Revisión Sistemática de Literatura: Inyección SQL en Aplicaciones web
Systematic Literature Review: SQL Injection in Web Applications

Jesennia Iñiguez-Banegas, Rene Guaman-Quinche, Robert Figueroa-Diaz and Freddy Ajila-Zaquinaula

***Resumen*— La inyección SQL es una vulnerabilidad de seguridad que afecta a las aplicaciones web. Esto ocurre cuando se inserta una consulta SQL (código malicioso), por medio de las entradas de una interfaz de cliente permitiendo leer y modificar la información. El presente artículo detalla el proceso de la revisión sistemática de literatura sobre estudios primarios que plantean propuestas y solución acerca de inyección SQL. Se siguió el protocolo propuesto por Bárbara Kitchenham y se revisó un total de 9 estudios de varias revistas y conferencias. Las investigaciones sobre inyecciones SQL es todavía un tema abierto, se ha obtenido propuestas para la prevención y detección de la misma. Una de ellas es Hibrid Modeling Framework que hace frente a las vulnerabilidades de inyección SQL en la fase de diseño. Las soluciones expuestas son muchas y diversas, enfocadas en la prevención y detección de vulnerabilidades de inyección SQL.**

***Palabras clave*— mecanismos de seguridad, inyección SQL, frameworks de desarrollo, ataques inyección SQL, seguridad de aplicaciones web.**

***Abstract*— SQL injection is a security vulnerability that affects web applications. This occurs when a SQL (malicious code) query is inserted through the inputs of a client interface allowing you to read and modify information. This article details the process of systematic review of literature on primary studies that raise proposals and solution about SQL injection. Barbara Kitchenham proposed protocol was followed and a total of 9 studies of various journals and conferences was reviewed. Research on SQL injections is still an open issue, it has been obtained proposals for the prevention and detection of it. One is Hibrid Modeling Framework that addresses SQL injection vulnerabilities in the design phase. Exposed solutions are many and diverse, focused on prevention and detection of SQL injection vulnerabilities.**

***Index Terms*— security mechanisms, SQL injection, development frameworks, SQL injection attacks, web application security**.

Este artículo fue enviado para su revisión el 15 de Agosto de 2016

J. Iñiguez-Banegas egresada de la Carrera de Ingeniería en sistemas de la Universidad Nacional de Loja (e-mail:jesenniaib@gmail.com)

R. Guaman-Quinche y R. Figueroa-Diaz, Docentes Investigadores de la Carrera de Ingeniería en Sistemas de la Universidad Nacional de Loja (e-mail: rguaman, roberth.figueroa, pfordoñez@unl.edu.ec)

F. Ajila docente Investigador de la Escuela de Ingeniería Industrial, Facultad de Mecánica de la Escuela Superior Politécnica de Chimborazo (e-mail: freddy.ajila@espoch.edu.ec)

I. INTRODUCCIÓN

LA seguridad del software es una inquietud cada vez más significativa para las instituciones del sector público o privado. Sin embargo, pocos programadores abordan este carácter de calidad de forma estratégica [1].

Los arquitectos y desarrolladores continuamente ponen un énfasis mayor en satisfacer los requerimientos prácticos y funcionales, y la seguridad usualmente es aplicada como un "adicional" para arreglar una vulnerabilidad durante o después de que la aplicación ha sido desarrollada [2].

Desarrollar código encaminado a la seguridad es una tarea que pocos la realizan por ser compleja y [3], por ello, asiduamente se recurre a la adopción y uso de frameworks que se orientan en satisfacer distintas áreas de la seguridad como por ejemplo el control de acceso a los distintos sistemas, el cifrado de la información y la validación de entradas, entre las más importantes [4].

Para mejorar la seguridad en los framework en el diseño de arquitectura, se consideran tres enfoques:

- *Ninguna adopción*: cuando la seguridad no se considera para el diseño de la arquitectura, sino, soluciones adicionales para cubrir aspectos puntuales;
- *Adopción a medias:* se usan frameworks de seguridad luego del diseño inicial de la arquitectura; y
- *Adopción total:* considera la seguridad desde el inicio dentro del diseño de la arquitectura e influye en todo el proyecto [5] [6].

Un framework es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado, por lo tanto se depende de lo sólido y flexible del mismo a la vez lo cual es un problema para el programador si no lo sabe utilizar [7][8]. Siendo así que el 80% de problemas de seguridad de sistemas web se debe que los programadores no configuran los mecanismos de seguridad de frameworks [9].

Las vulnerabilidades de aplicaciones Web se han convertido, en los últimos años, en una gran amenaza para la seguridad de sistemas informáticos [10]. Esta situación se explica por el aumento de la complejidad de tecnologías de la Web [11], por la evolución frecuente de estas tecnologías, por los ciclos cortos de desarrollo de aplicaciones Web durante el cual las actividades de prueba y validación son limitados, y también, en algunos casos, por la falta de seguridad habilidades y cultura de los desarrolladores [12].

