Análisis de vulnerabilidades en aplicaciones Web desarrolladas en PHP Versión 5.6.24 con base de datos MYSQL Versión 5.0.11 a partir de ataques SQL Inyección

Modalidad de grado para pregrado en ingeniería de sistemas

Universidad Cooperativa De Colombia
Sede Bucaramanga
Programa de Ingeniería de Sistemas

Asesor de modalidad de grado:
Msc. Castellanos Guarín Luis Fernando

Presentado por:
Amado Ballen John William
Ayala Calderón Carlos Andrés
Sierra Morales Armando Andrés

Bucaramanga
Enero
2017

Tabla de contenido.

pág.

1. Introducción 6

2. Planteamiento del problema. 9

3. Justificación. 10

4. Objetivos. 11

4.1 Objetivo general 11

4.2 Objetivos específicos 11

5. Marco conceptual. 12

6. Metodología 14

7. Resultados 17

7.1. Historia de las Bases de datos 17

7.2. Historia de ataques de SQL inyección 18

7.3. Tipos de Pruebas de penetración 20

7.4. Impacto de un ataque SQL inyección 21

7.5. Ejecución del análisis 23

8. Conclusiones. 31

9. Recomendaciones 32

9.1. Perspectiva como resultado de la prueba 32

9.2. Perspectiva obtenida desde la investigación literaria 32

10. Discusión. 37

11. Referentes bibliográficos 39

12. Anexos. 43

Lista de figuras

pág.

Ilustración 1. Ilustración ataque Inyección. 13

Ilustración 2. Imagen “index.php” 25

Ilustración 3. Imagen “Tabla productos phpMyAdmin” 26

Ilustración 4. Imagen “Tabla usuario phpMyAdmin”. 26

Ilustración 5. Imagen “Panel de control de XAMPP V.3.2.2, con servicio activo de Apache y MySQL” 27

Ilustración 6. Imagen “login” de usuario aplicación web”. 28

Ilustración 7. Imagen “Herramienta en proceso de escaneo para hallar vulnerabilidades en la aplicación” 29

Ilustración 8. Imagen “Respuesta Escaneo de la Herramienta a la aplicación web”. 29

Ilustración 9. Imagen “Alertas de la Herramienta al finalizar el escaneo de la aplicación web” 30

Lista de tablas

pág.

Tabla 1. Pruebas de Validación 16

Tabla 2. Objetivos cumplidos sobre la investigación. 44

Tabla 3. Presupuesto Global 45

Tabla 4. Gastos de Personal 45

Tabla 5. Equipos 46

Tabla 6. Materiales 46

1. Introducción

Actualmente las aplicaciones web continúan siendo un blanco fuerte para los atacantes, estas presentan vulnerabilidades por defectos previamente no descubiertos por los programadores al no poseer conocimientos de seguridad y poder hacer uso de complementos necesarios a estos sitios web o utilizar herramientas para testeo de aplicaciones con el fin de encontrar vulnerabilidades; han permitido un aumento en estas clases de aplicaciones que se vuelven vulnerables en la web al almacenar información importante, según estudios estadísticos desarrollados por la empresa Symantec Security, estudia el desarrollo y ataques que se presentan en el mundo informático, en el último informe dan a conocer que el año 2015, han aumentado estos ataques a las aplicaciones web en 78% informando que es el principal ataque de esta clase y catalogado como un ataque altamente peligroso y contundente, producido por inyección al ser una técnica que permite explotar la base de datos con código inyectado de SQL, se encuentran mal configurados y permiten ser vulnerables al dejar huecos o ventanas abiertas para permitir realizar estos tipos de ataques (symantec Corporation World Headquarters, 2016).

"SQL inyección es una vulnerabilidad ubicada en el TOP 10 de la OWASP (Open Web Application Security Project), reconocida plataforma a nivel mundial para el estudio y conocimiento de amenazas para la explotación de aplicaciones basadas en la web es un proyecto de código abierto dedicado a determinar y combatir las causas que hacen que el software sea inseguro, utilizada para su estudio en trabajos a nivel internacional, nacional y local para su conocimiento hacia los programadores o especialistas en seguridad informática" (Tarazona Amaya, 2010).

Esto ha permitido realizar análisis de ataques a las aplicaciones web con información y servicios dinámicos que contiene una bases de datos, existe en esta organización la herramienta libre de la organización OWASP- Zap "Zed Attack Proxy" con la versión 2.4 que permite ser utilizada en sistemas operativos como KALI LINUX y WINDOWS, realiza auditoría de la seguridad a las aplicaciones web, esta es una de las más potentes que posee la plataforma, está diseñada especialmente para monitorear la seguridad de las aplicaciones web de las compañías o entidades que lo requieran, realiza auditorías más activas con métodos de análisis automáticos y pasivos, que se darán a conocer en este trabajo enfocado en la seguridad en las aplicaciones web (OWASP, 2015).

Se han realizado trabajos para el análisis y clasificación de la técnica inyección de código SQL, por medio de comandos para encontrar la vulnerabilidad que se ejecute directamente en la base de datos por medio de consultas en formularios de su envío a través del protocolo HTTP, haciendo uso de métodos GET o POST a solicitudes en funciones de manera que interactúe a códigos usualmente reconocidos por la aplicación, inicia con el login para encontrar un acceso y realizar una análisis para la entrada a la aplicación web, permitiendo el acceso a la información y datos adyacentes (William G.J., Viegas, & Alessandro, 2006), este código hace uso de operadores lógicos como el "OR", "AND" y "NOT", con el fin de lograr una respuesta de la aplicación y permitir que la base de datos detalle esta información para el atacante (Su & Wasserman, 2006), (Puneet Sing, 2016).

La implementación de la seguridad en la aplicaciones se debe tener en cuenta en la parte legal, la legislación nacional de nuestro país obliga a las entidades que hacen uso de información de personas, clientes o usuarios a proteger la información y clasificarla como información sensible de acceso único por los usuarios y así mismo permite disponibilidad,

accesibilidad y seguridad, de acuerdo a la ley estatutaria 1581 de 2012 , establece diferenciar una base de datos empresarial o personal de uso doméstico, cualquier suministro de información de un cliente o usuario se deberá realizar protocolos para informar la autorización por parte del usuario, esta ley permite que se establezcan principios para el tratamientos de estos datos, legalidad, finalidad, libertad, calidad, transparencia, acceso y circulación restringida y algo muy importante la confidencialidad (Congreso de la Republica de Colombia, 2012).

2. Planteamiento del problema.

Actualmente el aumento masivo de la tecnología y de Internet ha facilitado al mundo el uso de los diferentes servicios como consulta de páginas estáticas, dinámicas y realizar reservas en línea para la compra de productos, asesorías online, entre otros. Algunos de estos servicios manejan información confidencial y de vital importancia, mantener esta información protegida para prevenir la pérdida o que se permita la manipulación o alteración de estos datos que se pueden relacionar como los activos de una entidad o empresa. Debido a los diferentes servicios mencionados generalmente están automatizados mediante herramientas con software especializado como las bases de datos, es importante asegurar este tipo herramientas para prevenir las intrusiones y la manipulación sean nulas. Este trabajo consiste en realizar una documentación que nos permita revisar y verificar la seguridad en las aplicaciones web creadas en PHP a través de “frameworks” que específicamente manejen bases de datos multidimensionales (Gómez, 2011).

¿El análisis de la herramienta OWASP-ZAP puede generar recomendaciones para contrarrestar los ataques a la técnica SQL inyección hacia aplicaciones web desarrolladas en el “framework” CodeIgnite para PHP v 5.6.24?

