	0	0	0	0
3	Semilla	TM de semilla certificada producida	0	0	0	0
3	Semilla	Numero de ha sembradas con el plan de alto	0	0	0	0
3	Semilla	Numero de beneficiarios del plan de alto rendimiento	0	0	0	0
3	Semilla	Numero de hectareas intervenidas en el componente semilla	0	0	0	0
3	Semilla	Numero de beneficiarios del componente semilla	0	0	0	0
4	Credito	Numero de expedientes de credito elaborados	5	0	0	0
5	Agricola	Numero de asistencias tecnicas a productores	180	15	0	0
5	Agricola	Numero de capacitaciones a organizaciones en temas agricolas realizadas	24	2	0	0
5	Agricola	Numero de promotores agricolas formados	0	0	0	0
5	Agricola	Numero de analisis de suelo realizados	0	0	0	0
5	Agricola	Numero de mediciones de ph realizados	12	1	0	0
5	Agricola	Numero de Parcelas demostrativas implementadas	4	1	0	0
5	Agricola	Numero de Centros de mecanización implementados	0	0	0	0
5	Agricola	Numero de Centros de Bioinsumos implementados	0	0	0	0
5	Agricola	Numero de Centros de Acopio agricola implementados	0	0	0	0
5	Agricola	Realizar seguimiento a la comercializacion de Centros de acopio	0	0	0	0
5	Agricola	Numero kits de riego (innovacion) implemetados	0	0	0	0
5	Agricola	Numero de hectareas intervenidas en el componente productivo agricola	27	2	0	0
5	Agricola	Numero de beneficiarios del componente productivo agricola	696	58	0	0

4. Cerrar Sesión

Para finalizar si deseamos salir del sistema damos clic en la opción Cerrar Sesión del menú principal.



Ministerio
de Agricultura, Ganadería,
Acuacultura y Pesca

SISEV

Sistema de Seguimiento y Evaluación - PASTAZA

Bienvenido: Joffre Stalin Monar Monar

Inicio

Registros

Planificación

Reportes

Cambio Contraseña

Sugerencias

Cerrar Sesión