Según Owasp en el año 2014 el 98% de las aplicaciones web son vulnerables, lo que da como resultados un promedio de vulnerabilidades por aplicación del 20%. El servicio de almacenamiento en la nube FireHost reporta que el número de ataques de inyección de código SQL fue de cerca del 69% en el año 2012. Según un reporte, los servidores localizados en centros de datos alrededor de Europa y EUA registraron al menos medio millón de estos ataques en abril y junio de 2012; menos de 300 mil fueron registrados durante el primer trimestre [13] [14].

De las estadista descritas sobre los ataque de Inyección SQL, es necesario que se generen investigaciones para contribuir a que la información de los usuario tenga los principios de integridad, disponibilidad y confidencialidad, por ello, el propósito de este artículo es mostrar el resultado de la revisión sistemática de literatura, que fue orientada en estudios actuales en inyección SQL, la sección II presenta la metodología para desarrollar la revisión sistemática y extracción de información basada en [15][16]. En la sección III presenta los resultados obtenidos en tablas de estudios relevantes y en la sección IV se discute los principales hallazgos y la sección V se define las conclusions del presente artículo.

II. METODOLOGÍA

Basado en la metodología de revisiones sistemáticas de Bárbara Kitchenham se elaboró un esquema para la revisión, selección y extracción de información quedando de la siguiente manera:

a. Pregunta de investigación.
b. Palabras clave.
c. Método de revisión.
- Fuentes y estrategias de búsqueda
- Cadenas de búsqueda,
- Criterios de selección de estudios.
- Extracción de información.

d. Estudios incluidos y excluidos

Además se utiliza Mendeley, como gestor bibliográfico para almacenar y organizar los estudios y sus referencias.

A. Pregunta de investigación.

Se dirigió el alcance de este trabajo sobre artículos relacionados a mecanismos de seguridad aplicando frameworks de desarrollo. La pregunta de investigación planteada es:

¿Qué tipos de estudios primarios existen sobre mecanismos de seguridad para inyección SQL en frameworks de desarrollo?

B. Palabras clave.

Se realizó una revisión de literatura previa, que consistió en analizar algunos documentos relacionados al tema que facilitan identificar las palabras claves obtenidas de los títulos, resúmenes e introducción.

En la tabla 1 se detalla la lista de palabras obtenidas a través del Keywords.Equations

TABLA 1. REVISIÓN PRELIMINAR Y TÉRMINOS.

Cód.	Titulo	Palabras clave
R01	Towards SQL Injection Attacks Detection Mechanism Using Parse Tree	SQL injection attacks, parse tree, detection, web environments.
R02	Securing Web Applications from Injection and Logic Vulnerabilities: Approaches and Challenges	SQL injection, Cross-site scripting, Business logic vulnerabilities, Application logic vulnerabilities, Web application security, Injection flaws
R03	Effective detection of vulnerable and malicious browser extensions	Browser extensions, Web security, Malware, Hidden Markov Model, JavaScript
R04	Mitigating SQL Injection Attacks Via Hybrid Threat Modelling	SQL Injection Attacks, Software Security, SDLC, SSDL, Hybrid Threat Modeling, Attack Trees, Misuse Cases, State Machines
R05	Securing Web Applications from Injection and Logic Vulnerabilities: Approaches and Challenges	SQL injection, Cross-site scripting, Business logic vulnerabilities, Application logic vulnerabilities, Web application security, Injection flaws

Una vez obtenidas las palabras claves descritas en la tabla 1, se puede realizar la construcción de la cadena de búsqueda.

C. Método de revisión

1) Fuentes y estrategias de búsqueda

- SCOPUS Library: https://www.scopus.com
- SCIENCEDIRECT Library: http://www.sciencedirect.com
- IEEEXPLORE Library: http://ieeexplore.ieee.org/

2) Cadenas de búsqueda

A partir de la pregunta de investigación, se definieron palabras clave para las búsquedas: Security mechanisms, SQL injection, development frameworks, SQL injection attacks, web application security.

Para generar la cadena de búsqueda se utilizaron los operadores lógicos “OR” y “AND”, quedando:
(Security mechanisms and SQL injection or SQL injection attacks and development frameworks or web application security).

Criterios de inclusión.

Es necesario aclarar que se consideró los siguientes criterios de búsquedas:
- Considerar sólo publicaciones desde el año 2014 en adelante.

- Los resultados de la búsqueda solo sean en el área de Ciencias y Computación.
- Las producciones científicas sean estudios primarios (artículos de conferencia, artículos de revista).
- La búsqueda por su relevancia científica será en el idioma inglés.
- Los estudios deben tener información relevante a la pregunta de investigación.

Criterios de exclusión.

Los estudios que no han sido relevantes en este estudio se han excluido mediante los siguientes criterios:

- Publicaciones informales que no siguen una metodología científica.
- Todas las que no cumplan con los criterios de inclusión.