3. Justificación.

La importancia de la realización del proyecto se centra en un aspecto que surge a partir de las buenas prácticas que se deben implementar a la hora de desarrollar aplicaciones y/o sistemas de información web. Tales prácticas que se desconocen o no son aplicadas permiten vulnerabilidades a la hora de buscar seguridad en estas (O.L., 2017).

En este sentido las principales necesidades detectadas que justifican la realización del proyecto son:

- La necesidad de comprender, documentar la gravedad y/o complejidad de un ataque de este tipo.
- La necesidad de investigar y adquirir conocimiento para implementar en el ámbito profesional como futuros Ingenieros de Sistemas.

La viabilidad del estudio es posible con la investigación de la literatura sobre la seguridad de la información y relación con el método de inyección SQL, esta se encuentra disponible en la web y en su mayoría está en revistas indexadas o investigaciones por parte de organizaciones especializadas que se dedican a estudios avanzados en seguridad informática como o empresas dedicadas a antivirus; las herramientas que se utilizan en el análisis son libres y se encuentran disponibles en sus sitios web.

4. Objetivos.

4.1 Objetivo general

✓ Analizar las vulnerabilidades que se pueden presentar en una aplicación web desarrollada en framework CodeIgnite v3.1 para PHP v5.6.24 y base de datos MySQL v5.0.11 mediante la técnica de “SQL Injection attack” con el fin de proponer buenas prácticas de seguridad informática.

4.2 Objetivos específicos

✓ Documentar la evolución de la técnica SQL inyección en ataques a páginas web desarrolladas en PHP 5.6.24

✓ Analizar una aplicación Web desarrollada en framework CodeIgnite v3.1 para PHP v5.6.24 y MySQL v5.0.11, utilizando la herramienta de la OWASP-ZAP v2.4, para detectar los posibles agujeros en dicha aplicación, identificando los riesgos que ocasiona la técnica SQL inyección.

✓ Proponer recomendaciones para la seguridad en aplicaciones Web desarrolladas en PHP v5.6.24 y MySQL v5.0.11, para contrarrestar métodos en la técnica SQL inyección, basada en el tipo de ataque: modificación en base de datos.

5. Marco conceptual.

"Un ataque de inyección SQL consiste en la inserción o "inyección" de una consulta SQL a través de los datos de entrada del cliente de la aplicación. Una inyección SQL con éxito de exploit puede leer los datos sensibles de la base de datos, modificar datos e insertar, actualizar, eliminar, ejecutar operaciones de administración en la base de datos (tales como apagar el DBMS), recuperar el contenido de un archivo determinado presente en el sistema de archivos DBMS y en algunos casos emitir comandos al sistema operativo. Los ataques de inyección SQL son un tipo de ataque de inyección en el que comandos SQL se inyectan en la entrada de datos de plano a fin de efectuar la ejecución de comandos predefinidos SQL" (OWASP, 2017).

En resumen:

- "Los ataques por inyección SQL permite al atacante suplantar identidad, alterar datos existentes, causar problemas de repudio, permite la revelación de todos los datos en el sistema, eliminarlos o volverlos inasequibles (sin acceso a los datos), llegar a convertirse en administradores del servidor de base de datos" (OWASP, 2017).
- "La inyección SQL es muy común con aplicaciones PHP y ASP. Debido a la naturaleza de las interfaces programáticas, las aplicaciones J2EE (www.j2ee.com/) y ASP.NET (https://www.asp.net/) tienen menor probabilidad de ser fácilmente atacadas por una inyección SQL" (OWASP, 2017).
- "La gravedad de una inyección SQL está limitada por la habilidad e imaginación del atacante y sus conocimientos en el tema, como por ejemplo las conexiones con bajo

privilegio al servidor de bases de datos, entre otras. En general, se considera a la inyección SQL de alto impacto" (OWASP, 2017).

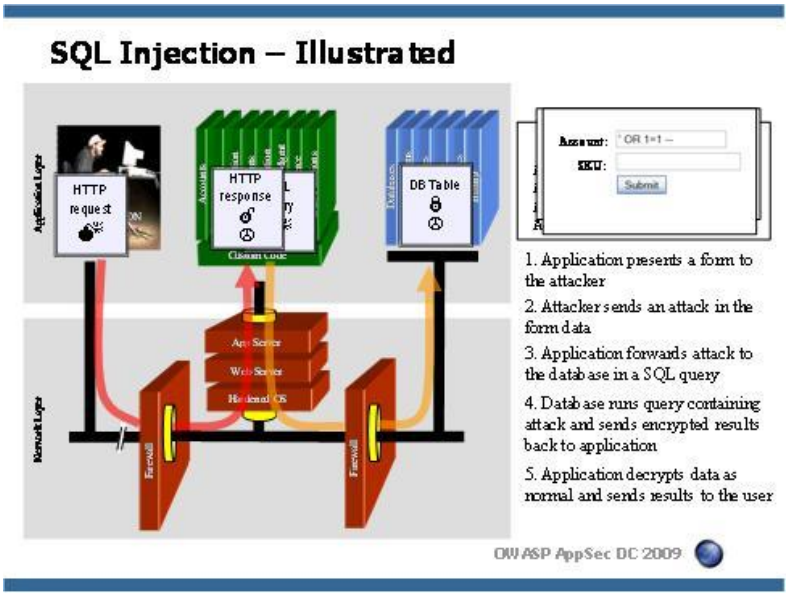


Ilustración 1. Ilustración ataque Inyección. Fuente: http://cis1.towson.edu/~cssecinj/wp-content/uploads/sqlInjections2.JPG.

La Ilustración 1 muestra cómo funciona el ataque a grandes rasgos. La aplicación presenta un formulario de "login" al atacante, el cual envía un "ataque" a través formulario. La aplicación reenvía el ataque a la B.D. (Base de Datos**),** una consulta SQL, posteriormente la B.D. ejecuta la consulta que contiene el ataque y envía los resultados cifrados a la aplicación donde por último dicha aplicación descifra datos de forma normal y envía resultados al atacante.

6. Metodología

"OWASP Testing Guide de OWASP (Versión 4)"

Esta guía presenta una metodología que recorre de forma organizada y sistemática todas las posibles áreas que sugieren vectores de ataque a una aplicación web.

De esta forma, siguiendo una lista de pruebas perfectamente organizada, se puede auditar de forma eficaz la seguridad de un desarrollo web.

Respecto la versión anterior, se han revisado y ampliado todos los puntos ya tratados. Se han añadido cuatro nuevas áreas de control (López, 2017):

- ❖ Gestión de Identidades
- ❖ Control de errores
- ❖ Criptografía
- ❖ Pruebas del lado cliente.

"De esta forma, para la realización del proyecto, se implementó la metodología expuesta anteriormente, ya que se estima conveniente para los objetivos a alcanzar, planteados anteriormente, es importante tener en cuenta que es la más usada en el mundo, por su enfoque wiki o libre; la comunidad o proyecto OWASP puede evolucionar y expandir la información en esta guía, con el fin de mantenerse en rápidos movimientos en amenazas de seguridad de las aplicación" (OWASP, OWASP Testing Project, 2017).

“La metodología propone dos fases en las pruebas de seguridad. La fase pasiva, optada a implementar al observar el funcionamiento de la aplicación, "se juega" con todas las funcionalidades posibles de la misma. El objetivo de esta fase es entender la lógica de operación e identificar los posibles vectores de ataque y/o vulnerabilidades. A continuación, en una segunda fase se ejecutarán de forma activa las pruebas propuestas según los vectores identificados en la fase anterior” (OWASP, OWASP Testing Project, 2017).