Las cadenas de búsquedas(C) utilizadas fueron las siguientes:

Biblioteca digital de SCOPUS Library:

C01: TITLE-ABS-KEY (security mechanisms AND sql injection OR sql injection attacks AND development frameworks OR web application security) AND (LIMIT-TO (PUBYEAR , 2016) OR LIMIT TO (PUBYEAR , 2015) OR LIMIT-TO (PUBYEAR , 2014)) AND (LIMIT-TO (SUBJAREA , "COMP"))

Biblioteca digital de SCIENCEDIRECT Library:

C02: ALL (Security mechanisms and SQL injection or SQL injection attacks and development frameworks or web application security) AND LIMIT-TO (yearnav, "2016, 2015, 2014") AND LIMIT-TO (cids, "271887","Computers & Security") AND LIMIT-TO (contenttype, "JL, BS","Journal")

Biblioteca digital de IEEEXPLORE Library:

C01: (Security mechanisms and SQL injection or SQL injection attacks and development frameworks or web application security)

3) Criterios de selección de estudios

Obtenidos los resultados de las búsquedas es conveniente describir el criterio a seguir para la selección de estudios primarios, considerando los siguientes:

- Presenten en el resumen, información actual de mecanismos de seguridad para inyección SQL en frameworks de desarrollo.
- Contener información relevante para la revisión en la introducción o conclusión.

4) Extracción de información

Los criterios de selección de estudios establecen la pauta de extracción de información relevante para este trabajo. Por cada artículo seleccionado, se sintetizará al menos uno de los siguientes elementos:

- Propuestas o modelos para prevenir inyecciones SQL
- Resultados
- Conclusiones relevantes.

D. Estudios incluidos y excluidos

El criterio utilizado para la selección de artículos fue que aportaran sobre la existencia de mecanismos de seguridad en los framework.

Las búsquedas realizadas generaron 24 artículos, de los cuales se registraron 5 coincidencias, es decir el número de artículos revisados fueron 19, de los cuales se seleccionaron 9 artículos de acuerdo al criterio ya mencionado.

TABLA 2. ARTÍCULOS INCLUIDOS Y EXCLUIDOS

	Artículos			
Base de Datos	Encontrados	Coincidencias	Revisados	Seleccionados
Scopus	10	2	8	6
ScienceDirect	13	3	10	2
IEEE	1	0	1	1
Total	24	5	19	9

III. RESULTADOS

Las siguientes tablas muestra la información relevante extraída de cada uno de los artículos seleccionados.

TABLA 3. RESULTADOS DEL ARTÍCULO SA01.

Mecanismos de seguridad	Se exhibe un esquema novedoso que transforma automáticamente las aplicaciones web, haciéndolas seguras contra ataques de inyección SQL. Esta técnica analiza dinámicamente el tamaño resultado de la consulta desarrollador destinados a cualquier entrada, y detecta los ataques de comparar esto contra el resultado de la consulta real.
Resultados	La evaluación empírica demostró que IDL (Injection Detector Libraries) consume más tiempo para la detección de los algoritmos existentes, ya que incluye muchos pasos que se consideran métodos sintácticos para dar resultados más precisos. Sin embargo este es un método que puede detectar con mayor precisión que SQLIA.
Conclusiones relevantes	Mediante el uso de una variable de entrada de sustitución y desinfección basada en el tamaño de la consulta, es posible detectar y prevenir las consultas SQL que incluyen vulnerabilidades de inyección.

TABLA 4. RESULTADOS DEL ARTÍCULO SA02.

Mecanismos de seguridad	Una nueva técnica o método para detectar SQLIA mediante el modelado de las consultas SQL como una gráfica de tokens y el uso de la medida de centralidad de los nodos para entrenar a una máquina de vectores soporte (SVM).
Resultados	Los resultados experimentales demuestran que esta técnica puede identificar con eficacia las consultas SQL maliciosas con sobrecarga de rendimiento insignificante.
Conclusiones relevantes	El sistema no requiere la construcción de un modelo de uso normal de las consultas, ni requiere el acceso al código fuente.

TABLA 5. RESULTADOS DEL ARTÍCULO SA03.

Mecanismos de seguridad	Propone una metodología de la vulnerabilidad y ataque de inyección para SQLi y XSS se puede aplicar a una variedad de configuraciones y tecnologías. Se basa en la idea de que podemos evaluar diferentes atributos de los mecanismos de seguridad de aplicaciones web existentes mediante la inyección de vulnerabilidades realistas en una aplicación web y atacar de forma automática.

Resultados	Los resultados muestran que la inyección de vulnerabilidades y ataques es de hecho una forma eficaz para evaluar los mecanismos de seguridad y para señalar no sólo sus debilidades, si no también formas para su mejora.
Conclusiones relevantes	Concluye que aproximadamente la mitad de las vulnerabilidades SQLi provienen de la explotación de los campos numéricos.

TABLA 6. RESULTADOS DEL ARTÍCULO SA04.