Los “test” o pruebas se agrupan en 11 categorías, las cuales son:

1. Information Gathering (Reunión de información)
2. Configuration and Deployment Management Testing (Pruebas de gestión en la configuración y la implementación)
3. Identity Management Testing (Pruebas de Gestión de Identidad)
4. Authentication Testing (Pruebas de autenticación)
5. Authorization Testing (Pruebas de autorización)
6. Session Management Testing (Pruebas de gestión de sesiones)
7. Input Validation Testing (Prueba de validación de entrada)
8. Error Handling (Manejo de errores)
9. Cryptography (Criptografía)
10. Business Logic Testing (Pruebas de lógica empresarial)
11. Client Side Testing (Pruebas de lado del cliente)

"Citado lo anterior, no todas las categorías se implementarán en la ejecución de la misma, por lineamientos establecidos para la realización del proyecto (herramientas y sus versiones) y grado de complejidad de la misma (No robusta)" (OWASP, OWASP Testing Project, 2017).

Solo se tendrá en cuenta la siguiente categoría en el test:

Tabla 1. Pruebas de Validación

4.8.5		Las pruebas para la inyección de SQL
4.4.8.5.2		Prueba de MySQL

Fuente: Según guía de pruebas OWASP V4.

La Tabla 1 representa las pruebas de validación de datos referentes o relevantes en la aplicación de la temática del proyecto.

7. Resultados

7.1. Historia de las Bases de datos

El termino bases de datos fue inducido por primera vez en un simposio celebrado en California EE.UU. en el año de 1963, donde se podía decir que una base de datos es un conjunto de información relacionada que se encuentra agrupada y estructurada, este término se remonta a la antigüedad donde ya existía bibliotecas y toda clase de registros, se utilizaban para recoger información, en 1984 el señor Hernán Hollerith creo la maquina automática de tarjetas perforadoras siendo nombrado el primer ingeniero estadístico de la historia (Palomares, 2011).

A finales del año 1973, el señor Donald D. Chamberlain y otro grupo de personas que trabajaban en los laboratorios de investigación de IBM Corporation, con el objetivo de desarrollar un lenguaje especifico que les permitiera acceder en forma estándar aquellas bases de datos, que nacidas quizás bajo otro concepto comenzaban a adecuarse al denominado modelo relacional, se perfilaban como un estándar en la generación de modelos complejos de datos.

La primera etapa de este proyecto arrojo como resultado un lenguaje denominado SEQUEL por “Structure English Query Languaje” el cual fue evolucionando en forma acelerada convirtiéndose hacia el año 1977 en SEQUEL/2, finalmente por motivos legales, en SQL (Estructural Query Languaje).

Iniciando los años ochenta, momento en que IBM lanzara su producto de base de datos relacional DB” orientado a equipos de rango medio / alto nivel, este lenguaje SQL se

incorporado en la mayoría de los productos de este tipo (Sybase, Oracle y más tarde Informix), finalmente para convertirse en un estándar ANSI en 1986 y estándar ISO a fines de 1987, la ANSI sufrió varias versiones y agregados a lo largo del tiempo desde el inicio del estándar en año 1986 con SQL-86, en ese tiempo era el alias de SQL-87 hasta la fecha que ofrece mayor facilidades en las aplicaciones.

En el trascurso de los años 80 varias compañías como Oracle y Sybase comercializaron productos basados en SQL que les permitió convertirse en el estándar industrial de las bases de datos relacionales.

Luego llegaría el turno de las revisiones SQL89 y SQL92 las cuales incorporaban mejora sustancial y corregían algunos inconvenientes encontrados en versiones anteriores.
A la fecha casi 30 años de su creación el "Puré SQL" ha demostrado que lejos de aquellos inicios como lenguaje embebido se ha convertido en una poderosa herramienta capaz de manejar estructuras lógicas complejas, así como cualquier tipo de dato imaginado como la versión en desarrollo denominado SQL3 comparte tantos atributos como lenguajes de programación convencionales y se considera esta nueva versión como un verdadero lenguaje standolone (Racciatti, 2002)

7.2. Historia de ataques de SQL inyección

JEFF FORRISTAL alias "puppy", fue quien inicio durante una auditoría a un sistema Windows NT, encontró un servicio web que interactuaba con una base de datos MS SQL Server 6.5, y revisando la posibilidad del motor de SQL, capto que era posible ejecutar sentencias encadenadas o batch. Debido a que se encontró una página web que

permitía introducir valores en parámetros para la realización de búsquedas, inicio a encadenar sentencias SQL para provocar la ejecución de consultas a la aplicación y permitiera esta que se concatenase dicha consulta con la original en el código fuente. Además, como no podía ver el código fuente de la sentencia en sí, pensó en incluir caracteres guion para evitar la ejecución del código posterior en lo que introdujo (Phrack Magazine, 1985-2016).

Inyección SQL es toda una regla este señor lo evidenció con lujo de detalles paso por paso, en la revista PHRACK número 54 publicada el 25 de diciembre de 1998 (Phrack Magazine, 1985-2016), (S.B.D. SecurityDefault.com, 2013).

Dentro de su contribución de Rain Forrest Puppy (pseudónimo que utilizó sin revelar su verdadera identidad durante muchísimos años), con todo el proceso de intrusión sobre sistemas Windows NT, se encuentra un apartado denominado "ODBC and MS SQL server 6.5" en el que comienza el proceso mencionado previamente donde plasmo el siguiente documento paso a paso “How i Hacked PacketStorm” (Packet Storm, 2016).

SQL inyección (SQL Injection), es una típica vulnerabilidad que se puede encontrar en cualquier auditoría web. Ha llegado a un punto que no sorprende encontrarnos con ella en todo tipo de aplicaciones, sean críticas o no, está claro que no sólo permite la obtención de información de la base de datos al igual la estructura de esta misma (S.B.D. SecurityDefault.com, 2013).

Lo más curioso de todo es que lleva entre nosotros más de 15 años y es de las vulnerabilidades más conocidas y siempre protagonista del Top 10 de amenazas OWASP, en esta última edición las inyecciones de código se encuentran en primer lugar (eSecurity Planet, 2013).

Jeff Forristal es un investigador de seguridad que lleva más de una década dedicado al mundo de la seguridad en diferentes vertientes, actualmente se encuentra trabajando en Bluebox Security con el fin de investigar vulnerabilidades en dispositivos móviles. Entre sus aportaciones destaca, la herramienta Whisker, de las primeras aplicaciones para análisis de vulnerabilidades web. En su sección de PacketStorm (lleva dado de alta desde 1999) veréis scripts y herramientas que ha compartido para la detección y explotación de vulnerabilidades sobre multitud de sistemas (S.B.D. SecurityDefault.com, 2013).

La inyección de SQL se ha convertido en el azote de la era de Internet. Año tras año, ha sido citada como una de las principales vulnerabilidades de seguridad en Internet, responsables de innumerables violaciones de datos (eSecurity Planet, 2013).

7.3. Tipos de Pruebas de penetración

Las pruebas de penetración se enfocan principalmente en las siguientes perspectivas:

- Pruebas de penetración con objetivo: se buscan las vulnerabilidades en partes específicas de los sistemas informáticos críticos de la organización (Ramos Ramos, 2015).
- Pruebas de penetración sin objetivo: consisten en examinar la totalidad de los componentes de los sistemas informáticos pertenecientes a la organización. Estas pruebas suelen ser las más laboriosas (Ramos Ramos, 2015).
- Pruebas de penetración a ciegas: en estas pruebas sólo se emplea la información pública disponible sobre la organización (Ramos Ramos, 2015).

- Pruebas de penetración informadas: aquí se utiliza la información privada, otorgada por la organización acerca de sus sistemas informáticos. En este tipo de pruebas se trata de simular ataques realizados por individuos internos de la organización que tienen determinado acceso a información privilegiada (Ramos Ramos, 2015).
- Pruebas de penetración externas: son realizas desde lugares externos a las instalaciones de la organización. Su objetivo es evaluar los mecanismos perimetrales de seguridad informática de la organización (Ramos Ramos, 2015).
- Pruebas de penetración internas: son realizadas dentro de las instalaciones de la organización con el objetivo de evaluar las políticas y mecanismos internos de seguridad de la organización (Ramos Ramos, 2015).

7.4. Impacto de un ataque SQL inyección

Aunque los efectos de un ataque de inyección SQL éxito varían en función de la aplicación específica y la forma en que la aplicación procesa los datos suministrados por el usuario, inyección SQL en general se puede utilizar para llevar a cabo las siguientes acciones sobre el sistema:

- Divulgación de información: Este ataque permite a un atacante obtener, ya sea directa o indirectamente, la información sensible en una base de datos (QuintanaLlanes, 2013).

- descubrimiento de información: las técnicas de inyección SQL pueden permitir a un atacante modificar consultas para acceder a registros y/o objetos de datos a los que inicialmente no tenía acceso (Alonso Cebrián, Guzman Sacristan, Laguna Durán, & Martin Bailón) .
- Comprometer la integridad de los datos: Este ataque consiste en la alteración de los contenidos de una base de datos. Un atacante podría utilizar este ataque para desfigurar una página web, o más probablemente para insertar contenido malicioso en páginas web (QuintanaLlanes, 2013).
- Elevación de privilegios: Todos los sistemas de autenticación que utilicen credenciales almacenados en motores de bases de datos hacen que una vulnerabilidad de inyección SQL pueda permitir a un atacante acceder a los identificadores de usuarios más privilegiados y cambiarse las credenciales (Alonso Cebrián, Guzman Sacristan, Laguna Durán, & Martin Bailón).
- Disponibilidad de datos: Este ataque permite a un atacante borrar información con la intención de causar daño o eliminar la información del registro de auditoría o en una base de datos (QuintanaLlanes, 2013).
- Denegación de servicio: La modificación de comandos SQL puede llevar a la ejecución de acciones destructivas como el borrado de datos, objetos o la parada de servicios con comandos de parada y arranque de los sistemas, asimismo, se pueden inyectar comandos que generen un alto computo en el motor de base de datos que haga que el servicio no responda en tiempos útiles a los usuarios legales (Alonso Cebrián, Guzman Sacristan, Laguna Durán, & Martin Bailón).

- Ejecución de comandos remotos: Uso de comandos a través de una base de datos puede permitir a un atacante poner en peligro el sistema operativo host. Estos ataques a menudo aprovechan un procedimiento existente, del sistema operativo para la ejecución del comando host (QuintanaLlanes, 2013).
- Suplantación de usuarios: Al poder acceder al sistema de credenciales, es posible que un atacante obtenga las credenciales de otro usuario y realice acciones con la identidad robada o "spofeada" a otro usuario (Alonso Cebrián, Guzman Sacristan, Laguna Durán, & Martin Bailón)

7.5. Ejecución del análisis

El análisis realizado a continuación, va relacionado a un ataque simulado a la aplicación web desarrollada con el framework codeIgnite v3.1, para PHP v5.6, utilizando una base de datos de MySQL v5.0.11, ejecutada desde el administrador PHPMyadmin v4.5.1, lo más semejante a un "pentesting" real, por motivos seguridad estas pruebas no se pueden realizar a cualquier aplicación web sin permiso de las empresas o dueños de estos dominios y aplicaciones, por tal motivo se estaría infringiendo normas y leyes penales. El ejercicio como tal, es simular la función de una aplicación para "encontrar" fallos en la comprobación de parámetros de entrada, se considera parámetro de entrada cualquier valor que provenga desde el usuario. Esta instancia se debe resumir en "un ataque inteligente", por lo que cualquier de estos parámetros pueden ser enviados con "malicia". Se debe asumir también que cualquier medida de protección implementada en el cliente puede

fallar. Como ejemplos de parámetros a considerar donde se suelen dar estos fallos (Alonso Cebrián, Guzman Sacristan, Laguna Durán, & Martin Bailón) .

Estos son algunos fallos:

- Campos de formularios: utilizados en métodos de llamadas POST.
- Campos de llamada GET pasados por variables.
- Parámetros de llamadas a funciones JavaScript.
- Valores en cabecera http.

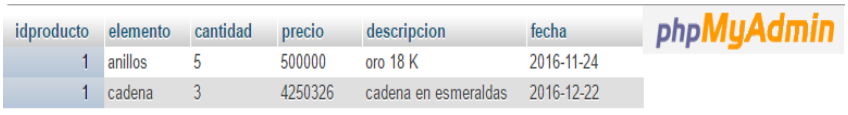
La siguiente practica se inicia con la creación de la aplicación web utilizando como framework CodeIgniter v3.1 que permite desarrollar la aplicación con un conjunto de herramientas más rápido con la ayuda de librerías, todo a su vez la reducción de código para realizar nuestra aplicación. CodeIgniter permite utilizar la arquitectura de Modelo Vista Controlador MVC, a tal punto que permite el desarrollo y validación de formularios al igual el enlace con la base de datos (EllisLAb - British Columbia Institute of Technology, 2017).

Esto permite la creación del “Login” de la aplicación web que está incluido en la mayoría de páginas web dinámicas, muestran un ingreso de usuarios por medio de su puerta principal que a su vez permitirá realizar pruebas de código SQL por medio del acceso de usuarios.

Ilustración 2. Imagen “index.php” http://localhost/sqlinyeccion/ - Fuente autores.

En la Ilustración 2, se refleja el ingreso de la aplicación web diseñada con el framework CodeIgniter v3.1, se tiene en el menú conectarse que permitirá llevar al usuario después de haber dado clic sobre este y lo enviara al formulario de ingreso al sistema, donde posteriormente esta URL (Uniform Resource Locator), es un identificador de recursos uniforme. Están formados por una secuencia de caracteres, de acuerdo a un formato modélico y estándar, que designa recursos en una red, como Internet, se usará para realizar el ataque a la herramienta de prueba (EllisLAb - British Columbia Institute of Technology, 2017).

Lo siguiente fue desarrollar una base de datos utilizando phpMyAdmin v4.5.1 la herramienta de desarrollo libre escrito en PHP, para el uso de MySQL v5.0.11, esta herramienta es compatible con varios gestores de bases de datos que permiten el uso de tablas, columnas y permisos. (phpMyAdmin, 2017).



idproducto	elemento	cantidad	precio	descripcion	fecha
1	anillos	5	500000	oro 18 K	2016-11-24
1	cadena	3	4250326	cadena en esmeraldas	2016-12-22

Ilustración 3. Imagen “Tabla productos phpMyAdmin” http://localhost/phpmyadmin/sql.php?server=1&db=inyeccion&table=producto&pos=0&token=69ee616df71074bcddbf6d4f4b4cf9bd – Fuente: autores.

Al implementar la base de datos se incluyeron dos tablas denominadas: “producto”, “usuario”. En la tabla producto se agregan seis (6) campos que refleja en la Ilustración 3: id producto, elemento, cantidad, precio, descripción, fecha, al igual dos registros para introducir algunos datos y poblar dicha tabla.

username	password	type
admin	25446782e2ccaf0afdb03e5d61d0fbb9	administrador
empleado	8fbbf95f1e5678899cb285b6051846a7	empleado

phpMyAdmin

Ilustración 4. Imagen “Tabla usuario phpMyAdmin” http://localhost/phpmyadmin/sql.php?server=1&db=inyeccion&table=usuario&pos=0&token=69ee616df71074bcddbf6d4f4b4cf9bd – Fuente: autores.