Mecanismos de seguridad	Un mecanismo de detección concreta basado en DSD (Dynamic Detección SQLIAs) se plantea para detectar SQLIAs mediante el uso de árbol de análisis sintáctico. La principal ventaja de la propuesta es que no requiere acceder al código fuente de las aplicaciones si no que es incorporado para entornos web existentes.
Resultados	Los resultados experimentales demostraron que el mecanismo tiene una mayor precisión (99.9%), menor tasa de falsos positivos (2%) y falsos negativos cuando se detecta SQLIAs. Por lo tanto es un eficiente mecanismo de detección SQLAS para entornos web.
Conclusiones relevantes	El mecanismo propuesto no requiere acceder al código fuente de las aplicaciones. Esto significa que DSD se puede aplicar directamente a aplicaciones web existentes. Por lo tanto es un eficiente mecanismo de detección de SQLIAs para entornos web.

TABLA 7. RESULTADOS DEL ARTÍCULO SA05.

Mecanismos de seguridad	Propone una amenaza Hybrid Modeling Framework, polilla, para hacer frente a vulnerabilidades de inyección SQL en la fase de diseño, una fase de desarrollo temprana del SDLC (ciclo vital del desarrollo/diseño de sistemas).
Resultados	Como resultado de los puntos de entrada ampliado, los investigadores han desplazado los engranajes de los enfoques reactivos prevalentes de SQLIAs la prevención de una estrategia de gestión de riesgos proactiva llamada de amenaza de modeling un ejercicio realizado en la fase de diseño del SDLC.
Conclusiones relevantes	La seguridad es un proceso continuo que debe ser integrado en las aplicaciones desarrolladas de solicitud a través de la liberación de mantenimiento.

TABLA 8. RESULTADOS DEL ARTÍCULO SA06.

Mecanismos de seguridad	Un enfoque basado en la técnica HMM (Modelo oculto de Markov) para detectar las extensiones del navegador vulnerable y malicioso, ampliando y complementando las técnicas existentes. Estas técnicas se centran principalmente en el análisis de flujo de información.
Resultados	Se implementa en una herramienta de prototipo y evaluó utilizando un número de 387 extensiones de Mozilla Firefox. Indican que el enfoque no sólo detecta extensiones vulnerables y maliciosos conocidas, sino que también identifica previamente no detectados, extensiones con una sobrecarga de rendimiento insignificante. La precisión de falso positivo 97,68%.
Conclusiones relevantes	El número de muestras utilizadas durante la evaluación es pequeño para apoyar la eficacia de HMM, nuestro enfoque se puede utilizar como una técnica complementaria a los enfoques existentes.

TABLA 9. RESULTADOS DEL ARTÍCULO SA07.

Mecanismos de seguridad	Una revisión de la literatura que resume el estado actual de la técnica para asegurar las aplicaciones web de los principales defectos tales como errores de inyección y lógicas. Aunque existen diferentes tipos de errores de inyección, el alcance se limita a la inyección de SQL (SQLI) y Cross-site scripting (XSS), ya que son calificados como los mejores entre la mayoría de las amenazas de los diferentes consorcios de seguridad.
Resultados	Se necesita más investigación en el área de los defectos de fijación en el código fuente de las aplicaciones. La mayoría de los artículos se centran en la detección de los defectos y la prevención de ataques contra las aplicaciones web.
Conclusiones relevantes	A pesar de que varios enfoques están disponibles para asegurar las aplicaciones web de SQLI y XSS, son todavía muy extendido debido a su impacto y la gravedad. Este articulo propocion una revisión integral de los recientes avances en la obtención de las vulnerabilidades de inyección y la lógica de negocio de las aplicaciones web, y señala los problemas no resueltos que deben abordarse.

TABLA 10. RESULTADOS DEL ARTÍCULO SA08.

Mecanismos de seguridad	Marco de seguridad adopción de Cloud Computing (CCAF) adecuada para negocios nubes. Se basa en el desarrollo y la integración de las tres principales tecnologías de seguridad: firewall, gestión de identidad y cifrado basado en el desarrollo de la empresa, sincronización de archivos y las tecnologías de Acciones.
Resultados	Los resultados en la primera hora y 24 horas pruebas mostraron que una CCAF protección de seguridad completa de múltiples capas podría bloquear y evitar la inyección de SQL para MySQL y MongoDB. En las pruebas de penetración, de seguridad en capas múltiples CCAF podría detectar y bloquear el 99,95%
Conclusiones relevantes	Una protección de seguridad multicapa completa CCAF podría bloquear toda inyección SQL y proporcionar una protección real a los datos.

TABLA 11. RESULTADOS DEL ARTÍCULO SA09.

Mecanismos de seguridad	Nueva metodología, basada en técnicas de agrupamiento de las páginas web, que está dirigido a identificar las vulnerabilidades de una aplicación web después de un análisis de cuadro negro de la aplicación de destino.
Resultados	Este enfoque también condujo al desarrollo de un nuevo escáner de vulnerabilidad denominada Wasapy.
Conclusiones relevantes	Enriquecer las gramáticas implementadas en Wasapy para permitir la generación de una variedad más grande de inyecciones que cubren las vulnerabilidades incluidos hasta el momento, así como las nuevas vulnerabilidades.

IV. DISCUSIÓN

Principales hallazgos

SA01: Se describe una técnica de adulteración positivo que caracterizan IZES el proceso de desinfección mediante el modelado de la forma en que una aplicación procesa los valores de entrada. En base al uso de una variable de entrada de sustitución y desinfección basada en el tamaño de la consulta, es posible detectar y prevenir las consultas SQL que incluyen vulnerabilidades de inyección. La evaluación empírica demostró que IDL (Injection Detector Libraries) a pesar de consumir más tiempo en la detección de algoritmos, es eficaz contra el conjunto de pruebas SQLIA. Los IDL tienen tres pasos principales: Paso 1 comprueba los patrones de ataque contra las expresiones regulares. Si ninguna regla coincide, entonces ese patrón de ataque se envía al sistema de detección. Paso 2 analiza el patrón de ataque utilizando nuestra base de datos interna, llamado un "conjunto de reglas," para clasificar la vulnerabilidad. Para evaluar la cadena de consulta, el IDL utiliza un analizador de SQL para dividirla en

una secuencia de símbolos que corresponden a palabras clave de SQL, operadores y literales. El IDL luego itera a través de los tokens y comprueba si las que no son literales contenen exclusivamente los datos de confianza. Si todas las fichas pasan esta comprobación, la consulta se considera segura y se puede ejecutar. Por último, el paso 3 se sutituye caracteres o cadenas vulnerables en la consulta SQL. En particular, esto incluye funciones que eliminan o sustituyen ciertos caracteres o cadenas de su entrada.

SA02: Muestra una nueva técnica para detectar SQLIA mediante el modelado de las consultas SQL como una gráfica de tokens y el uso de la medida de centralidad de los nodos para entrenar a una máquina de vectores soporte (SVM). El enfoque fue diseñado para trabajar en la capa de base de datos y servidor de seguridad se implementó en un prototipo llamado SQLiGoT, además utiliza las funciones disponibles en la mayoría de los lenguajes de programación modernos y puede ser portado a otras plataformas sin necesidad de grandes modificaciones. El sistema fue probado exhaustivamente el uso de grafos no dirigidos y dirigidas con dos diferentes métodos de borde de ponderación. Proponen diseños alternativos de la SVM clasificador, que consiste en simples y múltiples, probados y comparados. Los resultados experimentales obtenidos en cinco aplicaciones web totalmente vulnerables confirman la eficacia que tiene. El sistema no requiere la construcción de un modelo de uso normal de las consultas, ni requiere el acceso al código fuente.

SA03: La metodología propuesta ofrece un entorno práctico que se puede utilizar para probar mecanismos de contramedida (como los sistemas de detección de intrusos (IDS), escáneres de vulnerabilidades de aplicaciones web, paredes de fuego de aplicaciones web, analizadores de código estático, etc.), y el tren evaluar los equipos de seguridad, ayudar a estimar las medidas de seguridad (como el número de vulnerabilidades presentes en el código), entre otros. Esta evaluación de herramientas de seguridad puede realizarse en línea mediante la ejecución del inyector de ataque, mientras que la herramienta de seguridad también está en marcha; o fuera de línea mediante la inyección de un conjunto representativo de las vulnerabilidades que se pueden utilizar como banco de pruebas para evaluar una herramienta de seguridad. La metodología se llevó a cabo en una vulnerabilidad de hormigón y herramienta del inyector Ataque (Vait) para aplicaciones web. El primero en evaluar la eficacia de Vait en la generación de un gran número de vulnerabilidades realistas para la evaluación en línea de herramientas de seguridad, en particular los escáneres de vulnerabilidades de aplicaciones web. El segundo para mostrar cómo se puede explotar las vulnerabilidades inyectadas para lanzar ataques, permitiendo la línea evaluación de la eficacia de los mecanismos de contramedida instalados en el sistema de destino, en particular un sistema de detección de intrusos. Estos experimentos ilustran cómo propuesta se puede utilizar en la práctica, no sólo para descubrir debilidades existentes de las herramientas analizadas, sino también para ayudar a mejorarlas.

SA04: Propone un interesante mecanismo de detección concreta basada en DSD, se plantea para detectar SQLIAs mediante el uso de árbol de análisis sintáctico. La principal ventaja de la propuesta es que no requiere acceder al código fuente de las aplicaciones si no que es incorporado para entornos web existentes. El DDS consiste en cinco unidades: Collector1, Collector2, Repositorio1, Repositorio2, y Agente y SQLIAs. El mecanismo consta de dos fases: de clasifición y detección. Cuando un usuario envía una solicitud HTTP a una aplicación, la fase de clasificación está involucrado para identificar la solicitud si se trata de primer acceso en tiempo o tiempo de acceso no primero. Después de eso, la fase de detección proporciona una detección SQLIA para esta aplicación en los dos casos anteriores. La exactitud de este mecanismo es más de 99.9% para cada tipo de aplicación típica, la tasa de falsos positivos es menos de 2% para cada tipo de aplicaciones.

.