La Ilustración 4, se muestra la tabla usuario, se crean los siguientes campos: “username, password y type”, “username” , como referencia a los distintos tipos de usuarios, el password se cifra en encriptación MD5, significa en las siglas “Message Digest Algorithm 5” se utiliza como una función de codificación o huella digital de un archivo que permite para este caso codificar el dato (contraseña) en dicho campo y funciona de la siguiente manera; un algoritmo de codificación de 128 bits que genera un hexadecimal de 32 caracteres. Este algoritmo es irreversible, esto indica que es imposible descifrar el dato (contraseña) original. Se continua con el campo type este almacenará los distintos roles en el sistema. Todo esto con el fin de simular una aplicación web enlazada a una base de datos funcional, para el ataque, como se describe a continuación (MD5online org, 2017):

Se utiliza la plataforma de software libre XAMPP v3.2.2 que consiste en el sistema de gestión de bases de datos MySQL, el servidor web Apache e intérpretes de lenguajes como PHP y Perl. El nombre proviene del acrónimo de **X** (para cualquiera de los diferentes de sistemas operativos, **A**pache, **M**ySQL, **P**hp, **P**erl), (Apache Friends, 2017).

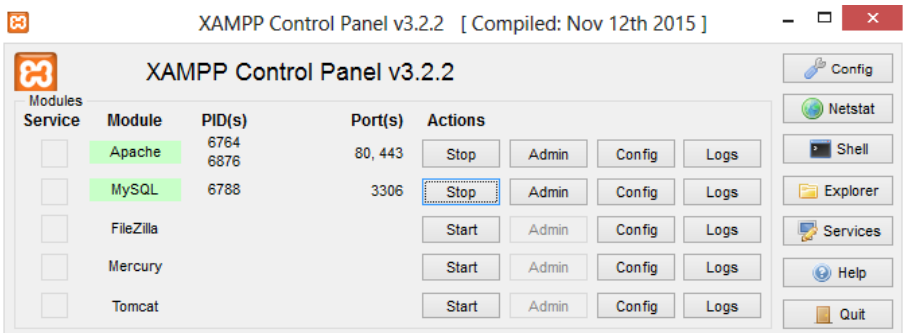


Ilustración 5. Imagen “Panel de control de XAMPP V.3.2.2, con servicio activo de Apache y MySQL” Imagen tomada del panel de control de XAMPP – Fuente autores.

Este software libre sirve para probar la aplicación en el ordenador sin la necesidad de ingresar a internet para lograr la práctica y poder simular el servidor. Se lanza y administra los servicios como se muestra en la Ilustración 5, el panel de control de XAMPP permite saber que puertos se están utilizando en nuestro equipo y también permite elegir qué otros servicios iniciar, según se requiera (Apache Friends, 2017).

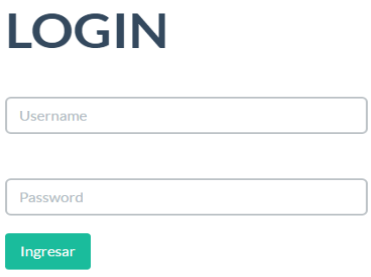


Ilustración 6. Imagen “login” de usuario aplicación web” http://localhost/phpmyadmin/login.php - Fuente autores.

El uso de XAMPP permite iniciar los servicios de la aplicación de prueba, posteriormente se ingresa al navegador Google Chrome v55.0.2883.77 m 64-bits. Se digita la siguiente ruta http://hostslocal/sqlinyeccion/index.php/login, esta muestra el modulo “login” para el acceso de usuarios. Tal y como se muestra en la anterior Ilustración 6.

Uso de la herramienta OWASP ZAP

Es una poderosa herramienta que permite realizar ataques de penetración también conocido como “pentesting”, usada para auditar aplicaciones en busca de vulnerabilidades, cabe resaltar que es una herramienta libre o gratuita de multiplataforma, permite el uso en varios sistemas operativos, como se menciona en esta prueba realizada sobre sistema operativo Windows 8.1. “Con la instalación de la herramienta OWASP-ZAP, como inicio a esta se debe tener la configuración deseada, para encontrar vulnerabilidades que tenga la aplicación de prueba, con relación a los ataques SQL Inyección” (OWASP, 2017).

Se debe iniciar la herramienta para su ejecución e ingresar la URL de la página o la ruta del servidor de la víctima, por consiguiente, el programa ejecuta el análisis para poder encontrar las vulnerabilidades, es de resaltar que esta herramienta muestra cuatro tipos de alertas, como a continuación se enumeran (OWASP, OWASP Testing Project, 2017).

1. Alertas con Alta prioridad.
2. Alertas con Prioridad media
3. Alertas con Baja prioridad.
4. Alertas informativas.

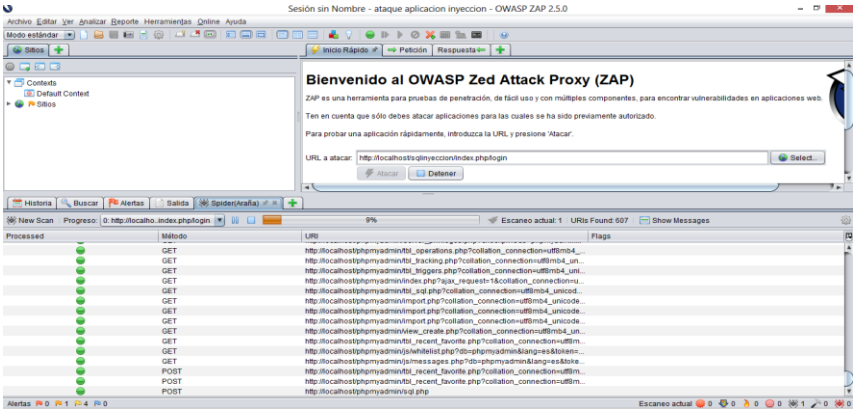


Ilustración 7. Imagen "Herramienta en proceso de escaneo para hallar vulnerabilidades en la aplicación" Tomado de la interfaz gráfica de OWASP-ZAP V2.4 – Fuente autores.

La Ilustración 7, muestra la herramienta realizando el escaneo de la aplicación, al hacer peticiones POST por medio del navegador hacia la aplicación para encontrar fallos o huecos de seguridad, haciendo uso de peticiones directamente a la aplicación web diseñada bajo el framework CodeIgniter.

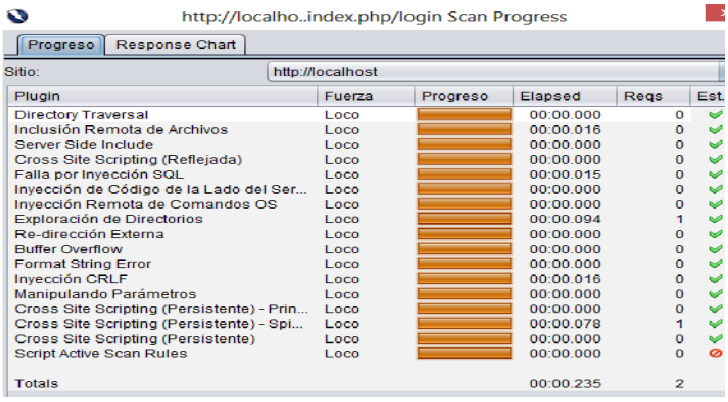


Ilustración 8. Imagen "Respuesta Escaneo de la Herramienta a la aplicación web" Tomado de la interfaz gráfica de OWASP-ZAP V2.4 – Fuente: autores.