SA05: Estudio de amenaza Hybrid Modeling Framework, polilla, para hacer frente a vulnerabilidades de inyección SQL en la fase de diseño, una fase de desarrollo temprana del SDLC, el modelado de amenazas implica el descubrimiento de la superficie de ataque explotable del activo de software mediante el examen de todos los límites de confianza, el flujo de datos, incluyendo los caminos de entrada y salida de todos los puntos de entrada. Los casos de mal uso (CUG), árboles de ataque (ATS) y máquinas de estado de comportamiento (MAN) se combinan en una técnica híbrida para diseñar un modelo de amenazas necesaria para proporcionar los requisitos de seguridad optimas y activos de software, concluyendo que la seguridad es un proceso continuo que debe ser integrado en las aplicaciones desarrolladas de solicitud a través de la liberación de mantenimiento.

SA06: Se enfoca en técnica basada en HMM (Modelo oculto de Markov) para detectar las extensiones del navegador vulnerable y malicioso, ampliando y complementando las técnicas existentes. Estas técnicas se centran principalmente en el análisis de flujo de información, para capturar los flujos de datos sospechosos, imponer la restricción de privilegios de llamadas a la API de extensiones maliciosos, aplicar firmas digitales para supervisar las actividades del proceso y el nivel de memoria, y permitir a los usuarios del navegador especificar las políticas con el fin de restringir las operaciones de extensiones. Se implementa en una herramienta de prototipo y evaluó utilizando un número de 387 extensiones de Mozilla Firefox. Indican que el enfoque no sólo detecta extensiones vulnerables y maliciosos conocidas, sino que también identifica previamente no detectados, extensiones con una sobrecarga de rendimiento insignificante. La precisión de falso positivo 97,68%.

SA07: Presenta una revisión sistemática de la literatura de los recientes avances en la obtención de las vulnerabilidades de inyección de las aplicaciones web. El objetivo de este estudio es resumir el estado actual de la técnica para asegurar las aplicaciones web de los principales defectos tales como errores de inyección y lógicas. Se exploran principalmente los siguientes puntos:

- Se analizan diversos tipos de vulnerabilidades y ataques que explotan estas vulnerabilidades en aplicaciones web.

- Se analizan los pros y los contras de los enfoques de mitigación para proteger las aplicaciones web de inyección y de negocios vulnerabilidades lógicas
- Proporciona información sobre las capacidades de los escáneres de vulnerabilidad existentes.
- Se destacan las aplicaciones web de código abierto que pueden utilizarse para la prueba y evaluación.

A pesar de que varios enfoques están disponibles para asegurar las aplicaciones web de SQLI y XSS, son todavía muy extendido debido a su impacto y la gravedad

SA08: framework de seguridad de varios niveles adopción de Cloud Computing (CCAF) adecuada para negocios en las nubes. Se basa en el desarrollo y la integración de las tres principales tecnologías de seguridad: firewall, gestión de identidad y cifrado basado en el desarrollo de la empresa, sincronización de archivos y las tecnologías de Acciones. Describe la tecnología de seguridad básica de Empresa sincronización de archivos y Compartir, la arquitectura y componentes en capas, y las tecnologías y resultados básicos de varias capas de experimentos a gran escala para las pruebas de penetración, inyección SQL y escaneo de datos. Dando como resultado en las pruebas de penetración que podría detectar y bloquear el 99,95% los virus troyanos y mantener un 85%, por encima de bloquear durante 100 horas continúas de ataques. Una protección de seguridad multicapa completa CCAF podría bloquear toda inyección SQL que proporciona una protección real a los datos. CCAF seguridad multicapa tenía tasa de 100% de no informar de falsa alarma.

SA09: nueva metodología que permite identificar automáticamente las vulnerabilidades residuales de una aplicación web a partir del análisis de la aplicación específica, siguiendo un enfoque cuadro negro puesto que no requiere detalles de la implementación del código fuente de la página Web. Está diseñado para poner de relieve los posibles escenarios de ataque, incluyendo la explotación de vulnerabilidades varios sucesivos que no son necesariamente independientes. Utilizan una herramienta Wasapy (Evaluación de la Seguridad Web de aplicaciones en Python.) de software utilizando el lenguaje Python, lo que facilita enormemente el manejo de conceptos HTTP (cookies, configuraciones, etc.). Se implementa un escáner de vulnerabilidades Web que contribuyen a enriquecer las gramáticas implementadas en Wasapy para permitir la generación de una variedad más grande de inyecciones que cubren las vulnerabilidades incluidos hasta el momento, así como las nuevas vulnerabilidades.

No hay una solución única hasta el momento que pueda eliminar las vulnerabilidades y prevenir ataques SQL. Por lo tanto, una serie de técnicas de mitigación debe ser empleado para frenar la propagación de los ataques y eliminar las vulnerabilidades SQL.