Una vez realizada la prueba a la aplicación y después de un tiempo, donde la herramienta OWASP ZAP realiza un escaneo de peticiones a toda la aplicación para encontrar fallos y obtener un 100% de escaneo, se permite obtener una serie de respuestas en cuanto a los tipos de amenazas y vulnerabilidades que esta misma encuentra, como lo demuestra la Ilustración 8.

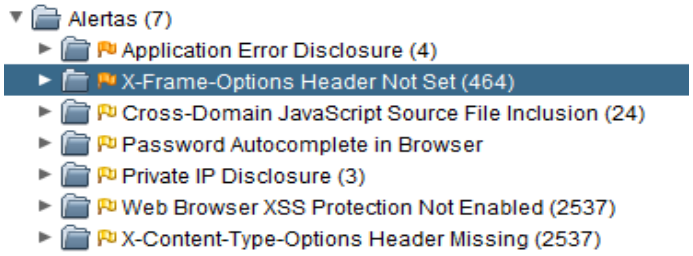


Ilustración 9. Imagen “Alertas de la Herramienta al finalizar el escaneo de la aplicación web” Tomado de la interfaz gráfica de OWASP-ZAP V2.4 – Fuente: autores.

En los detalles del proceso de escaneo se observa que la aplicación ejecuto seguridad por el uso del framework y no permitió resultados de ataque SQL inyección como se muestra en el informe, se puede observar que se encuentra otras clases de ataques que pueden ser explotados, pero no vienen a mención por los limites delineados para esta prueba. Siendo un indicio para no iniciar un ataque SQL inyección a la aplicación probada y así permite que un programador pueda detectar estas otras amenazas para evitar vulnerabilidades de seguridad de estos ataques. El escaneo que se muestra Ilustración 9 refleja las siguientes clases de alertas:

1. 2 alertas con prioridad media.
2. 5 siguientes son de alerta con baja prioridad.

Se puede observar que la herramienta cumple su cometido.

8. Conclusiones.

El campo de la inyección de SQL contiene numerosas técnicas de defensa (Anup Shakya, 2011), esto no significa que sea el ataque menos vulnerable en las aplicaciones web, no obstante y según el OWASP top ten del 2013 (OWASP, 2017), presento en su listado como la A1 Inyección, siendo SQL Inyección como la primera técnica.

La herramienta de la OWASP "Zed Attack Proxy" de uso gratuito detectó algunos agujeros en la seguridad del aplicativo creado para el proyecto, identificando cada uno de ellos, no obstante no se detectaron vulnerabilidades para ataques de SQL inyección a razón que el framework utilizado tiene funciones pre-establecidas evitando esta técnica, pero se encontraron otras vulnerabilidades como son Application Error Disclosure, X-frame-Options Header NotSet, Cross-Domain JavaScript, Password Autocomplete, Private Ip Disclosure, Web Browser XSS, X-Content-Type-Options

Se generaron doce (12) recomendaciones obtenidas desde la investigación literaria (OWASP y proyecto SQL rand, ids, metodología de autenticación con algoritmos, técnicas para prevenir las inyecciones de SQL, agente especializado en la detección de ataques, herramienta ardilla), desde la perspectiva como resultado de la prueba se estableció una recomendación (uso framework).

9. Recomendaciones

9.1. Perspectiva como resultado de la prueba

- El uso del framework en el desarrollo de aplicaciones web, se recomienda para implementar páginas web con php e incluir estas funciones, librerías, informe de errores, filtros para Scripting, protección CSRF, tratamiento de claves, que vienen incluidas para minimizar las vulnerabilidades de seguridad de un sitio web (EllisLAb - British Columbia Institute of Technology, 2017).

9.2. Perspectiva obtenida desde la investigación literaria

Se dan a conocer algunas recomendaciones hechas por diferentes autores, en cuanto a la evolución de la técnica SQL en ataques a páginas web.

SQL rand, un sistema para prevenir ataques de inyección SQL contra Servidores web. La intuición principal es que mediante el uso de un lenguaje de consulta SQL aleatorio, a una aplicación CGI (Interfaz de Gateway común) particular, es posible detectar y abortar consultas que incluyen código inyectado. SQLrand es un sistema muy práctico que soluciona un problema hasta ahora Ignorado, en preferencia a los ataques de desbordamiento de búfer "de alto perfil" (Stephen W. Boyd, 2004).

SQL inyección tiene un impacto perjudicial en las aplicaciones web que tendrán un impacto en los negocios en línea. La propuesta por los desarrolladores tiene su principal finalidad es neutralizar el ataque de inyección de SQL durante la fase de codificación y ejecución. Una de las soluciones a esto es mediante la detección de intrusiones del sistema. Entre todos los tipos de IDS (Detección de Intrusión del sistema), los IDS basados en la firma han sido La mayoría de los administradores de red por su fiabilidad y precisión (Dr. RajashreeShettar, 2014).

De otro modo se presenta una metodología en la cual proporcionan dos niveles de autenticación de usuario en el nivel de base datos:

1. Autenticación SQL.
2. Autenticación XML (Asha. norte, 2012).

Se presenta una visión general de los ataques SQL inyección (SQLIA) y los métodos de evitarlos. Se discuten los modelos propuestos para bloquear y prevenir inyecciones de SQL dinámicas en procedimientos almacenados en las bases de datos. Cuenta también con un enfoque que tiene tanto conceptual y práctico sobre la mayoría de las técnicas existentes (Tejinderdeep Singh Kalsi).

El uso de un agente especializado en la detección de ataques de inyección SQL, el agente incorpora un Motor de Razonamiento basado en casos que está equipado con un sistema de aprendizaje y adaptación capacidad de clasificación de códigos maliciosos. También incorpora algoritmos avanzados en las etapas del ciclo de razonamiento (Cristian Pinzón, Álvaro Herrero, Juan F. De Paz, Emilio Corchado, Javier Bajo, 2010).

Igualmente se presenta una técnica y una herramienta automatizada para encontrar vulnerabilidades de seguridad en aplicaciones web; ardilla (es una herramienta automatizada para la creación de inyección SQL y los tipos de ataque XSS en las aplicaciones web PHP) está diseñado para probar aplicaciones PHP Antes del despliegue. Vulnerabilidades de seguridad que ardilla permite identificar se puede arreglar antes que el software llegue a los usuarios, ardilla crea ataques concretos que explotan la vulnerabilidad (Adam Kieyzun, Mayo - 2009).

Se presenta un estudio de comparación de técnicas para la detección y prevención de SQLIAs (ataques de inyección de SQL), identificando por primera los diversos tipos de SQLIAs conocidos a la fecha. Se evalúan las técnicas consideradas en términos de su capacidad para detectar y/o prevenir este tipo de ataques, estudio de los diferentes mecanismos mediante los cuales SQLIAs puede ser inyectados en una aplicación e identificados, de igual manera mecanismos de detección y prevención podrían ser totalmente automatizada (William G. J. Halfond, 2006).

Se recomienda en el desarrollo de páginas web con PHP usar frameworks que permitan agregar funcionalidad extendida a un lenguaje de programación y automatizar los patrones de programación para orientarlos a un determinado propósito (Gustavo Martínez, Germán DaríoCamacho, Daniel Alberto Biancha, 2010). Para tal investigación minimizó las vulnerabilidades se seguridad de un sitio web.

La mayoría de los navegadores web modernos admiten encabezado HTTP, opciones de clases de ataques relacionados con SQL inyección, asegurándose de establecer en las páginas web devueltas por el sitio, un rastro para en enlazar el servidor con el navegador (OWASP Zed Attack Proxy Project, 2016).

- Asegúrese de que los archivos de origen de JavaScript se carguen solo desde fuentes de confianza y que los usuarios finales de la aplicación no puedan controlar las fuentes (OWASP Zed Attack Proxy Project, 2016).
- Desactive el atributo AUTOCOMPLETE en formularios o elementos de entrada individuales que contengan entradas de contraseña utilizando AUTOCOMPLETE='OFF' (OWASP Zed Attack Proxy Project, 2016).
- Elimine la dirección IP privada del cuerpo de respuesta HTTP. Para comentarios, utilice comentarios JSP/ASP en lugar de comentarios HTML/JavaScript que pueden ver los navegadores de cliente (OWASP Zed Attack Proxy Project, 2016).
- Asegúrese de que el servidor de aplicaciones web establezca el encabezado Content-Type, si es posible, asegúrese de que el usuario final utilice un navegador web moderno y compatible con estándares que no realicen en absoluto o que la aplicación web/servidor pueda no realizar Sniffing (OWASP Zed Attack Proxy Project, 2016).
- Asegúrese de que el filtro de XSS del navegador web este habilitado, estableciendo el encabezado de respuesta HTTP X-XSS-Protecion en '1' (OWASP Zed Attack Proxy Project, 2016).

10. Discusión.

Las principales razones del ataque SQL inyección es aprovechar las fallas en la lógica de validación de entrada de los componentes web por secuencias de comandos, se referencia el trabajo realizado con SQLrand que formula capturar las consultas inyectadas para realizar un posterior análisis por medio de un algoritmo e implementación de un proxy,

Con la finalidad de prevenir que la base de datos muestre alguna **información** (Stephen W. Boyd, 2004); en relación con nuestra herramienta de la OWASP "Zed Attack Proxy" permite a los programadores realizar testeos durante el desarrollo de la aplicación con el fin de encontrar vulnerabilidades y poder corregir alguna vulnerabilidad encontrada en la aplicación al igual permitir realizar la corrección o el análisis en el código directamente en un ambiente gráfico, caso contrario al SQL rand donde se debe implementar este modelo sobre la aplicación que debe estar actualizando este tipo de código para no ser vulnerable (OWASP Zed Attack Proxy Project, 2016).

Otro análisis en comparación es el trabajo de investigación de un detector de intrusión CBR para SQL ataques de inyección que incorpora un motor de razonamiento en un sistema de aprendizaje y adaptación, con la capacidad de clasificación de códigos maliciosos, este agente por medio de algoritmos avanzados revisa las consultas de un ciclo de razonamiento que asemeja a una red neuronal, permite clasificar consulta SQL ejecutada o una secuencia de SQL inyección, es importante resaltar que la implementación de la herramienta en una aplicación seria robusta y elevaría los costos por tratarse de temas de

inteligencia artificial y uso de neuronas (Cristian Pinzón, Álvaro Herrero, Juan F. De Paz, Emilio Corchado, Javier Bajo, 2010), caso contrario la implementación en nuestro proyecto del framework CodeIgniter v.3.1. permitió encontrar que la calidad de código que permite el desarrollo de una aplicación con parámetros de seguridad incluida como filtros para XSS, además manejo de las Url's que se basa en un enfoque de segmentos, filtrado en los datos que restringe, también adiciona protección de CSRF, para no permitir engaño en solicitudes, encriptación de claves de usuarios y algo muy importante opciones de consulta para regular y simplificar escapes de estas consultas sin necesidad de mostrar al usuario o atacante (EllisLAb - British Columbia Institute of Technology, 2017).

El uso de framework hoy en día es muy indispensable para el desarrollo y evitar amenazas en la web, prevenir ataques de SQL inyección permite que la aplicación no sea objeto de inicios a otros ataques.

11. Referentes bibliográficos

Phrack Magazine. (1985-2016). *phrack.org*. Obtenido de phrack.org: http://phrack.org/issues/54/8.html

Adam Kieyzun, P. J. (Mayo - 2009). Automatic creation of SQL Injection and cross-site scripting attacks. *ICSE '09 Proceedings of the 31st International Conference on Software Engineering - IEEE*, 10.

Alonso Cebrián, J. M., Guzman Sacristan, A., Laguna Durán, P., & Martin Bailón, A. (s.f.). Ataques a BB. DD. , SQL Injection. *Fundacion para la Universidad Abierta de Cataluña*, 68.

American Psychological Association. (2010). *Manual de publicaciones de la American Psychological Association.* México: Manual Moderno.

Anup Shakya, D. A. (2011). *A Taxonomy of SQL Injection Defense.* Suecia: School of Computing Blekinge Institute of Technology .

Apache Friends. (22 de 01 de 2017). *Apache Friends*. Obtenido de https://www.apachefriends.org/es/index.html

Asha. norte, M. V. (2012). Preventing SQL Injection Attacks. *Revista Internacional de Aplicaciones Informáticas*, 5.

Congreso de la Republica de Colombia. (2012). *Ley Estatutaria 158 de 2012.* Bogotá: Congreso de la Republica.

Cristian Pinzón, Álvaro Herrero, Juan F. De Paz, Emilio Corchado, Javier Bajo. (2010). CBRid4SQL: A CBR Intrusion Detector for SQL Injection Attacks. En *Hybrid Artificial Intelligence Systems.* San Sebastián, Spain: Springer Berlin Heidelberg.

Dr. RajashreeShettar, A. G. (2014). SQL Injection Attacks and Defensive Techniques. *International Journal of Computer Technology and Applications*.

EllisLAb - British Columbia Institute of Technology. (05 de 01 de 2017). *CodeIgniter*. Obtenido de CodeIgniter: www.codeigniter.com

eSecurity Planet. (25 de 11 de 2013). *eSecurity Planet*. Obtenido de http://www.esecurityplanet.com/network-security/how-was-sql-injection-discovered.html

Fundación Universitaria Konrad Lorenz. (10 de Julio de 2014). *Para Autores: Revista Latinoamericana de Psicología*. Obtenido de http://publicaciones.konradlorenz.edu.co/index.php/rlpsi/about/submissions#onlineSubmissions

Gómez, I. C. (2011). *Diseño de metodologia para verificar la seguridad en aplicaciones web contra inyecciones SQL.* Bogotá. D.C.: Universidad Militar Nueva Granada.

Gustavo Martínez, Germán DaríoCamacho, Daniel Alberto Biancha. (2010). DISEÑO DE FRAMEWORK WEB PARA EL DESARROLLO DINÁMICO DE. *Scientia et Technica - . Universidad Tecnológica de Pereira*, 6.

Hernández, R., Fernández, C., & Baptista, P. (2006). *Metodología de la investigación.* México: MacGraw-Hill.

López, A. (12 de 01 de 2017). *certsi_*. Recuperado el 13 de 12 de 2016, de certsi_: https://www.certsi.es/blog/owasp-4

MD5online org. (23 de 01 de 2017). *MD5 Online*. Obtenido de http://www.md5online.es/

O.L., A. (16 de 01 de 2017). *Google Academico.* Obtenido de Google Academico: http://www.dsi.uclm.es/personal/FranciscoMSimarro/cedasi/Olsina-tesis.pdf

OWASP. (22 de 01 de 2013). *OWASP*. Obtenido de OWASP: https://www.owasp.org/index.php/Automated_Audit_using_SQLMap

OWASP. (28 de 09 de 2015). *OWASP Top Ten Cheat Sheet.* Obtenido de OWASP Top Ten Cheat Sheet: https://www.owasp.org/index.php/OWASP_Top_Ten_Cheat_Sheet

OWASP. (21 de 01 de 2016). *OWASP PHP Security Project*. Obtenido de OWASP PHP Security Project: https://www.owasp.org/index.php/OWASP_PHP_Security_Project

OWASP. (22 de 01 de 2017). *OWASP Cheat Sheet*. Recuperado el 16 de 01 de 2017, de https://www.owasp.org/index.php/Inyecci%C3%B3n_SQL