La solución más ideal es eliminar vulnerabilidades SQL de la raíz, es decir, el código fuente. Sin embargo, en las aplicaciones web en el mundo real, la obtención del código fuente o parches puede ser difícil. Por lo tanto, las técnicas de análisis estático son más útiles durante el desarrollo de la aplicación y antes del despliegue. Las técnicas de análisis dinámicos como técnica de pruebas de penetración pueden ser utilizadas para explotar las aplicaciones web durante el tiempo de ejecución con el fin de determinar si todavía son vulnerables a ataques SQL.

V. CONCLUSIONES Y TRABAJOS FUTUROS

El trabajo se centra en 9 artículos relacionados con la investigación de inyección SQL. Se identifica las soluciones, métodos, técnicas propuestas en los estudios. Las soluciones propuestas son muchas y diversas, en su mayoría se enfocan en la prevención de ataques de inyección SQL y detección de vulnerabilidades. Solo el estudio SA05, discute la eliminación de vulnerabilidades de inyección SQL a partir del código fuente, lo que es importante para prevenir ataques y ahorrar recursos en post-implementación. Los artículos SA01, SA02, SA08, hacen un análisis sintáctico a través de tokens, SVM (máquina de vectores soporte), IDL (Injection Detector Libraries) para la detección de SQLIA. Los documentos SA03, SA04, SA06, SA09, realizan análisis semánticos de diferentes mecanismos de seguridad como DSD (Detección SQLIAs Dinámico), IDS (Sistema de detección de intrusos), HMM (Modelo oculto de Markov), CCAF (Framework de adopción de Cloud Computing) y Wasapy en la detección de vulnerabilidades. Y un estudio proporciona una revisión integral de los recientes avances en la obtención de las vulnerabilidades de inyección. Todos los estudios hacen frente a los problemas relacionados con inyecciones SQL para eliminarlos. Pero los ataques son cada vez más frecuentes, así que la seguridad debe ser tratada en todas los faces de desarrollo considerando la seguridad desde el principio y en todo el ciclo de vida de la aplicación y usar framework para soportarla puede dar excelentes resultados. El análisis de estudios primarios indican la facilidad con la que puede ser vulnerada una aplicación cunado no se le asigna una prioridad adecuada a los controles de seguridad en las distintas etapas de desarrollo.

En el futuro, realizar una nueva Revisión Sistemática en cuanto a los de mecanismos de seguridad para mitigar la inyección SQL en aplicaciones web, compáralo con nuestro estudio y saber cuáles han sido los avances en la seguridad a partir de la publicación de mismo. Realizar estudios secundarios para obtener información acerca de que estudios existen de los diferentes tipos de escáneres de vulnerabilidades existentes para mitigar la Inyección SQL en las aplicaciones web y las diferentes vulnerabilidades de las aplicaciones web.

VI. ARTÍCULOS SELECCIONADOS EN LA REVISIÓN SISTEMÁTICA.

SA01.-Y. S. Jang and J. Y. Choi, “Detecting SQL injection attacks using query result size,” Comput. Secur., vol. 44, pp. 104–118, 2014.

SA02.-D. Kar, S. Panigrahi, and S. Sundararajan,

"SQLiGoT: Detecting SQL Injection Attacks using Graph of Tokens and SVM," Comput. Secur., 2016.

SA03.-J. Fonseca, N. Seixas, M. Vieira, and H. Madeira, "Analysis of field data on web security vulnerabilities," IEEE Trans. Dependable Secur. Comput., vol. 11, no. 2, pp. 89–100, 2014.

SA04.-T. N. Aung and S. S. Khaing, "Genetic and Evolutionary Computing," Adv. Intell. Syst. Comput., vol. 388, pp. 405–411, 2016.

SA05.-H. Omotunde and R. Ibrahim, "Mitigating SQL injection attacks via hybrid threat modelling," 2015 IEEE 2nd Int. Conf. InformationScience Secur. ICISS 2015, pp. 15–18, 2016.

SA06.-H. Shahriar, K. Weldemariam, M. Zulkernine, and T. Lutellier, "Effective detection of vulnerable and malicious browser extensions," Comput. Secur., vol. 47, pp. 66–84, 2014.

SA07.-G. Deepa and P. S. Thilagam, "Securing web applications from injection and logic vulnerabilities: Approaches and challenges," Inf. Softw. Technol., vol. 74, pp. 160–180, 2016.

SA08.-V. Chang, Y. H. Kuo, and M. Ramachandran, "Cloud computing adoption framework: A security framework for business clouds," Futur. Gener. Comput. Syst., vol. 57, pp. 24–41, 2016.

SA09.-R. Akrout, E. Alata, M. Kaaniche, and V. Nicomette, "An automated black box approach for web vulnerability identification and attack scenario generation," J. Brazilian Comput. Soc., vol. 20, no. 1, p. 4, 2014.

REFERENCIAS

[1] R. A. Oliveira, N. Laranjeiro, and M. Vieira, "Assessing the security of web service frameworks against Denial of Service attacks," J. Syst. Softw., vol. 109, pp. 18–31, 2015.

[2] M. Castro-león, F. Boixader, M. Taboada, D. Rexachs, E. Universitària, and T. Cerdà, "Servicios y Seguridad , un enfoque basado en estrategias de ataque y defensa," pp. 39–48, 2015.

[3] D. CAMACHO, G. MARTINEZ, and D. BIANCHA, "Diseño De Framework Web Para El Desarrollo Dinamico De Aplicaciones," no. 44, pp. 178–183, 2010.

[4] M. D. P. Salas-Zárate, G. Alor-Hernández, R. Valencia-García, L. Rodríguez-Mazahua, A. Rodríguez-González, and J. L. López Cuadrado, "Analyzing best practices on Web development frameworks: The lift approach," Sci. Comput. Program., vol. 102, pp. 1–19, 2015.

[5] H. Cervantes, R. Kazman, and J. Ryoo, "Seguridad y uso de Frameworks _ SG." p. SG # 47, 2015.

[6] A. R. Sartorio, G. L. Rodríguez, and M. Vaquero, "Investigación en el diseño y desarrollo para el enriquecimiento de un framework colaborativo web sensible al contexto," XIII Work. Investig. en Ciencias la Comput., pp. 1–5, 2011.

[7] C. García, R. Hervás, and P. D. A.-/9 L. B.-G. Gervás, "Una Arquitectura Software para el Desarrollo de Aplicaciones de Generación de Lenguaje Natural," Soc. Española para el Proces. del Leng. Nat. Proces. Leng. Nat., vol. 33, pp. 111–118 ST – Una Arquitectura Software para el De, 2004.

[8] G. Martínez Villalobos, G. D. Camacho Sánchez, and D. A. Biancha Gutiérrez, "Diseño de Framework web para el desarrollo dinámico de aplicaciones," Sci. Tech., vol. XVI, no. 44, pp. 178–183, 2010.

[9] H. T. Quinche, René Guamán, "Seguridad en Entornos Web para Sistemas de Gestión Académica," pp. 1–47.

[10] R. Akrout, E. Alata, M. Kaaniche, and V. Nicomette, "An automated black box approach for web vulnerability identification and attack scenario generation," J. Brazilian Comput. Soc., vol. 20, no. 1, p. 4, 2014.

[11] A. María Reina Quintero, "Separación avanzada de conceptos en entornos WEB.," pp. 3–16.

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

[13] Owasp, "OWASP Top 10 - 2013," OWASP Top 10, p. 22, 2013.

[14] J. I. Calderón, "Seguridad en Aplicaciones Web."

[15] S. E. Group and R. Unido, "Directrices para la realización sistemática de la literatura críticas en Ingeniería de Software Sección de Control de Documentos," 2007.

[16] B. Kitchenham, "Procedures for performing systematic reviews," Keele, UK, Keele Univ., vol. 33, no. TR/SE-0401, p. 28, 2004.

Jesennia Iñiguez, Egresada de la Carrera de Ingieneria en Sistemas de la Univesidad Nacional de Loja. Líneas de interés seguridad web, redes y telecomunicacones. Docente en Unidad Educativa Calazanz (Septiembre 2013 – Enero 2014). Instructora de computación en Compucenter Technology (Sept. 2014 – Noviembre 2014). Técnico y operados de cibercafé en Gyg@net (Enero 2012 – Noviembre 2014). Ciudad Loja, Ecuador, 2016

Rene Guamán Quinché, Docente Investigador de la Carrera de Ingeniería en Sistemas de la Universidad Nacional de Loja. Líneas de interés en tecnologías web y móviles, sistemas distribuidos y paralelos. Máster en Sistemas Informáticos Avanzados en la Universidad del País Vasco. Ciudad Loja, Ecuador, 2016

Roberth Figueroa Díaz, Ingeniero en Sistemas Informáticos y Computación, graduado en la Universidad Técnica Particular de Loja - Ecuador, máster en Ingeniería del Software para la Web por la Universidad de Alcalá - España. Es certificado en Sistemas Linux otorgado por IBM Advanced Career Education y por Microsoft Certified Program. Docente en la Universidad Técnica Particular de Loja, Universidad Internacional del Ecuador y Universidad Nacional de Loja, así como analista desarrollador de software en la Unidad de Proyectos y Sistema Informáticos de la UTPL. Líneas de interés están la Ingenieria del Software, Ciencia de datos. Inteligencia Artificial, Internet de las cosas y Big Data.

Freddy Ajila, Docente Investigador de la Escuela de Ingeniería Industrial, Facultad de Mecánica de la Escuela Superior Politécnica de Chimborazo ESPOCH (Desde Octubre 2014 hasta la actualidad). Profesor de Sistemas Operativos, Arquitectura de Computadoras y Estructuras de Datos de la Carrera de Ingeniería en Sistemas de la Universidad Nacional de Loja – Ecuador (Abril 2013 – Julio 2014). Magister en Telemática graduado en la Universidad de Cuenca – Ecuador (Agosto 2011). Ingeniero en Informática graduado en la Universidad Técnica Particular de Loja – Ecuador (Junio del 2006). Activista de software libre, Administrador de servidores, redes y telecomunicaciones. Provincia de Chimborazo, Ciudad Riobamba, Ecuador, 2016.