OWASP. (17 de 01 de 2017). *OWASP Testing Project.* Recuperado el 14 de 12 de 2016, de OWASP Testing Project: https://www.owasp.org/images/1/19/OTGv4.pdf

OWASP. (05 de 02 de 2017). *OWASP Top Ten Cheat Sheet*. Obtenido de https://www.owasp.org/index.php/OWASP_Top_Ten_Cheat_Sheet

OWASP Zed Attack Proxy Project. (20 de 12 de 2016). *OWASP Zed Attack Proxy Project*. Recuperado el 2017 de 01 de 07, de https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project#tab=Main

Packet Storm. (2016). *Packet Storm*. Obtenido de Packet Storm: https://packetstormsecurity.com/files/10631/rfp2k01.txt.html

Palomares, M. D. (04 de 01 de 2011). *Blog Historia de la Informatica*. Obtenido de Universidad Politecnica de Valencia: http://histinf.blogs.upv.es/2011/01/04/historia-de-las-bases-de-datos/

phpMyAdmin. (23 de 01 de 2017). *phpMyAdmin*. Obtenido de https://www.phpmyadmin.net/

Puneet Sing, J. (2016). Analysis of SQL Ijection Detection Techniques. *CIISE, Concordia Universite, Montreal, Québec, Canada*, 10.

QuintanaLlanes, M. M. (2013). SQL INYECCION. *Universidad Mayor de San Andres*, 3.

Racciatti, H. M. (2002). *Tecnicas de SQL Injection: Un repaso Version 1.5*.

Ramos Ramos, J. L. (2015). PRUEBAS DE PENETRACION O PENT TEST. *Universidad Mayor de San Andres*, 3.

Ramos, J. L. (2013). PRUEBAS DE PENETRACIÓN O PENT TEST. *Revista de Información, Tecnología y Sociedad*, 3. Obtenido de revistasbolivianas.

S.B.D. SecurityDefault.com. (29 de 11 de 2013). *SecurityDefault.com*. Obtenido de http://www.securitybydefault.com/2013/11/quien-descubrio-las-inyecciones-de.html

Sarango, J., & Cevallos, A. (s.f.). *REDUCCIÓN DE ATAQUES INFORMÁTICOS EN APLICACIONES WEB BASADOS EN PHP*. Universidad de las Fuerzas Armadas - ESPE, Sangolquí - Ecuador.

Stephen W. Boyd, A. D. (2004). SQLrand: La prevención de ataques de inyección SQL. En M. Jakobsson, *criptografía aplicada y la seguridad de la red: segunda conferencia internacional*. Berlina: DOI.

Su, Z., & Wasserman, G. (2006). The Essence of Command Injection Attacks in Web Application. *Charleston, South Carolina, USA*, 11.

symantec Corporation World Headquarters. (2016). *Internet Security Threat Report*. Mountain View, CA 94043 USA: Symantec Corporation. Obtenido de https://www.symantec.com/content/dam/symantec/docs/reports/istr-21-2016-en.pdf?aid=elq_&om_sem_kw=elq_16758113&om_ext_cid=biz_email_elq_&elqTrackId=283a3acdb3ff42f4a70ab5a9f236eb71&elqaid=2902&elqat=2

Tarazona Amaya, C. A. (2010). Seguridad en aplicaciones WEB. *OWASP*, 85.

Tejinderdeep Singh Kalsi, N. K. (s.f.). INTENDED APPROACH FOR THE DETECTION AND PREVENTION OF SQL INJECTION ATTACKS. *International Journal of Advanced Engineering Technology*, 3.

William G. J. Halfond, J. V. (2006). A Classification of SQL Injection Attacks and Countermeasures. *Georgia Institute of Technology*.

12. Anexos.

Tabla 2. Objetivos cumplidos sobre la investigación.

Objetivo	Descripción	Cumplimiento
General	Analizar las vulnerabilidades que se pueden presentar en una aplicación web desarrollada en framework CodeIgnite PHP y base de datos MySQL mediante la técnica de "SQL Injection attack" con el fin de proponer buenas prácticas de seguridad informática.	✓
Especifico 1	Documentar la evolución de la técnica SQL inyección en ataques a páginas web desarrolladas en PHP	✓
Especifico 2	Analizar una aplicación Web desarrollada en framework CodeIgnite para PHP y MySQL, utilizando la herramienta OWASP-ZAP, para detectar los posibles agujeros en dicha aplicación, identificando los riesgos que ocasiona la técnica SQL inyección.	Ilustración 7. Imagen "Herramienta en proceso de escaneo para hallar vulnerabilidades en la aplicación" Tomado de la interfaz gráfica de OWASP-ZAP V2.4 – Fuente autores.
Especifico 3	Proponer recomendaciones para la seguridad en aplicaciones Web desarrolladas en PHP v5.6 y MySQL v2.4, para contrarrestar métodos en la técnica SQL inyección, basada en el tipo de ataque: modificación en base de datos.	✓

Fuente autores.

Tabla 3. Presupuesto Global

Rubros	**Fuentes**		**Total**
	Propios	**Contrapartida**	
Personal	$1’512.000		$4’536.000
Equipos	$5’030.000		$5’030.000
Materiales	$1.100.000		$1’080.000
Material bibliográfico	$40.000		$40.000
		Total	**$10’686.000**

Fuente: Autores.

Tabla 4. Gastos de Personal

Nombre	**Rol**	**Dedicación**	**Recursos**		**Total**
			Propios	**Contrapartida**	
Ayala Calderón Carlos	Estudiante	252 horas	$1’512.000		$1’512.000
Amado Ballén Jhon William	Estudiante	252 horas	$1’512.000		$1’512.000
Sierra Morales Armando A.	Estudiante	252 horas	$1’512.000		$1’512.000
				Total	**$4’536.000**

Fuente: Autores.

Tabla 5. Equipos

Equipo	Justificación	Recursos		Total
		Propios	Contrapartida	
Portátil Toshiba Satellite L505,core i3 tercera generación, quad Core 2.3 RAM 4Gb, Tarjeta video HD graphics integrada, SO Windows 10.	Equipo para la realización del proyecto	$1’780.000		$1’780.000
Portátil Asus N550L procesador X64 Core i7, RAM 8Gb, Tarjeta video nvidia GEFORCE 745M, SO Windows 8.	Equipo para realización del proyecto	$2’100.000		$2’100.000
Portatil Toshiba Satellite L4100, core i5, RAM 4Gb, SO Windows 7.	Equipo para la realización del proyecto	$1’150.000		$1’150.000
			Total	**$5’030.000**

Fuente: Autores.

Tabla 6. Materiales

RUBROS	JUSTIFICACIÓN	FUENTES (En pesos)		
		PROPIOS	**Contrapartida**	**Total**
DVD’s o medios magnéticos	Para la copia de los documentos del proyecto presentados a director y biblioteca.	$10.000		$10.000
Papelería	Papelería	$50.000		$50.000
Encuadernación Tesis	Impresión, encuadernación para la presentación del proyecto	$200.000		$200.000
Transporte del equipo de estudiantes (Se refiere a: gastos de transporte en la ciudad de Bucaramanga, para que los equipos puedan desplazarse y reunirse con la directora del proyecto a concretar el mismo)	Costo de transporte para las reuniones del equipo	$300.000		$300.000
Transporte del equipo de estudiantes en el territorio Nacional (Se refiere a: gastos de transporte nacional para la muestra de la ponencia de nuestro artículo científico)	Gastos de desplazamiento para asistencia seminario	$520.000		$520.000
			TOTAL	$1’080.000

Fuente: Autores.